Actes ROADEF 2010

11e congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision

Recueil des articles longs

LAAS-CNRS

INSTITUT de MATHEMATIQUES de TOULOUSE

IRIT CNRS - INPT - UPS - UT1 - UTM

Université de Toulouse

Avant-propos

Le 11e congrès annuel de la Société Française de Recherche Opérationnelle et d'Aide à la Décision est organisé en 2010 par le Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS), l'Institut de Mathématiques de Toulouse (IMT) et l'Institut de Recherche en Informatique de Toulouse (IRIT), avec la participation des principaux établissements du PRES Université de Toulouse : Université Toulouse 1 Capitole, Université Toulouse 2 Le Mirail, Université Paul Sabatier, Institut National Polytechnique de Toulouse (INPT), Institut National des Sciences Appliquées de Toulouse (INSAT), Institut Supérieur de l'Aéronautique et de l'Espace (ISAE) et École Nationale de l'Aviation Civile (ENAC). Le congrès a lieu du 24 au 26 février 2010 sur le site de la Manufacture des Tabacs de l'Université Toulouse 1 Capitole.

Les nombreux chercheurs de Toulouse (la ville ROse !) en RO-AD, dispersés dans les trois universités, les écoles d'ingénieurs et les autres centres de recherches, se sont réunis pour l'organisation de ROADEF 2010. La liste des membres du comité d'organisation apparaît en page ii de cet ouvrage.

Le traditionnel (depuis 2005 !) appel à articles longs, dont le premier auteur est un doctorant, a été lancé le 26 mai 2009 avec date limite de soumission fixée au 23 octobre 2009. L'objectif de ces actes est d'offrir aux doctorants la possibilité d'une publication rapide de leurs travaux, tout en assurant une évaluation sérieuse par des spécialistes de leur domaine. Cette année, nous avons explicitement sollicité des articles rédigés en français. Nous présentons dans le présent ouvrage le fruit d'une sélection de 14 articles longs parmi les 29 qui ont été soumis, ce qui donne un **taux de sélection de 48%**. Chaque article a été relu par deux arbitres anonymes. Les auteurs des articles retenus pouvaient rendre leur version définitive, prenant en compte les modifications mineures demandées par les arbitres, jusqu'au 15 janvier 2010.

Cette collection d'articles représente une grande diversité d'approches, de techniques et de domaines d'application de la RO-AD. Les problématiques abordées dans les articles que nous avons retenus relèvent de la théorie des graphes, des métaheuristiques, de la programmation non-linéaire mixte, de la théorie de la décision et des problèmes d'ordonnancement. Les domaines d'application incluent les transports, les télécommunications, les sciences du vivant et l'industrie pharmaceutique.

Nous tenons à remercier ici les arbitres qui ont accepté de rendre leur rapport dans un temps très court. Il s'agit des membres du comité scientifique dont la liste est présentée en page iii, ainsi que les arbitres suivants : Mohamed Ali Aloulou, Christian Artigues, Nahla Ben Amor, Olivier Briant, Alexandre Caminada, Yann Chevaleyre, Clarisse Dhanens, Didier Dubois, Niklaus Eggenberg, Dominique Feillet, Semi Gabteni, Adrien Goëffon, Philippe Lacomme, Wim Nuijten, André Rossi, Francis Sourd et Olivier Spanjaard. C'est leur minutieux travail qui a rendu possible la réalisation de cet ouvrage. Nous avons pu constater que leur contribution a été très constructive et ce, même pour les auteurs dont les articles ont dû être rejetés.

Les organisateurs remercient enfin les partenaires institutionnels et privés dont le soutien a permis l'édition de cet ouvrage et l'organisation du congrès.

Pierre Lopez et Marcel Mongeau

Co-présidents du comité d'organisation ROADEF 2010

Comité d'organisation

- Pierre Lopez (Chargé de Recherche, LAAS-CNRS), co-président
- Marcel Mongeau (Maître de Conférences, UPS, IMT), co-président
- Christian Artigues (Chargé de Recherche, LAAS-CNRS)
- Alain Berro (Maître de Conférences, UT1, IRIT)
- Cyril Briand (Maître de Conférences, UPS, LAAS-CNRS)
- Alain Haït (Professeur associé, ISAE)
- Laurent Houssin (Maître de Conférences, UPS, LAAS-CNRS)
- Sophie Jan (Maître de Conférences, UPS, IMT)
- Bertrand Jouve (Maître de Conférences, UTM, IMT)
- Nicolas Jozefowiez (Maître de Conférences, INSA, LAAS-CNRS)
- Catherine Mancel (Enseignant-chercheur, ENAC)
- Frédéric Messine (Maître de Conférences, INPT, IRIT)
- Aude Rondepierre (Maître de Conférences, INSA, IMT)
- Caroline Thierry (Maître de Conférences, UTM, IRIT)
- Christèle Soler-Dominguez (Secrétaire, LAAS-CNRS)
- Jean-Michel Pons (Ingénieur de Recherche, LAAS-CNRS)
- Isabelle Nolhier (Technicienne, LAAS-CNRS)

Comité scientifique

- Michel Bierlaire (Ecole Polytechnique Fédérale de Lausanne, Suisse)
- Jean-Charles Billaut (Ecole Polytechnique de l'Université de Tours, LI, Tours)
- Alain Billionnet (CNAM, CEDRIC, Paris)
- Frédéric Bonnans (Ecole Polytechnique, CMAP, Paris)
- Denis Bouyssou (Université Paris-Dauphine, LAMSADE, Paris)
- Nadia Brauner (Université Joseph Fourier, G-SCOP, Grenoble)
- Jacques Carlier (Université de Technologie de Compiègne, HEUDIASYC, Compiègne)
- Yves Caseau (Bouygues Telecom)
- Irène Charon (Ecole Nationale Supérieure des Télécommunications, Paris)
- Philippe Chrétienne (Université Pierre et Marie Curie, LIP6, Paris)
- Yves Crama (Université de Liège, HEC, Liège, Belgique)
- Stéphane Dauzère-Pérès (Ecole des Mines de Saint-Etienne, CMP G. Charpak, Gardanne)
- Alexandre Dolgui (Ecole des Mines de Saint-Etienne, Saint-Etienne)
- Xavier Gandibleux (Université de Nantes, LINA, Nantes)
- Michel Gendreau (Ecole Polytechnique de Montréal, CIRRELT, Montréal, Canada)
- Michel Gourgand (ISIMA, LIMOS, Clermont-Ferrand)
- Alain Guénoche (Institut de Mathématiques de Luminy, Marseille)
- Jin-Kao Hao (Université d'Angers, LERIA, Angers)
- Mohamed Haouari (Ecole Polytechnique de Tunisie, Unité ROI, Tunisie)
- Jean-Baptiste Hiriart-Urruty (Université Paul Sabatier, IMT, Toulouse)
- Eric Jacquet-Lagrèze (EURODECISION)
- Safia Kedad-Sidhoum (Université Pierre et Marie Curie, LIP6, Paris)
- Martine Labbé (Université Libre de Bruxelles, Belgique)
- Jean-Bernard Lasserre (LAAS-CNRS, Toulouse)
- Claude Lemaréchal (INRIA Grenoble Rhône-Alpes, Grenoble)
- Philippe Mahey (ISIMA, LIMOS, Clermont-Ferrand)
- Ridha Mahjoub (Université Paris-Dauphine, LAMSADE, Paris)
- Patrice Marcotte (Université de Montréal, DIRO, Montréal, Canada)
- Michel Minoux (Université Pierre et Marie Curie, LIP6, Paris)
- Aziz Moukrim (Université de Technologie de Compiègne, HEUDIASYC, Compiègne)
- Denis Naddef (ENSIMAG, G-SCOP, Grenoble)
- Vangelis Paschos (Université Paris-Dauphine, LAMSADE, Paris)
- Patrice Perny (Université Pierre et Marie-Curie, LIP6, Paris)
- Christophe Picouleau (CNAM, CEDRIC, Paris)
- Gérard Plateau (Institut Galilée, LIPN, Paris)
- Marie-Claude Portmann (Ecole des Mines de Nancy, LORIA, Nancy)
- Christian Prins (Université de Technologie de Troyes, LOSI, Troyes)
- Alain Quilliot (ISIMA, LIMOS, Clermont-Ferrand)
- Catherine Roucairol (Université de Versailles Saint-Quentin-en-Yvelines, PRISM, Versailles)
- Bernard Roy (Université Paris Dauphine, LAMSADE, Paris)
- Thomas Schiex (Institut National de la Recherche Agronomique, Toulouse)
- Frédéric Semet (Ecole Centrale de Lille, LAGIS, Lille)
- Patrick Siarry (Université Paris XII, LiSSi, Créteil)
- Jacques Teghem (Faculté Polytechnique de Mons, Mons, Belgique)
- Alexis Tsoukias (Université Paris Dauphine, LAMSADE, Paris)
- Pascal Van Hentenryck (Brown University, Providence, USA)
- Pascale Zaraté (Institut National Polytechnique de Toulouse, IRIT, Toulouse)

Sommaire

Programmation non-linéaire mixte sous contraintes "On/Off" : analyse de convexité et applications

Hassan Hijazi[1,2] *, Pierre Bonami[2*], Gérard Cornuejols[2] †, Adam Ouorou[1]

[1] Orange Labs R&D ; CORE-MCN ; 38-40 rue du Général Leclerc, 92794 Issy-Les-Moulineaux cedex 9, France
hassan.hijazi@orange-ftgroup.com

[2] Laboratoire d'Informatique Fondamentale de Marseille ; Universté de la Méditerranée ; Parc Scientifique et Technologique de Luminy, 163 avenue de Luminy - Case 901, F-13288 Marseille Cedex 9, France

Résumé : *Dans le cadre de la Programmation Non-Linéaire Mixte, nous rencontrons fréquemment des contraintes de type "on/off". Une contrainte est dite "on/off" si une variable booléenne lui est associée et cette contrainte doit être satisfaite si et seulement si cette variable vaut 1. Nous nous intéressons à l'ensemble simple défini par une seule contrainte "on/off" avec variables bornées. Ce dernier étant représenté par l'union de deux ensembles disjoints Γ_0 et Γ_1, des relaxations convexes telles que les formulations big-M sont courantes. Néanmoins, il est toujours possible d'écrire l'enveloppe convexe de cette union dans des espaces de dimensions supérieures, mettant en jeu un nombre important de variables. Ces formulations étant désavantageuses, on introduit dans ce travail des modèles définis dans des espaces de dimensions réduites. Sous des hypothèses spécifiques sur les fonctions non-linéaires définissant nos ensembles, on montre que l'enveloppe convexe $conv(\Gamma_0 \cup \Gamma_1)$ peut être caractérisée dans l'espace des variables d'origine. Une première application, puisée du monde des télécommunications, consiste à étudier des problèmes de routage sous contraintes de délais multiples. Tandis que les problèmes classiques de routage imposent une contrainte de délai moyen sur l'ensemble du réseau, notre modèle permet de prendre en compte un délai de bout en bout propre à chaque type de demande. Grâce à cette propriété, un opérateur bénéficiant de ces outils est en mesure de garantir une "qualité de service différenciée" à ses clients.*

Mots-Clés : *Programmation Non-Linéaire Mixte, contraintes on/off, contraintes disjonctives, programmation convexe, problème de routage, contrainte de délai, qualité de service différenciée.*

1 Introduction

De nos jours, plusieurs algorithmes de résolution ont été développés pour les Programmes Non-Linéaires Mixtes (PNLM) convexes, voir [11, 17, 20, 7]. Des solveurs tels que FilMINT [1] et Bonmin [6] ont permis de résoudre des problèmes de tailles raisonnables.
Néanmoins, il est toujours nécessaire de fournir à ces solveurs des modèles mathématiques offrant des relaxations continues de bonne qualité et faciles à résoudre.
Dans ce travail, nous étudions et comparons différentes relaxations continues convexes de PNLM sous contraintes "on/off". Une contrainte "on/off" est une contrainte algébrique qui doit être satisfaite si et seulement si la variable booléenne correspondante vaut 1. Étant données des fonctions convexes fermées $g : \mathbb{R}^{n+K} \to \mathbb{R}^m$, $h : \mathbb{R}^{n+K} \to \mathbb{R}$ et $f^k : \mathbb{R}^n \to \mathbb{R}$, $\forall k \in \{1,\ 2, ..., K\}$, on s'intéresse aux

†Subvention NSF CMMI0653419, Subvention ONR N00014-03-1-0133 et Subvention ANR ANR06-BLAN-0375.

problèmes d'optimisation de la forme :

$$\begin{aligned} & \min\ h(x,z) \\ t.q.\quad & g(x,z) \leq 0 \\ & f^k(x) \leq 0 \text{ si } z_k\ =\ 1,\ \forall k \in \{1,\ 2, ..., K\} \\ & l_i \leq x_i \leq u_i,\ \forall i \in \{1,\ 2,\ ...,\ n\} \\ & x \in \mathbb{R}^n, z_k \in \{0,1\},\ \forall k \in \{1,\ 2, ..., K\}. \end{aligned} \tag{1}$$

Chaque $f^k(x) \leq 0$ représente une contrainte "on/off", avec z_k la variable booléenne correspondante. les inégalités $g(x,z) \leq 0$, quant à elles, regroupent les contraintes convexes restantes. Les bornes sur les variables sont supposées finies.
La difficulté du problème (1) est due à la présence des contraintes de types "on/off". De telles contraintes définissent un ensemble réalisable non convexe, plus précisément une union de deux ensembles disjoints. Il est nécessaire de reformuler ces contraintes en vue de résoudre le modèle mathématique sous-jacent par un solveur quelconque.
Une façon de modéliser (1) en tant que PNLM convexe serait d'introduire des contraintes classiques de type big-M conduisant à une formulation compacte et facile à résoudre, offrant néanmoins une relaxation continue de qualité relativement mauvaise. Une méthode plus efficace inspirée de la décomposition de Benders applicable dans le contexte des Programmes Linéaires en Nombres Entiers a été proposée par Codato et Fischetti [9].
Nous étudions ici une autre alternative basée sur la Programmation Disjonctive (voir [5, 8, 18, 14, 13]). En effet, (1) peut être reformulé en tant que programme disjonctif :

$$\begin{aligned} & \min\ h(x,z) \\ t.q.\quad & g(x,z) \leq 0 \\ & (x, z_k) \in \Gamma_0^k \cup \Gamma_1^k,\ \forall k \in \{1,\ 2, ..., K\} \\ & \Gamma_0^k = \{\ (x,\ z_k) \in \mathbb{R}^n \times \mathbb{B} : z_k = 0,\ l_i \leq x_i \leq u_i,\ \forall i \in \{1,\ 2,\ ...,\ n\}\ \} \\ & \Gamma_1^k = \{\ (x,\ z_k) \in \mathbb{R}^n \times \mathbb{B} : z_k = 1,\ f^k(x) \leq 0,\ l_i \leq x_i \leq u_i,\ \forall i \in \{1,\ 2,\ ...,\ n\}\ \}. \end{aligned} \tag{2}$$

Balas ([4, 5]) a été le premier à introduire les formulations algébriques explicites des enveloppes convexes pour des unions de polyèdres. Plus récemment, plusieurs généralisations et extensions ont été établies pour des unions d'ensembles convexes quelconques, dans [8], [14] et [18]. Dans Stubbs et Mehrotra [18], Ceria et Soares [8], aussi bien que dans Grossman et Lee [14], on trouve les outils mathématiques d'analyse convexe nécessaires à la description de $conv\ (\Gamma_0^k \cup \Gamma_1^k)$, l'enveloppe convexe de $\Gamma_0^k \cup \Gamma_1^k$, dans un espace de dimension supérieure. On peut donc reformuler (2) sous la forme :

$$\begin{aligned} & \min\ h(x,z) \\ t.q.\quad & g(x,z) \leq 0 \\ & (x, z_k) \in conv(\Gamma_0^k \cup \Gamma_1^k),\ z_k \in [0,1],\ \forall k \in \{1,\ 2, ..., K\} \\ & \Gamma_0^k = \{\ (x,\ z_k) \in \mathbb{R}^n \times \mathbb{B} : z_k = 0,\ l_i \leq x_i \leq u_i,\ \forall i \in \{1,\ 2,\ ...,\ n\}\ \} \\ & \Gamma_1^k = \{\ (x,\ z_k) \in \mathbb{R}^n \times \mathbb{B} : z_k = 1,\ f^k(x) \leq 0,\ l_i \leq x_i \leq u_i,\ \forall i \in \{1,\ 2,\ ...,\ n\}\ \}. \end{aligned} \tag{3}$$

Il est clair que (3) est une formulation valide de (1), sa relaxation continue est un programme nonlinéaire convexe apportant des bornes de bonne qualité. Cependant, écrire $conv(\Gamma_0^k \cup \Gamma_1^k)$ dans un

espace de dimension supérieur conduit à un programme non-linéaire de grande taille.
Naturellement, on serait intéressé par des formulations de types (3) qui ne nécessitent pas l'ajout de variables supplémentaires. Günlük et Linderoth [15] ont montré que pour le cas particulier où Γ_0^k est réduit à un point unique, l'enveloppe convexe $conv\,(\Gamma_0^k \cup \Gamma_1^k)$ peut être formulée dans l'espace des variables d'origine en ayant recours aux fonctions perspectives. Sous ces mêmes hypothèses, Aktürk, Atamtürk et Gürel [2], avaient auparavant réussi à apporter une caractérisation complète de ces enveloppes convexes pour des fonctions particulières utilisées dans des problèmes d'ordonnancement.
Dans ce travail, nous étudions le cas général dans lequel Γ_0^k représente un hyper-rectangle défini par des bornes finies sur les variables x. Dans la section 2, en se basant sur les résultats de [8], $conv\,(\Gamma_0^k \cup \Gamma_1^k)$ est défini dans des espaces de dimensions supérieurs. Des formulations projetées de l'enveloppe convexe ainsi que des relaxations dans l'espace des variables d'origine sont établies.
Dans la section 3, nous nous intéresserons à une application dans les Télécommunications introduite en premier lieu par Ben Ameur et Ouorou [3] traitant du routage sous contrainte de délai. Ce problème associe une contrainte de délai propre à chaque type de demande, permettant aux opérateurs d'offrir une garantie de "qualité de service différenciée". Nous proposons des nouveaux modèles mathématiques basés sur les formulations introduites dans la section 2 et adaptés à ce problème. Dans la section 4, nous présentons des tests numériques sur des instances réelles de réseaux de Télécommunications ainsi que sur des instances générées aléatoirement, permettant de comparer l'ensemble des modèles.
Étant donné un ensemble $\Gamma \in \mathbb{R}^n$, on notera $cl(\Gamma)$ sa fermeture topologique et $proj_{(x^1,...,x^j)}(\Gamma)$ sa projection sur l'espace $(x^1, ..., x^j)$. Le domaine effectif d'une fonction f, noté $dom(f)$, représente l'ensemble des points $x \in \mathbb{R}^n$ pour lesquels $f(x) < +\infty$. Soit f une fonction convexe fermée, la contrainte $f(x) \leq 0$ admet une extension fermée unique $(cl\ f) \leq 0$, qui est obtenu en redéfinissant f aux points $x \notin dom(f)$ tels que : $\{x \in \mathbb{R}^n : (cl\ f)(x) \leq 0\} = cl(\{x \in \mathbb{R}^n : f(x) \leq 0\})$.
Veuillez noter que l'ensemble des preuves associées aux résultats avancés dans cet article sont présentés dans la section annexe.

2 Enveloppe convexe de $\Gamma_0 \cup \Gamma_1$

Considérons un premier exemple de PNLM sous contraintes "on/off" dans $\mathbb{R}^3$, avec K=1, i.e. faisant intervenir une seule contrainte "on/off" :

$$\begin{aligned} & \min_{(x,z)\in\mathbb{R}^2,\ z\in\mathbb{B}} h(x,z) \\ t.q.\quad & (x,z) \in conv\,(\Gamma_0 \cup \Gamma_1) \\ & \Gamma_0 = \{\ (x,\ z) \in \mathbb{R}^2 \times \mathbb{B} : z = 0,\ l_1 \leq x_1 \leq u_1,\ l_2 \leq x_2 \leq u_2\} \\ & \Gamma_1 = \{\ (x,\ z) \in \mathbb{R}^2 \times \mathbb{B} : z = 1,\ \tfrac{1}{c_1 - x_1} + \tfrac{1}{c_2 - x_2} \leq d,\ l_1 \leq x_1 \leq u_1 < c_1,\ l_2 \leq x_2 \leq u_2 < c_2\} \end{aligned} \tag{4}$$

La Figure 1 illustre les ensembles Γ_0 et Γ_1 dans $\mathbb{R}^3$. Tandis que Γ_0 est représenté par le rectangle apparaissant sur le côté droit, la figure convexe gauche désigne l'ensemble Γ_1. La Figure 2 offre une représentation de $conv(\Gamma_0 \cup \Gamma_1)$ dans $\mathbb{R}^3$ avec les courbes de niveau de la fonction perspective $\tilde{f}(x - (1-z)u, z)$ définie ci-dessous. Notons que la contrainte non-linéaire $conv\,(\Gamma_0 \cup \Gamma_1)$ peut s'inspirer de la fonction perspective de f partant du point x^* apparaissant dans le coin supérieur de Γ_0.
La fonction perspective $\tilde{f} : \mathbb{R}^{n+1} \to \mathbb{R} \cup \{+\infty\}$ de f est définie par : $\tilde{f}(x,z) \equiv \begin{cases} zf(x/z) & \text{si } z > 0, \\ +\infty & \text{si } z \leq 0. \end{cases}$
Dans [16] (Proposition VI.2.2.2), on démontre que la fermeture de $\tilde{f}$ s'écrit sous la forme suivante :

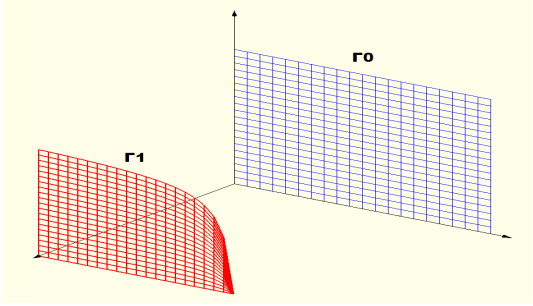
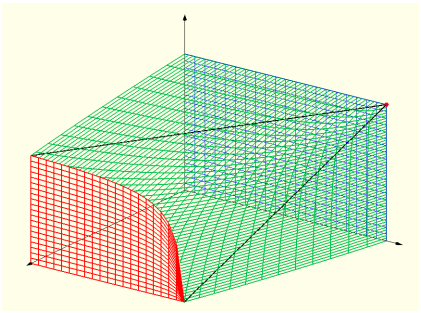


FIGURE 1 – Les ensembles Γ_0 et Γ_1.



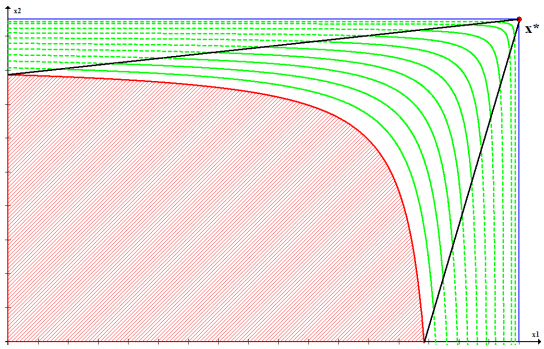


FIGURE 2 – conv($\Gamma_1 \cup \Gamma_0$).

$$(cl\ \tilde{f})(x,z) \equiv \begin{cases} zf(x/z) & \text{si } z>0, \\ \lim_{z\to 0^+} zf(\tilde{x}-x+x/z) & \text{si } z=0, \\ +\infty & \text{si } z<0, \end{cases}$$

où $\tilde{x}$ est un point quelconque de l'intérieur relatif du domaine effectif de f. Dans notre cas, la contrainte non-linéaire dans $conv\,(\Gamma_0 \cup \Gamma_1)$ est définie par $\tilde{f}(x-(1-z)u, z) \leq 0$.
Pour cet exemple spécifique, $conv\,(\Gamma_0 \cup \Gamma_1)$ s'écrit :

$$\begin{aligned} conv\,(\Gamma_0 \cup \Gamma_1) &= \left\{ \begin{array}{l} (x,\ z) \in \mathbb{R}^3 : \\ (cl\tilde{f})(x-(1-z)u,\ z) \leq 0 \\ l_1 \leq x_1 \leq u_1,\ l_2 \leq x_2 \leq u_2. \end{array} \right\} \\ &= \left\{ \begin{array}{l} (x,\ z) \in \mathbb{R}^3 : \\ cl\left(\frac{z}{zc_1-x_1+(1-z)u_1} + \frac{z}{zc_2-x_2+(1-z)u_2} - d\right) \leq 0 \\ l_1 \leq x_1 \leq u_1,\ l_2 \leq x_2 \leq u_2. \end{array} \right\}. \end{aligned}$$

Retournons à présent au cas général traitant une contrainte "on/off" dans $\mathbb{R}^n$. Se basant sur le Théorème 1 dans Ceria et Soares [8] caractérisant l'enveloppe convexe de l'union d'ensembles convexes fermés et l'appliquant au programme (3), on doit ajouter $|K|.(2n+4)$ variables pour définir l'ensemble des contraintes "on/off". De ce fait, réduire la dimension de l'espace en question a un impact important sur les problèmes d'optimisation ayant un nombre considérable de contraintes "on/off". Dans le lemme suivant, nous montrons que dans le cadre défini par nos hypothèses, la formule explicite de l'enveloppe convexe correspondant à une contrainte "on/off" peut être obtenue en introduisant seulement n variables nouvelles.

Lemme 1 *Soit :*

$f : E \to \mathbb{R},\ E \subseteq \mathbb{R}^n$, *une fonction convexe fermée,*

$\Gamma_0 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 0,\ l_i^0 \leq x_i \leq u_i^0,\ \forall i \in \{1, 2, ..., n\}\ \},$

$\Gamma_1 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 1,\ f(x) \leq 0,\ l_i^1 \leq x_i \leq u_i^1,\ \forall i \in \{1, 2, ..., n\}\ \}$, *non vide. Alors :*

$$\boldsymbol{conv\,(\Gamma_0 \cup \Gamma_1) = \{(x, z)\ |\ \exists y \in \mathbb{R}^n,}\ \textit{avec}\ \boldsymbol{(x, y, z) \in cl(\Gamma)}\},$$

$$\text{où } \Gamma = \left\{ \begin{array}{l} (x, y, z) \in \mathbb{R}^{2n+1} : \\ zf\,(y/z) \leq 0, \\ x_i - (1-z)u_i^0 \leq y_i \leq x_i - (1-z)l_i^0,\ \forall i \in \{1, 2, ..., n\}, \\ zl_i^1 \leq y_i \leq zu_i^1,\ \forall i \in \{1, 2, ..., n\}, \\ 0 < z \leq 1. \end{array} \right\}$$

Ci-dessous, on montre que si f est une fonction isotone (voir définition), les variables y peuvent entièrement être projetées.

Définition 1 *Soit $f : E \to \mathbb{R},\ E \subseteq \mathbb{R}^n$, f est indépendamment croissante (resp. décroissante) sur la ième coordonnée si :*
$\forall x = (x_1, x_2, ..., x_i, ..., x_n) \in dom(f),\ x' = (x_1, x_2, ..., x_i', ..., x_n) \in dom(f)$ *t.q.* $x_i' \geq x_i \Rightarrow f(x') \geq (\ resp.\ \leq)\ f(x)$.
On dit que f est indépendamment monotone sur la ième coordonnée si elle est indépendamment croissante ou décroissante sur cette coordonnée.
f est isotone si elle est indépendamment monotone sur toutes les coordonnées de x.
Les fonctions isotones les plus courantes sont les fonctions additivement séparables, sommes de fonctions réelles monotones.

Théorème 1 *Soient :*

$f : E \to \mathbb{R},\ E \subseteq \mathbb{R}^n$, *une fonction convexe fermée et isotone,*

$J^1(resp.\ J^2)$ *l'ensemble des indices où f est indépendamment croissante (resp. décroissante),*

$\Gamma_0 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 0,\ l_i^0 \leq x_i \leq u_i^0,\ \forall i \in \{1, 2, ..., n\}\ \},$

$\Gamma_1 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 1,\ f(x) \leq 0,\ l_i^1 \leq x_i \leq u_i^1,\ \forall i \in \{1, 2, ..., n\}\ \}$, *non vide. Alors :*

$$\boldsymbol{conv\,(\Gamma_0 \cup \Gamma_1) = cl(\Gamma')},$$

$$\text{où } \Gamma' = \left\{ \begin{array}{l} (x, z) \in \mathbb{R}^{n+1} : \\ zq_S\,(x/z) \leq 0,\ \forall S \subset \{1, 2, ..., n\}, \\ zl_i^1 + (1-z)l_i^0 \leq x_i \leq zu_i^1 + (1-z)u_i^0,\ \forall i \in \{1, 2, ..., n\}, \\ 0 < z \leq 1. \end{array} \right\},$$

avec $q_S = (f \circ h_S)$, $h_S(\mathbb{R}^n \to \mathbb{R}^n)$ *définie par,* $\forall t \in \mathbb{R}^n$, $(h_S(t))_i = \begin{cases} l_i^1 \ \forall i \in S \cap J_1, \\ u_i^1 \ \forall i \in S \cap J_2, \\ t_i - \frac{(1-z)u_i^0}{z} \ \forall i \in \bar{S} \cap J_1, \\ t_i - \frac{(1-z)l_i^0}{z} \ \forall i \in \bar{S} \cap J_2. \end{cases}$

Ce théorème apporte un nouveau moyen d'exprimer $conv\,(\Gamma_0 \cup \Gamma_1)$ dans (3). Cette formulation ne nécessite l'ajout d'aucune variable nouvelle, cependant, elle introduit un nombre exponentiel de contraintes non-linéaires, jusqu'à $|K|.(2^n - 1)$. On montre ci-dessous, qu'une seule contrainte non-linéaire est suffisante pour obtenir une formulation valide de notre problème (2) :

Corollaire 1 *Soient :*

$f : E \to \mathbb{R},\ E \subseteq \mathbb{R}^n$, *une fonction convexe fermée et isotone,*

J^1*(resp.* J^2*) l'ensemble des indices où* f *est indépendamment croissante (resp. décroissante),*

$\Gamma_0 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 0,\ l_i^0 \leq x_i \leq u_i^0,\ \forall i \in \{1, 2, ..., n\}\ \},$

$\Gamma_1 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 1,\ f(x) \leq 0,\ l_i^1 \leq x_i \leq u_i^1,\ \forall i \in \{1, 2, ..., n\}\ \}$, *non vide. Alors :*

1. $\boldsymbol{\Gamma''}$ *est une relaxation convexe valide de* $\boldsymbol{conv(\Gamma_0 \cup \Gamma_1)}$
2. $\boldsymbol{\{(x, z) \in \mathbb{R}^n \times \mathbb{B} : (x, z) \in \Gamma''\} \equiv \Gamma_0 \cup \Gamma_1}$

$$\text{où } \Gamma'' = \left\{ \begin{array}{l} (x, z) \in \mathbb{R}^{n+1} : \\ z(cl\ q')\,(x/z) \leq 0,\ \textit{t.q. :} \\ zl_i^1 + (1-z)l_i^0 \leq x_i \leq zu_i^1 + (1-z)u_i^0,\ \forall i \in \{1, 2, ..., n\}, \\ 0 \leq z \leq 1. \end{array} \right\},$$

avec $q' = (f \circ h_\emptyset)$, $h_\emptyset(\mathbb{R}^n \to \mathbb{R}^n)$ *définie par* $(h_\emptyset(y))_i = \begin{cases} y_i - \frac{(1-z)u_i^0}{z} \ \forall i \in J_1, \\ y_i - \frac{(1-z)l_i^0}{z} \ \forall i \in J_2. \end{cases}$

Remarque Günlük et Linderoth [15] ont caractérisé la formule explicite de l'enveloppe convexe de l'union d'un point et d'un ensemble convexe, le tout dans l'espace des variables d'origine.
On peut voir que cette caractérisation comme un cas particulier du Lemme 1. En effet, en fixant $l_0 = u_0 = 0$ dans l'ensemble définissant l'enveloppe convexe, on obtient :

$$conv(\Gamma_0 \cup \Gamma_1) = \left\{ \begin{array}{l} (x, y, z) \in \mathbb{R}^{2n+1} : \\ z(cl\ f)\,(y/z) \leq 0, \\ x \leq y \leq x, \\ zl^1 \leq y \leq zu^1, \\ 0 < z \leq 1. \end{array} \right\} \equiv \left\{ \begin{array}{l} (x, z) \in \mathbb{R}^{n+1} : \\ z(cl\ f)\,(x/z) \leq 0, \\ zl^1 \leq x \leq zu^1, \\ 0 \leq z \leq 1. \end{array} \right\},$$

ce qui correspond exactement au résultat établit dans [15].

3 Application : Problème de Routage sous Contraintes de Délais

Dans cette section, on étudie un problème de télécommunications intitulé “Problème de Routage sous Contraintes de Délais” introduit par Ben Ameur et Ouorou (2006) [3]. Parmi les services modernes offerts par les opérateurs, les applications temps réelles, telles que les jeux en ligne ou encore la vidéo conférence sont relativement sensibles aux délais de transmission. Ce type d'applications requiert une garantie sur le délai de bout en bout (i.e., délai de transmission du nœud source à toutes les destinations) aussi bien que sur la variation du délai inter-destinations (i.e., la différence en délai de transmission du noeud source à différentes destinations). En général, les problèmes de routages sont étudiés sous des contraintes de délai moyen de bout en bout, ignorant ainsi la nature hétérogène des réseaux actuels. Le problème considéré ici attribue un délai propre à chaque type de demande. On définit le délai de transmission sur un lien en fonction de la capacité installée sur ce dernier ainsi que du débit de trafic le traversant. Étant donné une demande d_k, un sous ensemble de chemins candidats est fourni, un nombre maximum de chemins actifs par demande est également désigné. Cette application, formulée en tant que programme non-linéaire mixte sous contraintes "on/off", correspond parfaitement à nos hypothèses. Notons que le modèle introduit dans [3] est un cas particulier du modèle ci-dessous, dans lequel on autorise un seul chemin actif par demande (mono-routage, N=1).

3.1 Modèles mathématiques

Soit $G = (V, E)$ un réseau orienté fini où V représente l'ensemble des sommets, E l'ensemble des arcs. On note K l'ensemble des demandes, d_k définissant la k_{ieme} demande, $v_k \in \mathbb{R}$ la quantité ou le débit à router et α_k la garantie de délai correspondant à cette demande, $\alpha_k \in \mathbb{R}$. On associe à toute demande un sous ensemble de chemins candidats $P(k) = \{P_k^1, P_k^2, ..., P_k^{n_k}\}$, chacun correspondant à un routage différent de d_k. On notera N le nombre maximum de chemins actifs autorisés par demande. Pour tout arc e, $c_e \in \mathbb{R}$ désigne la capacité installée et $w_e \in \mathbb{R}$ le coût de routage sur cet arc. On note ϕ_k^i la fraction de la k_{ieme} demande routée par son i_{eme} chemin, $\phi_k^i \in [0, 1]$.
z_k^i constituent les variables binaires indiquant si le chemin P_k^i est activé. Enfin, x_e représente la quantité totale de flot traversant l'arc e, $x_e \in \mathbb{R}$.

Programme initial (Pr) :

$$Min \sum_{e \in E} w_e x_e \tag{5}$$

$$\sum_{i=1}^{n_k} \phi_k^i \geq 1, \ \forall k \in K \tag{6}$$

$$\sum_{k \in K} \sum_{P_k^i \ni l} \phi_k^i v_k \leq x_e, \ \forall e \in E \tag{7}$$

$$x_e \leq c_e, \ \forall e \in E \tag{8}$$

$$\sum_{e \in P_k^i} \frac{1}{c_e - x_e} \leq \alpha_k, \ \forall k \in K, \ \forall P_k^i \in P(k) \text{ si } z_k^i = 1 \tag{9}$$

$$\sum_{P_k^i \in P(k)} z_k^i \leq N, \ \forall k \in K \tag{10}$$

$$\phi_k^i \leq z_k^i, \ \forall k \in K, \ \forall P_k^i \in P(k) \tag{11}$$

$$z_k^i \in \{0,1\},\ \phi_k^i \in [0,1],\ \forall k \in K,\ \forall P_k^i \in P(k) \tag{12}$$

$$\phi_k^i \in [0,1],\ \forall k \in K,\ \forall P_k^i \in P(k) \tag{13}$$

$$x_e \in \mathbb{R},\ \forall e \in E. \tag{14}$$

Étant donné un coût w_e correspondant à l'utilisation de l'arc e, la fonction objectif (5) consiste à minimiser le coût total de routage sur l'ensemble des liens utilisés. Pour toute demande d_k, dans (6) la fraction de la demande routée doit être supérieure à 1, garantissant, à l'optimalité, la satisfaction de toutes les demandes. Les contraintes (7) définissent les variables de flots x_e sur e comme étant la somme de toutes les quantités passant par cet arc. Dans (8), on impose les contraintes de capacités sur les variables de flot. (9) représente la contrainte "on/off" principale : la garantie de délai associée à une commodité doit être satisfaite sur un chemin candidat si et seulement si ce dernier est activé. Comme mentionné précédemment, ce modèle permet de fixer un nombre maximum de chemins candidats actifs par demande, ceci apparaît dans (10). Dans (11) le lien entre les variables binaires z_k^i et les variables ϕ_k^i est établi. Enfin, les bornes sur l'ensemble des variables sont introduites dans (12-14). Soulignons le fait que pour $N = 1$, on autorise l'activation d'un chemin unique par demande (mono-routage). Dans ce cas, la contrainte (10) s'écrit : $\sum_{P_k^i \in P(k)} z_k^i \leq 1,\ \forall k \in K$. En associant ces contraintes à (6) et (11), on peut déduire que, danse ce cas, les variables ϕ_k^i sont redondantes. Ben Ameur et Ouorou ont montré dans [3] que le problème de décision associé à (3.1) (ignorant la fonction objectif) était NP-complet avec seulement deux chemins candidats par demande.
La non convexité de ce modèle est due à la présence des contraintes "on/off" (9) ainsi qu'aux contraintes (10). On introduit, dans la suite, quatre modèles convexes équivalents à (3.1) donnant chacun une relaxation continue différente.

Relaxation big-M : ($\mathbf{P}_{bigM}$) Une relaxation convexe classique de (9) est la formulation big-M :

$$\begin{aligned} & \min\ \textstyle\sum_{e \in E} w_e x_e \\ t.q.\ & (6),\ (7),\ (8),\ (10),\ (11) \text{ et } (12-14) \\ & \textstyle\sum_{e \in P_k^i} \frac{1}{c_e - x_e} \leq M - z_k^i (M - \alpha_k),\ \forall k \in K,\ \forall P_k^i \in P(k). \end{aligned} \tag{9-a}$$

Cette formulation est exacte pour z_k^i binaire et M suffisamment grand. Notons que la qualité de la borne apportée par cette formulation dépend en premier lieu de la constante M. Il est donc important de calculer M de façon efficace. Ben Ameur et Ouorou [3] ont remarqué que le flot sur un arc donné e admet toujours une borne supérieure u_e vérifiant $u_e < c_e$. En effet, si un lien e est utilisé dans un chemin actif P_k^i, on peut écrire $\frac{1}{c_e - x_e} \leq \alpha_k - \sum_{e' \neq e} \frac{1}{c_{e'}}$. En se basant sur ces observations, on peut déduire une borne supérieure α_e sur le délai correspondant à chaque arc puis une borne sur le délai généré pour un chemin quelconque. En d'autres termes, la constante M peut être remplacée par l'expression suivante : $\alpha_k^i = \sum_{e \in P_k^i} \alpha_e$.
À présent, en se basant sur les résultats de la section 2, nous introduisons des nouvelles relaxations des contraintes "on/off" (9).
À tout chemin P_k^i est associé une contrainte de délai (9) dans (3.1) qu'on écrit sous la forme : $f^{(i,k)}(x) \leq 0$ si $z_k^i = 1$, avec $f^{(i,k)} : \mathbb{R}^n \to \mathbb{R}$, $f^{(i,k)}(x) = \sum_{e \in P_k^i} \frac{1}{c_e - x_e} - \alpha_k$. Les f_k^i étant toutes des fonctions convexes fermées, en se basant sur l'énoncé du Lemme 1, on introduit la proposition suivante :

Proposition 1 *Soit :*

$$f : \mathbb{R}^n_+ \to \mathbb{R},\ f(x) = \sum_{i=1}^n \left(\frac{1}{c_i - x_i}\right) - b,\ b \geq 0.$$

$$\Gamma_0 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 0,\ 0 \leq x_i \leq u_i < c_i,\ \forall i \in \{1, 2, ..., n\}\ \},$$

$$\Gamma_1 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 1,\ f(x) \leq 0,\ l_i \leq x_i \leq u_i,\ \forall i \in \{1, 2, ..., n\}\ \},\ \textit{non vide, et}$$

$$\Gamma^\epsilon = \left\{ \begin{array}{l} (x, y, z) \in \mathbb{R}^{2n+1} : \\ \sum_{i=1}^n \left(\frac{z^2}{zc_i - y_i + (1-z)\epsilon}\right) - zb \leq 0, \\ x_i - (1-z)u_i \leq y_i \leq x_i,\ \forall i \in \{1,\ 2,\ ...,\ n\}, \\ zl_i \leq y_i \leq zu_i,\ \forall i \in \{1,\ 2,\ ...,\ n\}, \\ 0 \leq z \leq 1. \end{array} \right\}$$

alors :

1. $\boldsymbol{proj_{(x,z)}(\Gamma^\epsilon)}$ *est une relaxation valide de* $\boldsymbol{conv(\Gamma_0 \cup \Gamma_1)}$.
2. $\boldsymbol{\{(x,z) \in \mathbb{R}^n \times \mathbb{B} : (x,z) \in proj_{(x,z)}(\Gamma^\epsilon)\} \equiv \Gamma_0 \cup \Gamma_1}$.

En se basant sur cette proposition, on introduit un nouveau PNLM équivalent à (3.1) et offrant une meilleure relaxation continue que celle du modèle big-M.

Modèle réduit : ($\mathbf{P}_{red}$) On remplace les contraintes (9) par les relaxations des enveloppes convexes définies dans la proposition précédente :

$$\begin{array}{ll} & \min\ \sum_{e \in E} w_e x_e \\ t.q. & (6),\ (7),\ (8),\ (10),\ (11) \text{ et } (12-14) \\ & \sum_{e \in P_k^i} \left(\frac{{z_k^i}^2}{z_k^i c_e - y_e^{(i,k)} + (1 - z_k^i)\epsilon}\right) - z_k^i \alpha_k \leq 0,\ \forall k \in K,\ \forall P_k^i \in P(k). \qquad \text{(9-b)} \\ & x_e - (1 - z_k^i)u_e \leq y_e^{(i,k)} \leq x_e,\ \forall k \in K,\ \forall P_k^i \in P(k),\ \forall e \in P_k^i. \\ & z_k^i l_e \leq y_e^{(i,k)} \leq z_k^i u_e,\ \forall k \in K,\ \forall P_k^i \in P(k),\ \forall e \in P_k^i. \end{array} \qquad (\mathrm{P}_{red})$$

Soit Nb_{max} le nombre maximum de chemins candidats par demande, jusqu'à $|E| \times |K| \times Nb_{max}$ variables peuvent être ajoutées dans ce nouveau modèle. Afin qu'un modèle puisse être compétitif avec la formulation big-M, il doit bénéficier de sa dimension réduite, en d'autre termes, aucune variable supplémentaire ne doit y être ajoutée. Se basant sur le corollaire 1, on introduit le résultat suivant, un premier pas vers un modèle défini dans l'espace des variables d'origine.

Corollaire 2 *Soit :*

$f : \mathbb{R}^n_+ \to \mathbb{R},\ f(x) = \sum_{i=1}^n \left(\frac{1}{c_i - x_i}\right) - b,\ b \geq 0.$

$\Gamma_0 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 0,\ 0 \leq x_i \leq u_i < c_i,\ \forall i \in \{1, 2, ..., n\}\ \},$

$\Gamma_1 = \{\ (x,\ z) \in \mathbb{R}^n \times \mathbb{B} : z = 1,\ f(x) \leq 0,\ l_i \leq x_i \leq u_i,\ \forall i \in \{1, 2, ..., n\}\ \}$, *non vide, et*

$$\Gamma_r^\epsilon = \left\{ \begin{array}{l} (x, z) \in \mathbb{R}^{n+1} : \\ \sum_{i=1}^n \left(\frac{z^2}{zc_i - x_i + (1-z)(u_i+\epsilon)}\right) - zb \leq 0, \\ zl_i \leq x_i \leq u_i,\ \forall i \in \{1, 2, ..., n\}, \\ 0 \leq z \leq 1. \end{array} \right\},$$

alors :

1. $\boldsymbol{\Gamma_r^\epsilon}$ *est une relaxation convexe valide de* $\boldsymbol{conv(\Gamma_0 \cup \Gamma_1)}$.
2. $\boldsymbol{\{(x, z) \in \mathbb{R}^n \times \mathbb{B} : (x, z) \in \Gamma_r^\epsilon\} \equiv \Gamma_0 \cup \Gamma_1}$.

Modèle projeté ($\mathbf{P}_{proj}$) et modèle étendu ($\mathbf{P}_{high}$) On remplace les contraintes (9) par les relaxations convexes définies dans le corollaire précédent :

$$\begin{array}{ll} & \min\ \sum_{e \in E} w_e x_e \\ t.q. & (6),\ (7),\ (8),\ (10),\ (11) \text{ et } (12-14) \\ & \sum_{e \in P_k^i} \left(\frac{{z_k^i}^2}{z_k^i c_e - x_e + (1 - z_k^i)(u_e + \epsilon)}\right) - z_k^i \alpha_k \leq 0,\ \forall k \in K,\ \forall P_k^i \in P(k). \qquad \text{(9-c)} \\ & 0 \leq x_e \leq u_e,\ \forall l \in E. \end{array}$$

(P_{proj})

Dans le Corollaire 1, une seule contrainte non-linéaire a été ajoutée dans la définition de la relaxation, sachant qu'un nombre exponentiel de contraintes pourraient être générées afin d'obtenir une meilleure borne (voir Théorème 1). Nous avons implémenté le modèle (P_{proj}) et un modèle où l'on génère $n \times |K|$ contraintes correspondant aux singletons ($|S| = 1$). Il ressort de cette expérience que la résolution du programme non-linéaire générant $n \times |K|$ contraintes est considérablement ralentie sans pour autant améliorer la borne de manière significative. Seule le modèle (P_{proj}) a donc été retenu pour les tests numériques.
Finalement, on note (P_{high}) le modèle mathématique basé sur la formulation de l'enveloppe convexe telle qu'elle est introduite dans la Programmation Disjonctive [8, 14, 18], ignorant les résultats du Lemme 1.
Par la suite, les quatre modèles sont comparés sur différentes instances de réseaux de télécommunications afin d'évaluer leur efficacité respectives.

État de l'art : application Ben Ameur et Ouorou introduisent dans [3] une reformulation convexe de la contrainte de délai "on/off" s'écrivant :

$$\sum_{e \in P_k^i} \left(\frac{{z_k^i}^2}{c_e - x_e}\right) - z_k^i \alpha_k \leq 0,\ \forall k \in K,\ \forall P_k^i \in P(k). \qquad (15)$$

On montre que les contraintes (9-c) introduites dans (P_{proj}) dominent ces dernières.

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}	P_{high}	P_{red}
rdata1	60	280	100	**(0.4 ; 0)**	**(0.4 ; 0)**	(35 ; 0)	(2 ; 0)
rdata2	61	148	122	(190 ; 5193)	**(155 ; 1012)**	(2997 ; 21312)	(1948 ; 15129)
adata3	100	600	200	(144 ; 335)	**(57 ; 0)**	(206 ; 556)	(159 ; 84)
rdata4	34	160	946	**(3 ; 0)**	**(3 ; 0)**	(2040 ; 11691)	(1485 ; 5845)
rdata5	67	170	761	([∞] ; 14788)	**(251 ; 3357)**	(1549 ; 2793)	([0.03%] ; 15697)
adata6	100	800	500	**(1065 ; 27991)**	(1470 ; 42244)	([1%] ; 6831)	([0.3%] ; 2499)

TABLE 1 – Mono-routage, 2 chemins candidats par demande

Proposition 2 *Les contraintes (9-c) dominent les contraintes (15).*

4 Résultats numériques

Une étude de performance permet de comparer les quatre modèles introduits dans la section 3 sur des réseaux réels (notés radatax) aussi bien que sur des instances générées aléatoirement (notées adatax) pouvant atteindre 100 noeuds et 1000 demandes. L'ensemble des modèles a été implémenté avec l'interface C++ de Bonmin (release 1.1.3) [6], un solveur libre de PNLM convexes (voir http ://www.coin-or.org/Bonmin). La limite en temps de calcul pour Bonmin est fixé à 2 heures, le solveur de Programmes Linéaires En Nombres Entiers (PLNE) sous-jacent est Cbc [12], tandis que Ipopt [19] représente le solveur de Programmes Non-Linéaires Continus.
Ces tests ont été effectués sur un Intel Xeon 1.6 Ghz. Bonmin offre la possibilité de choisir parmi cinq algorithmes de résolution : un Branch & Bound classique basé sur la programmation non-linéaire [10], une décomposition par Outer Approximation [11] ainsi que trois Branch & Cut basés sur la méthode de Quesada et Grossmann ([17]) dont une implémentation directe, une méthode hybride ajoutant une phase initiale d'Outer Approximation Decomposition tout en générant périodiquement des coupes Outer Approximation, et enfin une méthode ajoutant des coupes de plans coupants étendues [20] (similaire à la méthode proposée dans [1]).
Dans ce qui suit, nous rapportons les résultats des tests obtenus avec la méthode hybride qui a été retenue comme étant la plus efficace pour le problème considéré.
Dans les tableaux, sont indiqués le temps de calcul nécessaire à l'algorithme pour atteindre l'optimum ainsi que le nombre de noeuds exploré dans le Branch & Bound sous la forme suivante : **(Temps en secs ; Nombre de noeuds)**.
Si l'optimum n'est pas atteint avant la limite de temps, le gap entre la meilleure solution entière courante et la relaxation continue est indiqué entre crochets. Si aucune solution valide n'est trouvée, le symbole ∞ remplace le gap éventuel.
Dans les tableaux 1, 2 et 3, on considère N = 1, en d'autre termes un chemin candidat unique est autorisé à router l'ensemble de la demande (problèmes de mono-routage). Le nombre de chemins candidats par demande est respectivement de 2, 3 et 10 pour les tableaux 1, 2 et 3. Notons qu'avoir zéro noeuds exploré par Bonmin dans le Branch & Bound signifie que le problème a été résolu durant la première phase d'Outer Approximation Decomposition [11] de l'algorithme hybride. On observe que P_{proj}, représentant le modèle projeté, enregistre les meilleures performances. P_{proj} résout 14 instances parmi 18 tandis que P_{bigM}, P_{high} et P_{red} en résolvent respectivement 10, 11 et 10 instances. En considérant les moyennes géométriques, P_{proj} est respectivement 2.1, 6.7 et 6.3 plus rapide que P_{bigM}, P_{high} et P_{red}. L'avantage en terme de nombre de noeuds est comparable. Si on considère les quatre problèmes résolus par P_{proj} mais pas avec le modèle P_{bigM}, on remarque que P_{proj} est au moins un ordre de grandeur plus rapide que P_{bigM} avec 5 fois moins de noeuds explorés (il est au moins 5.45 et 7.17 plus rapide que P_{high} et P_{red}). Même si P_{high} et P_{red} apportent des relaxations continues

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}	P_{high}	P_{red}
rdata1	60	280	100	(2.7 ; 0)	**(2.4 ; 0)**	(2.8 ; 0)	(11.9 ; 0)
rdata2	61	148	122	(25 ; 0)	**(13 ; 0)**	(994 ; 3671)	(1396 ; 7815)
adata3	100	600	200	([0.28%] ; 157748)	(344 ; 5097)	(722 ; 3286)	**(312 ; 1124)**
rdata4	34	160	946	([0.001%] ; 79807)	**(1525 ; 50583)**	([0.04%] ; 28876)	([0.1%] ; 22438)
rdata5	67	170	761	([0.43%] ; 138618)	**([0.03%] ; 202122)**	([0.2%] ; 9472)	([0.14%] ; 9461)
adata6	100	800	500	([0.006%] ; 176413)	**(934 ; 19351)**	([0.6%] ; 16539)	([0.06%] ; 4067)

TABLE 2 – Mono-routage, 3 chemins candidats par demande

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}	P_{high}	P_{red}
rdata1	60	280	100	(568 ; 13649)	**(231 ; 4762)**	(1415 ; 9263)	(1209 ; 6485)
rdata2	61	148	122	(120 ; 0)	**(66 ; 0)**	(1527 ; 1599)	(1555 ; 2563)
adata3	100	600	200	**(534 ; 5118)**	(644 ; 14216)	(4866 ; 8841)	(6684 ; 11626)
rdata4	34	160	946	([1.9%] ; 79807)	([2.1%] ; 96212)	([∞] ; 3409)	**([1.8%] ; 3156)**
rdata5	67	170	761	([∞] ; 37446)	([∞] ; 30500)	([∞] ; 747)	([∞] ; 1568)
adata6	100	800	500	([2.7%] ; 35520)	**([1.5%] ; 2680)**	([∞] ; 2642)	([∞] ;1001)

TABLE 3 – mono-routage, 10 chemins candidats par demande

de meilleure qualité, ils demeurent des problèmes difficiles à résoudre vu leur nombre considérable de variables. L'instance adata3 du tableau 2 illustre cette réalité, en effet, l'optimum est atteint en développant 1124 noeuds avec P_{red}, la relaxation continue à la racine est résolue en 3.4 secs, tandis que le modèle P_{bigM} explore 157748 noeuds en deux heures sans converger vers l'optimum, le temps de résolution de la relaxation continue à la racine est de 0.5 secs (à titre indicatif, le modèle P_{proj} met 0.9 secs pour résoudre cette relaxation). Néanmoins, en règle général, la difficulté de résolution de P_{high} et P_{red} prend le dessus sur la qualité des relaxations qu'ils apportent. On déduit que le modèle projeté P_{proj} semble incorporer les bornes de bonne qualité des modèles étendus, tout en bénéficiant d'un nombre réduit de variables.

Dans ce qui suit, on considère le cas "multi-routage", permettant d'acheminer des fractions de la demande par plusieurs chemins actifs à la fois. Les tableaux 4, 5 et 6 rapportent respectivement les résultats obtenus pour les instances ayant 2, 3 et 10 chemins candidats par demande. En remarquant que les modèles P_{high} et P_{red} étaient considérablement lents dans les tests du mono-routage et sachant qu'il est nécessaire de rajouter des variables fractionnaires correspondant aux ϕ_k^i pour le cas multi-routage, ces modèles n'ont pas été implémenté pour le cas considéré. La comparaison est donc établie entre les modèles P_{proj} et P_{bigM}. Pour le multi-routage P_{bigM} et P_{proj} résolvent respectivement 14 et 17 instances parmi 18, sachant que pour l'instance non résolue par P_{proj}, le gap est dix fois plus petit que celui du modèle P_{bigM}.

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}
rdata1	60	280	100	(0.8 ; 0)	**(0.4 ; 0)**
rdata2	61	148	122	**(2.2 ; 0)**	(6.1 ; 0)
adata3	100	600	200	(20 ; 0)	**(5.4 ; 0)**
rdata4	34	160	946	(5.6 ; 0)	**(3.4 ; 0)**
rdata5	67	170	761	(81.7 ; 0)	**(77.8 ; 0)**
adata6	100	800	500	(63.9 ; 0)	**(59.1 ; 0)**

TABLE 4 – Multi-routage, 2 chemins candidats par demande

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}
rdata1	60	280	100	(2.7 ; 0)	**(1.2 ; 0)**
rdata2	61	148	122	**(10 ; 0)**	(29.7 ; 0)
adata3	100	600	200	(195.7 ; 164)	**(50.5 ; 0)**
rdata4	34	160	946	**(20.7 ; 0)**	(43 ; 0)
rdata5	67	170	761	(2503.1 ; 34783)	**(362.8 ; 4024)**
adata6	100	800	500	(1482.4 ; 6083)	**(224.7 ; 326)**

TABLE 5 – Multi-routage, 3 chemins candidats par demande

	$\|V\|$	$\|E\|$	$\|K\|$	P_{bigM}	P_{proj}
rdata1	60	280	100	(799.7 ; 12633)	**(220.8 ; 1922)**
rdata2	61	148	122	**(16.1 ; 0)**	(24.8 ; 0)
adata3	100	600	200	([0.08%] ; 94194)	**(768.6 ; 5207)**
rdata4	34	160	946	([0.4%] ; 40820)	**([0.04%] ; 45492)**
rdata5	67	170	761	([1.2%] ; 16106)	**(5467.7 ; 17347)**
adata6	100	800	500	([0.7%] ; 5880)	**(5392 ; 23867)**

TABLE 6 – Multi-routage, 10 chemins candidats par demande

5 Conclusion

L'étude et la définition des formulations explicites des enveloppes convexes associées à des unions d'ensembles constitue un défit majeur en programmation disjonctive. Tandis que plusieurs travaux abordent ce sujet pour des unions d'ensembles convexes quelconques, nous nous sommes penchés sur le cas particulier traitant de l'union d'un hyper-rectangle et d'un ensemble convexe fermé. Des formulations dans des espaces de dimensions inférieures ont été établies, aussi bien que des projections dans l'espace des variables d'origine. Ce cas considéré est souvent rencontré dans les problèmes d'optimisation ayant des contraintes de type "on/off". Comparés aux modèles existant, les nouvelles formulations introduites permettent de gagner jusqu'à un ordre de grandeur en temps de calcul, résolvant des instances non résolues auparavant. S'appliquant à tout PNLM sous contraintes "on/off", nos résultats permettent d'apporter des modèles mathématiques plus efficaces, améliorant les performances des algorithmes de résolution existants.

Références

[1] K. Abhishek, S. Leyffer, and J. T. Linderoth. FilMINT : An outer-approximation-based solver for nonlinear mixed integer programs. Preprint ANL/MCS-P1374-0906, Mathematics and Computer Science Division, Argonne National Laboratory, 2006.

[2] S. Aktürk, A. Atamtürk, and S. Gürel. A strong conic quadratic reformulation for machine-job assignment with controllable processing times. Technical Report BCOL Research Report 07.01, Industrial Engineering & Operations Research, University of California, Berkeley, 2007.

[3] W. Ben Ameur and A. Ouorou. Mathematical models of the delay constrained routing problem. *Algorithmic Operations Research*, 1(2) :94–103, 2006.

[4] E. Balas. Disjunctive programming. In *Annals of Discrete Mathematics 5 : Discrete Optimization*, pages 3–51. North Holland, 1979.

[5] E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM J. Alg. Disc. Math.*, 6 :466–486, 1985.

[6] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter. An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, 5(2) :186–204, 2008.

[7] P. Bonami, G. Cornuéjols, A. Lodi, and F. Margot. A feasibility pump for mixed integer nonlinear programs. *Mathematical Programming*, 119(2) :331–352, 2009.

[8] S. Ceria and J. Soares. Convex programming for disjunctive optimization. *Mathematical Programming*, 86 :595–614, 1999.

[9] G. Codato and M. Fischetti. Combinatorial bender's cuts for mixed-integer linear programming. *Operations Research*, 54(4) :756–766, 2006.

[10] R. J. Dakin. A tree search algorithm for mixed programming problems. *Computer Journal*, 8 :250–255, 1965.

[11] M. A. Duran and I. Grossmann. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming*, 36 :307–339, 1986.

[12] J. Forrest. CBC, 2004. Available from http ://www.coin-or.org/.

[13] A. Frangioni and C. Gentile. Perspective cuts for a class of convex 0-1 mixed integer programs. *Mathematical Programming*, 106(2) :225–236, 2006.

[14] I. Grossmann and S. Lee. Generalized convex disjunctive programming : Nonlinear convex hull relaxation. *Computational Optimization and Applications*, 26 :83–100, 2003.

[15] O. Günlük and J. Linderoth. Perspective relaxation of mixed integer nonlinear programs with indicator variables. *Lecture Notes in Computer Science*, 5035 :1–16, 2008.

[16] J. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*, volume 2. Springer-Verlag, Berlin Heidelberg New York, 1993.

[17] I. Quesada and I. E. Grossmann. An LP/NLP based branch–and–bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering*, 16 :937–947, 1992.

[18] R. Stubbs and S. Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86 :515–532, 1999.

[19] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1) :25–57, 2006.

[20] T. Westerlund and F. Pettersson. A cutting plane method for solving convex MINLP problems. *Computers and Chemical Engineering*, 19 :s131–s136, 1995.

6 Annexe

Preuve du Lemme 1 Le Théorème 1 dans [8] permet d'écrire la formulation exacte de $conv(\Gamma_0 \cup \Gamma_1)$ sous la forme : $conv(\Gamma_0 \cup \Gamma_1) = proj_{(x,z)}(\Gamma)$,

$$\text{où } \Gamma = \left\{ \begin{array}{l} (x, z, \lambda_0, \lambda_1, z_0, z_1, x^0, x^1) \in \mathbb{R}^{3n+5} : \\ x = x^0 + x^1, \\ z = z_0 + z_1, \\ \lambda_0 + \lambda_1 = 1, \\ (cl\ f)\left(x^1/\lambda_1\right) \leq 0, \\ l^0\lambda_0 \leq x^0 \leq u^0\lambda_0, \\ l^1\lambda_1 \leq x^1 \leq u^1\lambda_1, \\ z_0 = 0, \\ z_1 = \lambda_1, \\ 0 \leq \lambda^1,\ 0 \leq \lambda^0. \end{array} \right\} = \left\{ \begin{array}{l} (x, z, \lambda_0, x^0, x^1) \in \mathbb{R}^{3n+2} : \\ x = x^0 + x^1, \\ \lambda_0 + z = 1, \\ (cl\ f)\left(x^1/z\right) \leq 0, \\ l^0\lambda_0 \leq x^0 \leq u^0\lambda_0, \\ l^1 z \leq x^1 \leq u^1 z, \\ 0 \leq \lambda_0. \\ 0 \leq z. \end{array} \right\}.$$

Remplaçons $x^0 = x - x^1$ et $\lambda_0 = 1 - z$, on obtient alors :

$$\Gamma = \left\{ \begin{array}{l} (x, z, x^1) \in \mathbb{R}^{2n+1} : \\ (cl\ f)\left(x^1/z\right) \leq 0, \\ l^0\lambda_0 \leq x - x^1 \leq u^0\lambda_0, \\ l^1 z \leq x^1 \leq u^1 z, \\ 0 \leq 1 - z. \\ 0 \leq z. \end{array} \right\} = \left\{ \begin{array}{l} (x, y, z) \in \mathbb{R}^{2n+1} : \\ (cl\ f)\left(y/z\right) \leq 0, \\ x - (1-z)u^0 \leq y \leq x - (1-z)l^0, \\ zl^1 \leq y \leq zu^1, \\ 0 \leq z \leq 1. \end{array} \right\} = cl(\Gamma).$$

Preuve du Théorème 1 Nous allons démontrer que $cl(\Gamma')$ représente la projection de $cl(\Gamma)$, l'ensemble défini dans le Lemme 1, sur l'espace (x, z).

1. On montre que $(x, z) \in cl(\Gamma') \Rightarrow \exists y \in \mathbb{R}^n$ t.q. $(x, y, z) \in cl(\Gamma)$

   Pour tout point $(x, z) \in \Gamma'$, $z \neq 0$, soit (x, y, z) défini par :
   $y_i = \max\{zl_i^1,\ x_i - (1-z)u_i^0\},\ \forall i \in J^1$ et $y_i = \min\{zu_i^1,\ x_i - (1-z)l_i^0\},\ \forall i \in J^2$.
   On peut remarquer qu'il existe $S \subset \{1, 2, ..., n\}$ tel que $h_S(x/z) = y/z$. Ayant $zq_S\left(x/z\right) = zf\left(h_S(x/z)\right) \leq 0$ dans Γ' on a $zf(y/z) \leq 0$. Les contraintes restantes de Γ étant satisfaites par définition, on déduit que $(x, y, z) \in \Gamma$.
   Considérons les points restants $(x, 0) \in cl(\Gamma')$. Il existe une suite de points $(x^k, z^k) \in \Gamma'$ tel que $\lim_{k\to\infty}(x^k, z^k) = (x, 0)$. Soit $y^k = y\ \forall k \in \mathbb{N}$, on déduit que $(x^k, y^k, z^k) \in \Gamma$ et $\lim_{k\to\infty}(x^k, y^k, z^k) = (x, y, 0)$. Ceci suffit à prouver que $(x, y, 0) \in cl(\Gamma)$.

2. On montre que $(x, y, z) \in cl(\Gamma) \Rightarrow (x, z) \in cl(\Gamma')$

   Soit (x, y, z) un point dans Γ, $z \neq 0$. Par définition de Γ et des fonctions $h_S(x)$, on a $\forall S \subset \{1, 2, ..., n\}$

$$\frac{y_i}{z} \geq \max\{l_i^1,\ \frac{x_i}{z} - \frac{(1-z)u_i^0}{z}\} \geq (h_S(x/z))_i,\ \forall i \in J^1$$

et

$$\frac{y_i}{z} \leq \min\{u_i^1,\ \frac{x_i}{z} - \frac{(1-z)l_i^0}{z}\} \leq (h_S(x/z))_i,\ \forall i \in J^2.$$

f étant une fonction isotone, on peut écrire $zf(h_S(x/z)) \leq zf(y/z) \leq 0,\ \forall S \subset \{1,2,...,n\}$. Finalement, notons que les contraintes $zl^1 + (1-z)l^0 \leq x \leq zu^1 + (1-z)u^0$ sont obtenues par composition des deux derniers ensembles de contraintes dans Γ. L'extension à la fermeture est immédiate.

Preuve du Corollaire 1

1. $cl(\Gamma'')$ est clairement une relaxation convexe valide de $conv(\Gamma_0 \cup \Gamma_1)$ étant donné qu'elle est définie par un sous ensemble des contraintes de l'enveloppe convexe (voir Théorème 12).
2. Pour $z = 1$, on vérifie que $\Gamma'' \equiv \Gamma_1$. Pour $z = 0$, ayant démontré que Γ'' est une relaxation convexe valide de $(\Gamma_0 \cup \Gamma_1)$, on a $(\Gamma_0 \cup \Gamma_1) \subseteq \Gamma''$. En intersectant ce deux ensembles avec $\{(x,z) : z = 0\}$, on obtient : $(\Gamma_0 \cup \Gamma_1) \cap \{(x,z) : z = 0\} = \Gamma_0 \subseteq \Gamma'' \cap \{(x,z) : z = 0\}$. D'un autre côté, pour $z = 0$, par définition, toutes les contraintes de Γ_0 sont présentes dans Γ'', conduisant à $\Gamma'' \cap \{(x,z) : z = 0\} \subseteq \Gamma_0$.

Preuve de la Proposition 1 En premier lieu, on montre que toutes les contraintes de Γ^ϵ sont convexes, ensuite, on démontre que la projection de ce dernier sur l'espace (x,z) contient Γ_0 et Γ_1. L'unique contrainte non-linéaire dans Γ^ϵ est $g(y,z) \leq 0$ avec $g(y,z) = \sum_{i=1}^n g_i(y_i,z) - b$ et $g_i(y_i,z) = \frac{z^2}{zc_i - y_i + (1-z)\epsilon}$. La matrice hessienne de g_i s'écrit :

$$\mathcal{H}(g_i) = \begin{pmatrix} \frac{2(y_i-\epsilon)^2}{(z(c_i-\epsilon)-(y_i-\epsilon))^3} & \frac{-2z(y_i-\epsilon)}{(z(c_i-\epsilon)-(y_i-\epsilon))^3} \\ \frac{-2z(y_i-\epsilon)}{(z(c_i-\epsilon)-(y_i-\epsilon))^3} & \frac{2z^2}{(z(c_i-\epsilon)-(y_i-\epsilon))^3} \end{pmatrix}.$$

Ayant $y_i \leq zu_i \leq zc_i + (1-z)\epsilon,\ \forall i \in \{1,..,n\}$, il est facile de vérifier que $\mathcal{H}$ est semi-définie positive. Les fonctions g_i sont donc toutes convexes, g étant une somme de fonctions convexes est à son tour convexe. Considérons maintenant la projection de Γ^ϵ sur l'espace (x,z) :

$$\text{Pour } z = 0 \text{ on a } \Gamma^\epsilon = \left\{ \begin{array}{l} (x,y,0) \in \mathbb{R}^{2n+1} : \\ x_i \leq u_i,\ y_i = 0,\ \forall i \in \{1,\ 2,\ ...,\ n\}, \\ x_i \geq y_i,\ \forall i \in \{1,2,...,n\}, \end{array} \right\},$$

dans ce cas $proj_{(x,z)}(\Gamma^\epsilon) = \Gamma_0$.

$$\text{Pour } z = 1 \text{ on a } \Gamma^\epsilon = \left\{ \begin{array}{l} (x,y,1) \in \mathbb{R}^{2n+1} : \\ \sum_{i=1}^n \left(\frac{1}{c_i - y_i}\right) - b \leq 0, \\ x_i = y_i,\ l_i \leq y_i \leq u_i,\ \forall i \in \{1,\ 2,\ ...,\ n\}, \end{array} \right\},$$

dans ce cas $proj_{(x,z)}(\Gamma^\epsilon) = \Gamma_1$.

Finalement, ayant $\frac{z^2}{zc_i - y_i + (1-z)\epsilon} \leq \frac{z^2}{zc_i - y_i}$, la validité de ces contraintes est préservée.

Preuve Corollaire 2 f étant une fonction convexe fermée et isotone, le Corollaire 1 s'applique. En remplaçant la fonction f par son expression explicite dans ce corollaire on obtient la contrainte

suivante : $\sum_{i=1}^{n}\left(\frac{z^2}{zc_i-x_i+(1-z)u_i}\right)-zb\leq 0$. Ayant $\frac{z^2}{zc_1-x_1+(1-z)(u_1+\epsilon)}\leq\frac{z^2}{zc_1-x_1+(1-z)u_1}$, la validité de ces nouvelles contraintes est préservée, la convexité est également maintenue étant donné qu'on peut remplacer $(u_i+\epsilon)$ par v_i conduisant aux contraintes initiales. Parallèlement, il est facile de vérifier qu'en remplaçant z dans Γ_r^ϵ respectivement par 0 et 1, on retrouve les définitions de Γ_0 et Γ_1.

Preuve Proposition 2 Les contraintes (9-c) dominent les contraintes (15) si et seulement si $\sum_{e\in P_k^i}\left(\frac{{z_k^i}^2}{c_e-x_e}\right)-z_k^i\alpha_k\leq\sum_{e\in P_k^i}\left(\frac{{z_k^i}^2}{z_k^ic_e-x_e+(1-z_k^i)(u_e+\epsilon)}\right)-z_k^i\alpha_k,\ \forall k\in K,\ \forall P_k^i\in P(k)$.

Par définition des variables u_e, $\forall e\in E$, on peut écrire :

$u_e+\epsilon\leq c_e\Rightarrow u_e+\epsilon-c_e\leq 0\Rightarrow {z_k^i}^2(1-z_k^i)(u_e+\epsilon)-{z_k^i}^2(1-z_k^i)c_e\leq 0$

$\Rightarrow {z_k^i}^2(1-z_k^i)(u_e+\epsilon)-{z_k^i}^2c_e+{z_k^i}^3c_e\leq 0\Rightarrow {z_k^i}^2(1-z_k^i)(u_e+\epsilon)-{z_k^i}^2c_e+{z_k^i}^3c_e+{z_k^i}^2x_e-{z_k^i}^2x_e\leq 0$

$\Rightarrow {z_k^i}^2(z_k^ic_e-x_e+(1-z_k^i)(u_e+\epsilon))-{z_k^i}^2(c_e-x_e)\leq 0\Rightarrow\frac{{z_k^i}^2}{(c_e-x_e)}-\frac{{z_k^i}^2}{(z_k^ic_e-x_e+(1-z_k^i)(u_e+\epsilon))}\leq 0$

$\Rightarrow\frac{{z_k^i}^2}{(c_e-x_e)}\leq\frac{{z_k^i}^2}{(z_k^ic_e-x_e+(1-z_k^i)(u_e+\epsilon))}\Rightarrow\sum_{e\in P_k^i}\left(\frac{{z_k^i}^2}{c_e-x_e}\right)\leq\sum_{e\in P_k^i}\left(\frac{{z_k^i}^2}{z_k^ic_e-x_e+(1-z_k^i)(u_i+\epsilon)}\right)$

$\Rightarrow\sum_{e\in P_k^i}\left(\frac{{z_k^i}^2}{c_e-x_e}\right)-z_k^i\alpha_k\leq\sum_{e\in P_k^i}\left(\frac{{z_k^i}^2}{z_k^ic_e-x_e+(1-z_k^i)(u_i+\epsilon)}\right)-z_k^i\alpha_k.$

Résolution heuristique du *Stacker Crane Problem* préemptif et asymétrique à l'aide d'une *Arbre-représentation* des tournées

M. Lacroix[2*], H. Kerivin[1], A. Quilliot[2], H. Toussaint[2]

[1] Department of Mathematical Sciences, Clemson University, CLEMSON, O-326 Martin Hall, Clemson, SC 29634 - USA.
kerivin@clemson.edu

[2] LIMOS, CNRS UMR 6158, Université Blaise-Pascal - Clermont-Ferrand II, Complexe Scientifique des Cézeaux, 63177 Aubière, Cedex - France
{lacroix, quilliot, toussain}@isima.fr

Résumé : *Le Stacker Crane Problem est un problème de tournées impliquant l'utilisation d'un unique véhicule. Nous nous intéressons ici à la résolution de sa version préemptive et asymétrique. Cette résolution repose tout d'abord sur des résultats théoriques permettant de modéliser les solutions du problème sous forme d'arbre et de se ramener ainsi à un problème peu contraint. Ensuite cette représentation est utilisée pour réaliser une heuristique de construction gloutonne ainsi qu'une recherche locale simple et efficace. Enfin des résultats expérimentaux confirment l'efficacité de cette approche et l'impact de l'hypothèse de préemption.*

Mots-Clés : *Stacker Crane Problem, relais, tournée de véhicules, recherche locale, heuristique.*

1 Introduction

Les problèmes de Ramassage et Livraison (*Pickup and Delivery*), qui consistent à planifier le transport de biens ou de personnes depuis un nœud origine jusqu'à un nœud destination en utilisant un ensemble donné de véhicules, ont été largement étudiés. Plusieurs variantes ont été considérées, voir [7] et [26] pour des études et méthodes de résolution de ces problèmes. Parmi tous les problèmes de Ramassage et Livraison, le *Stacker Crane Problem* est caractérisé par le fait de disposer d'un unique véhicule, ne transportant qu'une demande à la fois. Ce problème tire son nom des grues-portiques utilisées notamment pour le chargement ou déchargement des navires. Ces grues ne peuvent manipuler qu'un conteneur à la fois, il faut donc porter une attention particulière à l'ordre des chargements et déchargements pour minimiser les déplacements de la grue.

Le *Stacker Crane Problem* (**SCP**) peut se définir de la manière suivante : étant donnés un graphe G dont les arcs sont valués par des longueurs (ou coûts) depuis un nœud "dépôt" spécifique et un ensemble de demandes K, il s'agit de déterminer la tournée d'un unique véhicule V de telle sorte

* nouvelle adresse : lacroix@lamsade.dauphine.fr, Laboratoire LAMSADE, Université Paris-Dauphine, Place du Maréchal de Lattre de Tassigny

que chaque demande $k \in K$ soit transportée de son nœud origine o_k à son nœud destination d_k. Chaque demande k est associée à une charge unitaire et le véhicule V ne peut transporter plus d'une charge à la fois. Il s'agit donc trouver une tournée Γ dans G, la plus courte possible, qui commence et finit au dépôt et permettant à V de satisfaire toutes les demandes. Dans le **SCP** préemptif (SCPP), chaque charge peut être déchargée en n'importe quel nœud du graphe avant d'être rechargée. Ce processus peut être effectué plusieurs fois avant que la charge arrive au nœud destination. On parle de **SCP** asymétrique lorsque la fonction qui à tout arc (x, y) fait correspondre un coût $Dist(x, y)$ est asymétrique.

Le **SCP** a tout d'abord été introduit par Frederickson et al. dans [19], sous sa forme symétrique et non préemptive. Ces auteurs fournissent une preuve de sa NP-complétude en utilisant une réduction à partir du problème du Voyageur de Commerce (*Traveling Salesman Problem*, TSP). Ils fournissent également un schéma d'approximation 9/5 pour ce problème. Atallah et Kosaraju [5] ont été les premiers à traiter le cas préemptif pour le **SCP** symétrique. Ils ont étudié à la fois le cas préemptif et non préemptif dans le cas où le graphe sous-jacent est un chemin élémentaire ou un cycle élémentaire. Ils ont prouvé que dans les deux cas le problème se résout en un temps polynomial. Frederickson et Guan [17], [18] ont étudié les versions préemptives et non préemptives du **SCP** symétrique dans le cas où le graphe sous-jacent est un arbre. Plusieurs variantes du problème de Ramassage et Livraison proches du **SCP** ont été étudiées. On peut mentionner le problème du Voyageur de Commerce qui correspond au **SCP** non préemptif et dans lequel il n'y a pas de contrainte de capacité (voir [28, 14, 27]). La version asymétrique du TSP a été traitée par des approches polyhédrales, des algorithmes de coupes et branchements (*branch and cut*) [3, 4, 6, 21], et par des heuristiques [10, 20, 24]. Hernández-Pérez et González [22] se sont intéressés au TSP avec contraintes de capacité. D'autres contraintes ont été considérées relatives à l'ajout de fenêtres de temps [26], à des relations de précédence imposées par les demandes [15], [16], à une politique de chargement de type LIFO [11]. Le cas où chaque demande possède plusieurs origines et destinations, et dans lequel, toute charge ramassée à une origine de la demande peut être utilisée pour livrer n'importe quelle destination de cette même demande, a également été considérée. Cette extension, appelée le problème d'échange (*Swapping Problem*), [1, 2, 8, 9], appartient à la classe des problèmes de Ramassage et Livraison multi-origines et multi-destinations (*many-to-many Pickup and Delivery problems*), voir [7]. Le problème avec transbordements, correspondant au transport d'une demande à l'aide de plusieurs véhicules, a été étudié dans [13, 23, 25].

Nous nous intéresserons ici au *Stacker Crane Problem* Préemptif et Asymétrique, que nous noterons **SCPPA**. Nous commencerons par donner une formulation du problème, puis nous montrerons qu'il est possible de modéliser les objets solutions sous forme d'arbres. Nous verrons que cette formulation permet à la fois une résolution par programmation linéaire en nombres entiers et par des heuristiques de construction et recherches locales fournissant des résultats numériques de bonne qualité que nous présenterons en dernière partie.

2 Description formelle du SCPPA

2.1 Notations préliminaires sur les listes

Soit $\Gamma = \{x_1, \ldots, x_n\}$ une liste constituée d'objets $x_i, i \in \{1, \ldots, n\}$.

– Pour tout élément $x = x_i$ de Γ, nous noterons $Succ_\Gamma(x)$ (resp. $Pred_\Gamma(x)$), le successeur x_{i+1} (resp. prédecesseur x_{i-1}) de x dans Γ, et par $Rg_\Gamma(x)$ le rang de x dans Γ. Une tournée constituée d'un unique élément x est notée $\{x\}$, et une tournée vide est notée $\emptyset$. Le premier (resp. dernier)

élément de Γ est noté $First(\Gamma)$ (resp. $Last(\Gamma)$). Le nombre d'éléments de Γ est noté $|\Gamma|$.

– On appelle **sous-tournée** de Γ une tournée Γ' telle que $\Gamma' = \{x_{i_1}, \ldots, x_{i_p}\}$ avec $i_1 < \cdots < i_p$.

– On notera $\oplus$ l'opérateur de **concaténation** qui transforme deux séquences $\Gamma = \{x_1, \ldots, x_n\}$ et $\Gamma' = \{y_1, \ldots, y_n\}$ en une unique séquence $\Gamma \oplus \Gamma' = \{x_1, \ldots, x_n, y_1, \ldots, y_n\}$.

– On appelle **coupe** de Γ une décomposition de Γ en une concaténation $\Gamma' \oplus \Gamma''$.

– Soient x_i et x_j deux éléments de Γ tels que $i = Rg_\Gamma(x_i) \leq j = Rg_\Gamma(x_j)$. On appelle **segment** de Γ la sous-liste $\{x_i, \ldots, x_j\}$ de Γ qui est définie par tous les z tels que $i \leq Rg_\Gamma(z) \leq j$.

2.2 Modélisation du SCPPA

Nous rappelons que le **SCPPA** peut se définir de la manière suivante :

– étant donné un graphe G, un véhicule V effectue une tournée dans G afin de satisfaire un ensemble K de demandes. Chaque demande $k \in K$ est exprimée à l'aide d'un couple (o_k, d_k) de nœuds de G, tel que o_k est l'origine de la demande k, d_k est la destination de la demande k, et V doit transporter exactement une unité de charge de o_k vers d_k ;

– la capacité de chargement de V est égale à 1 ;

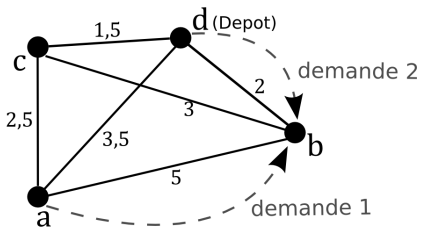
– le véhicule V est autorisé à traiter une demande de manière préemptive : lors du transport d'une charge C_1 il peut s'arrêter à un noeud x quelconque du réseau, décharger C_1, traiter une autre demande, puis revenir en x recharger C_1. Un tel nœud x est appelé **relais** pour la demande k ;

– V doit commencer et finir sa tournée à un dépôt, le nœud associé est noté *Depot*. La tournée doit être la plus petite possible au sens de la fonction coût *Dist* définie dans la partie 1.

Notons X l'ensemble des nœuds du graphe. Les nœuds origine et destination d'une demande pouvant être utilisés comme relais, on duplique ces noeuds afin de les rendre tous différents suivant leur fonction (dépôt, origine, destination ou relais). On peut ainsi décomposer X de la manière suivante : $X = \{Depot\} \cup X_O \cup X_D \cup X_R$ où :
$X_O = \{o_k, k \in K\}$;
$X_D = \{d_k, k \in K\}$;
X_R contient une copie de *Depot*, X_O et X_D et éventuellement un ensemble de relais.

Avec un calcul des plus courts chemins, on fait en sorte que ce graphe soit complet et sa description se ramène à une matrice de distance *Dist*. On suppose que cette fonction coût, non symétrique, *Dist* satisfait les inégalités triangulaires et qu'elle est telle que $Dist(x, x') = 0$ si x' est une copie de x.

La figure 1 (a) illustre un graphe à quatre nœuds "physiques" $\{a, b, c, d\}$ et deux "demandes". On notera que les arcs du graphe sont en traits continus tandis que les demandes sont symbolisées à l'aide de flèches en pointillés. Ainsi le nœud origine (o_1) de la demande 1 est situé en a, et le nœud destination (d_1) en b. Quant à la demande 2, son nœud origine (o_2) est situé en d et son nœud destination (d_2) en b. La matrice à droite de ce graphe donne pour chaque couple (x, y) de nœuds "logiques" le coût $Dist(x, y)$. Elle illustre également la décomposition de X en $X = \{Depot\} \cup X_O \cup X_D \cup X_R$.



(a)

		X_O		X_D		X_R			
	Depot	o_1	o_2	d_1	d_2	a	b	c	d
Depot	0	3.5	0	2	2	3.5	2	1.5	0
o_1	3.5	0	3.5	5	5	0	5	2.5	3.5
o_2	0	3.5	0	2	2	3.5	2	1.5	0
d_1	2	5	2	0	0	5	0	3	2
d_2	0	3.5	0	2	2	3.5	2	1.5	0
a	3.5	0	3.5	5	5	0	5	2.5	3.5
b	2	5	2	0	0	5	0	3	2
c	1.5	2.5	1.5	3	3	2.5	3	0	1.5
d	0	3.5	0	2	2	3.5	2	1.5	0

(b)

FIGURE 1 – (a) Problème à 4 sommets et 2 demandes (symbolisées par les flèches en pointillés) ; (b) Matrice des coûts ($Dist$) associée après duplication des sommets

2.3 Liens labellisés, tournées et tournées valides

On représente le déplacement du véhicule V d'un nœud x à un nœud y de X grâce à un lien labellisé : un lien labellisé est un triplet $r = (x, y, k)$, où x et y sont dans X et k est un label de l'ensemble $\{0\} \cup K$, x (resp. y) est noté $Start(r)$ (resp. $End(r)$) et k est appelé le label de r, noté $Label(r)$. La signification de k est alors que V est vide si $k = 0$ et contient la charge associée à la demande k sinon. Une tournée définie sur X est donc une suite Γ de liens labellisés.

Pour une telle tournée Γ et un label $k \in \{0\} \cup K$, on note Γ_k la suite des liens labellisés qui dérive naturellement de Γ en considérant dans Γ uniquement les liens labellisés r tels que $Label(r) = k$.

Le coût de Γ est donné par $Cout\_tournee(\Gamma) = \sum_{r \in \Gamma} Dist(Start(r), End(r))$.

Il est clair qu'une tournée ainsi définie ne représente pas forcément une solution du **SCPPA**. Il faut pour cela qu'elle représente le déplacement d'un véhicule qui traite toutes les demandes $k \in K$ de manière appropriée. Une telle tournée est dite **valide**, ce qui signifie que :
- pour chaque couple $(r, r' = Succ_\Gamma(r))$ de liens labellisés consécutifs dans Γ on a $End(r) = Start(r')$;
- $Start(First(\Gamma)) = End(Last(\Gamma)) = Depot$;
- chaque nœud $x \in X_O \cup X_D$ appartient à 2 liens labellisés r et $r' = Succ_\Gamma(r)$;
- le nœud $Depot$ appartient uniquement à $First(\Gamma)$ et $Last(\Gamma)$;
- pour chaque demande $k \in K$ la sous-tournée Γ_k associée est telle que :
  - $Start(First(\Gamma_k)) = o_k$;
  - $End(Last(\Gamma_k)) = d_k$;
  - pour chaque couple $(r, r' = Succ_{\Gamma_k}(r))$ de liens labellisés consécutifs dans Γ_k on a $End(r) = Start(r')$.

La figure 2 montre une tournée valide Γ = { ($Depot$, o_1, 0), (o_1, x, 1), (x, o_2, 0), (o_2, y, 2), (y, o_3, 0), (o_3, x, 3), (x, d_1, 1), (d_1, y, 0), (y, d_2, 2), (d_2, 0, x), (x, d_3, 3), (d_3, $Depot$, 0)}. Les valeurs sur les arcs indiquent l'ordre de parcours.

Le **SCPPA** peut alors être posé de la manière suivante : étant donnés l'ensemble des nœuds X et la matrice des coûts $Dist$, calculer une tournée valide Γ de coût minimal.

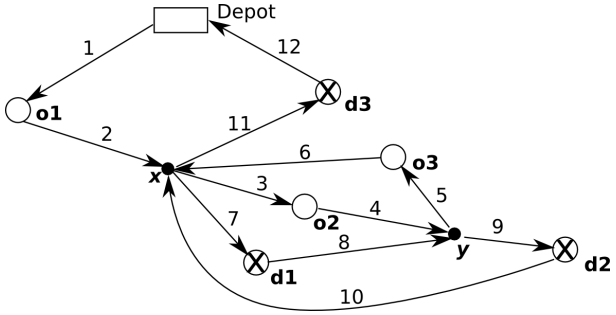


FIGURE 2 – Exemple de tournée valide

3 Résultats structuraux

Le **SCPPA** est un problème difficile à traiter en se basant sur la représentation usuelle de la notion de tournée du fait des contraintes qui pèsent sur ces tournées et de l'hypothèse de préemption. Nous allons ici établir quelques résultats qui permettront de nous ramener à un problème moins contraint dans lequel l'objet inconnu est un arbre.

3.1 Théorème de restriction

Soit Γ une tournée valide. Pour chaque lien labellisé $r = (x, o_k, 0)$ dans Γ, nous notons $\sigma_\Gamma(r)$ l'unique lien labellisé $(d_k, y, 0)$ également dans Γ. De même, si x est un nœud de X_R tel qu'il existe un lien labellisé $r = (y, x, k), k \geq 1$ dans Γ, alors on notera $\sigma_\Gamma(r)$ le premier triplet $r' = (x, z, k)$ qui existe dans Γ_k. Remarquons que $\sigma_\Gamma(r)$ n'est défini que pour certains liens r (ceux de la forme $(x, o_k, 0)$ et ceux de la forme $(y, x, k), k \geq 1$).

On dira que deux liens labellisés r et r', pour lesquels $\sigma_\Gamma(r)$ et $\sigma_\Gamma(r')$ sont définis, **se recouvrent** si $Rg_\Gamma(\sigma_\Gamma(r')) > Rg_\Gamma(\sigma_\Gamma(r)) > Rg_\Gamma(r') > Rg_\Gamma(r)$.

On peut alors établir le *théorème de restriction* dont la signification est que l'on peut restreindre la recherche de solutions optimales pour le **SCPPA** à un sous-domaine dont les éléments sont des tournées valides dotées de propriétés additionnelles qui nous permettent de les représenter de manière très simple.

Théorème 1 (Théorème de restriction) *Soit Γ une tournée optimale pour le **SCPPA**, telle que :*
- *(A) : $|\Gamma|$ est le plus petit possible ;*
- *(B) : Le nombre de liens labellisés r dans Γ tels que $Label(r) \neq 0$, est le plus petit possible, (A) étant supposé satisfait.*

Alors les 4 assertions suivantes doivent être satisfaites :
- *(S1) : Γ ne doit pas contenir 2 occurrences du même lien labellisé $r = (x, y, k)$, avec $k \neq 0$;*
- *(S2) : Γ ne doit pas contenir 2 liens labellisés consécutifs r et r' tels que $Label(r) = Label(r')$;*
- *(S3) : Γ ne doit pas contenir 2 liens labellisés r et r' qui se recouvrent ;*
- *(S4) : Γ ne doit pas contenir 2 liens labellisés r et r' tels que $End(r) = End(r')$ et qui soient tous deux de label non nul.*

Nous ne fournissons pas la preuve de ce résultat ici (voir [29] pour la démonstration).

3.2 *Arbre-Reformulation* du SCPPA

Le théorème précédent nous conduit à réduire nos recherches lorsqu'on traite le **SCPPA** à des tournées valides qui satisfont les propriétés (S1) à (S4). Nous nommerons de telles tournées des tournées **fortement valides**. La figure 3 (a) montre la tournée de la figure 2 transformée en tournée fortement valide sans aucune perte au niveau du coût.

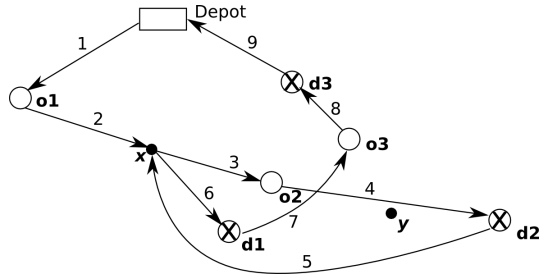


FIGURE 3 – Tournée de la figure 2 transformée en tournée fortement valide

Ainsi résoudre le **SCPPA** revient à trouver une tournée fortement valide et de coût minimum.

Nous allons voir maintenant comment une tournée fortement valide peut être représentée comme un arbre, ce qui nous fournira les bases pour l'algorithme résolvant le **SCPPA** que nous introduirons dans la partie suivante.

On appelle **arbre biparti ordonné** un arbre T tel que :
– Ses nœuds peuvent être ordonnés en deux classes A et B de telle manière que la classe A a ses fils dans la classe B et vice versa ;
– Pour chaque nœud x dans T qui n'est pas une feuille (nœud terminal) l'ensemble des fils associés à x est ordonné linéairement et décrit sous forme de liste.

On dira qu'un arbre T biparti ordonné est **consistant** avec le **SCPPA** défini par l'ensemble K des demandes et par l'ensemble X des nœuds ((X,K)-consistant) si :
– Un nœud dans T peut être identifié soit par une demande $k \in K$ (nous pourrons ainsi parler d'un nœud *demande*), soit par un nœud dans $\{Depot\} \cup X_R$ (nous parlerons alors d'un nœud *relais*). Notons que tous les nœuds *demande* existants doivent apparaître dans T alors qu'il peut n'y avoir que quelques (voire aucun) nœuds parmi les nœuds *relais* dans T. Les nœuds *relais* de T forment l'ensemble des nœuds *relais actifs* de T, noté $Actif(T)$;
– La racine de T est le *Depot* et les feuilles sont des nœuds *demande* ;
– Pour chaque demande k, son ensemble de fils linéairement ordonné $Relais_T(k)$ (qui peut-être vide) est composé de nœuds *relais* actifs et son père $Pere_T(k)$ est dans $Actif(T)$;
– Pour chaque nœud *relais* x, son ensemble de fils linéairement ordonné $Demande_T(x)$ est composé de nœuds *demande* et son père $Pere_T(x)$ est dans K.

Pour un tel arbre biparti ordonné T, on peut définir un coût $Cout\_arbre(T)$ de la manière suivante :
– Pour chaque nœud *demande* $k \in K$, on pose :

$$\begin{aligned} \text{Si k est une feuille } Cout_{Demande,T}(k) = \;& Dist(o_k, d_k)\ , \\ \text{sinon } Cout_{Demande,T}(k) = \;& Dist(o_k, First(Relais_T(k))) + Dist(Last(Relais_T(k)), d_k) + \\ & \sum_{\substack{x \in Relais_T(k),\\ x \neq Last(Relais_T(k))}} Dist(x, Succ_{Relais_T(k)}(x))\ ; \end{aligned}$$

– Pour chaque nœud $x \in \{Depot\} \cup X_R$, on pose :

$$Cout_{Relais,T}(x) = Dist(x, o_{First(Demande_T(x))}) + Dist(d_{Last(Demande_T(x))}, x) + \sum_{\substack{k \in Demande_T(x),\\ k \neq Last(Demande_T(x))}} Dist(d_k, o_{Succ(Demande_T(x),k)})\,;$$

– Finalement le coût de T s'écrit : $Cout\_arbre(T) = \sum_{k \in K} Cout_{Demande,T}(k) + \sum_{x \in Actif(T)} Cout_{Relais,T}(x)$.

En exemple, on donne les coûts associés à l'arbre représenté figure 4 (b) :
$Cost_{Demande,T}(1) = Dist(o_1, x) + Dist(x, d_1)\,;$
$Cost_{Demande,T}(2) = Dist(o_2, d_2)\,;$
$Cost_{Demande,T}(3) = Dist(o_3, d_3).$
Le coût des relais s'écrit :
$Cost_{Relais,T}(x) = Dist(x, o_2) + Dist(d_2, x)\,;$
$Cost_{Relais,T}(Depot) = Dist(Depot, o_1) + Dist(d_3, Depot) + Dist(d_1, o_3).$

Les résultats suivants vont permettre de transformer le **SCPPA** en un problème de recherche d'arbre spécifique, plus facile.

Théorème 2 *Il y a une bijection nommée Arbre entre les tournées fortement valides et les arbres bipartis ordonnés (X, K)-consistants. De plus, pour toute tournée fortement valide Γ associée à un arbre T on a $Cout\_arbre(T) = Cout\_tournee(\Gamma)$.* Voir [29] pour la démonstration de ce théorème.

Une tournée fortement valide et l'arbre biparti associé sont représentés figure 4.

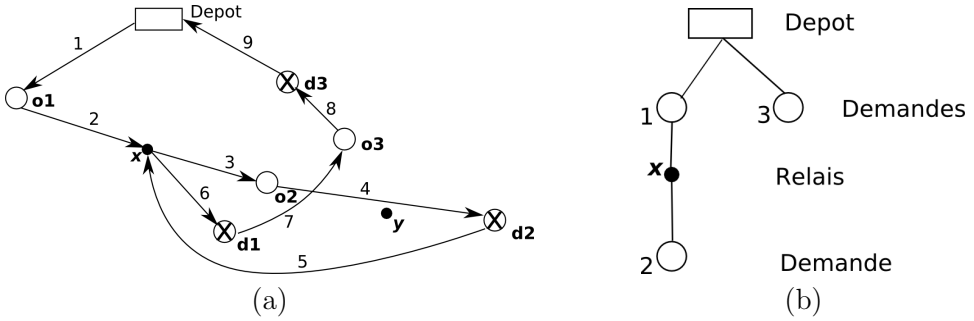


FIGURE 4 – (a) Tournée fortement valide ; (b) Arbre associé

Corollaire 1 *Résoudre un problème **SCPPA** signifie trouver un arbre biparti ordonné T consistant avec X, K et tel que $Cout\_arbre(T)$ est le plus petit possible.*

L'intérêt de ce dernier résultat est clairement de nous fournir une formulation du **SCPPA** grâce à des arbres bipartis, cette formulation étant moins contrainte que le problème original. On notera que les fils d'une demande correspondent aux sommets *relais* sur laquelle celle-ci est rechargée. Les fils d'un sommet *relais* actif x correspondent aux demandes transportées sur le circuit partant de x.

3.3 Formulation du SCPPA avec un programme linéaire en nombres entiers

Construisons tout d'abord un graphe auxiliaire $G = (X^*, E)$:
– $X^* = X \cup X_R^* \cup \{Depot^*\}$, où X_R^* est une copie de X_R et $Depot^*$ est une copie de $Depot$;
Pour chaque nœud x dans X_R, on note x^* sa copie dans X_R^*. De même, pour chaque origine $x = o_k$ dans X_O, on note x^* le nœud d_k associé dans X_D.

– $E = \{(Depot, x), x \in X_O\} \cup \{(x, Depot^*), x \in X_D\} \cup \{(o_k, d_k), k \in K\} \cup \{(d_k, o_k), k \neq k' \in K\}$
$\cup\{(x, y), (y, x), x \in X_O, y \in X_R\} \cup \{(x, y)(y, x), x \in X_R^*, y \in X_D\} \cup \{(x, y), x \in X_R^*, y \in X_R\}$.
Chaque arc $e = (x, y) \in E$ possède une longueur $Dist^*(e) = Dist(x, y)$.

Nous rappelons qu'un chemin γ d'un tel graphe G est une suite de nœuds telle que, pour chaque nœud x dans γ, le couple $(x, Succ_\gamma(x))$ définit un arc de E. Une tournée fortement valide Γ peut être transformée en un chemin Γ^* de G de telle sorte que :
– (S5) : Γ^* commence en *Depot* et finit en *Depot** et Γ^* est un chemin élémentaire c.-à-d. qu'il visite chaque nœud au plus une fois ;
– (S6) : pour chaque demande $k \in K$, Γ^* visite o_k et d_k dans cet ordre, et pour chaque relais $x \in X_R$, Γ^* visite x si et seulement s'il visite x^*, et dans ce cas, il le fait dans cet ordre ;
– (S7) : pour chaque couple (x, y), $x \neq y$, dans $X_R \cup X_O$ l'implication suivante doit être vraie :
$Rg_{\Gamma^*}(x) < Rg_{\Gamma^*}(y)$ ET $Rg_{\Gamma^*}(y) < Rg_{\Gamma^*}(x^*) \Rightarrow Rg_{\Gamma^*}(y^*) < Rg_{\Gamma^*}(x^*)$
On appelle cette condition la *condition de non recouvrement* ;
– (S8) : $Cout\_Tournee(\Gamma) = \sum_{x \in \Gamma^*, x \neq Depot^*} Dist(x, Succ_{\Gamma^*}(x))$.

Il vient alors à partir du théorème 1, qu'une tournée fortement valide peut être représentée d'une manière qui nous permette d'établir un programme linéaire en nombres entiers. Ce modèle implique un vecteur flot $z = (z_e, e \in E)$ à valeur dans $\{0; 1\}$, un vecteur rang $R = (R_x, x \in X)$ à valeurs entières, et un vecteur de décision t indexé sur les couples $(x, y), x \neq y, (x, y) \in X_R \cup X_O$ à valeur dans $\{0; 1\}$. Ces vecteurs ont la sémantique suivante :
– $\forall e \in E, z_e = 1$ ssi l'arc e est dans γ ;
– $\forall x \in \gamma, R_x$ est égal au rang de x dans γ ;
– $\forall (x, y), x \neq y, (x, y) \in X_R \cup X_O$:

$t_{x,y} = 1$ ssi $Rg_\gamma(y) < Rg_\gamma(x^*)$;

$t_{x,y} = 0$ ssi $Rg_\gamma(x^*) < Rg_\gamma(y)$;

On peut alors traduire les propriétés $(S5) \ldots (S8)$ à l'aide de ces vecteurs et on obtient le programme linéaire suivant (pour chaque contrainte la propriété qu'elle traduit est notée entre parenthèses) :

Vecteurs inconnus :
– $z = (z_e, e \in E)$, à valeur dans $\{0, 1\}$;
– $R = (R_x, x \in X)$, entier positif ;
– $t = (t_{x,y}, x \neq y, (x, y) \in X_R \cup X_O)$, à valeur dans $\{0, 1\}$;

Critère de performance :
– Minimiser $\sum_{e \in E} DIST^*(e).z_e$ *(S8)* ;

Contraintes :
– z est un vecteur flot, qui satisfait les lois de Kirchhoff à chaque nœud sauf en *Depot* et *Depot** ;
– Le flot entrant, induit par z, en *Depot* (respt. *Depot**) est égal à 0 (respt. 1), tandis que le flot sortant associé est égal à 1 (respt. 0) *(S5) et (S6)* ;
– Dans chaque nœud de $X_D \cup X_O$, le flot entrant induit par z est égal à 1 *(S5) et (S6)* ;
– Dans chaque nœud de $X_R \cup X_R^*$, le flot entrant induit par z est au moins égal à 1, et le flot entrant en x est égal au flot entrant en x^* *(S5) et (S6)* ;
– Pour chaque demande $k \in K$, on a $R_{o_k} \leq R_{d_k} - 1$ *(S6)* ;
– Pour chaque relais $x \in X_R$, on a $R_x \leq R_{x^*} - 1$ *(S6)* ;
– Pour chaque arc $e = (x, y) \in E$:

$z_e + (R_x + 1 - R_y)/|X^*| \leq 1$ (traduit l'implication $z_e = 1 \Rightarrow R_x + 1 - R_y \leq 0$) ;

– Pour chaque couple $(x, y), x \neq y, x, y \in X_R \cup X_O$ *(S7)* :

$t_{x,y} + (R_y + 1 - R_{x^*})/|X^*| \leq 1$ (traduit l'implication $t_{x,y} = 1 \Rightarrow R_y + 1 - R_{x^*} \leq 0$ (I_1)).
$t_{x,y} + (R_y - 1 - R_{x^*})/|X^*| \geq 0$ (traduit l'implication $t_{x,y} = 0 \Rightarrow R_y - 1 - R_{x^*} \geq 0$ (I_2)) ;
$t_{x,y} + (R_{y^*} + 1 - R_{x^*})/|X^*| \leq 1$ (traduit l'implication $t_{x,y} = 1 \Rightarrow R_{y^*} + 1 - R_{x^*} \leq 0$ (I_3)).

Notons que le vecteur t sert à traduire la condition de non recouvrement. Les deux premières inégalités (I_1) et (I_2) donnent la sémantique de t : $t_{x,y} = 1$ signifie $R_y < R_{x^*}$ (I_1) et $t_{x,y} = 0$ signifie $R_{x^*} < R_y$ (I_2). La troisième inégalité (I_3) interdit la configuration suivante : $R_x < R_y < R_{x^*} < R_{y^*}$ dans laquelle on a recouvrement. En effet si $R_y < R_{x^*}$, il vient de (I_3) que $R_{y^*} < R_{x^*}$. De même, en inversant les rôles de x et de y, on obtient $R_x < R_{y^*} \Rightarrow R_{x^*} < R_{y^*}$. Donc si on a $R_y < R_{x^*}$ (I_3) donne que les seuls rangs possibles pour y^* et x sont tels que : $R_y < R_{y^*} < R_x < R_{x^*}$ ou $R_x < R_y < R_{y^*} < R_{x^*}$.

Remarquons également que (I_1) et (I_3) sont redondantes. En effet si $R_{y^*} < R_{x^*}$ alors $R_y < R_{x^*}$ car par définition $R_y < R_{y^*}$.

4 Heuristiques à base d'arbres pour le SCPPA

Notation préliminaire : L'opérateur d'affectation sera noté $\leftarrow$. Ainsi $x \leftarrow y$ signifie que la variable x reçoit la valeur de y. Dans un souci de simplification on appelera simplement *relais* un nœud *relais* et *demande* un nœud *demande*.

Les algorithmes décrits et testés ici se déduisent facilement de la représentation à l'aide d'arbres des solutions optimales du **SCPPA** obtenues dans la partie 3.2. Ces algorithmes sont de simples algorithmes gloutons et algorithmes de descente basés sur deux types d'opérateurs : opérateurs d'insertion et opérateurs de transformation locale.

4.1 Solution initiale

Les opérateurs d'insertion opèrent sur un arbre biparti ordonné T, consistant avec l'ensemble des nœuds X et avec un sous-ensemble K' de l'ensemble des demandes K. T est alors un arbre partiel comprenant uniquement les demandes de K'. Ces opérateurs insèrent une demande $k \in K \setminus K'$ dans T. On utilisera deux opérateurs :

– **`Insert-simple`** : ses paramètres sont un relais actif x dans $\{Depot\} \cup X_R$, et une coupe (l_1, l_2) de la séquence $Demande_T(x) = l_1 \oplus l_2$. Il insère la demande k dans cette coupe : $Demande_T(x) \leftarrow l_1 \oplus \{k\} \oplus l_2$. Cet opérateur consiste à insérer la demande k dans la tournée. Celle-ci est alors transportée sans relais après la dernière demande de l_1 et avant la première demande l_2 ;
– **`Insert-avec-relais`** : ses paramètres sont une demande $k' \in K'$, une coupe $c = (l_1, l_2)$ de la suite $Relais_T(k')$, et un relais non actif x. Cet opérateur commence par insérer le relais $\{x\}$ dans la coupe c : $Relais_T(x) \leftarrow l_1 \oplus \{x\} \oplus l_2$. Par conséquent x devient actif. Puis il insère la demande k : $Demande_T(x) \leftarrow \{k\}$. Cet opérateur consiste à insérer la demande k dans la tournée. Pour cela, le véhicule décharge la demande k' au sommet x, va ensuite à l'origine o_k de la demande k et la transporte à sa destination d_k. Il revient enfin au sommet x pour recharger la demande k' et continue son trajet.

Nous pouvons alors proposer un premier algorithme `SCPPA-Insertion` (voir l'algorithme 1) d'insertion glouton, basé sur une arbre-représentation du problème, pour résoudre le **SCPPA** : l'arbre T

est construit en utilisant les opérateurs d'insertion. T est tout d'abord initialisé à la racine $\{Depot\}$. Les demandes sont insérées séquentiellement (dans un ordre aléatoire), en choisissant à chaque fois la meilleure insertion possible dans l'arbre partiel T au sens de $Cout\_arbre(T)$. Pour éviter des tests inutiles nous utilisons un système de filtrage afin de s'orienter plus rapidement vers les jeux de paramètres u intéressants. Cela peut être fait en utilisant, pour chaque couple de nœuds (x, y) dans X_R, des ensembles $V(x)$ et $V(y)$ de voisins de x et y, le milieu z de x et y et en imposant des conditions entre ces variables au moment où l'on calcule les différents composants de u.

Algorithme 1 : **SCPPA**-Insertion

Définir aléatoirement un ordre ρ sur les demandes de K ;
$T = \{Depot\}$;
Pour $k \in K$ suivant l'ordre ρ **faire**
  Choisir un opérateur d'insertion I et le jeu de paramètres u associé tels que l'insertion de k dans T via $I(u)$ induise une augmentation du coût $Cout\_arbre(T)$ la plus faible possible ;
  Appliquer $I(u)$ à T ;

Il est clair que l'algorithme 1 d'insertion s'intègre parfaitement dans un schéma de type Monte-Carlo décrit par l'algorithme 2.

Algorithme 2 : **SCPPA**-Insertion couplé à un schéma de type Monte-Carlo

Paramètre : Δ ;
Pour *i = 1 à* Δ **faire**
  Exécuter SCPPA-Insertion ;
Garder le meilleur résultat ;

4.2 Algorithme de descente

Les opérateurs de transformation locale opèrent sur un arbre biparti ordonné T, consistant avec l'ensemble des nœuds X et l'ensemble des demandes K et ils le modifient. On utilisera six opérateurs :

– **`Remplace-relais`** : ses paramètres sont un relais actif x et un relais non actif x'. Il remplace x par x' ;

– **`Deplace-relais`** : ses paramètres sont deux demandes k et k', un segment l de $Relais_T(k)$ et une coupe $c = (l_1, l_2)$ de $Relais_T(k')$. Il supprime l de $Relais_T(k)$ et l'insère dans la coupe c. On suppose pour utiliser cet opérateur que k ne **domine** pas k' dans T c'est-à-dire k ne doit pas pouvoir être obtenu à partir de k' grâce à une suite d'applications de l'opérateur $Pere$;

– **`Deplace-relais-l`** : ses paramètres sont une demande k, un segment l de $Relais_T(k)$ tel que $Relais_T(k) = l_3 \oplus l \oplus l_4$ et une coupe $c = (l_1, l_2)$ de $l_3 \oplus l_4$. Cet opérateur supprime l de $Relais_T(k)$ et le réinsère dans la coupe c : $Relais_T(k) \leftarrow l_1 \oplus l \oplus l_2$;

– **`Deplace-demande`** : ses paramètres sont deux relais actifs différents x et x', un segment l de $Demande_T(x)$ et une coupe $c = (l_1, l_2)$ de $Demande_T(x')$. Il supprime l de $Demande_T(x)$ et l'insère dans la coupe c. Si $Demande_T(x)$ devient vide, le relais x est alors supprimé de T et devient non actif. On suppose pour utiliser cet opérateur que x ne domine pas x' ;

– **`Deplace-demande-l`** : ses paramètres sont un relais x, un segment l de $Demande_T(x)$ tel que $Demande_T(x) = l_3 \oplus l \oplus l_4$ et une coupe $c = (l_1, l_2)$ de $l_3 \oplus l_4$. Cet opérateur supprime l de

$Demande_T(x)$ et le réinsère dans la coupe c : $Demande_T(x) \leftarrow l_1 \oplus l \oplus l_2$;

– **`Deplace-demande-relais`** : ses paramètres sont un relais actif x, un relais non-actif y , une demande k, un segment l de $Demande_T(x)$ et une coupe $c = (l_1, l_2)$ de $Relais_T(k)$. l est supprimé de $Demande_T(x)$, inséré dans $Demande_T(y)$ (y devient donc actif), y étant inséré dans c. Si $Demande_T(x)$ devient vide (cas où $l = Demande_T(x)$) alors x devient non actif. On suppose pour utiliser cet opérateur que la demande k n'est dominée par aucune des demandes k' dans l.

Les opérateurs de transformation locale nous permettent de construire un algorithme de descente `SCPPA-Descente` décrit par l'algorithme 3 : on construit tout d'abord un arbre T avec `SCPPA-Insertion`. Ensuite on cherche à l'améliorer (c'est-à-dire réduire $Cout\_arbre(T)$) en appliquant des transformations locales successives à l'aide des opérateurs décrits précédemment (voir instruction $(I1)$ dans l'algorithme 3). L'efficacité de l'algorithme dépend en grande partie du choix de l'opérateur I à appliquer et de son jeu de paramètres u. Le nombre de choix envisageables étant très grand, on évite de tester tous les paramètres possibles pour chaque opérateur : on utilise donc un mécanisme de filtrage similaire à celui proposé dans le cadre de `SCPPA-Insertion`. On introduit de plus une valeur seuil H tel que plus H est petit plus le domaine de recherche des paramètres est grand. H décroît alors au fur et à mesure des itérations permettant ainsi d'agrandir l'espace de recherche à chaque fois que la recherche dans l'espace restreint par H a échoué.

Algorithme 3 : **SCPPA**-Descente

Initialiser l'arbre T avec **SCPPA**-Insertion ;
Initialiser la valeur seuil filtrante H ;
stop $\leftarrow$ faux ;
tant que *stop = faux* **faire**
  Rechercher (en utilisant le filtre H) un opérateur I et un jeu de paramètres u tels que l'application de $I(u)$ sur T réduise $Cout\_arbre(T)$ $(I1)$;
  si *la recherche a échoué* **alors** $H \leftarrow H/2$;
  si H *devient trop petit* **alors** stop $\leftarrow$ vrai ;

5 Résultats expérimentaux

Les algorithmes décrits dans la section précédente ont été implémentés en C++ (le compilateur utilisé est celui de Visual Studio 2008), testés sur PC Intel Xeon, 1.86GHz, 3.25 GO de Ram.

Nous nous sommes concentrés sur plusieurs points :
– la capacité de `SCPPA-Insertion` et `SCPPA-Descente` à trouver de manière rapide des solutions proches de l'optimal théorique ;
– les caractéristiques des solutions : nombre de relais dans la solution (c'est-à-dire impact de l'hypothèse de préemption).
Dans ce but nous avons réalisé plusieurs tests. Nous avons utilisé les instances de la librairie TSPLIB (voir http ://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/) pour fournir les ensembles de nœuds X et la matrice de distance $Dist$, les couples origine / destination étant choisis aléatoirement parmi les nœuds de X. Nous avons traité des instances comportant de 20 à 300 nœuds et de 10 à 100 couples origine / destination. Les instances aléatoires ont été résolues de manière exacte par un algorithme de coupes et branchements à l'aide d'un programme linéaire similaire à celui donné dans la section 3.3.

Nous avons tout d'abord testé l'algorithme `SCPPA-Insertion` couplé au schéma Monte-Carlo avec

$\Delta = 100$. Les résultats sont présentés dans le tableau 1.

Nous utilisons les notations suivantes :
- *dem.* : nombre de demandes ;
- *som.* : nombre de sommets ;
- *inst.* : nom de l'instance ;
- *ref* : valeur de la solution optimale obtenue grâce au programme linéaire ;
- *min* : valeur de la meilleure solution obtenue parmi les Δ itérations de `SCPPA-Insertion` ;
- *écart* : écart en % entre *ref* et *min* ;
- *rel.* : nombre moyen de relais actifs dans les solutions produites par `SCPPA-Insertion` ;
- *cpu* : temps moyen en millisecondes d'une exécution de `SCPPA-Insertion`.

dem.	*som.*	*ref.*	*min.*	*écart (%)*	*rel.*	*cpu (ms)*
11	46	24654	25023	1.5	0.06	< 15
11	46	21395	21424	0.14	0.43	< 15
23	94	318959	325027	1.9	0.95	< 15
23	94	315118	322908	2.47	0.53	15
59	238	318099	337259	6.02	0.36	78

TABLE 1 – Tests de `SCPPA-Insertion` couplé au schéma Monte-Carlo avec $\Delta = 100$, réalisés sur 5 instances obtenues à partir des instances de la TSPLIB en choisissant aléatoirement les demandes (tous les nœuds, hormis le dépôt, sont donc soit l'origine d'une demande, soit la destination et les relais doivent être choisis parmi ceux-ci)

Le tableau 2 montre les résultats sur des instances *RELn* que nous avons construites de façon à ce que la solution optimale comporte des relais actifs, testant ainsi la capacité de l'algorithme à les trouver. Elles comportent les caractéristiques suivantes :
- l'instance comporte n nœuds, $n/2$ relais et $((n/2) - 1)/2$ demandes ;
- sa valeur optimale est $\sum_{k \in K} Dist(o_k, d_k)$, et la solution optimale est telle que tous les relais sont utilisés.

La figure 5 visualise sur un plan une tournée solution d'une instance de type *RELn* (avec $n = 26$), ainsi que son arbre associé. En chaque point $\{A \dots G\}$ du plan sont positionnés différents nœuds de l'ensemble X : les nœuds origines sont notés $o_i, i = 1 \dots 6$, les nœuds destinations sont notés $d_i, i = 1 \dots 6$ et le nœud relais représenté est l'unique relais actif noté r. Les valeurs sur les arcs de la tournée indique l'ordre de parcours.

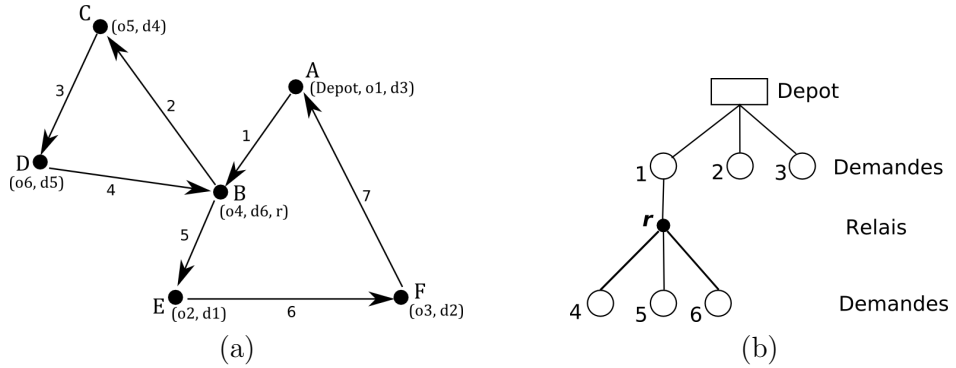


FIGURE 5 – (a) Tournée solution d'une instance "*RELn*" ; (b) Arbre associé

Cette heuristique nous permet d'obtenir des solutions d'assez bonne qualité de manière très rapide. Cependant on remarque que l'heuristique est de moins en moins efficace à mesure que le nombre de relais actifs augmente.

inst.	*ref.*	*min.*	*écart(%)*	*rel.*	*cpu (ms)*
REL62	11843	13045	10.1	0.11	< 15
REL110	17464	19554	12.0	0.35	16
REL158	21053	24235	15.1	0.54	31
REL182	22323	26321	17.9	0.54	47
REL230	24142	28117	16.5	0.91	78
REL326	26036	30734	18.0	1.76	141
REL374	26494	32077	21.1	2.87	187
REL422	26775	32830	22.6	3.39	250
REL518	27042	33773	24.9	5.92	375
REL566	27098	33664	24.2	6.86	453

TABLE 2 – Tests de `SCPPA-Insertion` effectués sur 10 instances "RELn"

Le second test consiste à exécuter `SCPPA-Descente` à partir d'une solution fournie par `SCPPA-Insertion`. On notera :

- *dem.* : nombre de demandes ;
- *som.* : nombre de sommets ;
- *ref* : valeur de la solution optimale obtenue grâce au programme linéaire ;
- *min* : valeur de la solution obtenue après l'application `SCPPA-Descente` ;
- *écart 1* : écart en % entre *ref* et la valeur de la solution initiale produite par `SCPPA-Insertion` ;
- *écart 2* : écart en % entre *ref* et *min* ;
- *rel.* : nombre de relais actifs dans la solution produite par `SCPPA-Descente` ;
- *cpu* : temps moyen en secondes de l'exécution de `SCPPA-Descente`.

dem.	*som.*	*ref.*	*min.*	*écart 1 (%)*	*écart 2 (%)*	*rel.*	*cpu (s)*
11	46	24654	24654	10.3	0	1	0.1
11	46	21395	21914	5.0	2.4	1	0.05
23	94	318959	322879	5.7	1.2	2	2.0
23	94	315118	316362	5.1	0.4	5	0.7
59	238	318099	31913	9.5	0.25	4	103

TABLE 3 – Tests de `SCPPA-Descente` ; réalisés sur 5 instances obtenues à partir des instances de la TSPLIB en choisissant aléatoirement les demandes (tous les nœuds, hormis le dépôt, sont donc soit l'origine d'une demande, soit sa destination, et les relais doivent être choisis parmi ceux-ci)

inst.	*ref.*	*min.*	*écart 1 (%)*	*écart 2 (%)*	*rel.*	*cpu (s)*
REL62	11843	11953	16.7	0.9	4	0.04
REL110	17464	17583	27	0.7	8	0.34
REL158	21053	21053	29.9	0	12	1.5
REL182	22323	22605	20.8	1.2	14	2.9
REL230	24142	24225	28.4	0.34	18	8.4
REL326	26036	26279	19.3	0.93	26	25
REL374	26494	26626	26.6	0.49	30	39
REL422	26775	26851	33.1	0.28	34	78
REL518	27042	27067	37	0.09	42	232
REL566	27098	27211	33.1	0.4	46	328

TABLE 4 – Tests de `SCPPA-Descente` effectués sur 10 instances "RELn"

On peut tout d'abord remarquer que l'écart entre préemption et non préemption est très diffé-

rent suivant la manière dont ont été générées les instances. Bien que notre algorithme de descente `SCPPA-Descente` n'utilise aucun des mécanismes habituels de contrôle destinés à gérer les optima locaux (recherche tabou, recuit simulé ...), on peut constater que les opérateurs qui dérivent de notre représentation en arbre fournissent de très bons résultats dans des temps de calcul très courts.

6 Conclusion

Nous avons traité ici un problème de Ramassage et Livraison avec des demandes préemptives et des contraintes de capacité et nous avons montré qu'il était possible de le transformer en un problème de construction d'arbre peu contraint, de telle sorte que nous puissons le résoudre grâce à de simples et efficaces algorithmes de construction gloutonne et de descente. Il serait intéressant d'étendre cette approche présentée ici dans un contexte spécifique, et d'en déduire des approches efficaces pour traiter les problèmes de préemption dans des problèmes de tournées ou d'ordonnancement plus généraux.

Références

[1] S. Anily, M. Gendreau, G. Laporte. The preemptive swapping problem on a tree. Technical report, Les Cahiers du GERAD, G-2005-69, 2006.

[2] S. Anily, R. Hassin. The swapping problem. Networks, 22(4) :419-433, 1992.

[3] N. Ascheuer, L. Escudero, M. Grötschel, M. Stoer. A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). SIAM Journal on Optimization, 3 :25-42, 1993.

[4] N. Ascheuer, M. Jünger, G. Reinelt. A Branch & Cut algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints. Computational Optimization and Applications, 17 :61-84, 2000.

[5] M.J. Atallah, S.R. Kosaraju. Efficient Solutions to Some Transportation Problems with Applications to Minimizing Robot Arm Travel. SIAM Journal on Computing, 17 :849, 1988.

[6] E. Balas, M. Fischetti, W. Pulleyblank. The precedence constrained asymmetric traveling salesman problem. Mathematical Programming, 68 :241-265, 1995.

[7] G. Berbeglia, J.F. Cordeau, I. Gribkovskaia, G. Laporte. Static pickup and delivery problems : a classification scheme and survey. TOP : An Official Journal of the Spanish Society of Statistics and Operations Research, 15(1) :1-31, July 2007.

[8] C. Bordenave, M. Gendreau, G. Laporte. A branch-and-cut algorithm for the preemptive swapping problem. Technical Report CIRRELT-2008-23, 2008.

[9] C. Bordenave, M. Gendreau, G. Laporte. Heuristics for the mixed swapping problem. Technical Report CIRRELT-2008-24, 2008.

[10] S. Chen, S.Smith. Commonality and genetic algorithms. Technical Report CMU-RI-TR-96-27, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1996.

[11] J.F. Cordeau, M. Iori, G. Laporte, J.J. Salazar-González. A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. Networks, 2008, in press. Voir http ://neumann.hec.ca/chairedistributique/common/tsppdl-bc.pdf.

[12] J.F. Cordeau, G.Laporte. The dial-a-ride problem : models and algorithms. Annals of Operations Research, 153(1) :29-46, 2007.

[13] C.E. Cortés, M. Matamala, C.Contardo. The Pickup and Delivery Problem with Transfers : Formulation and Solution Approaches. In VII French - Latin American Congress on Applied Mathematics. Springer, 2005.

[14] I. Dumitrescu. Polyhedral results for the pickup and delivery travelling salesman problem. Technical Report, CRT-2005-27, 2005.

[15] M.T. Fiala Timlin. Precedence constrained routing and helicopter scheduling. M. Sc. Thesis, Department of Combinatorics and Optimization University of Waterloo, 1989.

[16] M.T. Fiala Timlin, W.R. Pulleyblank. Precedence constrained routing and helicopter scheduling : heuristic design. Interfaces, 22 :100-111, 1992.

[17] G.N. Frederickson and D.J. Guan. Preemptive Ensemble Motion Planning on a Tree. SIAM Journal on Computing, 21 :1130, 1992.

[18] G.N. Frederickson and D.J. Guan. Nonpreemptive ensemble motion planning on a tree. Journal of Algorithms, 15(1) :29-60, 1993.

[19] G.N. Frederickson, M.S. Hecht, C.E. Kim. Approximation Algorithms for Some Routing Problems. SIAM Journal on Computing, 7 :178, 1978.

[20] L.M. Gambardella, M. Dorigo. An ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS Journal on Computing, 12(3) :237-255, 2000.

[21] L. Gouveia and P. Pesneau. On extended formulations for the precedence constrained asymmetric traveling salesman problem. Networks, 48(2) :77-89, 2006.

[22] H. Hernández-Pérez, J. Salazar-González. The multicommodity one-to-one pickup-and-delivery traveling salesman problem. European Journal of Operational Research, 196 :987-995 2009

[23] S. Mitrovic-Minic, G. Laporte. The pickup and delivery problem with time windows and transshipment. INFOR, 44 :217-227, 2006.

[24] R. Montemanni, D.H. Smith, L.M. Gambardella. A heuristic manipulation technique for the sequential ordering problem. Computers & Operations Research, 35(12) :3931-3944, 2008.

[25] P. Oertel. Routing with Reloads. Doktorarbeit, Universität zu Köln, 2000.

[26] S.N. Parragh, K.F. Doerner, R.F. Hartl. A survey on pickup and delivery problems : Part II : Transportation between pickups and delivery locations. Journal für Betriebswirtschaft, 58(1) :21-51, 2008.

[27] J. Renaud, F. Boctor, G. Laporte. Perturbation heuristics for the pickup and delivery traveling salesman problem. Computers & Operations Research, 29(9) :1129-1141, 2002.

[28] K.M. Ruland, E.Y. Rodin. The pickup and delivery problem : Faces and branch-and-cut algorithm. Computers and Mathematics with Applications, 33 :1-13, 1997.

[29] H. Kerivin, M. Lacroix, A. Quilliot, H. Toussaint. Heuristic for the preemptive asymmetric stacker crane problem. Rapport de recherche RR09-01, Laboratoire LIMOS, Université Blaise Pascal, Clermont-Ferrand, 2009.

Métaheuristiques pour l'allocation de mémoire dans les systèmes embarqués

María Soto, André Rossi, Marc Sevaux

Université de Bretagne-Sud. Lab-STICC, CNRS UMR 3192
Centre de Recherche B.P. 92116
F-56321 Lorient Cedex FRANCE
{maria.soto, andre.rossi, marc.sevaux}@univ-ubs.fr

Résumé :

La gestion de la mémoire cache a un impact significatif sur les performances et sur la consommation énergétique des systèmes embarqués. Cet article traite de l'allocation de mémoire des structures de données d'une application à la mémoire cache de manière à optimiser les performances du système. Les concepteurs de circuits souhaitent trouver un compromis entre le coût de l'architecture (le nombre de bancs mémoire à embarquer) et la consommation électrique. Le problème abordé consiste à allouer un banc mémoire à toute structure de données de manière à minimiser les conflits d'accès aux données. Le modèle proposé pour ce problème est le k-weighted graph coloring problem. Une formulation par PLNE et deux métaheuristiques basées respectivement sur une recherche taboue et sur un algorithme hybride à base de population sont comparées sur un ensemble d'instances. Les résultats obtenus sont encourageants et suggèrent que l'utilisation de méthodes issues de la coloration de graphes est une piste prometteuse pour l'allocation de mémoire dans les systèmes embarqués.

Mots-Clés : *Allocation de mémoire, métaheuristiques, coloration de graphes*

1 Introduction

Les progrès continus des technologies en microélectronique ont rendu possible le développement de puces miniatures permettant d'étendre considérablement les fonctionnalités des appareils multimédia portatifs (la téléphonie mobile en est un exemple spectaculaire). Alors que la technologie ouvre des perspectives et des marchés toujours plus vastes, la conception des circuits intégrés devient, elle, de plus en plus complexe. D'autant qu'une innovation permanente sur des produits à bas coût s'accompagne naturellement d'une obsolescence rapide des produits fabriqués, qui contraint les concepteurs de circuits à minimiser le temps de conception.

Traditionnellement, les circuits intégrés étaient conçus à la main par des spécialistes qui maîtrisaient la technologie et la complexité du produit dans son ensemble. Une telle façon de faire n'est plus possible aujourd'hui, et des outils de conception assistée par ordinateur comme Gaut [3] ont été introduits pour générer le design (ou l'architecture) d'un circuit intégré à partir de ses spécifications. Bien que les solutions produites par ce biais puissent être générées très rapidement, elles souffrent généralement d'un manque d'optimisation qui se traduit notamment par une consommation énergétique importante, ce qui est particulièrement préjudiciable pour les systèmes embarqués.

Cet article aborde la problématique de l'optimisation de l'allocation de mémoire cache des processeurs embarqués. En effet, la gestion de la mémoire cache a un impact significatif sur les performances et sur la consommation énergétique comme l'ont montré Wuytack *et al.* dans [15]. Lorsque l'on conçoit un circuit intégré pour un système embarqué (typiquement un décodeur MPEG, un filtre numérique ou tout autre algorithme de traitement du signal et de l'image), l'application à synthétiser est fournie sous la forme d'un code source écrit en langage C, et les structures de données manipulées doivent être placées dans des bancs mémoire (il s'agit de la mémoire cache du processeur qui exécute l'application).

Pour des raisons technologiques comme pour des raisons de coût, le nombre de bancs mémoire est limité. On fait également l'hypothèse que le processeur est capable d'accéder à tous ses bancs mémoire simultanément, ce qui autorise plusieurs chargements en temps masqué. Ainsi, les structures de données (ou plus simplement les variables) `a` et `b` peuvent être chargées en même temps lorsqu'on souhaite effectuer l'opération `a+b`, à condition que `a` et `b` se trouvent dans deux bancs mémoire différents. Si ces variables partagent le même banc mémoire, le processeur ne peut y accéder que séquentiellement, ce qui nécessite deux fois plus de temps qu'un accès en parallèle. On dira que `a` et `b` sont en conflit si elles apparaissent dans la même opération. Tout conflit a un coût, qui est proportionnel au nombre de fois où l'opération `a+b` est effectuée dans l'application (ce coût peut être un réel, dans le cas où le code a été "profilé" par un outil logiciel [7, 10] basé sur une estimation stochastique de la probabilité de branchement des instructions conditionnelles). Le recours à une estimation du nombre de fois où une opération est effectuée se présente lorsque l'opération apparaît dans une boucle `while` par exemple.

Un conflit entre deux structures de données sera dit *ouvert* si les deux structures de données partagent le même banc mémoire, il est dit *fermé* si les deux structures de données ont été affectées à deux bancs mémoire différents.

Ainsi, les concepteurs de circuits souhaitent trouver un compromis entre le coût de l'architecture (le nombre de bancs mémoires à embarquer) et la consommation énergétique de l'ensemble [1]. En effet, les électroniciens considèrent généralement que dans une certaine mesure, minimiser la consommation est équivalent à minimiser le temps d'exécution de l'application [2]. De plus, la consommation d'une architecture donnée peut également être estimée à l'aide d'un modèle empirique comme dans [8]. Ainsi, on a tout intérêt à paralléliser l'accès aux données de manière à minimiser le temps total d'exécution de l'application.

Le problème de l'allocation de mémoire peut donc être formulé ainsi : il convient d'allouer un banc mémoire à toute structure de données de l'application, de manière à minimiser le temps total d'accès aux données pour le processeur. Dans la section suivante, on propose un programme linéaire en nombres entiers pour résoudre ce problème de manière exacte. La section 3 présente deux métaheuristiques, une recherche taboue et une méthode à population, enfin la section 4 compare les résultats et discute les approches proposées.

2 Modélisation du problème par PLNE

Soit n le nombre de structures de données de l'application considérée. On désigne par a_i la structure de données i pour tout i dans $\{1, \ldots, n\}$. Le nombre de bancs mémoire disponibles est noté m. Par ailleurs, on note o le nombre de conflits entre deux structures de données. La paire $p_k = (a_{k1}, a_{k2})$ a le coût d_k pour tout k dans $\{1, \ldots, o\}$, ce coût s'exprime en unités de temps (typiquement la milliseconde). Le statut du conflit k dépend de l'allocation des structures de données a_{k1} et a_{k2}. Plus précisément, tout conflit se trouve nécessairement dans l'un des 2 statuts suivants :

– Statut 1 : a_{k1} et a_{k2} sont dans deux bancs mémoire différents. Le conflit est fermé, il n'engendre aucun coût,
– Statut 2 : a_{k1} et a_{k2} sont dans le même banc mémoire. Le conflit est ouvert, il engendre le coût d_k.

Il faut noter qu'une structure de données peut être en conflit avec elle-même. Le cas se produit typiquement lorsque la structure de données est un tableau et que l'application traduit une relation de récurrence comme `a[i] = a[i+1] ;`.

Toutes ces informations sont considérées comme des données fournies par le concepteur (ou plus exactement par un outil de *profiling* du code de l'application). L'entier n désigne le nombre de structures de données n, et m est le nombre de conflits entre ces structures de données. La liste des conflits (a_{k1}, a_{k2}) et leur coût d_k pour tout k dans $\{1, \ldots, o\}$ est une information qui caractérise à la fois l'architecture et l'application.

Les variables de décision du problème d'affectation des structures de données aux bancs mémoire sont :

– La matrice X, où $x_{i,j}$ indique que la structure de données a_i est affectée au banc mémoire b_j pour tout i dans $\{1, \ldots, n\}$ et pour tout j dans $\{1, \ldots, m\}$ ($x_{i,j} = 0$ sinon).
– Le vecteur Y représentant le statut des conflits. Pour tout k dans $\{1, \ldots, o\}$, l'élément y_k est égal à 1 seulement si le conflit k est ouvert, il vaut zéro sinon.

Le coût d'une solution (c'est à dire d'une affectation des structures de données aux bancs mémoire) est égal à la somme des coûts des conflits ouverts. Le coût d'une solution s'exprime comme suit :

$$\text{Minimiser} \sum_{k=1}^{o} y_k d_k \tag{1}$$

Les contraintes suivantes doivent être satisfaites pour que la solution soit réalisable.

Premièrement, toute structure de données est affectée à un banc mémoire :

$$\sum_{j=1}^{m} x_{i,j} = 1 \quad \forall i \in [1, n] \tag{2}$$

Deuxièmement, pour tout conflit, la variable y_k doit avoir la valeur numérique appropriée 0 ou 1. Cela est assuré à l'aide de la contrainte ci-dessous :

$$x_{a_{k1},j} + x_{a_{k2},j} \leq 1 + y_k, \quad \forall k \in [1, o], \forall j \in [1, m] \tag{3}$$

Le problème peut alors être formulé comme un programme linéaire en nombres entiers comprenant la fonction objectif et l'ensemble des contraintes :

$$\begin{aligned}
\text{Minimiser} \quad & \sum_{k=1}^{o} y_k d_k && && (1)\\
\text{sous les contraintes} \quad & \sum_{j=1}^{m} x_{i,j} = 1 && \forall i \in [1, n] && (2)\\
& x_{a_{k1},j} + x_{a_{k2},j} \leq 1 + y_k && \forall k \in [1, o],\ \forall j \in [1, m] && (3)
\end{aligned}$$

$$x_{i,j} \in \{0,1\} \qquad \forall i \in [1,n],\ \forall j \in [1,m] \qquad (4)$$
$$y_k \in \{0,1\} \qquad \forall k \in [1,o] \qquad (5)$$

Il faut noter que ce problème est identique au *k-weighted graph coloring problem* [14] : il s'agit de colorer un graphe non orienté dont les sommets représentent les structures de données et où chaque couleur représente un banc mémoire, de manière à minimiser la somme des coûts des arêtes dont les deux extrémités ont la même couleur. Une arête représente un conflit entre deux structures de données. Un problème proche consiste à déterminer le nombre minimum de bancs mémoire pour lequel aucun conflit n'est ouvert, il s'agit là d'un problème de coloration de graphes [4].

3 Solution du problème d'allocation mémoire

3.1 Méthode de résolution exacte

Une solution optimale peut être obtenue à l'aide d'un solveur, comme GLPK [6] ou Xpress-MP [16]. Mais comme le montrent les tests numériques de la section 4, une solution optimale ne peut pas être obtenue en temps raisonnable pour des instances moyennes. C'est la raison pour laquelle deux métaheuristiques sont proposées dans cet article.

3.2 Métaheuristiques pour le problème d'allocation mémoire

La première métaheuristique est une recherche taboue implémentée à partir d'une méthode taboue pour la coloration de graphes qui s'appelle *TabuCol* et décrit dans le chapitre 10 de [9]. L'autre métaheuristiques est un algorithme hybride à base de population qui s'appuie sur l'algorithme *Evocol (Evolutionary Hybrid Algorithm for Graph Coloring)* décrit dans [12]. Les sections suivantes présentent ces méthodes de manière détaillée.

3.2.1 Méthode taboue

La méthode taboue appartient aux métaheuristiques à base de voisinages qui s'appuient sur un même principe : l'algorithme passe de manière itérative de la solution courante à une solution voisine [13] . Le voisinage $\mathcal{N}(x)$ de la solution courante x dépend du problème traité. Généralement, les opérateurs de recherche locale s'arrêtent quand une solution localement optimale est trouvée. Il est alors nécessaire de sortir des minima locaux pour espérer trouver une solution de meilleure qualité. La méthode taboue y parvient en permettant que la solution courante soit temporairement dégradée. Pour ce faire, certaines modifications sur la solution courantes sont temporairement interdites (ou taboues), ce qui est à l'origine du nom de cette métaheuristique. On note NT le nombre maximal de mouvements interdits, il s'agit de la taille de la liste taboue.

La structure générale de la recherche taboue utilisée pour résoudre le problème est décrite dans l'algorithme 1, et commentée ci-après.

Les données de TabuMemexBasic sont les n structures de données, les m bancs mémoire, le coût des conflits entre les paires de structures de données concernées, deux paramètres de calibrage $Niter$ le nombre d'itérations de l'algorithme et NT la taille de la liste taboue. Le résultat de l'algorithme est la meilleure affectation mémoire trouvée, contenue dans $Best$.

Algorithme 1 : **TabuMemexBasic.**

Données : n structures de données, m bancs mémoire, le coût des conflits, $Niter$ nombre maximum d'itérations et NT la taille de la liste taboue.
Résultat : $Best$: la meilleure allocation trouvée.
`Initialisation`
Choisir une solution initiale s;
$Best \leftarrow s$;
$Iter = 0$;
`Phase itérative`
pour $((Iter \leq Niter)$ *ou* $(cost(s) > 0))$ **faire**
  Générer $s' \in \mathcal{N}(s)$, à partir de s en affectant la structure de données a_i au banc mémoire b_j de façon à ce que $cost(s') < cost(s'')$, $\forall s'' \in N(s)$;
  si (*la paire* (a_i, b_j) *n'est pas dans la liste taboue*) **alors**
    $s \leftarrow s'$;
    Mettre à jour la liste taboue;
    si $cost(s) < cost(Best)$ **alors**
      $Best \leftarrow s$;
    fin
  fin
  Recalculer la taille de la liste taboue NT;
  $Iter \leftarrow Iter + 1$.
fin

La solution initiale est choisie au hasard, c'est-à-dire que les structures de données sont affectées aux bancs mémoire de façon aléatoire. Toute solution initiale générée de cette façon est nécessairement admissible car la capacité des bancs mémoire n'est pas prise en compte.

Dans la phase itérative, l'algorithme effectue la recherche de la meilleure solution au voisinage de la solution courante pendant un nombre maximal d'itérations, $Niter$, mais la recherche peut s'interrompre avant si une solution sans conflit ouvert est trouvée (une telle solution est nécessairement optimale). Le voisinage considéré est constitué de toutes les solutions obtenues en changeant l'affectation d'une structure de données quelconque.

A chaque itération, l'algorithme cherche la solution voisine de coût minimum, même si la solution trouvée est de plus mauvaise qualité que la solution courante. La recherche taboue interdit à la structure de données a_i d'être affectée à son ancien banc mémoire b_j pendant NT itérations, c'est-à-dire qu'une nouvelle solution sera acceptée si la paire (a_i, b_j) n'est pas dans la liste taboue. Chaque fois que la solution courante change, la liste taboue est mise à jour sur le principe du FIFO (First In First Out), et $Best$ est mise à jour si la solution courante est de coût strictement inférieur au coût de $Best$.

La taille de la liste taboue n'est pas constante, elle est ajustée à chaque itération, comme dans les travaux de Porumbel, Hao et Kuntz sur la coloration de graphes [12].

La section 4 présente l'implémentation de TabuMemexBasic de manière détaillée.

3.2.2 Algorithme mémétique

Cette sous-section présente la dernière métaheuristique intitulé EvoMemexBasic qui est décrite dans l'algorithme 2. EvoMemexBasic prend en entrée les n structures de données, le nombre de

bancs mémoire m et deux paramètres liés à l'algorithme : r nombre des parents impliqués dans le croisement et p le nombre d'enfants produits à chaque itération. Il retourne $Best$, la meilleure solution trouvée.

Algorithme 2 : **EvoMemexBasic.**

Données : n structures de donnée, m bancs mémoire, le coûts des conflits, le nombre de parents, r, impliqués dans chaque itération, et le nombre d'enfants, p, produits pour chaque itération.
Résultat : $Best$: la meilleure allocation trouvée.
`Initialisation`
Choisir une population aléatoire de solutions;
`Phase itérative`
répéter
  répéter
    répéter
      Sélectionner r parents ($r > 2$);
      Croiser les parents pour produire un nouvel enfant s;
      Appliquer la méthode TabuMemexBasic à l'enfant s;
      Accepter ou pas l'enfant s;
    jusqu'à *un enfant est accepté* ;
  jusqu'à p *enfants produits* ;
  Actualiser la population de parents avec les p enfants.
jusqu'à *un critère d'arrêt est satisfait* ;

L'initialisation consiste à générer un ensemble de solutions aléatoirement, ce qui est aisé étant donné que toute allocation des structures de données aux bancs mémoire est admissible.

Le principe général de l'algorithme EvoMemexBasic consiste à obtenir p nouveaux enfants en croisant r parents différents, c'est à dire r éléments de la population courante. Le croisement des r parents consiste à choisir les meilleures affectations des structures de données aux bancs mémoire de chaque parent pour former l'enfant comme indiqué dans le prochain paragraphe. Chaque enfant (nouvelle solution) ainsi produit est lui-même amélioré en l'utilisant comme solution initiale de la recherche taboue TabuMemexBasic.

L'algorithme 3 décrit ce croisement, il prend pour chaque banc mémoire les meilleures affectations des structures de données de façon à minimiser le *taux de conflit*, c'est à dire le nombre de conflits du banc mémoire considéré divisé par le nombre de structures affectées à ce banc mémoire. Il choisit les meilleures affectations pour l'enfant et efface de chaque parent les structures déjà assignées. Finalement, afin d'assigner les structures de données qui ne le sont pas encore, le banc mémoire choisi est celui qui donne la plus petite somme des coûts des conflits ouverts.

EvoMemexBasic considère que la population s'améliore si le coût des conflits des solutions décroît à chaque itération, et si la population conserve une diversité suffisante pour assurer une couverture satisfaisante de l'espace des solutions. De ce fait, un enfant n'est accepté dans la population que si la distance à ses parents est supérieure à un certain seuil. La distance entre deux solutions s_a et s_b est définie par le nombre minimum de structures de données dont l'assignation doit changer pour passer de s_a à s_b. Cette définition est couramment utilisée en coloration de graphes [5].

Pour garantir l'amélioration et la diversité de la population, l'algorithme met à jour la population à chaque itération. Nous utilisons la variance statistique des coûts de conflits des solutions comme mesure de diversité de la population. Si cette variance est inférieure à notre seuil nous générons une nouvelle population aléatoirement. Dans chaque actualisation de population nous remplaçons les p

Algorithme 3 : **Croisement-EvoMemexBasic.**

Données : r parents
Résultat : Un enfant.
pour $j = 1, \ldots, m$ **faire**
  pour *chaque parent dans* $1, \ldots, r$ **faire**
    Calculer le taux de conflit du banc mémoire b_j
  fin
  Sélectionner les structures de données du parent avec le plus petit taux de conflit
  Assigner ces structures de données au banc memoire b_j pour construire le nouvel enfant
  pour *chaque parent dans* $1, \ldots, r$ **faire**
    Effacer les structures de données assignées au nouvel enfant
  fin
fin
Assigner les structures de données restantes de façon à minimiser le plus petit coût.

parents (solutions) dont le coût est le plus élevé, par les p enfants. Cette forme de mise à jour présente un bon compromis entre la diversité et la qualité de la population.

4 Résultats expérimentaux et discussion

Cette partie présente les aspects relevant de l'implémentation des algorithmes TabuMemexBasic et EvoMemexBasic, ainsi que les résultats obtenus sur un ensemble d'instances avec un PC basé sur un processeur Intel Pentium IV cadencé à 3 GHz, et 1 Go de mémoire vive (RAM).

L'algorithme TabuMemexBasic change itérativement la taille de la liste taboue toutes les N itérations de manière aléatoire en fonction de $a + N \times t$, où a est un nombre entier fixe et t est un nombre aléatoire entre 0 et 2. Nous avons choisi expérimentalement $N = 50$ et $a = 10$ pour notre méthode, et un nombre d'itérations maximum $Niter = 10000$.

Pour l'implémentation de EvoMemexBasic, nous avons repris les valeurs des paramètres issus des tests effectués dans [12] : une population de 15 éléments, le croisement avec $r = 3$ parents, la génération de $p = 3$ enfants, et un seuil d'acceptation de $R = 0,1 \times n$ pour la distance minimale entre deux éléments de la population. Dans notre expérience, nous avons fixé une limite de 100 itérations comme critère d'arrêt et un minimum de $0,3$ pour la variance de la population pour cette méthode.

La table 1 présente les résultats obtenus avec les paramètres fixés ci-dessus pour un ensemble d'instances. Ces instances proviennent de problèmes de coloration des graphes de DIMACS [11] que nous avons modifiés en donnant des coûts aléatoires aux conflits, sauf pour la première qui est une instance réelle provenant d'un problème d'électronique. Cette table donne le coût des meilleures solutions obtenues avec les deux métaheuristiques présentées ainsi qu'avec la formulation PLNE résolue avec GLPK. Le temps de calcul, exprimé en secondes, est reporté pour chaque méthode. Les deux premières colonnes présentent des informations de base sur les instances comme le nom, le nombre des structures de données, de conflits et de bancs mémoire. La dernière colonne indique, pour chaque instance, le statut (optimal ou non) de la meilleure solution retournée par GLPK après une heure de calcul.

Les résultats de cette expérience montrent que EvoMemexBasic trouve toujours la solution optimale lorsque le PLNE peut être résolu de façon optimale dans la limite d'une heure de calcul.

TabuMemexBasic est plus rapide que EvoMemexBasic, ce qui est normal puisqu'elle constitue un sous-programme de EvoMemexBasic. Il faut noter que la meilleure solution retournée par GLPK lorsque la recherche de la solution optimale n'a pas abouti après une heure est généralement de mauvaise qualité comparée aux résultats des deux métaheuristiques. Pour l'instance `r125.1c`, GLPK n'a pas obtenu de solution entière, même après une heure de calcul.

TABLE 1 – Résultats

Instances		TabuMemexBasic		EvoMemexBasic		GLPK		Optimal
Nom	$n\backslash o\backslash m$	coût	temps	coût	temps	coût	temps	
`mpeg2enc`	180 \227 \2	**32,09**	1,60	**32,09**	3,20	**32,09**	107,79	oui
`queen5_5`	25 \160 \3	**974**	0,73	**974**	3,68	**974**	492,38	oui
`queen6_6`	36 \290 \4	999	1,13	999	5,73	1253	3599,29	non
`queen7_7`	49 \476 \4	1896	1,46	1896	16,10	2430	3600,02	non
`queen8_8`	64 \728 \5	1617	2,06	1617	54,12	2443	3600,01	non
`myciel3`	11 \20 \2	**146**	0,26	**146**	1,82	**146**	0,03	oui
`myciel4`	23 \71 \3	**69**	0,52	**69**	2,47	**69**	1,16	oui
`myciel5`	47 \236 \3	591	0,81	591	3,60	591	3599,41	non
`myciel6`	95 \755 \2	9017	1,26	9017	18,52	9963	3600,43	non
`myciel7`	191 \2360 \4	2262	2,21	2262	55,93	4642	3607,71	non
`r125.1`	125 \209 \3	346	1,22	346	28,87	425	3599,73	non
`r125.1c`	125 \7501 \23	2719	11,27	2685	135,24	-	3820,08	non
`r125.5`	125 \3838 \18	785	8,58	734	156,97	1668	3648,99	non
`mug88_1`	88 \146 \2	**967**	1,09	**967**	15,72	**967**	157,23	oui
`mug88_25`	88 \146 \2	**881**	1,07	**881**	16,27	**881**	53,11	oui
`mug100_1`	100 \166 \2	1149	1,19	**1129**	26,79	**1129**	957,19	oui
`mug100_25`	100 \166 \2	**1142**	1,24	**1142**	18,54	**1142**	562,00	oui

5 Conclusion

Cet article présente deux métaheuristiques, EvoMemexBasic basée sur un algorithme hybride à base de population et TabuMemexBasic basée sur une recherche taboue. Il nous a semblé pertinent d'adapter ces algorithmes issus de la coloration de graphes car le problème abordé peut être vu comme un *k-weighted graph coloring problem*. Les meilleurs résultats sont obtenus avec EvoMemexBasic qui reprend les principales caractéristiques d'Evocol : un contrôle rigoureux de la diversité de la population et un croisement *multi-parent*. TabuMemexBasic se distingue d'une recherche taboue classique par la taille variable de la liste taboue.

Dans la table 1, nous avons comparé les résultats obtenus pour les métaheuristiques avec une formulation exacte résolue avec GLPK. Les résultats sont encourageants, et laissent supposer que les solutions retournées sont de très bonne qualité même pour les instances de grande taille pour lesquelles la solution optimale n'est pas connue.

L'utilisation de méthodes issues de la coloration de graphes pourrait ainsi être étendue avec succès à des modèles plus complexes de l'allocation de mémoire pour les système embarqués, ce qui permettrait d'apprécier les gains apportés par l'utilisation de ces méthodes à des cas concrets en terme de consommation énergétique.

Références

[1] D. Atienza, S. Mamagkakis, F. Poletti, J. Mendias, F. Catthoor, L. Benini, D. Soudris, Efficient system-level prototyping of power-aware dynamic memory managers for embedded systems. Integration, the VLSI Journal, 39(2), pp. 113-130 (2006).

[2] A. Chimientia, L. Fanucci, R. Locatellic, S. Saponarac, VLSI architecture for a low-power video codec system, Microelectronics Journal, 33(5), pp. 417-427, (2002).

[3] P. Coussy, E. Casseau, P. Bomel, A. Baganne, E. Martin, A formal method for hardware IP design and integration under I/O and timing constraints, ACM Transactions on Embedded Computing Systems, 5(1), pp. 29-53 (2006).

[4] R. Diestel, Graph Theory, Volume 98 of Graduate Texts in Mathematics. Springer-Verlag, Heidelberg (2005).

[5] P. Galinier, J-K. Hao. Hybrid Evolutionary Algorithms for Graph Coloring. Journal of Combinatorial Optimization 3, 379-397 (1999).

[6] GLPK (GNU Linear Programming Kit). URL : http ://www.gnu.org/software/glpk/

[7] M. Iverson, F. Ozguner, L. Potter, Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment, IEEE Transactions on Computers, 48(12), pp. 1374-1379 (1999).

[8] N. Julien, J. Laurent, E. Senn, E. Martin, Power consumption modeling and characterization of the TI C6201, IEEE Micro 23(5), pp. 40-49 (2003).

[9] P. Lacomme, C. Prins, M. Sevaux. Algorithmes de Graphes. Eyrolles, Paris (2003).

[10] W. Lee, M. Chang, A study of dynamic memory management in C++ programs, Computer Languages Systems & Structures, 28(3), pp. 237-272 (2002).

[11] D. Porumbel, DIMACS Graphs : Benchmark Instances and Best Upper Bounds, http ://www.info.univ-angers.fr/pub/porumbel/graphs/.

[12] D. Porumbel, J-K Hao, P Kuntz, Diversity control and multi-parent recombination for evolutionary graph coloring algorithms, Evolutionay Computation in Combinatorial Optimization, 9th Proceedings of the European Conference, EvoCOP 2009 Tübingen, Germany, pp. 121-132, (2009).

[13] K. Sörensen. A framework for robust and flexible optimization using metaheuristics. PhD thesis Universiteit Antwerpen (2003).

[14] T. Vredeveld, J.K Lenstra. On local search for the generalized graph coloring problem. Operations Research Letters vol. 31, pp. 28-34 (2003)

[15] S. Wuytack, F. Catthoor, L. Nachtergaele, H. De Man, Power exploration for data dominated video application, Proceedings of IEEE Symposium on Low Power Design, pp. 359-364 (1996).

[16] Dash Optimization joins FICO. URL : http ://www.dashoptimization.com/

Système d'Aide à la Décision pour la gestion d'un réseau hydrographique en périodes de crues

Majdi Argoubi[1,2], Abdelkader El Kamel[1], Khaled Mellouli[2]

[1] LAGIS ; Ecole Centrale de Lille ; Cité scientifique, BP48, Villeneuve d'Ascq, 59650, France
mejdiargoubi@yahoo.fr ; abdelkader.elkamel@ec-lille.fr
[2] LARODEC ; Institut Supérieur de Gestion de Tunis ; 41 rue de la liberté, Le Bardo, 2000, Tunisie
khaled.mellouli@ihec.rnu.tn

Résumé : *Un Système d'Aide à la Décision (SAD) est développé pour l'optimisation de la gestion d'un réseau hydrographique en périodes de crues. Le SAD repose sur trois composantes : un module d'optimisation, un Système d'Information Géographique (SIG) ainsi qu'une interface à l'usager. La Programmation Dynamique Neuronale (PDN), permettant de surmonter les limites associées à l'utilisation de la Programmation Dynamique Stochastique (PDS) classique, est utilisée pour la génération des lâchers optimaux. Les résultats ainsi obtenus sont validés sur le réseau hydrographique du nord tunisien.*

Mots-Clés : *Réseau hydrographique, PDN, SIG, SAD.*

1 Introduction

Le contrôle des crues repose sur les ressources et les équipements dont dispose le barragiste pour répondre à la forte croissance des apports en eau vers les réservoirs. Diverses installations sont souvent prévues. Souvent, et dans beaucoup de pays, certains réservoirs servent d'une façon continue pendant toute l'année dans le laminage des crues alors que d'autres ne sont utilisées qu'à des moments où les apports en eau atteignent leurs plus hauts sommets. Pour répondre à des mesures de sécurité, une politique ou autrement dit un sorte de "modèle" doit être élaboré. Cette politique permettra d'indiquer, par exemple, quelle quantité d'eau doit-on laisser s'échapper du réservoir. Il s'agit tout simplement de *l'optimisation de la gestion des réservoirs en périodes de crues* [1].

Dans le cadre d'une gestion d'un système multi-réservoirs en périodes de crues, l'objectif principal est de se prémunir au maximum contre :

– Un dépassement des capacités d'écrêtement des réservoirs

– Les débordements au niveau des sections du fleuve

Une stratégie de gestion de crues cherchera alors, à chaque pas de temps, un compromis entre un stock de sécurité à atteindre en un nombre de pas de temps faible et un écrêtement efficace. Dans leur version originelle, les SIG permettent d'acquérir toutes les données relatives au réseau hydrographique, sous diverses formes (données topologiques, apports en eau, cartographies. . .). Bien que novateur dans le contexte de la gestion des réseaux hydrographiques, ces outils ont d'ores et déjà besoin d'être enrichi de fonctionnalités d'optimisation pour devenir des systèmes daide à la décision.

Un SAD est défini comme “un outil qui permet aux décideurs de combiner jugement personnel et soutien informatique, dans une interface machine pour produire des informations utiles au processus décisionnel” [2]. Les travaux abordés dans cette étude concernent la mise en oeuvre d'un SAD donnant plus de souplesse et de cohérence au processus de décision en périodes de crues. Il existe un certain nombre d'actions qui améliorent la flexibilité des SAD face aux problèmes réelles. Ceux-ci peuvent être résumés comme suit :

– Intégration de l'incertitude dans les variables système

– Intégration des paramètres spatiaux

Ce travail s'articule autour de quatre sections. La première section est un aperçu des applications des SAD dans la gestion des réservoirs en périodes de crues. La deuxième section est consacrée à la présentation des caractéristiques de notre réseau hydrographique. Dans la troisième section nous présentons les différents modules du SAD. Enfin une mise en oeuvre portant sur la gestion du réseau hydrographique du Nord de la Tunisie est réalisée.

2 Aperçu sur les SAD de contrôle des crues

Plusieurs SAD de gestion de réservoirs ont été développés dans la littérature, néanmoins seulement une petite fraction s'est intéressée au contrôle des crues [3],[4]. Le modèle BRASS [5] est un SAD dédié au contrôle des réservoirs et à la lutte contre les inondations dans le bassin de la rivière Savannah, en Géorgie. BRASS combine simulation hydrologique et modèle de prévision. Bien qu'il n'intègre pas un modèle d'optimisation, BRASS s'appuie sur l'amélioration en temps réel des prévisions des débits des crues pour corriger les scénarios des lâchers.

Un SAD a été développé pour les opérations de lutte contre les crues dans le bassin de la rivière Trinity, au Texas [6]. Le système est conçu pour intégrer un système de gestion de base de données dans des versions spécialisées de HEC et HEC-1-5, qui sont des modèles de simulation hydrologiques des bassins hydrographiques. Encore une fois, pour générer les stratégies de lâchure, ce système repose sur des prévisions en temps réel des apports en eau, plutôt que sur des techniques d'optimisation.

La modèle CALSIM [7], développé par le Département d'Etat des Ressources en Eau de Californie, en tant que modèle générale basé sur la simulation et sur la Programmation Linéaire Mixte et permettant aux utilisateurs de décrire facilement leurs objectifs et contraintes sans préparer des codes étendues.

Un autre exemple de SAD utilisant un programme d'optimisation est le MODSIM [8], qui est actuellement utilisé par l'US Bureau of Reclamation pour la gestion des réservoirs implémentés sur la rivière Snake. L'avantage de ce système est que ces règles sont explicites et donc aisément communicables aux opérateurs.

Dans [9], un SAD a été élaboré pour un réservoir à fonctions multiples. La Programmation Multi-Objectif (PMO) avec un pas de temps mensuel est utilisée pour la gestion. La PMO permet aux décideurs de tenir compte des différents objectifs dans le développement des politiques optimales. Le SAD est appliqué au système de ressources en eau de surface le plus important en Iran, à savoir le système de réservoirs de la rivière Karoun Dez. Les résultats de l'étude de cas ont montré que le SAD est en mesure d'augmenter sensiblement la production d'électricité à long terme du système tout en satisfaisant les contraintes liées aux débits maximum causant les crues.

3 Réseau hydrographique de Medjerda

La Tunisie fait partie des pays semi-arides de la rive sud méditerranéenne. Par le climat et la nature très fragile des sols, l'agriculture tunisienne, est constamment confrontée à un problème crucial : la conservation des eaux et du sol. D'une part, les pluies peu abondantes et irrégulières incitent au stockage de l'eau pour permettre de mieux la répartir et de prolonger son utilisation, d'autre part, ces précipitations sont fréquemment torrentielles, engendrant des crues brutales.

Le système Medjerda est localisé au Nord de la Tunisie et contrôle plus de 80% des eaux de surface du pays [10]. La superficie combinée de ces bassins versants est d'environ 23.700 km^2. Le réseau hydrographique est équipé de 5 réservoirs en arborescence (figure 1). Parmi eux, on trouve celui de Sidi Salem, le plus grand réservoir du pays. Dans le tableau 1, on résume leurs caractéristiques physiques et hydrologiques.

Réservoirs	Bassin versant (Km^2)	Apports annuels (Mm^3)	Apports min (Mm^3)	Capacité du réservoir (Mm^3)	Lâchers max (m^3/s)
Beni Metir	103	41,187	3,740	57,630	610
Mellègue	10300	117,25	36,200	147,54	5400
Kasseb	101	46,288	7,840	69,620	460
Bou Heurtma	390	120,789	8,388	109,800	2500
Sidi Salem	7950	447,000	94,288	762,000	5260

TABLE 1 – Caractéristiques physiques et hydrologiques du système Medjerda

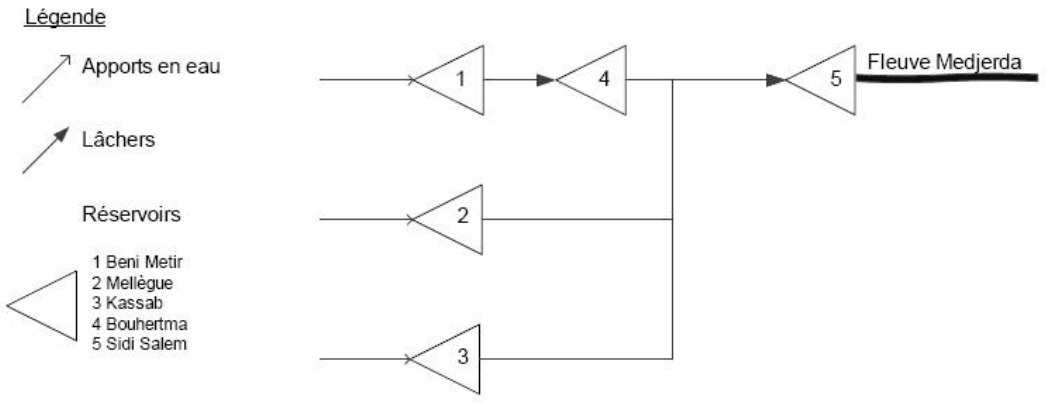


FIGURE 1 – Réseau hydrographique du Nord tunisien.

4 Modules du SAD

4.1 Architecture

Le contrôle des crues ainsi que des objectifs environnementaux sont considérés dans le développement des politiques de gestion via le SAD. Ce dernier se compose de trois modules :

– Module de simulation : Système d'Information Géographique

– Module d'optimisation : Programmation Dynamique Neuronale

– Interface graphique de dialogue

L'approche de modélisation du SAD utilise un couplage d'un module d'optimisation avec un module de simulation, ceci est réalisé à travers un échange dynamique de données pour garantir la rétroaction dans le temps et dans l'espace. Initialement, le module d'optimisation fourni les règles de décision au module de simulation. Le module de simulation identifie le changement dans les caractéristiques spatiales en un pas de temps et les communique au module d'optimisation. A travers l'optimisation, ces changements dans l'espace auront un impact sur les règles de décision dans le pas de temps suivant. De cette manière, le processus peut être résolu dans le temps et dans l'espace d'une façon intégrée tout en assurant la rétroaction.

Le mode de couplage adopté est le mode connecté où le module d'optimisation et le module de simulation utilisent chacun un code de programmation propre à lui et une base de données commune. Une interface permet de reformuler et de restructurer les données afin de rendre possible la communication entre les deux systèmes (figure 2).

Nous avons considéré que cette technique de couplage était la plus efficace car elle évite toute conversion de données lorsqu'il s'agit de faire communiquer les modules entre eux. Aussi, l'indépendance entre les codes de programmation a été souhaitée parce qu'elle permet tout d'abord de ne pas limiter notre module d'optimisation aux possibilités du module de simulation, quitte à proposer ensuite des modifications selon nos besoins. En l'occurrence, dans le cadre du prototype que nous avons développé, nous avons fait le choix de n'apporter dans la programmation que des modifications ou adaptations mineures.

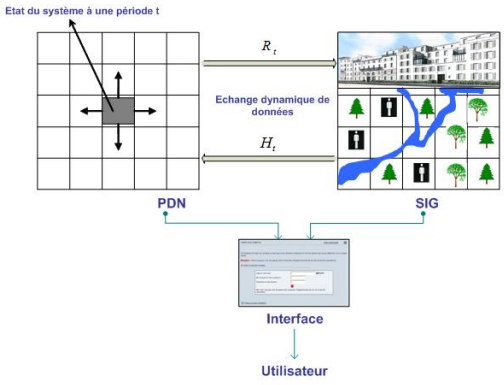


FIGURE 2 – Architecture du SAD.

4.2 Module de simulation : Système d'Information Géographique

Dans le cadre des sciences de l'environnement, les SIG représentent des outils de simulation appropriés. En effet, les SIG ont attiré l'attention des chercheurs en hydrologie et autres sciences de l'environnement, du fait qu'ils permettent de modéliser et de visualiser les processus complexes à évolution spatiale, telles que la modélisation de la propagation d'une onde de crues le long d'une rivière.

Plusieurs applications sont évoquées dans la littérature reliant les SIG à des modèles hydrologiques [11]. Toutefois, malgré le caractère fondamental de l'analyse spatiale des SIG, plusieurs auteurs soulignent quelques limites. Ces derniers concernent en premier lieu le manque de fonctionnalités d'optimisation. Des tentatives d'implémentation des équations des programmes d'optimisation au sein du code source des SIG ont été proposées pour surmonter cette limite. Le simulateur est alors développé comme un module du SIG et modifie directement les données pendant la simulation [12]. Cependant, en périodes de crues, cette intégration s'avère inappropriée à cause de la différence intrinsèque entre le pas de temps requit pour l'optimisation et celui requit pour la simulation par les SIG. La lenteur des SIG fait perdre aux programmes d'optimisation leur force d'interaction et limite ainsi cette approche aux modèles à taille et à variabilité spatiale limitées et aux périodes caractérisées par une relative stabilité climatologique.

Ce constat met en évidence le fait que malgré leur énorme potentiel, les SIG ne contribuent que fort peu à la solution des problèmes spécifiques des décideurs. Ainsi, il serait intéressant dans un premier temps de proposer une solution intermédiaire. Cette solution consiste à coupler les SIG et les méthodes d'optimisation ayant pour finalité l'aide à la décision.

Dans notre cas d'étude, la modélisation hydrologique fut effectuée au Laboratoire des Sciences et Techniques de l'Eau (LSTE) de l'Institut National Agronomique de Tunisie (INAT) et mena au développement d'un SIG. Le logiciel Mike 11 [13] a été utilisé comme support de ce SIG. L'atout majeur de Mike 11 en hydrologie est qu'il propose des fonctionnalités de suivi de la dynamique fluviale [1]. Mike 11 résout les équations de base de l'hydraulique classique : il s'agit des équations de Barré de Saint-Venant. L'équation (1) exprime la conservation de la masse et l'équation (2) exprime la conservation de l'énergie :

$$S\frac{dU}{dx} + U\frac{dS}{dx} + B\frac{dh}{dt} = 0 \tag{1}$$

$$\frac{1}{g}\frac{dU}{dt} + \frac{U}{g}\frac{dU}{dx} + \frac{dU}{dx} - J_f + J_e = 0 \tag{2}$$

où S est la surface mouillée, U la vitesse moyenne de l'écoulement, x la coordonnée curviligne du tronçon, B la largeur de la section, h la hauteur d'eau et t le temps, g l'accélération terrestre, J_f la pente de fond et J_e la pente d'énergie.

Des exemples d'utilisation des SIG se trouvent en Allemagne pour ses deux plus importants cours d'eau que sont le Rhin et le Danube [14]. Aussi, la modélisation de la dynamique fluviale de la Medjerda pendant les inondations de Janvier 2003 en Tunisie a été effectuée selon le même principe [15].

1. L'évolution des hauteurs en eau au niveau des sections d'un fleuve

4.3 Module d'Optimisation : Programmation Dynamique Neuronale

Bien que les opérations de contrôle des crues du système Medjerda soient jusqu'à présent jugé satisfaisantes, il est possible d'apporter des améliorations. Des techniques d'optimisation peuvent être introduites au niveau du processus de décision et déterminer ainsi les meilleurs scénarios de lâchers.

L'étude des méthodes d'optimisation existantes révèle leur multitude, leur variété et leurs compétences particulières. En vue de justifier le recours à une technique au dépend de l'autre, un classement est effectué en fonction des objectifs visés, la précision de modélisation requise ainsi que l'étendue et le comportement général du système (figure 3), [16]. Ceci nous a orienté vers la Programmation Dynamique Stochastique (PDS). De plus, le caractére Markovien du système de niveaux de stock d'eau dans les réservoirs s'impose comme un atout majeur pour le passage par les Processus de Décision Markovien (PDM), [17].

Nous utilisons la notation de [16]. L'horizon de l'optimisation comporte T périodes, notées par $t = \dots T$, chaque période représente un pas de temps. L'évolution du système est décrite à chaque pas de temps en utilisant quatre variables. S_t : le vecteur des stocks d'eau des M réservoirs du réseau au début de la période t, c'est la variable d'état du problème ; H_t : le vecteur des hauteurs d'eau dans les sections du fleuve ; R_t : le vecteur des lâchers des M réservoirs de la période t, c'est la variable de décision ; AP_t : le vecteur des apports en eau vers les réservoirs.

Le comportement du système hydraulique est caractérisé par ses dynamiques au niveau des réservoirs. Pour tout réservoir m du réseau hydrographique (avec $m = 1 \dots M$), la dynamique du système pour une période donnée est gouverné par :

$$s_{t+1}^m = s_t^m + ap_t^m - r_t^m \tag{3}$$

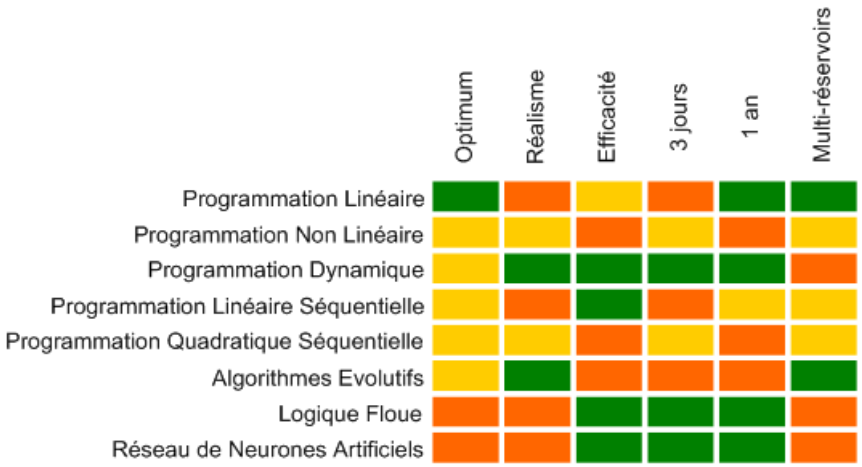


FIGURE 3 – Performance relative des méthodes d'optimisation de la gestion des réservoirs. foncé : approprié ; moyen : possible ; clair : déconseillé, [16]

Des bornes inférieure et supérieure sur les stocks et les lâchers doivent être imposées. Ceci est généralement lié à des seuils imposés pour éviter une situation de vidange totale du réservoir et garantir un certain équilibre écologique dans le fleuve :

$$s_{min}^m \leq s_t^m \leq s_{max}^m \tag{4}$$

$$r_{min}^m \leq r_t^m \leq r_{max}^m \tag{5}$$

Considérons un problème avec T périodes où R_t l'action du système est prise suite à l'observation de son état S_t. Comme conséquence de l'action prise, le système passe de l'état S_t à l'état S_{t+1} qui va être observé à la prochaine période. Souvent en gestion de réservoirs, le nombre des variables d'état est supposé fini. Une propriété fondamentale du système est la propriété markovienne qui stipule que le comportement futur, conditionné par l'action et l'état présents, est indépendant des états et actions passés. En terme de distribution de probabilité, cette condition peut être interprétée par :

$$P(S_{t+1} = j/S_t = i, S_{t-1} = i-1, S_{t-2} = i-2, \ldots; R_t) = P(S_{t+1} = j/S_t = i; R_t) \tag{6}$$

La transition du système est régie par l'équation (3), la probabilité de transition de l'état i à l'état j sera alors celle de la réalisation de l'apport en eau permettant cette transition. Notre modèle est donc un modèle de PDM en horizon fini qui peut être décomposé en T sous-problèmes d'une période qui seront résolus par récursion via le principe d'optimalité de Bellman qui s'énonce comme suit :

Lemme : *"Dans un processus d'optimisation dynamique, une suite de décisions est optimale si, quels que soient l'état et l'instant considérés sur la trajectoire qui lui est associée, les décisions ultérieures constituent une suite optimale de décisions pour le sous-problème dynamique ayant cet état comme conditions initiales "*. [18]

Notre programme d'optimisation doit représenter de manière mathématique les attentes du barragiste. Le choix est délicat puisqu'il est le centre de la gestion et exclut toutes autres caractères qu'il n'explicite pas. Un événement de crues peut difficilement être pris en compte par une fonction économique : on cherche plus à l'interdire qu'à rembourser les dégâts en cas de débordement. Explicitement, il s'agit de se prémunir au maximum contre un dépassement des capacités d'écrêtement des barrages et avoir un minimum de dégâts en aval. Notre démarche consiste alors à imposer une pénalité à chaque déviation du système suite au dépassement des normes de sécurité au niveau des réservoirs et au niveau des sections du fleuve.

Soit $V_t(S_t, H_t)^\star$ la valeur minimale du coût reçu à partir de la période t jusqu'à la fin de l'horizon de l'optimisation. Pour $T, T-1, \ldots, 1$ et on a la récursion (l'équation de Bellman) suivante :

$$V_t(S_t, H_t)^\star = \min_{R_t} \{\alpha L_t(H_t, R_t) + (1-\alpha) Q_{t+1}(S_t, R_t)\} \tag{7}$$

sous les contraintes :

$$s_{t+1}^m = s_t^m + ap_t^m - r_t^m$$

$$s_{min}^m \leq s_t^m \leq s_{max}^m$$

$$r_{min}^m \leq r_t^m \leq r_{max}^m$$

où $m = 1, 2, \ldots, M$ et α une pondération départageant nos deux objectifs.

L_t représente le coût immédiat de débordement au niveau du fleuve subi suite à un lâcher R_t avec :

$$L_t(H_t, R_t) = \frac{\sum_{\varsigma}(h_t^{\varsigma} - h_{alt}^{\varsigma})^2}{\sum_{\varsigma}(h_{déb}^{\varsigma} - h_{alt}^{\varsigma})^2} \tag{8}$$

où $h_{déb}^{\varsigma}$ et h_{alt}^{ς} sont respectivement la hauteur de débordement et la hauteur d'alerte de la section du fleuve ς.

D'autre part, Q_{t+1} décrit l'espérance du coût futur de passage de l'état $S_t = i$ à l'état j sous la décision de lâchure R_t avec :

$$Q_{t+1}(S_t = i, R_t) = \sum_j P_{ij}^{R_t} \left[\frac{\sum_{m=1}^{M}(s_{t+1}^m - s_{sécu}^m)^2}{\sum_{m=1}^{M}(s_{max}^m - s_{sécu}^m)^2} + V_{t+1}^{\star} \right] \tag{9}$$

où $P_{ij}^{R_t}$ la probabilité de passage de l'état i à l'état j (voir équation 6) et $s_{sécu}^m$ est le stock de sécurité consigné au-dessus duquel l'écrêtement de la crue devient de plus en plus difficile.

Pour chaque période t, on discrétise les variables du modèle selon la méthode décrite dans [16]. Partant de l'hypothèse que les coûts optimaux terminaux $V_T(S_T, H_T)^{\star}$ sont connus, pour tout état i du système, la fonction $V_t^{\star}$ est obtenue par récursion à rebours en commençant par $t = T + 1$ et cela en appliquant l'algorithme de la PDS. Une action optimale pour l'état S_t est donnée par le vecteur des lâchers optimaux $R_t^{\star}(S_t)$. La règle de décision optimale $R_t^{\star}(S_t)$ donne l'action qui minimise (7) pour chaque état S_t. Nous devons donc obtenir les T décisions à la fin de notre optimisation.

Néanmoins, cette approche de résolution souffre du problème de dimensionnalité c'est à dire l'explosion du vecteur des états possibles tout au long des périodes. Un autre inconvénient associé à la PDS est l'effort de modélisation. En effet, il n'y a pas de formulation générale qui puisse être appliquée à tout problème à résoudre. Les algorithmes de la PDS doivent être adaptés à chaque type de problème. Aussi, une autre difficulté réside au niveau de la technique de discrétisation des variables et dans la détermination d'un modèle explicite du système des probabilités de transition. Enfin, une difficulté concernant la mise sous forme d'un problème de décision de Markov du problème à résoudre. Des méthodes spécifiques ont été proposées pour résoudre les PDM de grandes tailles et dont la modélisation mathématique nécessite un effort considérable de modélisation. Des travaux récents dans ce contexte utilisent le cadre des PDM Factorisés [19]. Pour la gestion de réservoirs en périodes de crues, la plus grande faiblesse de cette approche est qu'elle utilise (dans la phase d'agrégation) une supposition très arbitraire quant à la distribution de l'eau stockée dans le système des réservoirs agrégés pour la détermination des lâchers des réservoirs. La perte des caractéristiques du problème originel reste le souci majeur lors de la phase de désagrégation.

Ces dernières années, on assiste à l'émergence de nouvelles approches combinant la PDS et les Réseaux de Neurones (RN) [20]. Ces approches demandent moins d'efforts de modélisation mais elles donnent une solution proche de l'optimum. L'idée de combiner la PDS aux RN a été introduite, pour la première fois, par la communauté de l'intelligence artificielle sous le nom de Programmation Dynamique Neuronale [21]. La PDN est proposée comme une approche utilisée pour approximer la fonction coût par une fonction neuronale et résoudre le problème de contrôle stochastique. La méthodologie associée à la PDN exploite donc la propriété d'approximateur universel des RN et permet aux systèmes d'apprendre de leur comportement à travers des simulations et d'améliorer ainsi itérativement leurs performances. L'avantage de la PDN est qu'en gestion de réservoirs, elle ne nécessite pas une discrétisation fine contrairement à la PDS et qu'elle accepte des données incomplètes, incertaines ou bruitées d'ou le gain important en terme de temps de calcul (voir section 5.1).

Pour résoudre le problème (7) de gestion d'un réseau hydrographique en périodes de crues, une approche basée sur la PDN est proposée . L'idée originelle de l'approche est basée sur l'algorithme

d'approximation de la fonction coût par des RN présenté dans [21] (itération approximative sur la fonction coût). Par contre nous proposons d'approximer, par des RN, les fonctions de coût espéré Q_{t+1} au lieu des fonctions V_t. Le coût immédiat L_t, quant à lui, est calculé quant à lui directement à partir des données fournies par le SIG. C'est pour ça qu'on doit composer dans le présent travail avec une difficulté supplémentaire par rapport à l'utilisation classique des RN : il s'agit de combiner le processus d'approximation des fonctions Q_{t+1} avec un algorithme d'optimisation qui permet le calcul de la fonction coût total V_t.

Pour commencer, on suppose que notre problème (7) est à T étapes et procédons par récursion arrière pour sa résolution. Le coût espéré associé à la fin de l'horizon est donné par la fonction Q_{t-1}. Les valeurs terminales sont supposées connues pour tous les états du système. En fait, pour respecter la condition limite en fin d'horizon et qui consiste à avoir des réservoirs pleins, la fonction coût en fin d'horizon est modélisée par :

$$V^{\star}_{T+1}(S_{T+1}, H_{T+1}) = \kappa \sum_{m=1}^{M} \frac{1}{(s^m_{T+1})^2} \tag{10}$$

où κ est une constante qu'on a fixé par calibration à 10^4. Cette condition nous permet d'avoir, en fin d'horizon, des probabilités de transition proches de 1 pour les états de réservoirs pleins.

Ainsi, les valeurs $Q^{\star}_{T+1}$ seront déterminées aux différents vecteurs i de l'ensemble des états par la formule :

$$Q^{\star}_{T+1}(S_{T+1} = i) = \underset{R_T}{min} \sum_j P^{R_T}_{ij} \left[\frac{\sum_{m=1}^{M} (s^m_{T+1} - s^m_{sécu})^2}{\sum_{m=1}^{M} (s^m_{max} - s^m_{sécu})^2} + V^{\star}_{T+1}(S_{T+1} = j) \right] \tag{11}$$

Le vecteur d'apprentissage du réseau de neurones est composé des entrées du réseau qui sont les états du système et le vecteur cible composé des valeurs $Q^{\star}_{T+1}$ données par l'équation (11). L'apprentissage du réseau de neurones nous permet d'obtenir une fonction et cela suite à la résolution du problème

$$\underset{r_{T+1}, b_{T+1}}{min} \sum_i \left(Q^{\star}_{T+1}(S_{T+1} = i) - \widetilde{Q}_{T+1}(S_{T+1} = i, r_{T+1}, b_{T+1}) \right)^2 \tag{12}$$

où r_{T+1}et b_{T+1} regroupent respectivement les poids et les biais du réseau de neurones.

En considérant la fonction neuronale $\widetilde{Q}_{T+1}$, la politique "optimale" $\widetilde{R}_T$ pour la période T est déduite. La fonction coût pourra être obtenue par la suite en intégrant $\widetilde{R}_T$ dans l'expression de $V_T(S_T)$.

La fonction coût est donnée dans ce cas par

$$\widetilde{V}_T(S_T, H_T) = \alpha L_T(H_T, \widetilde{R}_T) + (1 - \alpha) \widetilde{Q}_{T+1}(\widetilde{R}_T, r_{T+1}, b_{T+1}) \tag{13}$$

Ayant l'expression de $\widetilde{V}_T(S_T, H_T)$, les valeurs $Q^{\star}_T$ sont obtenues aux différents vecteurs i de l'ensemble des états et cela en utilisant la formule

$$Q^{\star}_T(S_T = i) = \underset{R_{T-1}}{min} \sum_j P^{R_{T-1}}_{ij} \left[\frac{\sum_{m=1}^{M} (s^m_T - s^m_{sécu})^2}{\sum_{m=1}^{M} (s^m_{max} - s^m_{sécu})^2} + V^{\star}_T(S_T = j) \right] \tag{14}$$

Ce sont ces valeurs qui sont utilisées comme valeurs cibles pour la détermination de la fonction neuronale $\widetilde{Q}_T$ et qui est obtenue suite à la résolution, via les réseaux de neurones, du problème suivant :

$$\underset{r_T, b_T}{min} \sum_i \left(Q_T^\star(S_T = i) - \widetilde{Q}_{T+1}(S_T = i, r_T, b_T) \right)^2 \tag{15}$$

Ayant l'expression $\widetilde{Q}_T$, la politique optimale $\widetilde{R}_{T-1}$ est déterminée tout en ajustant l'indice t pour la période appropriée. L'expression $\widetilde{V}_{T-1}(S_{T-1}, H_{T-1})$ est aussi déduite de la même manière que (13) tout en tenant compte de l'indice de la période. Ainsi, les approximations des fonctions Q et V pour les autres périodes sont déduites récursivement en combinant à chaque période t la résolution de (14) et (15) tout en tenant compte de l'équation (13). Si l'architecture d'approximation est assez riche et si la méthode d'apprentissage est performante dans la résolution du problème des moindres carrées, la fonction $\widetilde{Q}_t$ générée à chaque période sera très proche de la fonction optimale $Q_t^\star$, [21].

Lors de l'implémentation, le réseau représenté sur la figure 4 ci-dessous a été utilisé pour l'approximation de la fonction Q. La couche cachée est composée de 3 neurones à fonctions d'activation sigmoïdes et une couche de sortie comportant un neurone linéaire. Nous avons constaté qu'un réseau de neurones composé de trois neurones donne des résultats très acceptables. Les essais numériques ont été abordés au début avec un réseau ayant un plus grand nombre de neurones. Par contre, il a été constaté qu'en diminuant le nombre de neurones à 3, la différence est négligeable. Il est à rappeler qu'il n'y a pas de recette générale relative à la détermination du nombre optimal de neurones à utiliser. L'ajustement de ce nombre se fait en examinant l'amélioration ou la dégradation des résultats avec l'augmentation ou la diminution du nombre de neurones. L'expression de la fonction neuronale est

$$\widetilde{Q_t}(S_t, r_t, b_t) = \sum_{k=1}^{3} f\left(\sum_m r_{m,k} s_t^m + b_k \right) r_k^{'} + b^{'} \tag{16}$$

L'algorithme d'apprentissage employé est celui de Lavenberg-Marquardt. La méthode employée pour éviter le surapprentissage est celle de la régularisation Bayesienne telle que décrite par [22].

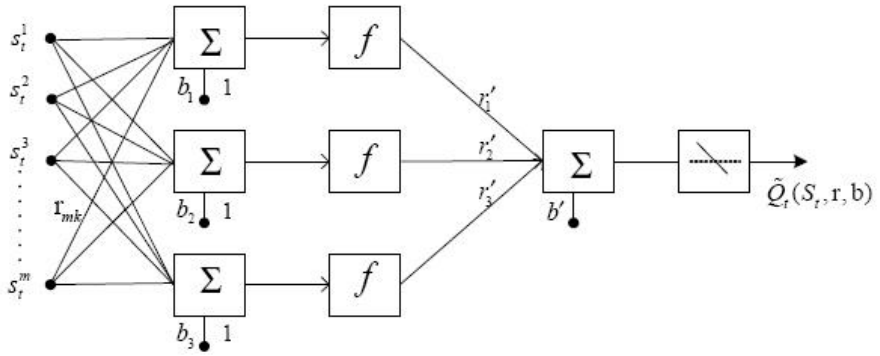


FIGURE 4 – Réseau de neurones employé.

5 Mise en œuvre

Nous présentons un exemple concret d'utilisation du SAD. L'exemple que nous traitons concerne la gestion du réseau hydrographique du Nord tunisien pendant la crue de Janvier 2003. Le gestionnaire parcourt le SAD via son interface graphique de dialogue et rempli un à un l'ensemble des données recueillis sur ses ouvrages. Le SAD repose sur une modélisation conceptuelle qu'on a souhaité la plus générique possible, c'est à dire capable de représenter les différentes situations et les différents résultats. Nous avons opté pour l'outil de développement Delphi [23]. Ce dernier désigne à la fois un langage de programmation et un environnement de développement d'interfaces graphiques.

Une discrétisation grossière des retenues des réservoirs est effectué pour la PDN contrairement à la discrétisation fine utilisée par la modélisation originelle (PDS). La période d'étude s'étale du 25 Janvier 2003 à 20 h jusqu'à 29 Janvier 2003 à 14 h. On a pu ainsi balayer une des trois crues enregistrées pendant l'hiver de 2003. Un horizon de 10 périodes est considéré pour l'optimisation, avec un pas de temps de 10 heures. L'unité de discrétisation correspond à un débit de 200 m^3/s, soit un volume de 7.2 million de m^3 pour un pas de temps de 10 heures.

5.1 Résultats relatifs à la PDN

Le barrage de Sidi Salem est la pièce maîtresse du réseau hydrographique du Nord Tunisien. C'est le seul réservoir ayant des lâchers directs vers le fleuve Medjerda (voir figure 1). Pour cela, on va uniquement se focaliser sur les résultats techniques relatifs à ce réservoir.

Les différentes trajectoires des fonctions Q obtenues avec la PDN sont comparées à celles obtenues avec la PDS. On rappelle qu'une discrétisation très fine a été utilisée dans la modélisation originelle (10455333 points de discrétisation) alors que 31360 points de discrétisation ont été utilisés pour l'approximation des fonctions Q. Nous avons constaté que les fonctions Q obtenues par la PDN sont pratiquement confondues avec celles obtenues via la PDS. Les courbes donnant cette différence (c'est à dire $Q - \widetilde{Q}$) sont tracées sur la figure 5 et ceci pour 3 périodes différentes prisent au hasard durant l'évolution des deux programmes d'optimisation (PDS/PDN). La différence maximale est d'environ une unité (pour $t = 11$). La moyenne de l'erreur relative (en considérant les 31360 états) ne dépasse pas les 1%. L'erreur générée lors de l'approximation de la fonction Q est minime.

Sont représentées dans la figure 6 les trajectoires des lâchers $\widetilde{R}_t$ et R_t pour le barrage de Sidi Salem obtenues respectivement via la PDN et la PDS pour $t = 3$, 6, 9 et 11. Les deux trajectoires se confondent généralement. Par contre les écarts constatés sur les trajectoires des politiques sont plus importants que ceux enregistrés pour Q et $\widetilde{Q}$. Les erreurs relatives moyennes des politiques optimales pour $t = 3$, 6, 9 et 11 sont données au tableau 2. Ces erreurs sont calculées en considérant les valeurs des lâchers obtenues avec la PDN et la PDS. La formule employée pour le calcul de ces erreurs est

$$\frac{1}{N+1} \sum_{i=1}^{N} \sum_{m=1}^{M} \frac{R_t(s_t^{m,i}) - \widetilde{R}_t(s_t^{m,i})}{R_t(s_t^{m,i})} \tag{17}$$

où N représente le nombre d'états et m le nombre de réservoir dans le système.

La plus petite erreur est enregistrée à $t = 3$ et elle vaut 0.3% alors que la plus grande erreur est enregistrée pour $t = 11$ et vaut 3.12%. Il est à constater d'après ce tableau que l'erreur diminue au fil des itérations puisque c'est la récursion à rebours qui est utilisée pour résoudre le problème : la période 3 est exécutée après la période 6.

Période t	Erreur relative (en %) pour $R_t(s_t^m)$
$t = 3$	0.30
$t = 6$	0.37
$t = 9$	0.83
$t = 11$	3.12

TABLE 2 – Erreur relative des politiques optimales

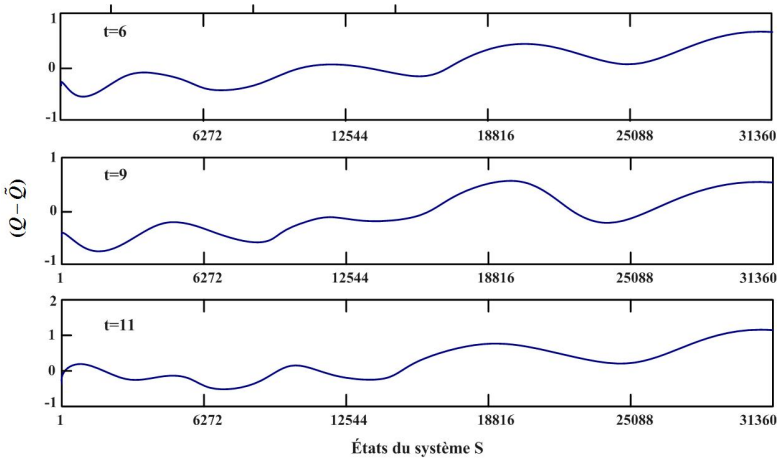


FIGURE 5 – Différences entre Q (obtenue par PDS) et $\widetilde{Q}$(obtenu par PDN).

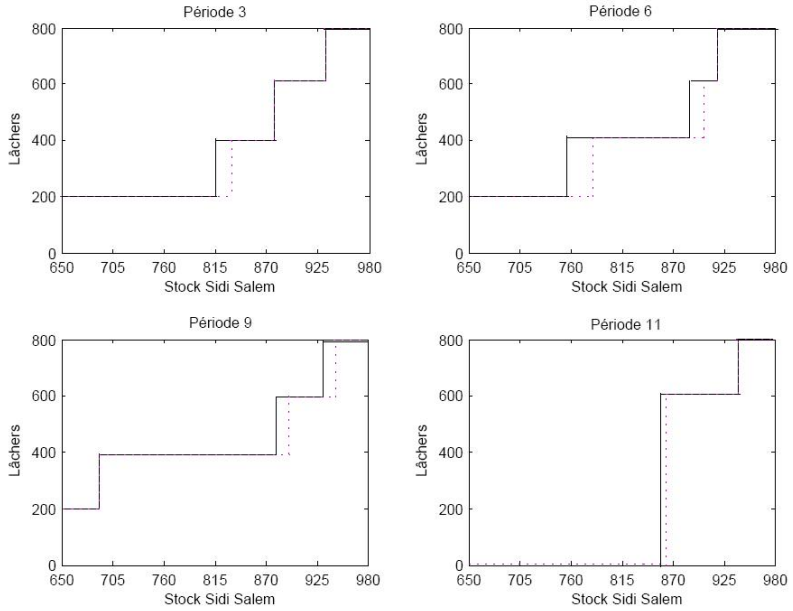


FIGURE 6 – Comparaison des politiques optimales pour différentes périodes (PDN/PDS).

5.2 Résultats relatifs à notre cas d'étude

Lorsque l'implémentation ainsi que la sélection des données au niveau du SAD est achevée, l'opérateur saisit les résultats acquis. Pour cela il introduit les apports en eau observés à chaque pas de temps et choisit le réservoir sur lequel il souhaite récupérér les résultats.

Pour le réservoir de Sidi Salem, au fur et à mesure de l'évolution de la crue, notre approche donne plus de poids au non dépassement des capacités d'écrêtement en effectuant des lâchers importants au début de la période de gestion (figure 7). Le réservoir a tendance à être bien rempli et son niveau d'eau évolue très lentement. On a remarqué que toute baisse du stock du réservoir de Sidi Salem est compensée par une hausse des lâchers calculés qui atteignent les 500 m^3/s le 25/01/03 à 21h jusqu'au 26/01/03 à 16h. On reste toujours loin des 740 m^3/s lâchés réellement par l'opérateur. La méthode de contrôle proposée est donc la plus systématique, dans la mesure où elle effectue des lâchers proportionnels aux apports en eau. Néanmoins, le second pic de lâchers, du 26/01/03 à 18h au 27/01/03 à 02h, présente un léger déficit en termes de débordement avec un débit de 460 m^3/s, ce déficit a un coût faible coïncidant avec la baisse des apports en eau.

Enfin, la performance opérationnelle du réseau hydrographique est bonne. En effet, les décisions prises auraient conduit à une meilleure gestion des aménagements lors de la crue de Janvier 2003, même si d'importantes erreurs dans les inputs sont à attendre. En fait, les périmètres inondés par le pic de lâchers observés sont plus vastes que ceux du modèle (figure 8).

Néanmoins, ilconvient de rappeler que les dégâts réellement observés liés aux débordements de Medjerda n'auraient pas pu être évités car les stocks d'eau des réservoirs au début de la crue étaient très importants.

Conclusion

Nous avons présenté en détail le SAD, ses finalités, son architecture, son contenu informationnel et ses fonctionnalités. Cet outil constitue une avancée significative dans le cadre de la gestion des réservoirs en périodes de crues. La PDN a alors été proposée comme outil d'optimisation. C'est une approche alternative à des techniques souffrant de problèmes de dimensionnalité et de modélisation. D'autre part, un SIG a été introduit pour la prévision des différentes variables spatiales du modèle. Grâce au couplage SIG / PDN, notre SAD permet d'améliorer les actions de lâchure. Une politique efficiente a ainsi été obtenue en termes de pic de lâchers et en termes de terres potentiellement inondables. D'autre part, le SAD contribue à positionner les gestionnaires dans leurs choix, leur fournie une vision plus synthétique de leurs décision, sous forme de figures et de cartographies et leur apporte une vision globale de l'état et de la performance du système.

Néanmoins, de nombreuses actions futures doivent être entreprises pour décrire plus fidèlement le modèle physique. Les plus pertinents nous semblent être les suivants : la modélisation d'un système multi-réservoirs avec des apports en eau reçus de façon continue et l'adaptation de la PDN pour l'optimisation de la gestion d'un système multi-réservoirs à buts multiples tenant compte des conflits entre des objectifs complémentaires voir contradictoires (contrôle de crue, production hydroélectrique...).

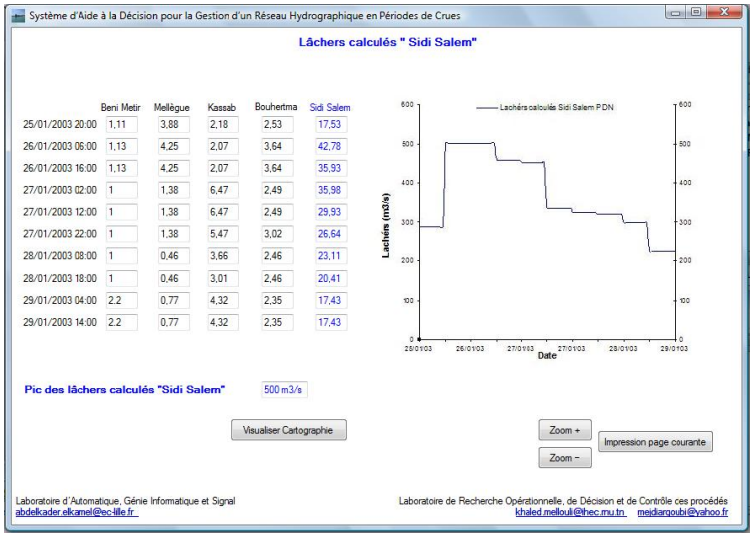


FIGURE 7 – Lâcher calculés du réservoir de Sidi Salem sur la période d'étude.

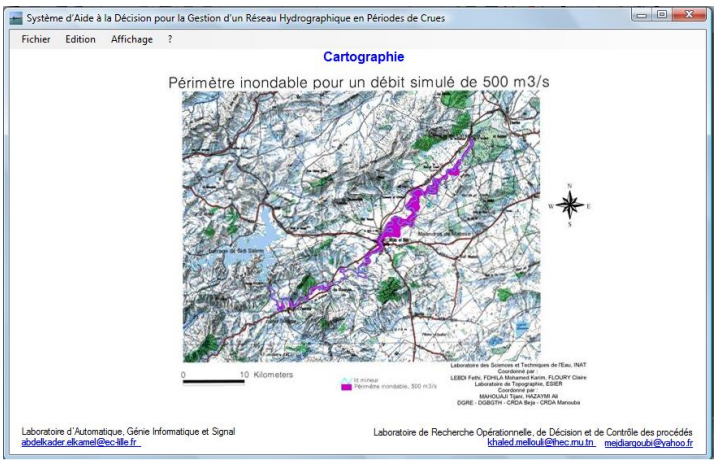


FIGURE 8 – Cartographie de la zone potentiellement inondable par le pic des lâchers (PDN) : 500 m^3/s.

Références

[1] R.Wurbs. Technical Report 283. *Fundamentals of Water Availability Modeling with WRAP*, Texas Water Resources Institute, 4th Edition, March 2008.

[2] S. Simonovic. Decision support systems for sustainable management of water resources : 1. General principles. *Water Resources*, 214 : 223-232, 1996.

[3] L. Mays, Y. Tung. *Hydrosystems engineering and management*. McGraw-Hill, New York. 1992.

[4] J. Labadie. Reservoir system optimization models. *Water Resources Update Journal*, No. 107, 1998.

[5] R. Colon, M. McMahon. BRASS Model : Application to Savannah River system reservoirs. J*ournal of Water Resources Planning and Management*, 177-190, 1987.

[6] J. Killen. PC-based decision support system for Trinity River, Texas. J*ournal of Water Resources Planning and Management*, 375-381, 1995.

[7] D. Munevar, F. Chung. Modeling California's water resource systems with CALSIM. *Proc., 26th Annual Water Resources Planning and Management Conference*, 1999.

[8] J. Labadie, M. Baldo, R. Larson. *MODSIM : Decision support system for river basin management.* Documentation and user manual, Dept. of Civil Engineering, Colorado State Univ, 2000.

[9] M. Karamouz, M. Houck, J. Delleur. Optimization and simulation of multiple reservoir systems. *Journal of Water Resources Planning and Management*, 71-81, 1992.

[10] H. Abida, M. Ellouze. Probability distribution of flood flows in Tunisia. *Hydrology and Earth System Sciences*, 12 : 703-714, 2008.

[11] M. Shamsi. Storm-water management implementation through modeling and GIS. *Journal of Water Resources Planning and Management*, 6(3)114-127, 1996.

[12] P. Burrough. Methods of spatial analysis in GIS. *International Journal of Geographical Information System*, 4(3) : 221-223, 1990.

[13] Mike 11 & 21. Reference Manual, *DHI Software*, 2000.

[14] P. Homagk, K. Ludwig. Operational use of river flood forecasting models in Baden-Württemberg. *Wasserwirtschaft*, 88(4) : 160-167, 1998.

[15] C. Floury, F. Lebdi. La modélisation de la dynamique fluviale de l'oued Medjerda pendant les inondations de Janvier Février 2003 en Tunisie. *PFE INAT*, 2004.

[16] M. Argoubi, A. El Kamel, K. Mellouli and F. Lebdi. Modélisation et optimisation d'un réseau hydrographique en périodes de crues. *Journal Européen des Systèmes Automatisés*, 41(6) : 733-761, 2007.

[17] D. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Bellmont, 1995.

[18] J. Culioli. *Introduction à l'optimisation.* Ellipses, Paris, 1994.

[19] C. Boutilier, R. Dear- den and M. Goldszmidt. Stochastic Dynamic Programming with Factored Representations. *Articial Intelligence*, 121(1) : 49-107, 2000.

[20] J. Teghem,M. Pirlot. *Optimisation approchée en recherché opérationnelle.* Eyrolles, Paris, 2002.

[21] D.P. Bertsekas, J.N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Bellmont, MA, 1996.

[22] H. Demuth, M. Beale. *Neural Netwotk Toolbox for use with Matlab.* The Mathworks : Natick, Massachusetts, 2001.

[23] Delphi. *Prise en main.* Borland® Delphi™ 7 pour Windows™, 2002.

Définition d'un schéma d'optimisation GRASP×ELS pour le 2L-CVRP avec rotations de boîtes

H. Toussaint, C. Duhamel, P. Lacomme, A. Quilliot

LIMOS, Université de Clermont-Ferrand II, Campus des Cézeaux, BP 10125, 63173 Aubière Cedex France.
{toussain, duhamel, placomme, quilliot}@isima.fr

Résumé : *Le 2L-CVRP est une extension du problème de VRP dans lequel la demande de chaque client est constituée d'un ensemble de boîtes définies par un poids et des dimensions (largeur et longueur dans le cas 2D). Comme pour le VRP, il s'agit de concevoir des tournées de coût minimal mais aussi d'organiser le placement des boîtes dans chaque véhicule, en autorisant les rotations de chaque boîte. La contrainte de placement peut être relâchée de sorte que le problème devienne un problème de RCPSP, autorisant le fractionnement des boîtes. Nous proposons la résolution du 2L-CVRP en déterminant d'abord une solution du problème relâxé RCPSP-CVRP puis en transformant cette solution en solution du 2L-CVRP. Le schéma utilisé est de type GRASP×ELS. La possibilité de rotation sur les boîtes nécessite un algorithme de résolution du RCPSP rapide pour gérer le grand nombre de placements possibles. Notre approche est comparée à la méthode de colonies de fourmis de Fuellerer et al [10] sur les instances de la littérature. Les résultats obtenus montrent que le schéma d'optimisation fournit de meilleurs résultats dans des temps de calcul comparables pour les instances de classe 1 à 4.*

Mots-Clés : *VRP, GRASP, ELS, 2L-CVRP.*

1 Introduction

Le CVRP (Constrained Vehicle Routing Problem) est un problème de tournées sur nœuds dans lequel une flotte homogène de véhicules doit traiter les demandes de clients définies au niveau des nœuds. La demande de chaque client représente indifféremment un volume ou un poids et la capacité du véhicule s'exprime dans cette même unité de mesure. Le 2L-CVRP est une extention du CVRP dans laquelle on prend en compte le placement en deux dimensions (2D) de boîtes dans chaque véhicule, ce qui correspond à un problème de livraison. De ce fait, la demande d'un client est définie par un ensemble de boîtes à livrer, chaque boîte ayant son poids et ses propres dimensions rectangulaires. Lorsque toutes les boîtes sont de dimension 1×1, le 2L-CVRP se réduit trivialement au CVRP. En plus des contraintes de placement, des caractéristiques additionnelles peuvent être prises en compte. On dira que le problème est :

- *séquentiel* si les boîtes sont chargées dans le camion en fonction de l'ordre de passage chez les clients (facilite le déchargement) ;
- *orienté* si l'orientation des boîtes est fixe.

Ceci donne lieu à quatre problèmes, notés selon la classification proposée par Fuellerer et *al.* [9] :

- 2|SO|L : problème séquentiel et orienté (2D Sequential Oriented Loading) ;
- 2|UO|L : problème non séquentiel et orienté (2D Unrestricted Oriented Loading) ;

– 2|SR|L : problème séquentiel et non orienté (2D Sequential Rotated Loading) ;
– 2|UR|L : problème non séquentiel et non orienté (2D Unrestricted Rotated Loading).

Certains auteurs, dont Fuellerer *et al.* [10] ont envisagé d'autres contraintes telles que l'extention à trois dimensions, la fragilité des boîtes, la possibilité de placement en porte-à-faux ou encore l'équilibrage du chargement, rendant le modèle d'autant plus réaliste. Nous considérons ici le cas non séquentiel et non orienté (2|UR|L).

Plusieurs métaheuristiques ont été proposées pour le 2L-CVRP, en particulier les algorithmes tabous [11, 20], les algorithmes génétiques, le recuit simulé et les colonies de fourmis [10]. La meilleure méthode publiée à ce jour est l'algorithme de type colonie de fourmis de Fuellerer *et al.* [10].

La principale difficulté du 2L-CVRP réside dans l'intégration d'une composante de tournées et d'une composante de placement 2D (de type orthogonal packing problem). Les travaux récents permettent de traiter de manière très efficace les problèmes de type VRP (Vehicle Routing Problem). Par contre, les problèmes de placement en deux dimensions sont notoirement difficiles. Les meilleures méthodes exactes ne peuvent traiter des instances de plus d'une vingtaine de boîtes en un temps acceptable et les méthodes approchées peinent à produire des solutions de bonne qualité pour les instances de grande taille en un temps de calcul raisonnable [21]. Ce problème de placement s'apparente à un problème d'*orthogonal packing*, problème qui consiste à déterminer si un ensemble de rectangles peuvent être placés sans rotation dans un rectangle plus grand. On peut citer les travaux récents de Aziz et al. [22], de Harren [23] et de Chlebík [24] pour les problèmes d'*orthogonal packing*. On peut noter que, quelque soit l'approche proposée, le problème de placement est souvent résolu de manière heuristique, compte tenu des impératifs de temps de calcul et de la difficulté de résolution. Dans cet article, nous proposons de relâcher la composante placement 2D en un problème de RCPSP (Resource Constrained Project Scheduling Problem) à une ressource (le problème est noté RCPSP-CVRP). Bien que NP-difficile [1], le RCPSP est moins contraint qu'un problème de placement 2D car il ne tient pas compte de la géométrie des boîtes, ou activités dans la terminologie du RCPSP.

Formellement, le 2L-CVRP est défini sur un graphe orienté complet $G = (V, E)$ où V est l'ensemble des $n+1$ nœuds, constitué du dépôt 0 et des clients $1 \dots n$. E est l'ensemble des arcs du graphe avec $c_{ij} \geq 0$ le coût de l'arc (i, j). Une flotte homogène de N véhicules est localisée sur le dépôt. On note D la capacité en poids des véhicules et L et W la longueur et la largeur de leur zone de chargement. La surface utile de transport des véhicules est donc $A = L \times W$. La demande d'un client i est un ensemble de n_i boîtes tel que :
– le poids total des boîtes vaut d_i ;
– la boîte k est de dimension $l_{ik} \times w_{ik}$, $k = 1..n_i$.

Une solution du 2L-CVRP est un ensemble de tournées tel que chaque client est servi par une et une seule tournée. Une tournée est une suite ordonnée $t = (0, t_1, ..., t_{n_t}, 0)$ de nœuds commençant et terminant au dépôt, n_t étant le nombre de clients de la tournée. En fonction des contraintes qu'elle vérifie, une tournée est :
– réalisable en terme de poids si la contrainte de poids est vérifiée, $\sum_{i=1...n_t} d_i \leq D$;
– réalisable en terme de surface si la contrainte de surface est vérifiée, $\sum_{i=1...n_t} \sum_{k=1...n_i} (l_{ik} \times w_{ik}) \leq A$;
– réalisable pour le RCPSP si les boîtes des clients de la tournée satisfont les contraintes de RCPSP (voir section 2.5).
– réalisable pour le 2L-CVRP s'il existe un placement valide des boîtes des clients de la tournée.

2 Un schéma d'optimisation de type *GRASP×ELS*

2.1 Principe

Les contraintes liées au placement des boîtes dans le camion induisent un niveau de complexité supplémentaire dans le problème de tournées de véhicules. En effet, le problème de placement 2D (de type 2D bin-packing) est connu pour être difficile et seules de petites instances peuvent être résolues de manière exacte dans des temps de calcul raisonnable. Les approches heuristiques ou métaheuristiques ne produisent de bons résultats, dans le cas général, que sur des instances de taille moyenne et pour un temps de calcul souvent conséquent [21].

Nous proposons de relâcher les contraintes de cohérence géométrique des boîtes dans le placement. Ainsi, en autorisant les boîtes à être découpées selon leur largeur, le problème se ramène à un RCPSP avec une ressource. La durée représente la longueur alors que la ressource joue le rôle de la largeur. La disponibilité en ressource est constante et égale à la largeur du véhicule. Ainsi, résoudre le RCPSP associé revient à rechercher une solution telle que la date de fin des boîtes est inférieure à la longueur du camion tout en respectant la largeur du camion en chaque instant (figure 1). Une telle solution est dite « réalisable pour le RCPSP » comme vu précédemment. On obtient donc un problème de tournées avec contraintes de type RCPSP que nous notons RCPSP-CVRP.

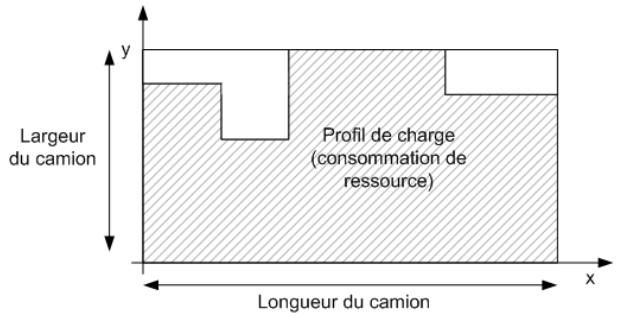


FIGURE 1 – Une solution réalisable pour le RCPSP

Durant le processus d'optimisation (voir figure 2), on impose que toutes les tournées soient réalisables pour le RCPSP, ce qui exige une procédure efficace de résolution du RCPSP ou tout au moins un test de réalisabilité efficace. Pendant l'exécution du GRASP×ELS, les m meilleures solutions du RCPSP-CVRP sont conservées. À la fin du processus, on tente de transformer successivement chacune de ces m solutions en une solution du 2L-CVRP, jusqu'à ce qu'une transformation réussisse.

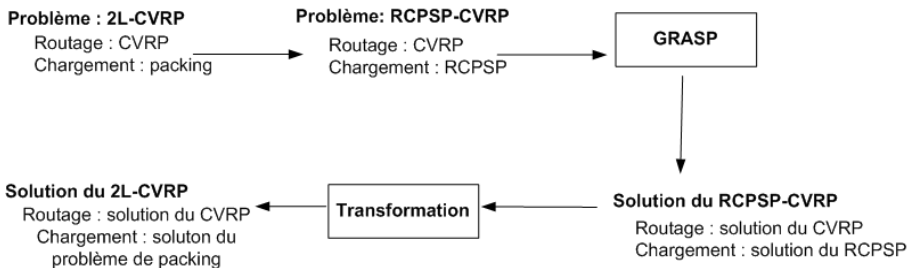


FIGURE 2 – Résolution d'un problème relâché

L'efficacité de l'approche repose *in fine* sur une procédure de transformation efficace d'une solution du RCPSP-CVRP en une solution du 2L-CVRP. Une telle approche a été développée pour le 2L-CVRP non orienté [6] (les rotations de boîtes ne sont pas autorisées). La prise en compte de cette

contrainte supplémentaire oblige à prévoir, dans le schéma du GRASP×ELS initial :
- une fonction de hachage ;
- une procédure de résolution du RCPSP ;
- une procédure de transformation de solution du RCPSP-CVRP en solution du 2L-CVRP.

2.2 Présentation du GRASP×ELS

La méthode *GRASP×ELS* (Greedy Random Adaptative Search Procedure × Evolutionary Local Search) est une hybridation entre un *GRASP* et des techniques de recherche locale itérée. Ce schéma est particulièrement intéressant car il permet d'échapper, dans une certaine mesure, aux bassins d'attraction constitués par les minima locaux (figure 3).

– GRASP [7] repose sur un schéma itératif. À chaque itération une solution initiale est générée par une heuristique randomisée, puis améliorée par une Recherche Locale. La meilleure solution obtenue à la fin de chaque itération est conservée. On note *np* le nombre d'itérations du GRASP.
– ELS [19] est une généralisation des recherches locales itérées (ILS - Iterative Local Search). À chaque itération de ELS, la solution courante sert à générer *nd* solutions voisines par mutation. Chacune de ces solutions est ensuite améliorée par une Recherche Locale. La meilleure solution obtenue devient alors la nouvelle solution courante. On note *nc* le nombre d'itérations de ELS.

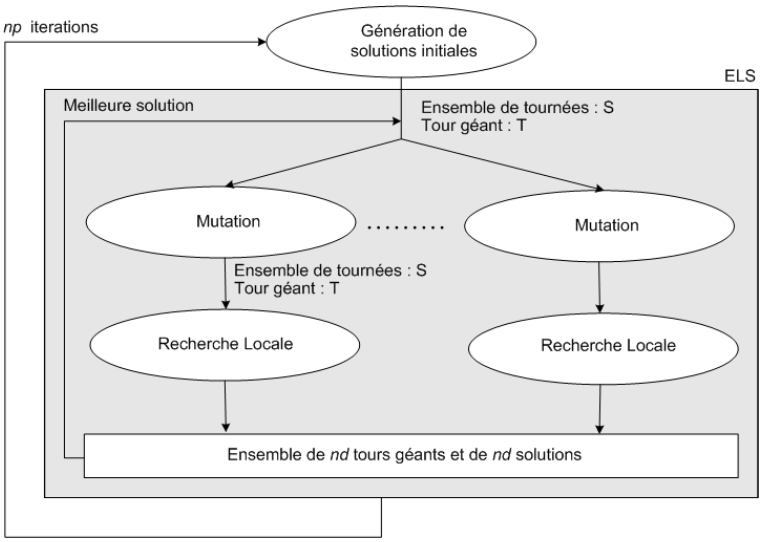


FIGURE 3 – Principe du GRASP-ELS

Ce schéma d'optimisation repose en plus sur la définition de deux espaces de solutions, chacun étant exploré par un processus de recherche distinct (voir figure 4) :
– l'espace des tours géants est constitué de l'ensemble des séquences complètes de clients, c'est-à-dire de l'ensemble des solutions du TSP (Traveling Salesman Problem). Il est exploré par le GRASP×ELS ;
– l'espace des solutions du RCPSP-CVRP contient l'ensemble des solutions dont les tournées sont réalisables pour le RCPSP. La recherche locale s'applique à la solution du RCPSP-CVRP.

Un tour géant est donc défini comme une séquence des clients. Il peut être optimalement découpé en sous-tournées par un algorithme polynômial, Split [2]. Des extensions de Split permettent de prendre en compte les contraintes du problème, comme la capacité des camions ou les fenêtres de temps. Ce type d'approche a fait ses preuves sur le CARP (Capacitated Arc Routing Problem) [17], sur le VRP [18] et récemment sur le Location Routing Problem [5].

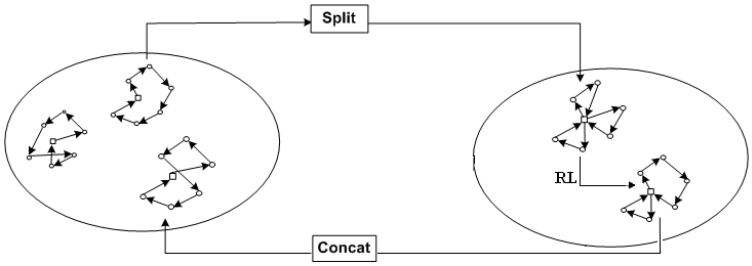


FIGURE 4 – Les deux espaces de recherche

Split permet la transformation d'une solution de type tour géant en une solution de type tournées. Réciproquement une procédure de concaténation permet de transformer une solution de type tournées en une solution de type tour géant. Ainsi, GRASP×ELS travaille au niveau supérieur (génération, mutation) sur les tours géants. Avant chaque appel à la Recherche Locale, le tour géant est transformé en solution du RCPSP-CVRP pour que les modifications de type déplacement de clients inter et intra-tournées puissent opérer efficacement. À la sortie de la Recherche Locale, la solution obtenue est retransformée en un tour géant.

2.3 Les heuristiques de construction

Path-Scanning et *Augment-Merge* de Golden et al. [12] et de Golden et Wong [13] sont des heuristiques déterministes souvent utilisées pour les problèmes de tournées. *Path-Scanning* est une heuristique gloutone partant d'une solution vide et réalisant l'ajout d'un client à chaque itération, uniquement à la fin de la tournée en cours de construction. Le choix du client à insérer repose sur des règles de priorité et une nouvelle tournée est initiée lorsque plus aucun client n'est insérable dans la tournée en cours de construction. *Augment-Merge* est aussi appelée 'méthode de la marguerite'. Elle construit initialement une tournée par client. Les deux étapes suivantes consistent à fusionner des tournées en minimisant l'augmentation de coût.

Une version randomisée de ces deux heuristiques est utilisée pour générer les solutions initiales du GRASP. Les contraintes de RCPSP ne sont pas prises en compte par ces deux procédures. Les tournées obtenues sont donc concaténées puis soumises à la procédure Split pour obtenir une solution du RCPSP-CVRP.

2.4 La Recherche Locale

Notre Recherche Locale est une procédure de recherche itérative reposant sur des mouvements de type 2-Opt et Swap, connus pour être efficaces sur les problèmes de tournées. Quatre types de voisinages spécifiques sont envisagés : mouvements 2-Opt au sein d'une même tournée, mouvements 2-Opt entre deux tournées différentes, permutations de clients au sein d'une même tournée et permutations de clients entre deux tournées différentes. L'utilisation des voisinages permet de réaliser des améliorations successives de la solution initiale et ceci tant que c'est possible et tant que le nombre maximal d'itérations dans la recherche locale n'est pas atteint.

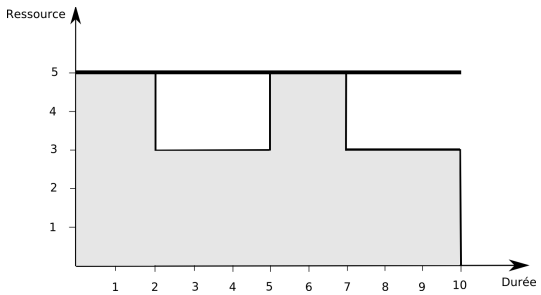
2.5 Réalisabilité du RCPSP

Nous donnons ici une définition simplifiée du RCPSP à une ressource, certaines contraintes n'étant pas nécessaires dans le cadre de notre problème. Pour une définition plus détaillée voir [1].

Le RCPSP à une ressource consiste à planifier un ensemble d'activités $j = \{1 \dots n\}$ de manière à former un projet (planning) de durée la plus courte possible (notée C_{max}). Une activité est caractérisée par sa durée et par sa consommation en ressource. La quantité de ressource est limitée (et vaut R) et la ressource est renouvelable, c'est-à-dire qu'une activité j utilise une certaine quantité de ressource R_j pendant sa durée d'exécution avant de la restituer. Une solution de ce problème est complètement définie par la date de début (notée ES_j) de chaque activité j. On connaît ainsi à chaque instant la quantité de ressource utilisée et on peut tracer le *profil de charge* qui indique la consommation en ressource au fil du temps. La figure 5(a) présente un problème à 6 activités $\{a, b, c, d, e, f\}$, une ressource et 5 unités de ressource disponibles. La date de début de chaque activité est donnée dans la colonne de droite. La figure 5(b) montre le profil de charge associé.

activité	durée	conso.	début
a	4	2	2
b	2	5	0
c	3	1	2
d	1	2	6
e	2	3	8
f	3	3	5

(a)



(b)

FIGURE 5 – (a) Solution (dates de début) d'un problème à 6 activités - (b) Profil de charge associé

Une manière de représenter graphiquement le planning d'une solution du RCPSP est d'associer, pour chaque activité j, un ensemble de rectangles r_{jk}, $k = 1 \dots R_j$ tous de longueur égale à la durée de l'activité j et de hauteur unitaire. Ces rectangles sont représentés dans un plan (x, y), le côté gauche du rectangle r_j devant être positionné à l'abscisse ES_j et l'ordonnée étant libre. La position d'un rectangle r_{jk} est donc définie par son abscisse $x_{jk} = ES_j$ et par son ordonnée y_{jk}. Notons qu'une solution du RCPSP donne les ES_j, les x_{jk} sont donc imposés, tandis que les y_{jk} ne sont pas connus : il faut déterminer les y_{jk} pour obtenir une représentation graphique. La figure 6 montre deux représentations de la solution décrite figure 5. Pour une meilleure lisibilité des figures nous fusionnons, quand c'est possible, les rectangles correspondants à une même activité. S'il existe une représentation graphique, telle que, pour toutes les activités j on a $y_{j(k+1)} = y_{jk} + 1$, $k = 1 \dots R_j - 1$, alors la représentation graphique de la solution du RCPSP est aussi une solution du problème suivant : « Trouver un placement des rectangles r_j dans le plan $[0, C_{max}] \times [0, R]$ tel que les rectangles ne se superposent pas ». Ce problème est un problème de placement 2D.

Notons qu'il n'est pas toujours possible de trouver une représentation d'une solution (respectant les dates de début pour chaque activité), pour laquelle les boîtes sont pas divisées. La figure 7 donne un exemple de solution du RCPSP pour lequel il n'est pas possible de trouver une représentation dans laquelle les rectangles ne sont pas divisés. En d'autres termes il n'existe pas de solution au problème de placement en utilisant les positions en abscisses fournies par le tableau de la figure 7.

Pour résoudre efficacement le 2L-CVRP, nous le remplaçons par le RCPSP-CVRP dans le processus d'optimisation, en considérant que chaque boîte à placer dans le camion correspond à une activité dont la durée est égale à la longueur de la boîte et la consommation en ressource est la largeur de la boîte. Une solution du RCPSP pour laquelle la date de fin de la dernière activité n'excède pas

la longueur du camion est dite réalisable pour le RCPSP. Par rapport à une solution de placement 2D, nous autorisons donc éventuellement le découpage des activités comme illustré dans la Figure 6(a). À l'issue de l'optimisation, nous appliquons une procédure de transformation sur les meilleures solutions trouvées, en considérant qu'une solution réalisable pour le RCPSP est susceptible de mener à un placement de boîtes satisfaisant. Effectivement, on constate en pratique qu'une solution du RCPSP-CVRP est très souvent transformable en solution du 2L-CVRP.

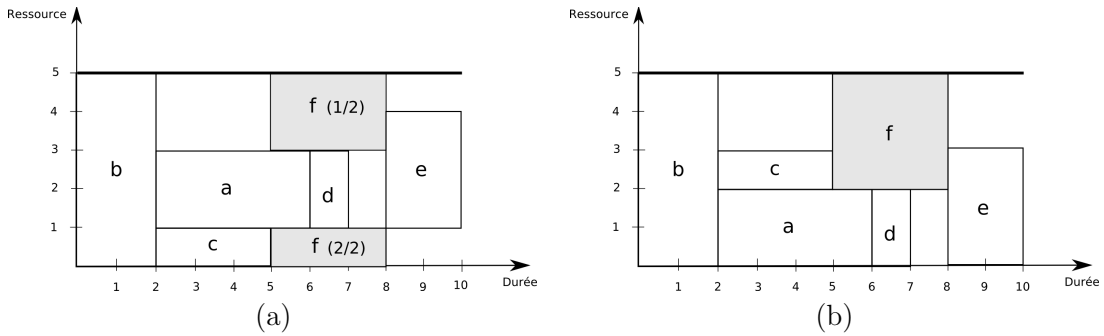


FIGURE 6 – (a) Représentation d'une solution du RCPSP (non valide pour le problème de placement) - (b) Autre représentation de la même solution (valide pour le problème de placement)

La réalisabilité du RCPSP associé à une tournée est vérifiée dans la procédure `Tester_RCPSP` (algorithme 1). Le critère le plus important pour cette procédure est la limite sur le temps de calcul. En effet, toutes les tournées générées dans la procédure Split doivent être vérifiées par `Tester_RCPSP`, ce qui induit un très grand nombre d'appels.

Nous avons choisi une heuristique rapide et efficace (voir l'algorithme 2) pour résoudre le RCPSP. Elle est inspirée des règles de priorité proposées par Kolisch *et al* en 1996 (voir [14]). Cette heuristique fonctionne de manière gloutonne. Elle s'appuie sur l'utilisation de deux ensembles A et D qui ont la signification suivante, t étant une date :

– A contient les jobs actifs à t, c.-à-d les jobs commencés en t mais pas encore terminés ;

– D contient les jobs qu'on peut ordonnancer à t, c.-à-d les jobs non encore traités tels que leur hauteur ne dépasse pas la quantité de ressource non utilisée à t.

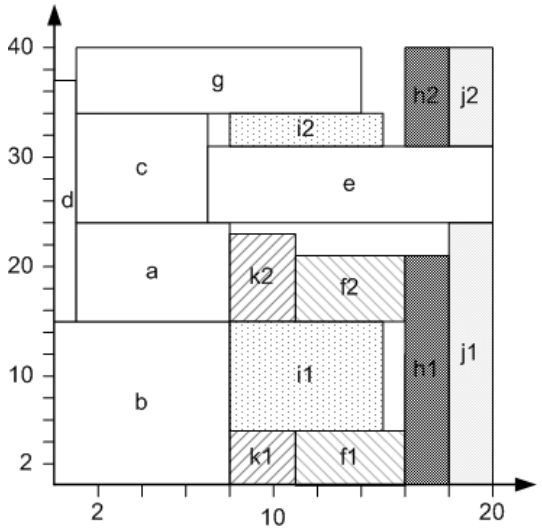
Étant donné que les boîtes (représentées par les activités dans le RCPSP) peuvent subir une rotation de $\pi/4$ pour être placées dans le camion, on ajoute dans D l'ensemble des activités *inverses* E' correspondant aux boîtes du problème initial dont on a permuté la longueur et la largeur. Bien entendu, les boîtes dont la longueur est plus grande que la hauteur du camion ne peuvent subir cette rotation. En outre une seule des deux activités, entre l'activité initiale et l'activité inverse, doit être insérée dans A.

L'heuristique procède de la manière suivante : initialement la date courante est fixée à $t = 0$. Une première activité est choisie aléatoirement dans D et insérée à la date courante. Cette activité passe dans l'ensemble A. L'activité *inverse* est supprimée de la liste des activités restant à traiter. L'ensemble D est mis à jour de telle sorte que toutes les activités qui ne peuvent plus être insérées à la date courante soient supprimées de D. Si $D = \emptyset$ alors on incrémente la date courante t jusqu'à la prochaine date à laquelle une ou plusieurs activités peuvent être insérées. Sinon on choisit une activité de D pour l'insérer à la date courante. On continue ainsi de suite jusqu'à ce que toutes les activités aient été traitées ou que la date courante dépasse la date limite t_l (correspondant à la longueur du camion). Le schéma est repris dans l'Algorithme 2. Comme l'algorithme repose sur un choix aléatoire, on relance la vérification tant qu'on ne réussit pas ou que le nombre de tests n'est pas atteint.

Les activités sont repérées par leur date de début. Leur date de fin est obtenue par l'ajout de la

activité	durée	conso.	début
a	6	9	2
b	8	15	0
c	5	10	2
d	2	22	0
e	13	7	7
f	5	11	11
g	12	6	2
h	2	29	16
i	7	13	8
j	2	31	18
k	3	13	8

(a)



(b)

FIGURE 7 – (a) Solution d'un problème de RCPSP à 11 activités et une ressource disponible en quantité 40 - (b) Une représentation graphique de cette solution (il n'existe pas de représentation de cette solution valide pour le problème de placement)

durée de l'activité. La date de début « au plus tôt » est la plus petite date à laquelle l'activité peut commencer. La date au plus tard est la plus grande date à laquelle l'activité peut commencer sans décaler les activités suivantes. Les algorithmes du RCPSP utilisent les notations suivantes :

- ES_j : date de début au plus tôt de l'activité j ;
- EF_j : date de fin au plus tôt de l'activité j ;
- LS_j : date de début au plus tard de l'activité j ;
- LF_j : date de fin au plus tard de l'activité j.

Algorithme 1 : Tester_RCPSP

Données : E : ensemble des activités $j = 1 \dots n$
Résultat : *reussite*

$k \leftarrow 0$;
Repeter
  $reussite =$ **heuristique_RCPSP**(E) ;
  $k \leftarrow k + 1$;
tant que *(k < max_iter) ET (reussite = faux)* ;

Par exemple, sur la figure 6, on a $ES_d = 6$, $EF_d = 7$, $LS_d = 7$ et $LF_d = 8$.

L'heuristique que nous proposons pour le RCPSP est inspirée des heuristiques proposées par Kolisch dans [15]. Les heuristiques proposées par Kolisch utilisent une règle de priorité pour choisir la prochaine activité j dans D. Nous avons remplacé la règle de priorité par un choix aléatoire. En effet, étant donné les caractéristiques de notre problème (une seule ressource et pas de contrainte de précédence) les règles de priorité donnent des résultats peu satisfaisants car elles sont toutes basées sur les contraintes de précédence et exploitent le fait qu'il y ait plusieurs ressources.

Algorithme 2 : heuristique_RCPSP

Données : E : ensemble des activités $j = 1 \dots n$, t_l : date limite
Résultat : *reussite*

Les activités j *orientables* sont dupliquées puis subissent une rotation de $\pi/4$, ces nouvelles activités forment l'ensemble E' ;
$t = 0$;
$A = \emptyset$;
$D = E \cup E'$;
$k = 1$;
$reussite$ = vrai;
Tant que $k \leq n$ *&* $reussite = vrai$ **faire**
  Choisir aléatoirement j dans D ;
  $ES_j = t$;
  Mettre à jour les ensembles A et D ;
  Tant que $D = \emptyset$ **faire**
    $t \leftarrow \min_{j \in A} EF_j$;
    Mettre à jour les ensembles A et D ;
  $k = k + 1$;
  si *il existe* j *telle que* $EF_j > t_l$ **alors**
    $reussite$ = faux

2.6 Transformation d'une solution du RCPSP-CVRP en solution du 2L-CVRP

On recherche une solution au problème de placement 2D, c.-à-d. les positions (x, y) des boîtes à partir des dates au plus tôt calculées par le RCPSP. Le RCPSP donne non seulement l'abscisse de chaque boîte mais aussi son orientation selon que la tâche appartient à E ou E'. Le problème est donc simplifié car on élimine 2 degrés de liberté, ce qui réduit considérablement le nombre de solutions envisageables. Le problème résultant est résolu grâce à une heuristique qui reprend la même philosophie que `heuristique_RCPSP` : dans `heuristique_RCPSP`, les boîtes sont insérées le plus à gauche possible. Ici les boîtes sont insérées le plus en bas possible. La figure 8 illustre le fonctionnement de l'algorithme sur le problème de la figure 6 : on choisit b de manière aléatoire puis on la place en $y = 0$, puis on place a, d et e car leur abscisse leur permet d'être toutes simultanément placées en $y = 0$. Pour finir, on devra augmenter y jusqu'à $y = 2$ pour pouvoir placer les boîtes restantes.

On utilise une fois de plus les ensembles A et D qui prennent ici la signification suivante, étant donné l'ordonnée y :

- A contient les boîtes j actives en y, c.-à-d les boîtes telles que $y_j < y < y_j + hauteur_j$
- D contient les boîtes j qu'on peut ordonnancer en y, c.-à-d les boîtes non encore traitées telles que leur placement en (x_j, y) n'induit pas une superposition avec une boîte déjà placée.

Ainsi les boîtes sont insérées séquentiellement. La première boîte j est placée en $(x = x_j, y = 0)$. Cette boîte passe dans l'ensemble A. L'ensemble D est mis à jour de telle sorte que toutes les boîtes qui ne peuvent plus être insérées en $y = 0$ sont supprimées de D. Si $D = \emptyset$ alors on incrémente y jusqu'à ce que y soit suffisament grand pour que $D \neq \emptyset$. On continue ainsi de suite jusqu'à ce que toutes les boîtes aient été placées ou que y dépasse la hauteur du camion. Si y dépasse la hauteur du camion alors on peut relancer la procédure heuristique `RCPSP_to_packing` avec les mêmes dates de début ou des dates de début différentes.

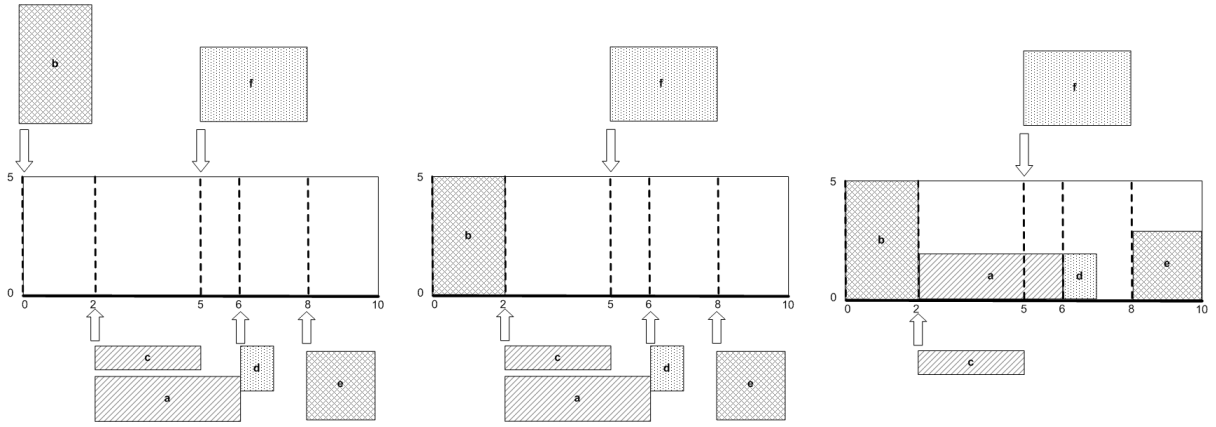


FIGURE 8 – Illustration du placement des boîtes à différentes étapes de l'exécution de l'algorithme 3

Algorithme 3 : RCPSP_to_packing

Données : B, ensemble des boîtes orientées$j = 1 \dots n$ avec leur date de début x_j
Résultat : *reussite*

$yCour = 0$;
$A = \emptyset$;
$D = B$;
Pour k = *1 à* n **faire**
  Choisir aléatoirement $j \in D$;
  $y_j = yCour$;
  Mettre à jour A et D ;
  Tant que $D = \emptyset$ **faire**
    $y \leftarrow \min_{j \in A} \{y_j\}$;
    Mettre à jour A et D ;
si *toutes les boîtes ont été placées* **alors**
  reussite = vrai
sinon
  reussite = faux

3 Expérimentations numériques

3.1 Jeux de données

Les trois dernières publications [11] [20] [10] sur le 2L-CVRP utilisent les mêmes jeux de données. Ces jeux de données se composent de 180 instances réparties dans cinq classes, chaque classe étant composée de 36 instances. Le nombre et la dimension des boîtes d'une instance varient d'une classe à l'autre. La classe 1 correspond à des instances de type CVRP (boîtes de taille 1×1, c'est-à-dire qu'il n'y a pas de problème de placement à résoudre). Dans la classe i, $i = 2 \dots 5$, le nombre de boîtes affectées à chaque client a été choisi suivant une loi uniforme entre 1 et i (voir [11]). Les instances peuvent être téléchargées à l'adresse suivante : http ://www.or.deis.unibo.it/research.html. Les meilleures solutions trouvées par Fuellerer *et al.* [10] sont disponibles à l'adresse : http ://prolog.univie.ac.at/research/VRPandBPP/.

Le temps de calcul accordé à la méthode de colonie de fourmis dans [10] est limité à une heure.

La machine utilisée est plus performante que celle que nous utilisons (tableau 1), un Pentium IV à 2.1Ghz sous Linux. Pour harmoniser les temps de calcul, nous avons donc multiplié par 1,5 la limite de temps pour notre méthode GRASP×ELS. Nous disposons donc de une heure trente, compte tenu de la différence de puissance de processeur.

	colonie de fourmis [10]	GRASP×ELS
processeur	PIV 3.2 GHz	Opteron 2.1 GHz
système	linux	linux
langage	C++	C++
facteur de vitesse	1	0.66
limite de temps	1h	1h30

TABLE 1 – Caractéristiques des machines utilisées

Toujours dans l'objectif de fournir une comparaison claire des méthodes, nous utilisons les mêmes réglages que [10] : 10 exécutions sont réalisées par le GRASP×ELS et la meilleure solution trouvée est reportée dans les tableaux comparatifs. Les résultats sur chaque instance sont disponibles à l'adresse suivante : http ://www.isima.fr/˜lacomme/2lcvrp/2lcvrp_rotation.html.

3.2 Instances de la classe 2 à 5

Les résultats du tableau 2 font référence au rapport technique de Fuellerer *et al.* (voir [9]). Les temps de calcul de la colonne CPU sont des temps normalisés : les temps de calcul de Fuellerer ont été multipliés par 1,5 et ils sont exprimés en secondes. La colonne nb_m représente le nombre de fois, sur 36, où la méthode considérée est la meilleure. La colonne coût est le coût moyen des 36 instances sur la classe considérée. Pour la ligne « classe 2-5 » les valeurs des colonnes coût et CPU sont les moyennes sur les classes 2 à 5, et la colonne nb_m représente le nombre de fois où la méthode considérée a été meilleure sur la moyenne des classes 2 à 5 d'une instance.

	colonie de fourmis [10]			GRASP×ELS		
	nb_m	coût	CPU	nb_m	coût	CPU
classe 2	7	1113.44	1473.07	29	1109.94	2266.84
classe 3	2	1141.25	1466.33	34	1132.41	1835.27
classe 4	7	1162.75	1474.03	29	1157.72	2000.13
classe 5	24	1040.08	1523.51	12	1052.72	1867.55
classe 2-5	12	1114.38	1484.23	24	1113.19	1992.45

TABLE 2 – Résultats obtenus sur les classes 2 à 5

Les résultats montrent que pour les classes 2, 3 et 4, la méthode GRASP×ELS fournit de meilleurs résultats que la méthode de Fuellerer. GRASP×ELS fournit une moyenne de 1109 contre 1113 sur la classe 2, 1132 contre 1141 sur la classe 3 et 1157 contre 1162 sur la classe 4. Les performances de GRASP×ELS sont bien moins bonnes pour la classe 5 puisque la méthode de colonie de fourmis domine régulièrement GRASP×ELS. Enfin, les classes 2, 3, 4 et 5 représentent un total de 144 instances. GRASP×ELS a obtenu de meilleurs résultats 104 fois sur 144 par rapport à la méthode de Fuellerer *et al.* Notons que la classe 1 n'est pas testée car elle correspond au cas sans orientation, pour lequel nous obtenons déjà de bons résultats [6].

3.3 Exemple de résultat

Considérons la solution obtenue sur l'instance 1 de la classe 3 dans le cas où les rotations de boîtes ne sont pas autorisées. La solution obtenue est présentée sur la figure 9. Le coût de cette solution est de 284,52.

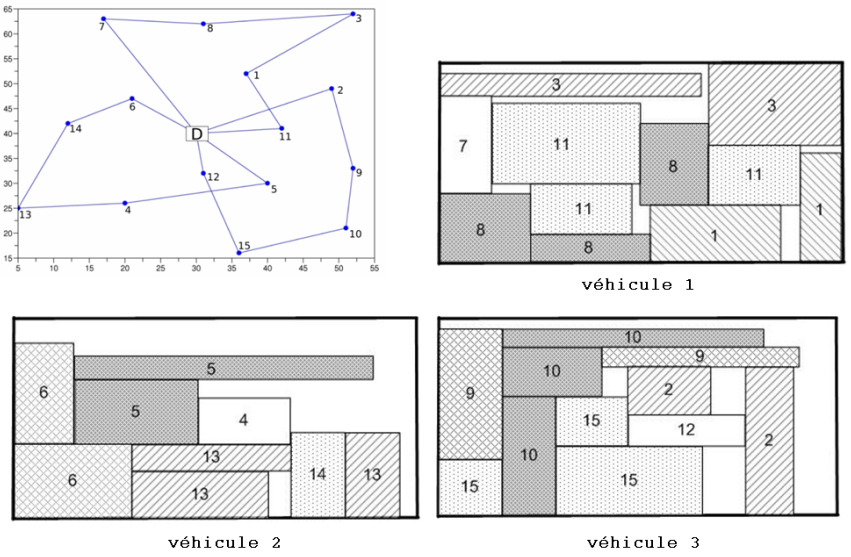


FIGURE 9 – Solution de l'instance 1 classe 3 (sans rotation)

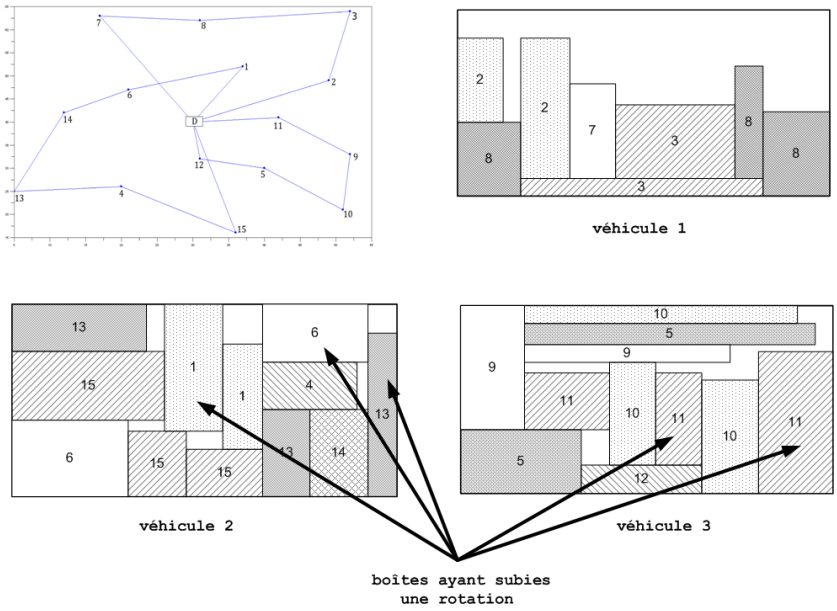


FIGURE 10 – Solution de l'instance 1 classe 3 (avec rotations)

En autorisant les rotations, GRASP×ELS obtient une solution de valeur 284,23. Plusieurs boîtes ont subi des rotations. En particulier les boîtes 11 et 6 ont subi une rotation comme le montre la figure 10. Comme on peut le constater, même si les deux solutions sont proches en terme de valeur (284,52 et 284,23) elles sont radicalement différentes en terme de tournées et de chargement des véhicules.

4 Conclusion

Cet article aborde la résolution du 2L-CVRP avec rotation de boîtes. Le 2L-CVRP est une extention du CVRP dans laquelle on intègre des considérations géométriques sur la demande des clients : la demande d'un client consiste en un ensemble de boîtes rectangulaires à placer dans le véhicule, défini par sa surface de chargement. Le 2L-CVRP est un problème de tournées avec des sous-problèmes de type placement 2D. L'originalité de l'approche consiste à relâcher le problème initial en RCPSP-CVRP et à transformer la solution obtenue à la fin du processus d'optimisation en une solution du 2L-CVRP. Un schéma d'optimisation de type GRASP×ELS est proposé pour le RCPSP-CVRP. La principale difficulté de résolution réside dans la rotation possible des boîtes, ce qui complique singulièrement la résolution du RCPSP. Une procédure particulière a été définie pour tester la faisabilité du RCPSP dans des temps de calcul très courts. Cette contrainte de temps de calcul est très forte dans notre schéma d'optimisation car la faisabilité des tournées est systématiquement testée. Les résultats obtenus sont meilleurs que ceux proposés par Fuellerer *et al.* [10] sur les classes 2 à 4. De manière générale, sur les 144 instances du jeu d'essai, la méthode GRASP×ELS fournit de meilleurs résultats que leur méthode à colonies de fourmis (104 fois contre 40). Ainsi, GRASP×ELS montre sa capacité à prendre en compte la contrainte de rotation de boîtes et devrait maintenant être étendue aux 2L-CVRP séquentiels.

Références

[1] C. Artigues, P. Michelon and S. Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research, 148 :249-267, 2003.

[2] J.E. Beasley. Route-first cluster-second methods for vehicle routing. Omega, 11 :403-408, 1983.

[3] G. Clarke and J.W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. Opererationnal Research, 12 :568-581, 1964.

[4] J.J. Dongarra. Performance of Various Computers Using Linear Equation Software. University of Manchester, 2009.

[5] C. Duhamel, P. Lacomme, A. Quilliot and H. Toussaint. 2L-CVRP : a GRASP resolution scheme based on RCPSP. CIE39 Conference - International Conference on Computers & Industrial Engineering, Troyes, France, 1106-1111, 2009.

[6] C. Duhamel, P. Lacomme, A. Quilliot and H. Toussaint. A GRASP×ELS approach for the 2L-CVRP. Soumis à Computers and Operations Research.

[7] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. Journal of Global Optimization, 6 :109-33, 1995.

[8] M. Fisher. A polynomial algorithm for the degree-constrained minimum k-tree problem. Operations Research, 42(4) :775-779, 1994.

[9] G. Fuellerer, K.F. Doener, R.F. Hartl and M. Iori. Ant colony optimization for the two-dimensional loading vehicle routing problem. Technical report, University of Vienna, Faculty of Business, Economics and Statistics, Department of Business Administration, 2006. http ://prolog.univie.ac.at/research/VRPandBPP/2l_CVRP/TR_2LCVRP.pdf

[10] G. Fuellerer, K.F. Doener, R.F. Hartl and M. Iori. Ant colony optimization for the two-dimensional loading vehicle routing problem. Computers and Operations Research, 36(3) :655-673, 2009.

[11] M. Gendreau, M. Iori, G. Laporte and S. Martello. A Tabu Search Heuristic for the Vehicle Routing Problem with Two-Dimensional Loading Constraints. Networks, 51(1) :4-18. 2008.

[12] B.L. Golden, J. DeArmon and E.K. Baker. Computational experiments with algorithms for a class of routing problems. Computers and Operations Research, 10 :47-59, 1983.

[13] B.L. Golden and R.T. Wong. Capacitated arc routing problems. Networks, 11 :305-15, 1981.

[14] R. Kolisch. Serial and parallel resource-constrained project scheduling methods revisited : theory and computation. European Journal of Operational Research, 90 :320-333, 1996.

[15] R. Kolisch. Efficient priority rules for the resource-constrained project scheduling problem. Journal of Operations Management, 14 :179-192, 1996.

[16] P. Lacomme, C. Prins and W. Ramdane-Chérif. Evolutionary algorithms for periodic arc routing problems. European Journal of Operational Research, 165 :535-553, 2005.

[17] P. Lacomme, C. Prins and W. Ramdane-Chérif. Competitive memetic algorithms for arc routing problems. Annals of Operations Research, 131 :159-85, 2004.

[18] C. Prins. A GRASP x evolutionary local search hybrid for the vehicle routing problem. In : Pereira FB, Tavares J, editors. Bio-inspired algorithms for the vehicle routing problem, Studies in Computational Intelligence 161. Berlin : Springer, 35-53, 2009.

[19] S. Wolf and P. Merz. Evolutionary local search for the super-peer selection problem and the p-hub median problem. Lecture Notes in Computer Science 4771. Berlin : Springer, 1-15, 2007.

[20] E.E. Zachariadis, C.D. Tarantilis and C. Kiranoudis. A guided Tabu Search for the Vehicle Routing Problem with two dimensional loading constraints. European Journal of Operational Research, 3(16) :729-743, 2009.

[21] A. Lodi. Algorithms for two-dimensional bin packing and assignment problems. Dottorato di Ricerca in Ingegneria dei Sistemi, Università di Bologna, 1999.

[22] F. Clautiaux , J. Carlier and A. Moukrim. A new exact method for the two-dimensional orthogonal packing problem. European Journal of Operational Research, 183 :1196-1211, 2007.

[23] R. Harren. Approximation algorithms for orthogonal packing problems for hypercubes. Theoretical Computer Science, 410(44) :4504-4532, 2009.

[24] M. Chlebík and J. Chlebíková. Hardness of approximation for orthogonal rectangle packing and covering problems. Journal of Discrete Algorithms, 7(3) :291-305, 2009.

Analyse inverse robuste à partir d'informations préférentielles partielles

Thomas Veneziano[1,2], Patrick Meyer[2,3], Raymond Bisdorff[1]

[1] University of Luxembourg
ILIAS - CSC
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
{raymond.bisdorff,thomas.veneziano}@uni.lu
[2] Institut Télécom ; Télécom Bretagne
UMR CNRS 3192 Lab-STICC
Technopôle Brest Iroise CS 83818 F-29238 Brest Cedex 3, France
patrick.meyer@telecom-bretagne.eu
[3] Université européenne de Bretagne

Résumé : *Dans cet article, nous proposons une méthode indirecte d'élicitation robuste de l'importance des critères à partir d'informations préférentielles partielles exprimées par un décideur, dans un contexte de relations valuées bipolaires de surclassement. Nous montrons que la modélisation de telles informations par un modèle mathématique linéaire en variables mixtes suffit à estimer l'importance des critères de façon adéquate.*

Mots-Clés *: aide multicritère à la décision, robustesse, analyse inverse, surclassement, élicitation de poids*

1 Introduction

Nous considérons une situation de décision dans laquelle un ensemble d'alternatives (actions potentielles) est évalué sur une famille finie et cohérente (c'est à dire préférentiellement indépendante, minimale et exhaustive) de critères de performance. Un décideur est amené à comparer par paires ces alternatives en s'appuyant sur les principes des méthodes de surclassement. On considère qu'une alternative *a surclasse* une alternative *b* lorsqu'une *majorité significative* de critères valide le fait que *a* est au moins aussi bonne que *b* et qu'il n'existe aucun critère sur lequel *a* montre une contre-performance notoire par rapport à *b* [1, 2]. La notion de majorité significative est directement liée à la connaissance de l'importance (ou poids) de chaque critère [3]. Ces paramètres peuvent être déterminés :

– soit *directement*, les poids étant donnés par le décideur et permettent ainsi le calcul de la relation de surclassement [4] ;
– soit *indirectement*, le décideur étant amené à fournir des connaissances partielles, permettant d'inférer les poids des critères [5, 6].

Dans cet article, nous nous concentrons exclusivement sur l'approche indirecte. Des approches similaires, nombreuses dans le domaine de la théorie de l'utilité multiattribut, on été généralement publiées sous les termes de méthodes de *désagrégation / agrégation* ou *régression ordinale* [7, 8,

9, 10, 11, 12]. En adéquation avec les techniques correspondantes en statistique inférentielle, nous préférons regrouper l'ensemble des techniques de modélisation d'information de préférence indirecte sous le terme d'*analyse inverse* en aide multicritère à la décision.

L'analyse inverse proposée dans cet article se base sur le concept de robustesse de la validation du surclassement par rapport à un ensemble potentiel de poids [3]. Nous parlerons de robustesse de CONDORCET pour désigner ce type d'informations, conformément à [3]. Cet article s'inscrit dans la suite de nos travaux (voir [6]) et nous montrons ici comment déduire de telles affirmations robustes à partir de préférences partielles exprimées par un décideur, en vue d'estimer numériquement des poids de critères.

L'article s'articule de la façon suivante : nous introduisons ci-après la dénotation de robustesse de CONDORCET d'une relation valuée bipolaire de surclassement, suivi d'une explication rapide sur la façon d'obtenir cette dénotation. Ensuite, en section 3, nous présentons le modèle mathématique d'estimation du poids de signification des critères, suivi d'une simulation d'expérience en section 3.5. Nous terminons par des conclusions et des perspectives de travaux futurs.

2 La robustesse de CONDORCET d'une relation valuée de surclassement

Dans cette section, nous introduisons une mesure de la qualité de la robustesse d'une relation nette obtenue à partir de la coupe majoritaire d'une relation valuée bipolaire de surclassement. Dans un premier temps, nous définissons cette mesure, puis montrons comment, à partir d'un graphe de surclassement valué et d'un ensemble de poids de signification des critères, nous pouvons réaliser cette mesure, appelée *dénotation de robustesse de* CONDORCET [6]. Enfin, nous proposons une nouvelle propriété concernant cette dénotation de robustesse.

2.1 Définitions

Soient $A = \{x, y, z, \ldots\}$ un ensemble fini de $n > 1$ alternatives potentielles et $F = \{g_1, \ldots, g_m\}$ une famille cohérente de $m > 1$ critères.

Pour chaque critère g_i, on évalue les alternatives sur des échelles de performance réelles, auxquelles on associe des seuils de discrimination : un seuil d'indifférence q_i et un seuil de préférence p_i [4]. On note x_i la performance de l'alternative x sur le critère g_i.

Afin de caractériser la proposition *"l'alternative x est au moins aussi bonne que l'alternative y sur le critère g_i"*, on associe à chaque critère $g_i \in F$ un ordre à deux seuils S_i dont la représentation numérique est obtenue par :

$$S_i(x,y) = \begin{cases} 1 & \text{si} \quad x_i + q_i \geqslant y_i \ , \\ 0 & \text{si} \quad x_i + p_i \leqslant y_i \ , \\ 0.5 & \text{sinon.} \end{cases}$$

De plus, nous associons à chaque critère $g_i \in F$ un poids de signification rationnel w_i, qui représente la contribution de g_i dans la validation (ou non) de la proposition "x est au moins aussi bonne que y", pour toutes les paires d'alternatives. Soit $W = \{w_i : g_i \in F\}$ l'ensemble des poids associé à F, tel que $0 < w_i < 1$ $(\forall g_i \in F)$ et $\sum_{g_i \in F} w_i = 1$ et notons $\mathcal{W}$ l'ensemble de ces ensembles de poids.

La *relation valuée bipolaire de surclassement globale*[1], notée $\widetilde{S}^W$, agrégeant les situations de surclassement locales, en vertu de la cohérence de la famille de critères, est donnée par :

$$\widetilde{S}^W(x,y) = \sum_{w_i \in W} w_i \cdot S_i(x,y),\ \forall (x,y)\ \in\ A \times A.$$

$\widetilde{S}^W(x,y)$ est alors évaluée sur un intervalle rationnel $[0,1]$ avec la sémantique suivante [2] :

– $\widetilde{S}^W(x,y) = 1$ lorsque l'ensemble des critères valide les situations de surclassement locales entre x et y ;
– $\widetilde{S}^W(x,y) > 0.5$ lorsqu'une majorité de critères valide la proposition "x est au moins aussi bonne que y" ;
– $\widetilde{S}^W(x,y) = 0.5$ dans le cas d'une situation d'indétermination, lorsque le poids des critères en faveur du surclassement est exactement balancé par celui des critères en défaveur ;
– $\widetilde{S}^W(x,y) < 0.5$ lorsqu'une majorité de critères ne valide pas la proposition "x est au moins aussi bonne que y" ;
– $\widetilde{S}^W(x,y) = 0$ lorsqu'aucun des critères ne valide les situations de surclassement locales entre x et y.

Soit $\succsim_W$ le préordre[2] sur F associé à la relation habituelle $\geqslant$ sur l'ensemble W des poids de signification des critères. $\sim_W$ induit r classes d'équivalences ordonnées $\Pi_1^W \succ_W \ldots \succ_W \Pi_r^W$ $(1 \leq r \leq m)$. Tous les critères d'une même classe d'équivalence ont des poids de même valeur dans W et, pour $i < j$, les critères de Π_i^W ont un poids plus grand que ceux de la classe Π_j^W. Nous dénotons $W_{\mathbb{1}}$ le jeu de poids représentant des critères équi-signifiants.

On note $\mathcal{W}_{\succsim_W} \subset \mathcal{W}$ l'ensemble de tous les ensembles de poids respectant le préordre $\succsim_W$ et soit $W \in \mathcal{W}$. La *dénotation de robustesse* de Condorcet [3] de $\widetilde{S}^W$, notée $[\![\widetilde{S}^W]\!]$, est définie, pour tous $(x,y) \in A \times A$ par :

$$[\![\widetilde{S}^W]\!](x,y)\ =\ \begin{cases} 3 & \text{si } \widetilde{S}^V(x,y) = 1\ \ \forall V \in \mathcal{W}\,; \\ 2 & \text{si } \left[\widetilde{S}^V(x,y) > 0.5\ \ \forall V \in \mathcal{W}_{\succsim_W}\right]\ \wedge\ \left[\exists V' \in \mathcal{W}\ :\ \widetilde{S}^{V'}(x,y) < 1\right]; \\ 1 & \text{si } \left[\widetilde{S}^W(x,y) > 0.5\right]\ \wedge\ \left[\exists V' \in \mathcal{W}_{\succsim_W}\ :\ \widetilde{S}^{V'}(x,y) \leqslant 0.5\right]; \\ 0 & \text{si } \widetilde{S}^W(x,y) = 0.5\,; \\ -1 & \text{si } \left[\widetilde{S}^W(x,y) < 0.5\right]\ \wedge\ \left[\exists V' \in \mathcal{W}_{\succsim_W}\ :\ \widetilde{S}^{V'}(x,y) \geqslant 0.5\right]; \\ -2 & \text{si } \left[\widetilde{S}^V(x,y) < 0.5\ \ \forall V \in \mathcal{W}_{\succsim_W}\right]\ \wedge\ \left[\exists V' \in \mathcal{W}\ :\ \widetilde{S}^{V'}(x,y) > 0\right]; \\ -3 & \text{si } \widetilde{S}^V(x,y) = 0\ \ \forall V \in \mathcal{W}\,; \end{cases}$$

avec la sémantique suivante :

– $[\![\widetilde{S}^W]\!](x,y) = \pm 3$ lorsque tous les critères *valident unanimement* (resp. *invalident*) la situation de surclassement entre x et y ;
– $[\![\widetilde{S}^W]\!](x,y) = \pm 2$ si une *majorité significative* de critères *valide* (resp. *invalide*) la situation de surclassement entre x et y pour tous les jeux de poids de même préordre que W ;
– $[\![\widetilde{S}^W]\!](x,y) = \pm 1$ si une *majorité significative* de critères *valide* (resp. *invalide*) la situation de surclassement entre x et y pour W mais pas pour tous les jeux de poids de même préordre ;
– $[\![\widetilde{S}^W]\!](x,y) = 0$ si le poids des critères en faveur du surclassement est exactement balancé par celui des critères en défaveur pour W.

Le lecteur attentif remarquera qu'en présence d'une situation de veto dans la comparaison d'un

1. Nous ignorons dans cette étude les situations de veto.
2. $\succ_W$ représente la partie asymétrique de $\succsim_W$, alors que $\sim$ représente sa partie symétrique.

TABLE 1 – Tableau de performance

crit. (F)	dir. pref.	poids (W)	alternatives (A) a	b	c	d	e	seuils indiff.	pref.
g_1	max	5/15	70.9	61.8	90.2	31.2	33.1	5.0	8.0
g_2	min	1/15	20.9	17.1	76.3	69.2	35.5	3.0	6.0
g_3	max	3/15	1	4	6	8	6	0	1
g_4	max	4/15	17.3	46.3	24.5	40.6	68.2	6.0	7.0
g_5	max	2/15	2	1	8	2	6	0	1

couple d'alternatives [3], la valeur de robustesse de CONDORCET associée sera -3, étant donné que la relation de surclassement global $\widetilde{S}^W$ est égale à 0, indépendamment des poids des critères.

2.2 Détermination de la dénotation de robustesse de CONDORCET

Dans cette section, nous expliquons brièvement comment obtenir la dénotation de robustesse de CONDORCET à partir d'une relation de surclassement valuée et d'un ensemble de poids associés à des critères. De plus amples détails pourront être trouvés dans [3].

En vue d'illustrer nos propos tout au long de cet article, nous considérons l'exemple numérique suivant.

Exemple *Soit $A = \{a, b, c, d, e\}$ un ensemble de cinq alternatives et F l'ensemble composé de trois critères cardinaux $\{g_1, g_2, g_4\}$, mesurant les performances des alternatives sur des échelles rationnelles allant de 0.0 à 100.0 et de deux critères ordinaux $\{g_3, g_5\}$, mesurant les performances sur une échelle ordinale discrète allant de 0 à 10. g_2 est un critère de coût, pour lequel les performances doivent être minimisées, alors que les quatre autres sont des critères de bénéfice, pour lesquels on privilégie les valeurs élevées.*

La table 1 représente les performances de l'ensemble des alternatives sur chaque critère. On peut remarquer l'ensemble des poids W, en troisième colonne, qui induit l'ordre d'importance des critères suivant : $\{g_1\} \succ_w \{g_4\} \succ_w \{g_3\} \succ_w \{g_5\} \succ_w \{g_2\}$.

Nous introduisons ici des notations qui permettront de détailler la construction de la dénotation de CONDORCET associée à une relation de surclassement $\widetilde{S}^W$ et un ensemble de poids W.

Soit $c_k^W(x, y)$ la somme des $S_i(x, y)$ pour tous les critères $g_i \in \Pi_k^W$, et $\overline{c_k^W}(x, y)$ la somme de la négation $(1.0 - S_i(x, y))$ de ces caractéristiques.

De plus, soit $C_k^W(x, y) = \sum_{i=1}^{k} c_i^W(x, y)$ la somme cumulée des caractéristiques "*au moins aussi bon que*" pour tous les critères ayant une importance au moins égale à celle associée à la classe Π_k^W, et $\overline{C_k^W}(x, y) = \sum_{i=1}^{k} \overline{c_i^W}(x, y)$ la somme cumulée des négations de ces caractéristiques, pour tous les $k \in \{1, \ldots, r\}$.

Pour les paires ordonnées (x, y), en l'absence de dénotation égale à ± 3 (*i.e.* aucune des deux alternatives ne surclasse trivialement l'autre et aucun veto n'est levé), la proposition suivante nous permet de tester la présence d'une dénotation égale à ± 2 :

TABLE 2 – Sommes cumulées pour les couples (b, c) et (a, d)

	$c_i^W(b,c)$	$\overline{c_i^W}(b,c)$	$C_i^W(b,c)$	$\overline{C_i^W}(b,c)$	$c_i^W(a,d)$	$\overline{c_i^W}(a,d)$	$C_i^W(a,d)$	$\overline{C_i^W}(a,d)$
Π_1^W	0	1	0	**1**	1	0	**1**	0
Π_2^W	1	0	**1**	**1**	0	1	**1**	**1**
Π_3^W	0	1	1	**2**	0	1	1	**2**
Π_4^W	0	1	1	**3**	1	0	**2**	**2**
Π_5^W	1	0	2	**3**	1	0	**3**	2

TABLE 3 – Surclassement et dénotation de CONDORCET associée

	$\widetilde{S}^W$					$[\![\widetilde{S}^W]\!]$				
A	a	b	c	d	e	a	b	c	d	e
a	-	.50	.07	.53	.40	-	0	-2	1	-1
b	.53	-	.33	.67	.40	1	-	-2	2	-1
c	.93	.67	-	.47	.67	2	2	-	-1	2
d	.60	.60	.53	-	.53	1	1	1	-	1
e	.60	.60	.53	.80	-	1	1	1	2	-

Proposition 1 (Bisdorff [3])

$$[\![\widetilde{S}^W]\!](x,y) = 2 \iff \begin{cases} \forall k \in 1,...,r : C_k^W(x,y) \geqslant \overline{C_k^W}(x,y)\,; \\ \exists k \in 1,...,r : C_k^W(x,y) > \overline{C_k^W}(x,y). \end{cases}$$

La dénotation négative -2 est obtenue dans des conditions similaires, en renversant les inégalités.

Le test de la dénotation ± 2 en Proposition 1 traduit la vérification de conditions de *dominance stochastique* (voir [3]).

La dénotation ± 1, correspondant à une simple observation de majorité (resp. minorité) pondérée, est vérifiée, dans le cas où la dénotation ± 2 n'est pas applicable, ainsi :

$$[\![\widetilde{S}^W]\!](x,y) = \pm 1 \iff \big((\widetilde{S}^W(x,y) \gtrless 0.5) \wedge [\![\widetilde{S}^W]\!](x,y) \neq \pm 2 \big).$$

Exemple *En revenant à notre exemple, nous pouvons désormais obtenir la dénotation de robustesse de* CONDORCET *associée à la relation de surclassement. Nous détaillons simplement les calculs pour les couples (b, c) et (a, d). Pour rappel, le préordre sur les importances induit un préordre en cinq classes sur les critères : $\{g_1\} \succ_w \{g_4\} \succ_w \{g_3\} \succ_w \{g_5\} \succ_w \{g_2\}$.*

On vérifie aisément dans la Table 2 que $[\![\widetilde{S}^W]\!](b,c) = -2$. Parallèlement, on note que $[\![\widetilde{S}^W]\!](a,d) \neq \pm 2$. Comme $\tilde{S}^W(a,d) = 0.53 > 0.5$, nous en déduisons $[\![\widetilde{S}^W]\!](a,d) = 1$. La table 3 récapitule la relation de surclassement $\widetilde{S}^W$, ainsi que la dénotation de CONDORCET *correspondante $[\![\widetilde{S}^W]\!]$ pour l'ensemble des paires d'alternatives dans $A \times A$.*

2.3 Propriété de la dénotation

Soit $W_{\mathbb{1}}$ le jeu de poids associant la valeur de signification 1 à tous les critères. On observe la propriété suivante :

Proposition 2

$$\widetilde{S}^{W_{\mathbb{1}}}(x,y) < 0.5 \Rightarrow \forall W \in \mathcal{W},\ [\![\widetilde{S}^{W}]\!](x,y) \leq 1$$
$$\widetilde{S}^{W_{\mathbb{1}}}(x,y) > 0.5 \Rightarrow \forall W \in \mathcal{W},\ [\![\widetilde{S}^{W}]\!](x,y) \geq -1$$

Preuve : Si $\widetilde{S}^{W_{\mathbb{1}}}(x,y) < 0.5$: Si l'on veut obtenir une dénotation $+2$ pour la paire ordonnée (x,y), selon la Proposition 1, on doit vérifier la condition suivante : $C_k^W(x,y) > \overline{C_k^W}(x,y)$, pour tous les k, notamment $k = r$, l'indice de la classe la plus faible. Or, $C_r^W(x,y)$ représente le nombre de critères (parmi l'ensemble complet des critères) en faveur du surclassement. Si $\widetilde{S}^{W_{\mathbb{1}}}(x,y) < 0.5$, alors cela signifie que moins de la moitié des critères sont en faveur du surclassement et nous obtenons : $C_r^W(x,y) < \overline{C_r^W}(x,y)$, quel que soit le jeu de poids W.

La démonstration est similaire pour la deuxième partie de la proposition. □

En d'autres termes, s'il n'y a pas au moins la moitié des critères en faveur du surclassement, il est impossible de trouver un jeu de poids robuste au sens de Condorcet, c'est à dire en ne considérant que le préordre des poids. On ne peut garantir la robustesse d'un surclassement allant à l'encontre du surclassement obtenu avec des poids tous équi-signifiants.

3 Analyse inverse pour l'estimation des poids

Dans [6] nous avons montré comment déterminer un ensemble de poids associé aux critères compatible avec une dénotation de Condorcet donnée. En pratique, il est cependant difficile d'imaginer qu'un décideur soit capable de donner tout ou même une partie de la dénotation de Condorcet associée à la relation de surclassement valuée observée dans son problème. Cependant, on peut facilement concevoir qu'un décideur puisse exprimer d'autres informations préférentielles utiles (comparaisons par paires, signification différenciée des critères), permettant l'estimation numérique d'un jeu de poids compatible en passant par la dénotation de robustesse de Condorcet.

Dans un souci de compréhension, nous redonnons dans une première partie les contraintes mathématiques associées à chaque valeur de robustesse, hormis les valeurs ± 3, ainsi que de rapides explications. De plus amples détails pourront être trouvés dans [6]. Dans une seconde partie, nous détaillons le type d'informations préférentielles que nous pouvons intégrer dans le modèle de détermination des poids, et la façon dont nous proposons d'utiliser la dénotation de Condorcet en vue de garantir une plus grande robustesse des résultats.

3.1 Contraintes inhérentes à la robustesse de Condorcet

Comme nous l'avons exprimé précédemment, aucune contrainte nécessaire à la résolution de notre problème ne peut être dérivée dans le cas d'une valeur de robustesse de ± 3. En effet, nous pouvons ignorer les situations d'unanimité, positive ou négative, car elles concernent des comparaisons par paires triviales entre alternatives qui sont soit Pareto dominantes, soit Pareto dominées. La situation

de surclassement global est alors unanimement garantie (resp. non garantie), indépendamment de tout jeu de poids sur les critères. En outre, ces dénotations n'apportent aucune information spécifique pour estimer la signification de chaque critère pris séparément.

Notons $A^2_{\pm 2}$ (resp. $A^2_{\pm 1}$ ou A^2_0) l'ensemble des pairs (x, y) d'alternatives telles que $[\![\widetilde{S}^w]\!](x, y) = \pm 2$ (resp. ± 1 ou 0).

Les poids des critères étant supposés rationnels, nous pouvons, sans perte de généralité, restreindre notre problème d'estimation à des ensembles d'entiers. Ainsi, un poids entier $w_i \in [1, M]$ sera associé à chaque critère g_i, où M représente la valeur maximale admissible. En limitant notre objectif à la résolution de problèmes réels, nous pourrons en pratique fixer cette borne comme étant égale au nombre m de critères.

Posons $P_{m\times M}$ une matrice Booléenne de terme générale $[p_{i,u}]$, qui caractérise par ligne le nombre d'unités de poids alloué au critère g_i. Formellement, la ligne i représente la décomposition du poids associé à g_i sur M bits dans une base unaire (avec les bits de poids fort le plus à gauche), de sorte que $\sum_{u=1}^{M} p_{i,u} = w_i$. Par exemple, si g_i est associé à un poids entier de 3 et que l'on a fixé $M = 5$, alors la $i^{ème}$ ligne de $P_{m\times 5}$ sera $(1, 1, 1, 0, 0)$.

Comme à chaque critère g_i doit être associé un poids strictement positif, nous pouvons en déduire qu'une unité de poids au minimum est allouée à chaque critère, *i.e.* $p_{i,1} = 1$ pour tous les $g_i \in F$. Cela résulte en la contrainte suivante :

$$\sum_{g_i \in F} p_{i,1} = m.$$

Les contraintes suivantes garantissent l'écriture correcte des lignes de P, les bits de poids fort étant tous regroupés à gauche du vecteur ligne :

$$p_{i,u} \geqslant p_{i,u+1} \quad (\forall i = 1, ..., m,\ \forall u = 1, ..., M-1).$$

3.1.1 Contraintes dans les cas où $[\![\widetilde{S}^w]\!](x, y) = \pm 2$

Pour toutes les paires $(x, y) \in A^2_{\pm 2}$, nous introduisons alors l'ensemble de contraintes suivant, modélisant les conditions $[\![\widetilde{S}^w]\!](x, y) = \pm 2$:

$$\sum_{g_i \in F} \Big(p_{i,u} \cdot \pm [S_i(x, y) - \overline{S_i}(x, y)] \Big) \ \geqslant\ b_u(x, y) \quad (\forall u = 1, ..., M),$$

où $\overline{S_i}$ représente la négation $(1 - S_i)$ de la fonction caractéristique d'ordre à deux seuils du critère g_i, et où les $b_u(x, y)$ sont des variables booléennes définies pour chaque paire d'alternatives et chaque niveau d'équi-signifiance $u \in \{1, \ldots, M\}$. Ces variables binaires imposent au moins un cas d'inégalité stricte pour tous les $(x, y) \in A^2_{\pm 2}$, comme requis par la Proposition 1, via les contraintes suivantes :

$$\sum_{u=1}^{m} b_u(x, y) \ \geqslant\ 1, \quad (\,\forall (x, y) \in A^2_{\pm 2}\,).$$

3.1.2 Contraintes dans les cas où $[\![\widetilde{S}^w]\!](x,y) = \pm 1$

Afin de modéliser les contraintes inhérentes aux conditions $[\![\widetilde{S}^w]\!](x,y) = \pm 1$, nous formulons l'ensemble suivant de contraintes, pour toutes les paires $(x,y) \in A^2_{\pm 1}$:

$$\sum_{g_i \in F} \Big(\sum_{u=1}^{M} p_{i,u} \Big) \cdot \pm\big(S_i(x,y) - \overline{S}_i(x,y) \big) \ \geqslant \ 1 \ \ \forall (x,y) \in A^2_{\pm 1},$$

où le facteur $(\sum_{u=1}^{M} p_{i,u})$ représente la valeur entière de l'estimation du poids w_i du critère g_i.

3.1.3 Contraintes dans les cas où $[\![\widetilde{S}^w]\!](x,y) = 0$

A l'instar de la section précédente, pour toutes les paires $(x,y) \in A^2_0$, nous formulons l'ensemble suivant de contraintes :

$$\sum_{g_i \in F} \Big(\sum_{u=1}^{M} p_{i,u} \Big) \cdot \big(S_i(x,y) - \overline{S}_i(x,y) \big) \ = \ 0.$$

3.1.4 Relaxation des contraintes à l'aide de variables d'écart

Comme nous l'expliquerons plus en détail dans la section suivante, en pratique il est impossible de demander directement cette dénotation à un décideur, en vue de déterminer la signification des critères. Il nous faudra donc inférer, à partir de ses préférences, les contraintes jugées nécessaires. Dans la pratique, nous devons alors faire face à la fois à des incompatibilités inhérentes aux préférences du décideur (celui-ci ayant exprimé un ensemble de préférences dont la réalisation simultanée est impossible) et aussi à des difficultés pour assurer la robustesse du surclassement lié au jeu de poids résultant. Pour ce dernier point en effet, il ne sera pas toujours possible de garantir tous les surclassements robustes souhaités, lorsque ceux-ci sont incompatibles entre eux ou bien, lorsque selon la Proposition 2, ils ne peuvent tout simplement jamais être satisfaits.

On introduit alors la notion de *contraintes relaxées*, que l'on dérive des *contraintes originelles* (ou contraintes fortes) en y ajoutant des *variables d'écart*. Ces variables, réelles positives que l'on cherchera à minimiser, permettent la satisfaction des contraintes de la dénotation de robustesse en cas d'impossibilité, et offrent la possibilité d'identifier les contraintes problématiques lors de l'analyse de la solution. En effet, si la valeur d'une variable d'écart n'est pas nulle à la fin de la résolution, alors la contrainte associée n'a pas pu être satisfaite. Selon la valeur de la dénotation, les conséquences diffèrent :

– Sur une contrainte de type ± 1 ou 0, une variable d'écart non nulle indique le changement de sens du surclassement, dans le cas où l'on ne parvient pas à trouver un jeu de poids vérifiant tous les surclassements ;
– Sur une contrainte de type ± 2, les variables d'écart permettent d'assurer qu'il n'y aura pas de blocage dans la résolution si l'on souhaite de la robustesse là où le modèle ne pourra la garantir.

Ainsi, nous utilisons des variables d'écart sur les contraintes de type ± 1 et 0 lorsque le décideur n'est pas certain d'un surclassement ou bien lorsqu'il est prêt à revoir son jugement sur certains surclassements si ceux-ci sont incompatibles avec d'autres auxquels il accorde une plus forte crédibilité. Sur les contraintes de type ± 2, elles permettent d'éviter le blocage de la résolution du problème lorsque l'on n'a pas réussi à assurer la robustesse. Il est à noter que, lorsqu'une contrainte relaxée de type

± 2 est violée, le surclassement n'est plus assuré ; il faut par conséquent coupler une telle contrainte relaxée avec une contrainte forte de type ± 1 de même sens si l'on veut garantir le surclassement.

On redonne ici les contraintes sous leurs versions relaxées :

$$\sum_{g_i \in F} \Big(p_{i,u} \cdot \pm [S_i(x,y) - \overline{S_i}(x,y)] \Big) \pm s_u^{\pm 2}(x,y) \geqslant b_u(x,y) \qquad \forall u = 1,...,M \quad \forall (x,y) \in A^2_{\pm 2}$$

$$\sum_{g_i \in F} \Big(\sum_{u=1}^{M} p_{i,u} \Big) \cdot \pm \big(S_i(x,y) - \overline{S}_i(x,y) \big) \pm s^{\pm 1}(x,y) \geqslant 1 \qquad \forall (x,y) \in A^2_{\pm 1}$$

$$\sum_{g_i \in F} \Big(\sum_{u=1}^{M} p_{i,u} \Big) \cdot \big(S_i(x,y) - \overline{S}_i(x,y) \big) + s^0_+(x,y) - s^0_-(x,y) = 0 \qquad \forall (x,y) \in A^2_0$$

où $s_u^{\pm 2}(x,y)$ (resp. $s^{\pm 1}(x,y)$) sont les variables d'écart associées au couple (x,y). Notons que pour les contraintes associées à une dénotation de valeur 0, il faut à chaque fois deux variables d'écart, $s^0_+(x,y)$ et $s^0_-(x,y)$ afin de connaître, en cas de non-satisfaction, dans quel sens le surclassement a été validé.

3.2 Prise en compte des préférences d'un décideur

Dans la section précédente nous avons décrit les contraintes induites par les valeurs de robustesse spécifiques sur l'obtention des poids des critères. Cependant, comme déjà signalé précédemment, en pratique il est difficile d'imaginer qu'un décideur soit capable de fournir des valeurs de robustesse pour tout couple d'alternatives. Nous proposons dans cette section d'identifier l'information préférentielle qu'un décideur est susceptible de fournir dans le cadre du problème de décision, et de présenter la façon dont cette information est intégrée dans le modèle d'analyse inverse.

3.2.1 Identification des préférences

La technique d'acquisition de cette information via un questionnement particulier du décideur n'est pas abordée ici. En effet, nous avons prévu d'étudier cette tâche dans nos travaux futurs.

De manière générale, dans le cadre que nous avons établi aux sections précédentes, on peut supposer qu'un décideur soit en mesure de fournir les informations préférentielles suivantes :

- un sous-ensemble E de $A \times A$ de couples ordonnés d'alternatives (a,b) pour lesquels le décideur est en mesure d'indiquer un sens de préférence strict ou une indifférence ;

  Exemple : a est préférée à b et c est indifférente à d ;
- un préordre partiel $\succeq_N$ sur les poids d'un sous-ensemble de critères $N \subseteq F$;

  Exemple : le critère g_1 a un poids plus important que le critère g_4 ;
- des valeurs numériques associées aux poids de certains critères ;

  Exemple : le poids du critère g_2 vaut 3 ;
- des contraintes sur les valeurs numériques associées aux poids de certains critères ;

  Exemple : le poids du critère g_1 est entre 2 et 4 ;
- un préordre partiel entre des ensembles de critères exprimant des préférences sur les sommes des poids de certains critères ;

  Exemple : les critères g_1 et g_3 pris ensemble sont plus importants que le critère g_2 ;
- des ensembles de critères pouvant valider ou invalider le surclassement ;

Exemple : si une alternative x est au moins aussi bonne qu'une alternative y sur les critères g_1, g_2 et g_3, alors le surclassement de x sur y est validé.

Nous détaillons dans la section suivante la traduction de ces informations préférentielles en contraintes linéaires dans le cadre du modèle d'analyse inverse que nous présentons dans cet article.

3.2.2 Traduction des préférences en contraintes linéaires

Le premier type d'information préférentielle concerne des préférences strictes et des indifférences sur un sous-ensemble d'alternatives E. Soit P la relation de préférence stricte sur E et I la relation d'indifférence sur E. Si le décideur estime que aPb alors on a nécessairement $\widetilde{S}^W(a,b) > 0.5$ et $\widetilde{S}^W(b,a) < 0.5$, où W est le jeu de poids recherché. De même, si le décideur exprime que cId alors on a $\widetilde{S}^W(c,d) > 0.5$ et $\widetilde{S}^W(d,c) > 0.5$.

En vue de fournir une réponse robuste au décideur, nous optons pour la traduction suivante de l'information préférentielle en contraintes linéaires :

- Une contrainte forte de type $+1$ pour le couple (a,b) afin de garantir le surclassement ;
- Une contrainte relaxée de type $+2$ pour le couple (a,b) pour viser un surclassement robuste ;
- Une contrainte forte de type -1 pour le couple (b,a) ;
- Une contrainte relaxée de type -2 pour le couple (b,a).

De façon similaire, on traduit les préférences sur l'indifférence entre les alternatives c et d de la façon suivante :

- Des contraintes fortes de type $+1$ pour les couples (c,d) et (d,c) afin de garantir l'indifférence ;
- Des contraintes relaxées de type $+2$ pour les couples (c,d) et (d,c) pour viser une indifférence robuste.

Il est à noter que, selon la Proposition 2, si le surclassement va à l'encontre du surclassement élémentaire, il ne sera pas possible de garantir la robustesse au sens de Condorcet et donc inutile de rajouter les contraintes relaxées de type ± 2.

Les informations sur les poids des critères sont plus aisées à représenter en contraintes linéaires. Ainsi, si un décideur souhaite fixer le poids du critère g_i à la valeur entière v, on ajoute la contrainte :

$$\sum_{c=1}^{M} p_{ic} = v$$

Si le décideur désire encadrer la valeur du critère g_i entre les entiers u et v, nous pouvons ajouter les contraintes suivantes :

$$\sum_{c=1}^{M} p_{ic} \geq u \quad \text{et} \quad \sum_{c=1}^{M} p_{ic} \leq v$$

Le fait que le critère g_i est, selon le décideur, plus important que le critère g_j se traduira par l'ajout de la contrainte :

$$\sum_{c=1}^{M} p_{ic} \geq \sum_{c=1}^{M} p_{jc} + 1$$

On généralise cette formule aux sous-ensembles de critères : Si un sous-ensemble H de critères a

une importance plus grande que celle du sous-ensemble K, alors on ajoute la contrainte :

$$\sum_{g_i \in H} (\sum_{c=1}^{M} p_{ic}) \geq \sum_{g_j \in K} (\sum_{c=1}^{M} p_{jc}) + 1$$

On peut aussi modéliser le fait qu'un sous-ensemble H soit, selon le décideur, suffisant pour valider un surclassement (la somme de ses poids est strictement supérieure à la moitié de la somme des poids de tous les critères) :

$$\sum_{g_i \in H} (\sum_{c=1}^{M} p_{ic}) \geq \frac{1}{2} \cdot \sum_{g_j \in F} (\sum_{c=1}^{M} p_{jc}) + 1$$

3.3 Fonction objectif

Nous faisons ici le choix de ne pas explorer l'ensemble du polytope des solutions viables, mais de déterminer un ensemble de poids W^* :

- satisfaisant l'ensemble des contraintes $[\![\widetilde{S}^{W^*}]\!](x,y) = \pm 1$,
- respectant au mieux les contraintes $[\![\widetilde{S}^{W^*}]\!](x,y) = \pm 2$, et
- avec les poids $w_i^* \in W^*$ $(\forall g_i \in F)$ les plus petits possibles (ce qui, en pratique, tend à utiliser le moins de classes d'équi-signifiance possible).

En conséquence, nous introduisons la fonction objectif suivante, devant être minimisée :

$$K_1\Big(\sum_{g_i \in F} \sum_{u=1}^{M} p_{i,u}\Big) \; + \; K_2\Big(\sum_{(x,y) \in A^2_{\pm 2}} \sum_{u=1}^{M} s_u^{\pm 2}(x,y)\Big) \; - \; K_3\Big(\sum_{u=1}^{M} \Big(\sum_{(x,y) \in A^2_{\pm 2}} b_u(x,y)\Big)\Big)$$

où $K_1 ... K_3$ sont des constantes paramétriques utilisées pour ordonner correctement les trois sous-objectif. Notons que la troisième partie de l'objectif n'est pas nécessaire à la résolution du problème, mais elle garantit un renforcement maximal des contraintes $[\![\widetilde{S}^{W}]\!](x,y) = \pm 2$ en forçant si possible le maximum d'inégalités strictes.

3.4 Le programme mathématique d'analyse inverse

Soit $\mathfrak{S}$ l'ensemble des couples (x,y) tel que x surclasse y, c'est à dire que le décideur préfère x à y ou bien qu'ils lui sont indifférents, auquel cas l'ensemble contient aussi le couple (y,x). Soit $\overline{\mathfrak{S}}$ l'ensemble des couples (x,y) pour lesquels le décideur préfère y à x. En résumé, nous donnons ici le programme mathématique linéaire en variables mixtes :

PLVM	
Variables :	
$p_{i,u} \in \{0,1\}$	$\forall g_i \in F,\ \forall u = 1,..,M$
$b_u(x,y) \in \{0,1\}$	$\forall (x,y) \in \mathfrak{S} \cup \overline{\mathfrak{S}}, \forall u = 1,..,M$
$s_u^{\pm 2}(x,y) \geqslant 0$	$\forall (x,y) \in \mathfrak{S} \cup \overline{\mathfrak{S}}, \forall u = 1,..,M$
Paramètres :	
$K_i > 0$	$\forall i = 1...3$
Fonction objectif :	

$$\min \quad K_1\Big(\sum_{g_i\in F}\sum_{u=1}^{M} p_{i,j}\Big) - K_2\Big(\sum_{u=1}^{M}\sum_{(x,y)\in A^2_{\pm 2}} b_u(x,y)\Big) + K_3\Big(\sum_{(x,y)\in \mathfrak{S}\ \cup\overline{\mathfrak{S}}} s_u^{\pm 2}(x,y)\Big)$$

Contraintes :

$$\begin{array}{lll}
s.t. & \sum_{g_i\in F} p_{i,1} = m & \\
& p_{i,u} \geqslant p_{i,u+1} & \forall g_i \in F,\ \forall u = 1,..,M-1 \\
& \sum_{g_i\in F}\Big(p_{i,u}\cdot\big[\,S_i(x,y) - \overline{S_i}(x,y)\,\big]\Big) + s_u^{+2}(x,y) \ \geq\ b_u(x,y) & \forall (x,y)\in \mathfrak{S},\ \forall u = 1,..,M \\
& \sum_{g_i\in F}\Big(p_{i,u}\cdot\big[\,S_i(x,y) - \overline{S_i}(x,y)\,\big]\Big) - s_u^{-2}(x,y) \ \leq\ -b_u(x,y) & \forall (x,y)\in \overline{\mathfrak{S}},\ \forall u = 1,..,M \\
& \sum_{u=1}^{M} b_u(x,y) \ \geqslant\ 1 & \forall (x,y)\in \mathfrak{S}\ \cup\overline{\mathfrak{S}} \\
& \sum_{g_i\in F}\Big(\big(\sum_{u=1}^{M} p_{i,u}\big)\cdot\big(S_i(x,y) - \overline{S}_i(x,y)\big)\Big) \ \geqslant\ 1 & \forall (x,y)\in \mathfrak{S},\ \forall u = 1,..,M \\
& \sum_{g_i\in F}\Big(\big(\sum_{u=1}^{M} p_{i,u}\big)\cdot\big(\overline{S}_i(x,y) - S_i(x,y)\big)\Big) \ \geqslant\ 1 & \forall (x,y)\in \overline{\mathfrak{S}},\ \forall u = 1,..,M
\end{array}$$

Contraintes (informelles) sur les poids inhérentes aux préférences du décideur :

$\sum_{c=1}^{M} p_{ic} = v_i$	Pour certains critères g_i
$\sum_{c=1}^{M} p_{ic} \geq u$ et $\sum_{c=1}^{M} p_{ic} \leq v$	Pour certains critères g_i
$\sum_{c=1}^{M} p_{ic} \geq \sum_{c=1}^{M} p_{jc} + 1$	Pour certains couples (g_i, g_j)
$\sum_{g_i\in H}(\sum_{c=1}^{M} p_{ic}) \geq \frac{1}{2}\cdot\sum_{g_j\in F}(\sum_{c=1}^{M} p_{jc}) + 1$	Pour certains sous-ensembles de critères
$\sum_{g_i\in H}(\sum_{c=1}^{M} p_{ic}) \geq \sum_{g_j\in K}(\sum_{c=1}^{M} p_{jc}) + 1$	Pour certains sous-ensembles de critères

3.5 Implémentation et expérimentations

Dans cette section nous commençons par présenter rapidement le logiciel permettant de procéder à une analyse inverse robuste à partir d'informations préférentielles partielles d'un décideur, et nous détaillons ensuite la méthodologie proposée sur l'exemple présenté précédemment dans cet article.

Le logiciel mettant en oeuvre le programme mathématique présenté à la section précédente est disponible sous la forme d'un service web dans le cadre du projet Decision Deck [3]. Ce service fait appel au solveur CPlex 11.0 via son interface de programmation AMPL. Les données d'entrée et de sortie sont modélisées au format XMCDA 2.0.0 [4] en vue de permettre sa composition avec d'autres services du projet Decision Deck (en particulier, calcul de la dénotation de robustesse et tracé du graphe de surclassement). Ce service web est entre autres disponible dans la plateforme diviz [5] qui permet de construire des workflows d'exécution d'algorithmes d'aide à la décision multicritère.

En vue de montrer la pertinence de notre discours, revenons à l'exemple fictif introduit en début d'article. Supposons que le décideur ait en sa possession la description de l'ensemble A des cinq alternatives, celle de l'ensemble F des cinq critères, le tableau de performances, ainsi que les seuils de préférence et d'indifférence des critères (voir tableau 1). En vue de déterminer la relation de surclassement permettant de résoudre son problème de décision, nous allons tenter d'obtenir un ensemble de poids des critères aussi *robuste* que possible à partir d'informations préférentielles partielles.

3. `http://www.decision-deck.org/ws`
4. `http://www.decision-deck.org/xmcda`
5. `http://www.decision-deck.org/diviz`

TABLE 5 – Surclassement et dénotation de robustesse associée

	$W_1 = \{2,1,1,1,1\}$										$W_2 = \{3,1,2,2,1\}$									
	$\widetilde{S}^{W_1}$					$[\![\widetilde{S}^{W_1}]\!]$					$\widetilde{S}^{W_2}$					$[\![\widetilde{S}^{W_2}]\!]$				
A	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e
a	-	.58	.17	.67	.50	-	2	-2	2	0	-	.50	.11	.56	.44	-	0	-2	1	-1
b	.58	-	.33	.75	.50	1	-	-1	2	0	.61	-	.33	.72	.44	1	-	-2	2	-1
c	.83	.67	-	.50	.67	2	2	-	0	2	.89	.67	-	.44	.67	2	2	-	-1	2
d	.50	.50	.50	-	.50	0	0	0	-	0	.56	.56	.56	-	.56	1	1	1	-	1
e	.50	.50	.50	.83	-	0	0	0	2	-	.56	.56	.56	.78	-	1	1	1	2	-

Dans un premier temps, le décideur indique des préférences partielles sur un couple d'alternatives et l'importance d'un critère de la façon suivante :

– l'alternative c est préférée à a ;
– le critère g_1 est le critère le plus important.

Ces informations sont traduites en un ensemble de contraintes linéaires (voir section 3.5 ci-avant) et la solution optimale du programme nous fournit l'ensemble de poids $W_1 = \{2,1,1,1,1\}$. La relation de surclassement ainsi que la dénotation de robustesse associée sont repris dans la première partie du tableau 5. On note le fait que la résolution fixe toutes les variables d'écart à 0, ce qui prouve que la contrainte souhaitée par le décideur a pu être satisfaite de façon robuste (cPa).

En analysant la dénotation de robustesse, on remarque que beaucoup de valeurs sont en situation d'indétermination. Il est donc nécessaire que le décideur fournisse un peu plus d'informations en vue de déterminer un ensemble de poids qui soit plus discriminant pour la comparaison des alternatives.

Après avoir réétudié son problème de décision, le décideur estime qu'il peut nous fournir les informations supplémentaires suivantes :

– l'alternative b est considérée comme indifférente à l'alternative d ;
– la validation de la concordance pour les 3 critères $\{g_2, g_3, g_4\}$ est suffisante pour valider un surclassement.

Ces nouvelles informations sont retraduites en contraintes linéaires, et la résolution du programme mathématique nous donne l'ensemble de poids $W_2 = \{3,1,2,2,1\}$. La relation de surclassement et sa robustesse sont présentées dans la partie droite du tableau 5. On observe qu'un grand nombre d'indéterminations ont été levées. Notons également que, d'après la relation de surclassement, on a bien bId, cependant ceci n'est pas une information très robuste (on a $[\![\widetilde{S}^{W_2}]\!](b,d) = 2$, mais $[\![\widetilde{S}^{W_2}]\!](d,b) = 1$). Ceci se traduit également par une variable d'écart non nulle au niveau des contraintes linéaires. En pratique, cela implique qu'un léger changement dans le jeu de poids pourrait invalider l'affirmation du décideur au sujet de l'indifférence entre b et d.

Nous observons qu'en procédant ainsi, on peut arriver à obtenir un jeu de poids du décideur, via un questionnement indirect. Cette recherche se fait en essayant de maximiser la robustesse de la relation de surclassement, tout en respectant les préférences exprimées par le décideur.

4 Conclusions et perspectives

Dans cet article, nous avons proposé une nouvelle approche d'analyse inverse permettant d'obtenir une estimation numérique robuste des poids des critères dans une analyse multicritère à partir d'informations préférentielles partielles qu'un décideur peut fournir, en se basant sur la notion de robustesse de Condorcet d'une relation valuée bipolaire de surclassement.

Cette approche est actuellement distribuée par le projet Decision Deck sous la forme d'un service web, accessible entre autres à partir de la plateforme diviz. Les expérimentations pratiques que nous prévoyons dans nos travaux futurs nous indiqueront les voies de développement et d'amélioration, notamment quant aux méthodes de questionnement du décideur. En outre, un raffinement de la fonction objectif, permettant d'atteindre d'autres jeux de poids viables, ou plus généralement l'exploration du polytope des solutions possibles, devra être considéré.

Références

[1] Roy B. Critères multiples et modélisation des préférences (l'apport des relations de surclassement). *Revue d'Economie Politique*, Volume 84, n°1, 1-44, janvier/février 1974.

[2] R. Bisdorff. Logical Foundation of Multicriteria Preference Aggregation. In : *Bouyssou D et al. (eds) Aiding Decisions with Multiple Criteria. Kluwer Academic Publishers*, pp. 379-403, 2002.

[3] R. Bisdorff. Concordant Outranking with multiple criteria of ordinal significance. *4OR : A Quarterly Journal of Operations Research*, 2(4) :293–308, 2004.

[4] Roy B, Bouyssou D (1993) Aide Multicritère à la Décision : Méthodes et Cas. Economica, Paris

[5] Mousseau V and Slowinski R. Inferring an ELECTRE TRI model from assignment examples. *Journal of Global Optimization*, 12(2) :157-174, 1998.

[6] R. Bisdorff, P. Meyer and T. Veneziano. Inverse analysis from a Condorcet robustness denotation of valued outranking relations. *F. Rossi and A. Tsoukiás (Eds.), Algorithmic Decision Theory. Springer-Verlag Berlin Heidelberg, LNAI 5783*, pp. 180–191, 2009.

[7] Jacquet-Lagrèze E, Siskos Y. Assessing a set of additive utility functions for multicriteria decision making : the UTA method. *The European Journal of Operational Research*, 10 151–164, 1982.

[8] Mousseau V, Figueira J, Dias L C, Gomes da Silva C, Clímaco J N. Resolving inconsistencies among constraints on the parameters of an MCDA model. *The European Journal of Operational Research*, 147 (1) 72–93, 2003.

[9] Siskos Y, Grigoroudis E, Matsatsinis N F. UTA Methods. In : Figueira J, Greco S, Ehrgott M (eds) State of the Art in Multiple Criteria Decision Analysis. *Springer Berlin*, chapter 8 297–343, 2005.

[10] Grabisch M, Kojadinovic I, Meyer P. A review of capacity identification methods for Choquet integral based multi-attribute utility theory : Applications of the Kappalab R package. *European Journal of Operational Research*, 186 (2), 766-785, 2008.

[11] Greco S, Mousseau V, Slowinski R. Ordinal regression revisited : Multiple criteria ranking using a set of additive value functions. *The European Journal of Operational Research*, 191 416–436, 2008.

[12] P. Meyer, J.-L. Marichal, R. Bisdorff. Disaggregation of bipolar-val-ued outranking relations. In : *L.T. Hoai An, P. Bouvry, P.D. Tao, editors, Modelling, Computation and Optimization in Information Systems and Management Sciences, Communications in Computer and Information Science, Springer, Proc. of MCO'08 conference*, Metz, France, 8-10 September 2008, 204 - 213, 2008.

La Condition Not-First/Not-Last en Ordonnancement Cumulatif

Roger Kameugne[1], Ervée Kouakam[2], Laure Pauline Fotso[2]

[1] Université de Maroua, Ecole Normale Supérieure, Département de Mathématiques, BP 46 Maroua-Cameroun, rkameugne@yahoo.fr

[2] Université de Yaoundé 1, Département d'Informatique ; BP 812 Yaoundé-Cameroun
kouakamerve@yahoo.fr, lpfotso@ballstate.bsu.edu

Résumé : *Cet article s'intéresse à la technique de propagation de la contrainte de ressource en ordonnancement connue sur le nom de Not-First/Not-Last. La règle du Not-First/Not-Last détermine si une tâche i ne peut être ordonnancée en premier (Not-First) ou en dernier (Not-Last) parmi les tâches de l'ensemble de tâches $\Omega \cup \{i\}$ tous s'exécutant sur la même ressource (Ω étant un ensemble de tâches). Nous proposons dans cet article un algorithme itératif pour la règle du Not-First/Not-Last de complexité $O(n^3)$ où n est le nombre de tâches, améliorant ainsi la complexité de la meilleur implémentation de cette règle proposé par Schutt et al [1] du facteur $\log(n)$. Les résultats expérimentaux sont présentés.*

Mots-Clés : *Ordonnancement cumulatif, Propagation des contraintes, Intervalle de tâches, Not-First/Not-Last.*

1 Introduction

L'edge-finding, le timetable sont les techniques de propagation des contraintes de ressource les plus utilisées. A coté de ces contraintes, on rencontre aussi le Not-First/Not-Last. Comme l'edge-finding ; elle détermine la position relative d'une tâche par rapport à un ensemble de tâches tous s'exécutant sur une même ressource. Ces règles sont bien connues dans le cas disjonctif (ressource de capacité l'unité) [2]. Dans le cas cumulatif, elles sont plus complexes car plusieurs tâches peuvent s'exécuter en parallèle sur une même ressource. A la connaissance des auteurs, le Not-First/Not-Last cumulatif a été introduite pour la première fois par Win Nuijten [5]. Ce dernier propose un algorithme de complexité $O(n^3k)$ où n désigne le nombre de tâches et k le nombre des différentes consommations de la ressource. Mais en 2005, Schutt et al [1] montrent que cet algorithme est incorrect et incomplet. Ils proposent alors un algorithme complet de complexité $O(n^3 \log(n))$. Il est mentionné dans [3] l'existence d'un algorithme de Not-First/Not-Last cubique proposé par Win Nuijten. A notre connaissance un tel algorithme n'a pas encore été publié. Dans [4], les auteurs proposent un algorithme de Not-First/Not-Last cumulatif incomplet de complexité $O(n^2h)$ où h est le nombre de différentes dates de fin au plus tôt des taches (Nous reviendrons plus lois sur l'incomplétude de cet algorithme). L'algorithme de Not-First/Not-Last s'applique sur les instances de CuSP pour Cumulative Scheduling Problems.

Une instance de CuSP est définie par la donnée d'un ensemble T de tâches devant s'exécuter sur une ressource de capacité C, chaque tâche i s'exécutant durant p_i unités de temps entre sa date de début au plus tôt r_i et sa date de fin au plus tard d_i, et nécessitant une consommation en ressource

notée c_i. On note $H = \{r_i + p_i, i \in T\}$ l'ensemble des différentes dates de fin au plus tôt des tâches et $h = |H|$. On note également $K = \{c_i, i \in T\}$ l'ensemble des différentes cosommations de la ressource par les tâches et $k = |K|$. On suppose que toutes ces données sont des entiers positifs et qu'on a toujours $r_i + p_i \leq d_i$ et $c_i \leq C \ \forall i \in T$ faute de quoi le problème serait insoluble. Une solution du CuSP est un ordonnancement dans lequel on attribue à chaque tâche i un date de début s_i telle que :

$$\forall i \in T : \quad r_i \leq s_i \leq s_i + p_i \leq d_i \tag{1}$$

$$\forall t \qquad \sum_{\substack{i \in T \\ s_i \leq t < s_i + p_i}} c_i \leq C \tag{2}$$

On note $e_i = c_i p_i$, l'énergie de la tâche i. On étend ces notations de la tâche aux ensembles de tâches non vides Ω en posant $r_\Omega = \min_{j \in \Omega} r_j$, $d_\Omega = \max_{j \in \Omega} d_j$, $e_\Omega = \sum_{j \in \Omega} e_j$. Par convention, lorsque Ω est vide, $r_\Omega = +\infty$, $d_\Omega = -\infty$ et $e_\Omega = 0$. Le CuSP est un sous problème du RCPSP pour Ressources Constraint Project Schedule Problems dans lequel la contrainte de précédence est relaxée et une seule ressource est considérée. C'est un problème NP-complet [3].

Dans cette article, nous proposons un nouvel algorithme de Not-First/Not-Last cumulatif de complexité $O(n^3)$, améliorant ainsi la meilleur implémentation de cette règle proposée par Schutt et al [1] du facteur $\log(n)$. Notre algorithme détecte toutes les conditions de la règle de Not-First pour une instance de CuSP, mais parfois effectue un faible ajustement. Cette procédure est donc utilisée itérativement jusqu'au point fixe. Cette approche n'est pas nouvelle, elle a déjà utilisée précédemment dans le cas du Not-First/Not-Last disjonctif [10]. Cet algorithme est complet dans le sens où il atteint le même point fixe que celui de la littérature. Les résultats expérimentaux confirment les résultats théoriques obtenus. Le reste de l'article est organisé comme suit : le paragraphe 2 est consacré aux propriétés de dominance du Not-First/Not-Last cumulatif. Au paragraphe 3, nous présentons notre algorithme de Not-First/Not-Last cumulatif et le paragraphe 4 est consacré aux résultats expérimentaux. Le paragraphe 5 conclue l'article et donne quelques perspectives.

2 Propriétés de Dominance du Not-First/Not-Last

Dans cette section, nous présentons la notion d'intégrité des fenêtres de tâches [6] aussi connue sous le nom de E-Feasibility [8] ou overlaoding checking [2] et donnons une spécification de la règle du Not-First/Not-Last cumulatif.

2.1 E-Feasibility

On montre que s'il existe un ordonnancement faisable pour une instance de CuSP alors pour tout sous ensemble de tâche Ω, l'énergie e_Ω est plus petit que C multipliée par la fenêtre de temps de Ω. Cette contrainte est donnée dans la définition suivante :

Définition 1 (E-faisable) *Une instance de CuSP est E-faisable si*

$$\forall \Omega \subseteq T \qquad C(d_\Omega - r_\Omega) \geq e_\Omega$$

Pour tester L'E-faisable, on utilise le concept d'intervalle de tâches

Définition 2 (Intervalles de tâches) *Soient i et k deux tâches (éventuellement les mêmes tâches). Si $r_i \leq r_k$ et $d_i \leq d_k$ alors l'intervalle de tâches $\Omega_{i,k}$ est l'ensemble de tâches*

$$\Omega_{i,k} = \{j \in T | r_i \leq r_k \wedge d_i \leq d_k\}$$

si non $\Omega_{i,k} = \emptyset$.

Nous illustrons ces définitions dans l'exemple suivant.

Exemple 1 *Considérons l'instance de CuSP donnée dans le tableau Table 1*

Tâches	r	d	p	c
a	2	100	3	1
b	2	9	1	1
c	0	11	3	1
d	1	9	3	1

TABLE 1 – Une instance E-faisable de CuSP

D'après la définition 2, $\Omega_{c,c} = \{b, c, d\}$, $\Omega_{a,c} = \emptyset$ et $\Omega_{c,a} = \{a, b, c, d\}$. On a $e_{\Omega_{c,c}} = 7 \leq C(d_{\Omega_{c,c}} - r_{\Omega_{c,c}}) = 11$ et $e_{\Omega_{c,a}} = 10 \leq C(d_{\Omega_{c,a}} - r_{\Omega_{c,a}}) = 100$. Après avoir tester tous les intervalle de tâches on conclut que cette instance de CuSP est E-faisable.

Il existe des algorithmes de complexité $O(n \log(n))$ efficaces pour tester cette condition [9]. Dans toute la suite les instances de CuSP que nous utiliserons sont supposées E-faisables.

2.2 Règle du Not-First/Not-Last

Comme le edge-finding [5, 8, 7] la règle Not-first/Not-last est importante pour un filtrage de qualité des domaines des tâches. Il est prouvé dans [3] que l'edge-finding n'est pas dominé par le Not-First/Not-Last et vice versa. De ce fait, pour un filtrage maximal du domaine des tâches, il est nécessaire d'utiliser la conjonction des deux algorithmes de filtrage. La proposition 1 donne les conditions pour qu'une tâche i s'exécute avant ou après au moins une tâche d'un ensemble Ω tous s'exécutant sur une même ressource.

Proposition 1 *Soient $\Omega \subset T$ un ensemble non vide d'une instance E-faisable de CuSP et $i \notin \Omega$ une tâche.*

1. *Si*
$$r_i < min_{j \in \Omega}(r_j + p_j) \wedge e_\Omega + c_i(\min(r_i + p_i, d_\Omega) - r_\Omega) > C(d_\Omega - r_\Omega) \quad (3)$$
alors toutes les dates de début de la tâche i inférieures à $min_{j \in \Omega}(r_j + p_j)$ sont inconsistantes.
2. *Si*
$$max_{j \in \Omega}(d_j - p_j) < d_i \wedge e_\Omega + c_i(d_\Omega - \max(d_i - p_i, r_\Omega)) > C(d_\Omega - r_\Omega) \quad (4)$$
alors toutes les dates de fin de la tâche i supérieures à $min_{j \in \Omega}(d_j - p_j)$ sont inconsistantes.

Preuve : Voir [1]. □

La règle qui permet d'ajuster toutes les fenêtres de temps en utilisant la condition (3) de la Proposition 1 est appelée Not-First. Elle permet d'ajuster la date de début au plus tôt de toute tâche i qui est non première dans l'exécution de l'ensemble de tâches $\Omega \cup \{i\}$. Un algorithme de Not-First est toute procédure qui effectue toutes ces déductions et ajustements. De même la règle qui permet d'ajuster les fenêtres de temps en utilisant la condition (4) de la Proposition 1 est appelée Not-last. Elle permet d'ajuster la date de fin au plus tard de la tâche i qui est non dernier dans l'exécution de l'ensemble de tâche $\Omega \cup \{i\}$. Un algorithme de Not-last est toute procédure qui effectue ces déductions et ajustements. Dans la suite nous nous limiterons à la règle Not-First utilisant la première condition de la Proposition 1. Les résultats similaires peuvent être établis sur la version symétrique.

Définition 3 (Algorithme du Not-first) *Un algorithme du Not-first/Not-last prend en entrée une instance E-faisable de CuSP et renvoie en sortie un vecteur*

$$\langle \overline{LB}(1), ..., \overline{LB}(n) \rangle$$

où

$$\overline{LB}(i) = \max(r_i, LB(i))$$

et

$$LB(i) = \max_{\substack{\Omega \subseteq T \\ i \notin \Omega \\ \gamma(\Omega, i)}} \min_{j \in \Omega}(r_j + p_j)$$

avec

$$\gamma(\Omega, i) \stackrel{def}{=} \left(\begin{array}{c} \left(r_i < \min_{j \in \Omega}(r_j + p_j) \right) \\ \wedge \\ (e_\Omega + c_i (\min(r_i + p_i, d_\Omega) - r_\Omega) > C(d_\Omega - r_\Omega)) \end{array} \right).$$

Exemple 2 *Considérons une instance du CuSP définie par le tableau Table 2*

Tâche	r	d	p	c
a	0	46	22	2
b	0	26	24	2
c	0	26	24	2

TABLE 2 – Instance E-faisable de CuSP

Ici $C = 5$; la relation (3.1) est vérifiée pour $i = a$ et $\Omega = \{b, c\}$. En effet $r_\Omega = 0$ et $r_a = 0, min_{j \in \Omega}(r_j + p_j) = 24$ et $C(d_\Omega - r_\Omega) = 130, e_\Omega + c_i(\min(r_i + p_i, d_\Omega) - r_\Omega) = 96 + 44 = 140$ et par suite r_a doit être ajusté à 24.

La figure 1 est une illustration graphique des données de l'instance du tableau TABLE 2 Philippe Baptiste [3] mentionne l'existence d'un algorithme de Not-First/Not-Last cubique proposé par Win Nuijten. A notre connaissance un tel algorithme n'a pas encore été publié. Pour ce qui est de l'incomplétude de l'algorithme de Not-First/Not-Last proposé dans [4], considérons l'instance de CuSP donné par le tableau TABLE 3 où deux tâches s'exécutent sur une ressource unaire.

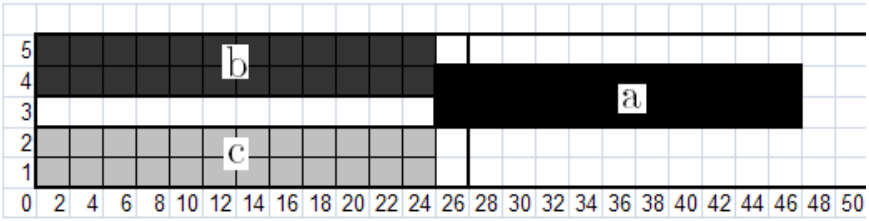


FIGURE 1 – illustration graphique des données du tableau 2

Tâche	r	d	p	c
a	0	10	6	1
b	3	10	3	1

TABLE 3 – An E-feasible instance of CuSP

Considerons l'ensemble de tâches $\Omega = \{a\}$. La condition $\gamma(\Omega, b)$ est vérifiée car $r_b < \min_{j\in\Omega} r_j + p_j$ et $e_\Omega + c_b(\min(r_b + p_b, d_\Omega) - r_\Omega) = 12$ alors que $C(d_\Omega - r_\Omega) = 10$. Ce qui implique l'ajustement suivant $LB(b) \geq max(r_a + p_a, r_b) = 6$. Cet ajustement n'est pas détecté par l'algorithme CompLB4 de [4] car la condition de la ligne 14 est toujours vérifiée.

3 Algorithme du Not-First Cumulatif

Dans cette section nous présentons notre algorithme du Not-First/Not-Last cumulatif de complexité $O(n^3)$. Dans toute la suite X désigne le vecteur de tâches rangées dans l'ordre croissant des dates de début au plus tôt. Ainsi dans X si $x < x'$ alors $r_{X(x)} \leq r_{X(x')}$. $X'(i)$ désigne la position de la tâche i dans le vecteur X. La construction du vecteur X se fait en $O(nlog(n))$.

Pour un indice donné $x \in X$, une tâche donnée j et une date de fin au plus tôt r_z+p_z, $\Omega(x, r_z+p_z, j)$ désigne l'ensemble des tâches k vérifiant $(r_{X(x)} \leq r_k) \wedge (r_z+p_z \leq r_k+p_k) \wedge (d_k \leq d_j)$. $\Omega(i, x, r_z+p_z, j)$ désigne $\Omega(x, r_z + p_z, j) - \{i\}$. Avec cette notation, la Proposition 2 montre que pour appliquer la règle de Not-First à une instance de CuSP, on doit considérer ce type d'intervalle de tâches.

Proposition 2 *Soient i une tâche et Ω un ensemble de tâches d'une instance E-faisable de CuSP tels que $i \notin \Omega$. Si la règle not-first appliquée à la paire (Ω, i) permet d'ajuster la date de début au plus tôt de la tâche i à $r_z + p_z$ alors il existe un indice $x \in X$, une tâche j avec $x \leq X'(z)$ telle que la paire $(\Omega(i, x, r_z + p_z, j), i)$ permette d'ajuster la date de début au plus tôt de la tâche i à $r_z + p_z$.*

Preuve : Soient j une tâche de Ω vérifiant $d_\Omega = d_j$. Pour $x = min\{X'(s) | r_\Omega = r_s\}$, l'intervalle de tâches Ω est inclus dans $\Omega(i, x, r_z + p_z, j)$. Comme $e_\Omega \leq e_{\Omega(i,x,r_z+p_z,j)}$, il vient que la condition $\gamma(\Omega(i, x, r_z + p_z, j), i)$ est satisfaite et la paire $(\Omega(i, x, r_z + p_z, j), i)$ permette d'ajuster la date de début de la tâche i à $r_z + p_z$. □

Soient j une tâche donnée et $ECT \in H$ une date de fin au plus tôt donnée. D'après la Proposition 2, pour calculer les $LB(i)$ pour une instance donnée de CuSP E-faisable, il suffit de considérer tous les $\Omega(i, x, ECT, j)$ où x est un indice. L'algorithme correspondant obtenu en utilisant les $\Omega(i, x, ECT, j)$ est en $O(hn^3)$ ce qui demeure très couteux. Pour améliorer, nous changeons la forme des intervalles de tâches sans perdre la complétude de l'algorithme.

Soient i et j deux tâches, $x \in X$ un indice et ECT une date de fin au plus tôt, au lieu d'utiliser les intervalles de tâches de la forme $\Omega(i, x, ECT, j)$, nous utilisons l'intervalle de tâche $\Phi(i, x, j)$ qui

est l'ensemble des tâches k vérifiant $i \notin \Phi(i,x,j) \wedge r_{X(x)} \leq r_k \wedge r_i < r_k + p_k \wedge d_k \leq d_j$. Cette famille d'intervalles peut être utilisée pour détecter la condition de Not-First et ajuster itérativement les fenêtres des tâches comme le montre la Proposition 3.

Proposition 3 *Soient i et j deux tâches, $x \in X$ un indice et ECT une date de fin au plus tôt. Si la condition $\gamma(\Omega(i,x,ECT,j),i)$ est satisfaite, alors $\gamma(\Phi(i,x,j),i)$ l'est aussi et $ECT \geq \min_{l \in \Phi(i,x,j)} r_l + p_l$*

Preuve : Par définition de $\Phi(i,x,j)$, on a $r_i \leq min_{l \in \Phi(i,x,j)} r_l + p_l$ et comme $\Omega(i,x,ECT,j) \subseteq \Phi(i,x,j)$, il s'en suit que la condition $\gamma(\Phi(i,x,j),i)$ est vérifiée et que $ECT \geq min_{l \in \Phi(i,x,j)} r_l + p_l$. □

D'après la Proposition 3, les intervalles de tâches $\Phi(i,x,j)$ détectent toutes les conditions de la règle de Not-First pour une instance E-faisable de CuSP, mais parfois permettront d'effectuer un faible ajustement. Cette procédure peut être utilisée itérativement jusqu'au point fixe. Cette approche n'est pas nouvelle, elle a déjà utilisée précédemment dans le cas du Not-First/Not-Last disjonctif [10]. On obtient ainsi l'Algorithme 1. Avec cette procédure, il n'est pas nécessaire d'obtenir le meilleur filtrage au premier passage car ce filtrage peut être amélioré aux autres passages. C'est pour cette raison qu'on parle d'algorithme itératif. La faible complexité $O(n^3)$ de l'algorithme 1 fait de lui un bon candidat si on s'en tient aux résultats expérimentaux.

Algorithm 1 *CompLB* : Algorithme de Not-First

Require: X vecteur de tâches triées par ordre croissant des dates de début au plus tôt ;
Ensure: Les $LB(i)$ sont calculés $\forall i \in T$.

```
1  for i := 1 to n do
2    LB(i) := r_i
   end for
   for all i ∈ T do
     for all j ∈ T do
       W := 0, minEct := +∞
       for x := n to 1 do
         if (r_i < r_{X(x)} + p_{X(x)}) ∧ (d_{X(x)} ≤ d_j) ∧ (X(x) ≠ i) then
           W := W + e_{X(x)}
           minEct := min(minEct, r_{X(x)} + p_{X(x)})
         end if
         if (W + c_i(min(r_i + p_i, d_j) − r_{X(x)})) > C(d_j − r_{X(x)})) then
           LB(i) := max(LB(i), minEct)
           break
         end if
       end for
     end for
   end for
```

Dans cet algorithme :
– La boucle 1-3 permet d'initialiser le vecteur LB
– Les lignes 7,8,9,10 permettent de calculer l'énergie et la date de fin au plus tôt des $\Phi(i,x,j)$.
– La ligne 12 teste la condition (3) de la Proposition 1.
– A la ligne 13, la date de debut au plus tôt de la tâche i est mise à jour.
– A la ligne 14 le **break** permet de passer à l'ajustement de la prochaine tâche car tout autre détection conduira à un faible ajustement.

4 Résultats Expérimentaux

Afin de confirmer les résultats théoriques présentés plus haut, nous avons fait une comparaison expérimentale de notre algorithme avec celui de Schutt et al [1]. Chacun de ces algoritmes a été appliqué à des instances de RCPSP pour Resources Constraint Project Schedule Problems. Une instance de RCPSP est définie par la donnée de :

- Un ensemble fini de ressources toutes de capacités limitées
- Un ensemble fini de tâches non préemtives toutes de durée détermnées
- Chaque tâche nessecite une consommation constante sur chacune des ressources

L'enjeu dans RCPSP est de trouver un ordonnancement minimal tout en respectant l'ensemble des contraintes. Pour l'implémentation, nous avons utilisé le système de contraintes CHOCO [11] qui est une librairie Java pour la resolution de problèmes combinatoires. Notre solveur a été configuré de la manière suivante :

- A chaque noeud, nous utilisons la stratégie de branchement par défaut de CHOCO i.e., le **MinDomain**, il s'agit de selectioner la variable qui a le plus petit domaine et l'attribuer la plus petite valeur de son domaine.
- L'échec est constaté lorsque le domaine de l'une des variables se vide, dans ce cas on effectue un backtrack chronologique.
- Le solveur minimise une variable qui représente la date de fin du projet.
- Le solveur ne s'arrête pas à la première solution mais cherche à améliorer cette solution.
- Le solveur s'arrête après une heure de recherche infructueuse.

Pour cette comparaison, deux versions ont été implementées sur un intel(R)Core(TM) 2 Duo, 1.8Ghz, 2Go de RAM tournant sur Windows XP.

- La première version utilise l'algorithme de du Not-First/Not-Last en $O(n^3 \log(n))$ de Schutt et al [1].
- La deuxième version utilise l'algorithme en $O(n^3)$ décrit dans cet article.

Ces deux versions ont été évaluées sur les instances BL [12] qui est un ensemble de quarante instances de RCPSP variant entre 25 et 20 tâches s'exécutant sur 3 ressources. D'après Baptiste et al [12], parmi les instances de RCPSP, on distingue les instances fortement disjonctives des instances fortement cumulatives. Une instance de RCPSP est dite fortement disjonctif lorsque plusieurs pairs de tâches ne peuvent pas s'exécuter en parallèle sur une même ressource, il est dit fortement cumulatif si plusieurs tâches peuvent s'exécuter en parallèle sur une même ressource. Pour formaliser cette notion, ils ont introduit la notion de ratio de disjonction qui est le rapport entre le nombre de pairs de tâches ne pouvant pas s'exécuter en parallèle sur le nombre total de pairs de tâches. La moyenne des ratios pour les instances de BL est de 0.33 et de ce fait, ces instances sont qualifiées de fortement cumulatives. Le tableau TABLE 4 suivant donne pour chaque version le Makespan Mks, le Nombre de Bactracks Bks, le temps moyen CPU en seconde CPU et le nombre d'appels du Not-First/Not-Last au cours de la recherche du Makespan $NbrIt$.

Vingt Quatre instances ont été résolues par les deux versions. Les deux versions ont le même makespan et le même nombre de backtracks, preuve que notre algorithme atteint le même point fixe que la règle. Le temps moyen d'exécution d'une instance par la $version_1$ est de $124,401$ sec alors qu'il est de $62,546$ sec pour la $version_2$. La $version_2$ est donc deux fois plus rapide que la $version_1$. Le nombre moyen d'appels du Not-First/Not-Last par instance dans la $version_2$ est de 289 alors qu'il est de 278 dans la $version_1$. Le Not-First/Not-Last que nous proposons dans ce papier est donc appelé $1,04$ fois plus celui de la littérature [1].

Consistency enforcing algorithms applied								
Instance	$version_1$				$version_2$			
Name	Mksp	Bks	CPU	NbrIt	Mksp	Bks	CPU	NbrIt
bl2001	16	24	5.844	25	16	24	2.844	23
bl2002	16	679	33.718	934	16	679	18.000	937
bl2003	15	2483	163.453	273	15	2483	87.625	383
bl2004	18	0	1.406	4	18	0	0.750	4
bl2006	16	2	1.984	0	16	2	1.016	0
bl2008	20	1181	43.781	0	20	1181	22.343	0
bl2009	14	107	10.781	0	14	107	5.313	0
bl2010	16	39	4.250	111	16	39	2.157	101
bl2011	17	3	1.766	0	17	3	0.859	0
bl2013	17	914	62.859	698	17	914	32.969	703
bl2014	18	24	3.891	0	18	24	1.766	0
bl2015	13	808	37.359	48	13	808	19.125	48
bl2019	14	4	2.328	0	14	4	1.204	0
bl2020	15	1013	49.516	890	15	1013	26.468	1001
bl2504	22	1108	110.234	1478	22	1108	55.172	1506
bl2506	33	124	13.750	179	33	124	7.297	175
bl2509	27	1469	200.625	661	27	1469	104.578	675
bl2510	23	28	6.531	0	23	28	3.172	0
bl2511	24	3260	209.313	0	24	3260	101.000	0
bl2514	30	4291	378.874	995	30	4291	197.593	1010
bl2515	20	577	99.594	6	20	577	53.203	6
bl2516	20	357	52.938	0	20	357	26.015	0
bl2517	26	10325	1057.657	255	26	10325	507.782	254
bl2519	23	2588	433.188	115	23	2588	222.859	114

TABLE 4 – Resultats expérimentaux

5 Conclusion et Persperctives

Dans cet article, nous proposons un nouvel algorithme de complexité $O(n^3)$ pour la règle Not-First/Not-Last utilisée pour la propagation de la contrainte de ressource en ordonnancement cumulatif. Notre algorithme améliore ainsi la complexité de la meilleur implémentation de cette règle proposée par Schutt et al [1] du facteur $\log(n)$. Les résultats expérimentaux ont été effectués pour confirmer le résultat théorique. Malgré le fait que notre algorithme est un peu plus appelé que celui de la littérature, il est deux fois plus rapide que celui-ci. Malgré ces efforts, beaucoup reste à faire car cette complexité reste encore très élevée comparé à la faible complexité de l'edge-finding [8, 7]. Nous nous proposons d'adapter la structure de données "cumulative Θ-tree" introduire par Petr Vilim dans [9] a fin de réduire la complexité de cet algorithme à $O(n^2 \log(n))$.

Références

[1] A. Schutt, A. Wolf and G. Schrader. Not-First and Not-last Detection for Cumulative Scheduling in $O(n^3log(n))$. Proceedings 16th International Conference on Applications of Declarative Programming and Knowledge, INAP 2005, 66-80, Fukuoka, Japan. (2005).

[2] Petr Vilim. Global Constraints in Scheduling. PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, Department of Theoretical Computer Science and Mathematical Logic, 2007.

[3] Ph. Baptiste. Résultats de Complexité et Progammation par Contraintes pour l'Ordonnancement. HDR thesis. Compiègne University of Technology. (2002).

[4] R. Kameugne and L.P. Fotso. Not-First/Not-Last Algorithm For Cumulative Scheduling Problems. In Proceedings of the join International Conference FRANCORO/ROADEF 149-155, (2007).

[5] W. Nuijten. Time and resource constrained Scheduling : A constraint Satisfaction Approach. PhD thesis, Eindhoven University of Technology. (1994).

[6] Y. Caseau and F. Laburthe. Improved CLP Scheduling with Task Intervals. In Proceedings of the Eleventh International Conference on Logic Programming, Santa Margherita Ligure, Italy. (1994).

[7] Petr Vilim. Edge Finding Filtering Algorithm for Discrete Cumulative Resources in $O(kn \log n)$. In : Ian P. Gent (ed) CP 2009. LNCS 5732. pp. 802-816. Springer, Heidelberg (2009).

[8] L. Mercier and Pascal Van Hentenryck. Edge Finding for Cumulative Scheduling. INFORMS Journal on Computing 20(1), pp. 143-153, (2008).

[9] Petr Vilim. Max energy filtering algorithm for discrete cumulative resources. In Acceptted to CP-AI-OR 2009, 2009.

[10] P. Torres and P. Lopez. On Not-First/Not-Last Conditions in Disjunctive Scheduling. European Journal of Operational Research, Vol 127(2) :332-343. (2000).

[11] F. Laburthe. CHOCO : implementing a CP kernel. In Proc. of TRICS 2000, Workshop on techniques for implementing Constraint Programming systems, Singapore, (2000).

[12] Ph. Baptiste. C. Le Pape. & W. Nuijten. Satisfiability Test and Time-Bound Adjustements for Cumulative Scheduling Problems, Annals of Operations Research 92. 305-333. (1999).

Une Méthode Tabou pour l'Optimisation des VPN

M. Z. Ben Hamouda[1,2], O. Brun[1], J. M. Garcia[2]

[1] CNRS ; LAAS ; 7 avenue du Colonel Roche, F-31077 Toulouse, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France
{zied.ben-hamouda,brun}@laas.fr
[2] QoS Design, 6, Avenue Marcel Doret, F-31500 Toulouse, France
jmgarcia@qosdesign.com

Résumé : *Nous proposons dans cet article une heuristique à base de recherche tabou pour résoudre le problème de planification des VPN spécifiés suivant le modèle Hose avec un monoroutage. L'originalité de cette méthode est d'utiliser une borne supérieure sur la bande-passante à réserver pour explorer le voisinage de la solution courante. Une série de résultats expérimentaux démontrent l'efficacité de cette heuristique.*

Mots-Clés : *VPN, Modèle Hose, Recherche tabou, Optimisation Robuste.*

1 Introduction

Les opérateurs de télécommunications proposent, depuis quelques années, des services VPN (Virtual Private Network). Les entreprises souscrivant à ces offres peuvent ainsi disposer d'une infrastructure privée et virtuelle de communication longue distance. Le client VPN peut ainsi connecter différents sites répartis géographiquement sans avoir à investir dans des liaisons louées et sans avoir à supporter les coûts liés à la gestion et à la maintenance d'une telle architecture.

L'architecture d'un VPN est décrite sur la figure 1. Pour l'entreprise souscrivant l'offre VPN, les noeuds du VPN apparaissent comme étant interconnectés suivant une topologie complètement maillée grâce à des liens virtuels. En fait, les noeuds VPN sont directement reliés à des routeurs IP du réseau opérateur, et les liens virtuels correspondent à des chemins entre ces routeurs dans le réseau IP.

Du point de vue de l'entreprise cliente, le VPN doit donc apparaître comme un réseau privé, offrant le même niveau de qualité de service qu'un réseau constitué de lignes louées, mais à un prix très inférieur. Pour cela, il est essentiel que l'opérateur garantisse la bande-passante allouée aux liens virtuels du VPN. Ceci suppose tout d'abord de spécifier la bande-passante nécessaire entre les différents noeuds du VPN, pour ensuite la réserver dans le réseau opérateur le long des chemins correspondant aux liens virtuels. La réservation peut être réalisée par l'opérateur en utilisant des technologies IP telles que MPLS ou RSVP.

Deux approches ont été proposées pour permettre à un client VPN de spécifier la capacité de communication dont il a besoin. La première, connu sous le nom de modèle "Pipe" [5] est illustrée sur la figure 2(a). Elle suppose que le client VPN décrive la demande maximale entre chaque paire

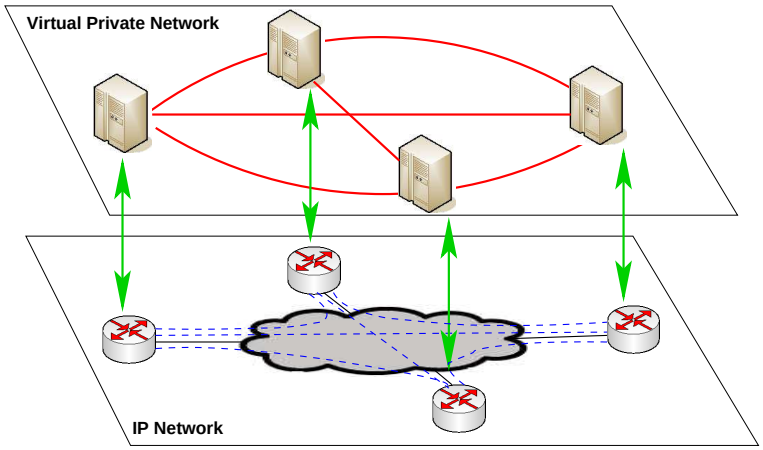
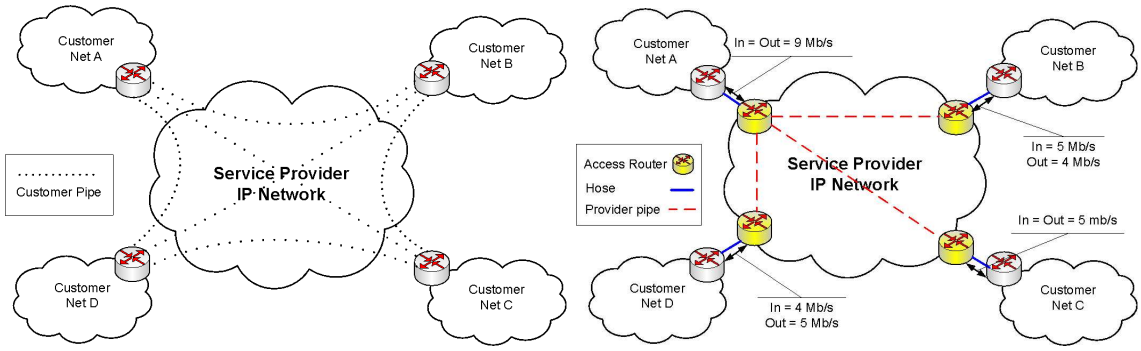


FIGURE 1 – Architecture d'un réseau privé virtuel.

de noeuds du VPN. S'il veut interconnecter N noeuds, cela suppose donc qu'il fournisse $N\,(N-1)$ valeurs de capacité, ce qui peut être fastidieux quand le VPN relie plusieurs dizaines de noeuds. Surtout, le modèle "Pipe" a l'inconvénient majeur d'exiger du client VPN une connaissance précise de la demande entre ses différents sites, ce qui n'est généralement pas le cas.



(a) Modèle Pipe (b) Modèle Hose

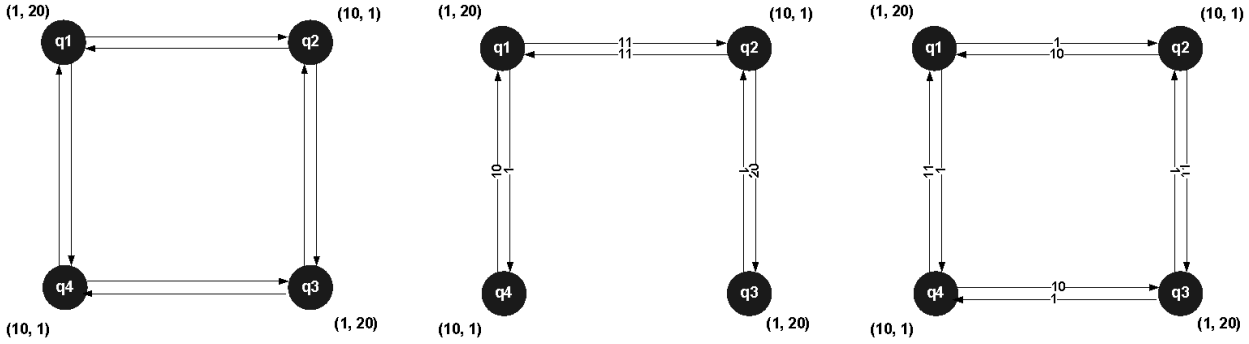
FIGURE 2 – Modèles Pipe et Hose de spécification de la bande-passante d'un VPN.

Le modèle "Hose" (cf. figure 2(b)), introduit par Duffield et al. en 1999 [12], est beaucoup plus simple et plus flexible pour le client VPN. Celui-ci n'a qu'à spécifier une bande-passante en entrée et une autre en sortie pour chaque noeud. Ainsi, avec N noeuds dans le VPN, il suffit de spécifier $2\,N$ valeurs. La bande-passante de sortie d'un noeud VPN représente le débit maximal auquel il pourra émettre vers les autres, tandis que sa bande-passante d'entrée représente le débit maximal auquel il pourra recevoir des données en provenance des autres noeuds. Le modèle "Hose" est de loin le plus utilisé actuellement, et dans la suite de cet article, nous supposerons que la demande en bande-passante du VPN est décrite suivant ce modèle.

Pour l'opérateur, la mise en place d'un VPN dédié à un client exige la réservation des ressources nécessaires dans son réseau, le long de chemins reliant les différents sites du client. Il est évidemment essentiel pour l'opérateur de minimiser la capacité totale réservée pour chaque VPN, afin d'utiliser au mieux les ressources disponibles. Cette minimisation de la bande-passante réservée suppose de déterminer le routage optimal des liens virtuels dans le réseau opérateur, c'est à dire d'affecter un

chemin dans le réseau à chaque lien virtuel de manière à minimiser la bande-passante totale reservée.

Ce problème est illustré sur la figure 3. Dans cette figure, nous présentons dans 3(b) et 3(c) les capacités reservées pour deux plans de routage différents du problème représenté en 3(a). Chaque noeud est étiqueté par une paire (in, out) indiquant les bandes passantes en entrée et en sortie sur ce noeud. La capacité de chaque lien physique du réseau est supposée infinie. Les valeurs indiquées sur les liens dans les figures 3(b) et 3(c) correspondent à la bande passante à réserver sur chacun de ces liens. Nous n'avons pas représenté les liens pour lesquels la réservation est nulle. Dans la solution représentée sur la figure 3(b), la capacité totale reversée est de 54 ; les trafics sont routés selon les chemins de l'arbre représenté sur cette figure. Dans la solution représentée sur la figure 3(c), la capacité totale réservée est de 46 ; le plan de routage est défini de la manière suivante : les trafics sont routés selon les plus courts chemins ; s'il existe deux plus courts chemins entre deux noeuds, le trafic est routé suivant le sens des aiguilles d'une montre.



(a) Réseau sous-jacent (b) Solution 1 (coût = 54) (c) Solution 2 (coût = 46)

FIGURE 3 – Exemple d'instances de notre problème

Le problème tel que nous venons de le présenter correspond au problème d'optimisation des VPN avec contraintes de monoroutage. Ce problème a été considéré par Gupta et al. [3, 4] qui, dans le cas de demandes en bande passante d'entrée-sortie symmétriques et de capacitée de liens finies, ont montré que la détermination d'une solution admissible est déjà un problème NP-difficile. Ils ont également présenté une 5.55-approximation. Dans le cas de demandes en bandes-passantes entrée/sortie asymmétriques et de capacitée de liens finies, Altin et al. ont proposé un algorithme de type branch-and-price [1]. D'autre auteurs ont considéré ce problème sous d'autres contraintes de routage, et notamment dans le cas d'un routage arborescent [6, 3, 8] et dans celui d'un routage multi-chemin par partage de charge [13].

Dans cet article, nous étudions le problème de la conception des VPN Hose avec un plan de routage en mono-chemin. C'est le cas des VPNs MPLS qui sont de nos jours largement répandus. La difficulté principale est le fait que spécifier les demandes suivant le modèle Hose revient en fait à spécifier un polytope de matrices de trafic possibles. Le plan de routage et la réservation de bande-passante établis par l'opérateur doivent permettre d'acheminer le trafic de n'importe quelle matrice du polytope, sans que le trafic sur un lien n'excède la bande-passante reservée. Il s'agit donc d'un problème de routage robuste, l'ensemble d'incertitude étant donné par le polytope des matrices de trafic valides.

Ce problème étant NP-difficile, nous proposons une approximation basée sur une méthode tabou. L'originalité de cette méthode est d'utiliser une borne supérieure sur la bande-passante à réserver pour explorer le voisinage de la solution courante.

Cet article est organisé de la façon suivante. Dans la partie 2, nous introduisons nos principales notations et formulons le problème de routage sous la forme d'un programme linéaire mixte. Ce problème étant NP-difficile, nous proposons dans la partie 3 une approximation basée sur la recherche tabou. Nous évaluons les résultats obtenus avec cette approximation dans la partie 4 avant de conclure et de présenter quelques perspectives à ce travail dans la partie 5.

2 Formulation du Problème

Le réseau opérateur sous-jacent sur lequel un VPN doit être provisionné est représenté par un graphe orienté $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Les noeuds du graphe correspondent aux routeurs et les arcs aux liens de communication entre les routeurs. L'ensemble des arcs issus du noeud $v \in \mathcal{V}$ sera noté $\delta^+(v)$, tandis que $\delta^-(v)$ représentera l'ensemble des arcs dont v est la destination. On notera C_e la capacité de transmission du lien $e \in \mathcal{E}$. On notera c_e le coût induit par la réservation d'une unité de bande-passante sur ce lien.

Soit $\mathcal{Q} \subseteq \mathcal{V}$ l'ensemble des noeuds faisant partie du VPN. On parlera dans la suite de noeuds terminaux. La demande en bande-passante du VPN est décrite suivant le modèle Hose, ce qui signifie que pour chaque noeud terminal $v \in \mathcal{Q}$, on connaît le débit maximal en émission $b^+(v)$ et le débit maximal en réception $b^-(v)$. Dans la suite, nous ferons l'hypothèse suivante de conservation du trafic : $\sum_{v\in\mathcal{Q}} b_v^+ = \sum_{v\in\mathcal{Q}} b_v^-$. Une matrice de trafic $\mathbf{D} = (d_{s,t})_{s,t\in\mathcal{Q}}$ est valide si et seulement si elle vérifie les deux inéquations suivantes :

$$\sum_{t\in\mathcal{Q}} d_{s,t} \leq b^+(s) \quad \text{et} \quad \sum_{t\in\mathcal{Q}} d_{t,s} \leq b^-(s).$$

Nous noterons par la suite $\mathcal{D}$ le polytope des matrices de trafic valides.

L'opérateur doit déterminer un plan de routage (en mono-chemin) pour le VPN, c'est à dire affecter un et un seul chemin dans le graphe $\mathcal{G}$ à chaque couple de noeuds terminaux u, v de $\mathcal{Q}$. Soit $f_{u,v}^e$ la fraction du trafic allant de u vers v passant sur le lien $e \in \mathcal{E}$. On peut définir un plan de routage comme un flot dans le réseau $\mathcal{G}$, c'est à dire comme un vecteur $\mathbf{f} = \left(f_{u,v}^e\right)_{u,v\in\mathcal{Q},e\in\mathcal{E}}$ vérifiant les contraintes suivantes :

$$(1)\begin{cases} \sum_{e\in\delta^+(v)} f_{s,t}^e - \sum_{e\in\delta^-(v)} f_{s,t}^e = 0, & s,t \in \mathcal{Q}, v \neq s,t \quad (1a) \\ \sum_{e\in\delta^+(v)} f_{s,t}^e - \sum_{e\in\delta^-(v)} f_{s,t}^e = 1, & s,t \in \mathcal{Q}, v = s \quad (1b) \\ \sum_{e\in\delta^+(v)} f_{s,t}^e - \sum_{e\in\delta^-(v)} f_{s,t}^e = -1, & s,t \in \mathcal{Q}, v = t \quad (1c) \\ f_{s,t}^e \in \{0,1\}, & e \in \mathcal{E}, s,t \in \mathcal{Q} \quad (1d) \end{cases}$$

Dans la suite, on notera $\mathcal{F}$ l'ensemble des plans de routage possibles, c'est à dire l'ensemble des vecteurs $\mathbf{f}$ vérifiant les contraintes ci-dessus.

Notons x_e la quantité de bande passante réservée par l'opérateur sur le lien $e \in \mathcal{E}$ et $\mathbf{x} = (x_e)_{e\in\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|}$ le vecteur de réservation de bande-passante. L'objectif de l'opérateur est de déterminer un plan

de routage minimisant le coût total $\sum_{e\in\mathcal{E}} c_e x_e$ de la bande-passante réservée, tout en garantissant que la bande-passante réservée sur chaque lien sera suffisante quelle que soit la matrice de trafic $D \in \mathcal{D}$. Le problème peut être formulé sous la forme d'un programme linéaire mixte :

$$(2)\begin{cases} \min_{\mathbf{x},\mathbf{f}} & \sum_{e\in E} c_e x_e & & \\ \text{s.t.} & & & \\ & \sum_{u,v\in Q} f^e_{u,v}\, d_{u,v} \le x_e, & D\in\mathcal{D}, e\in\mathcal{E} & \text{(2a)} \\ & 0 \le x_e \le C_e, & e\in\mathcal{E} & \text{(2b)} \\ & \mathbf{x} \in \mathbb{R}^{|\mathcal{E}|}, & & \text{(2c)} \\ & f \in \mathcal{F}, & & \text{(2d)} \end{cases}$$

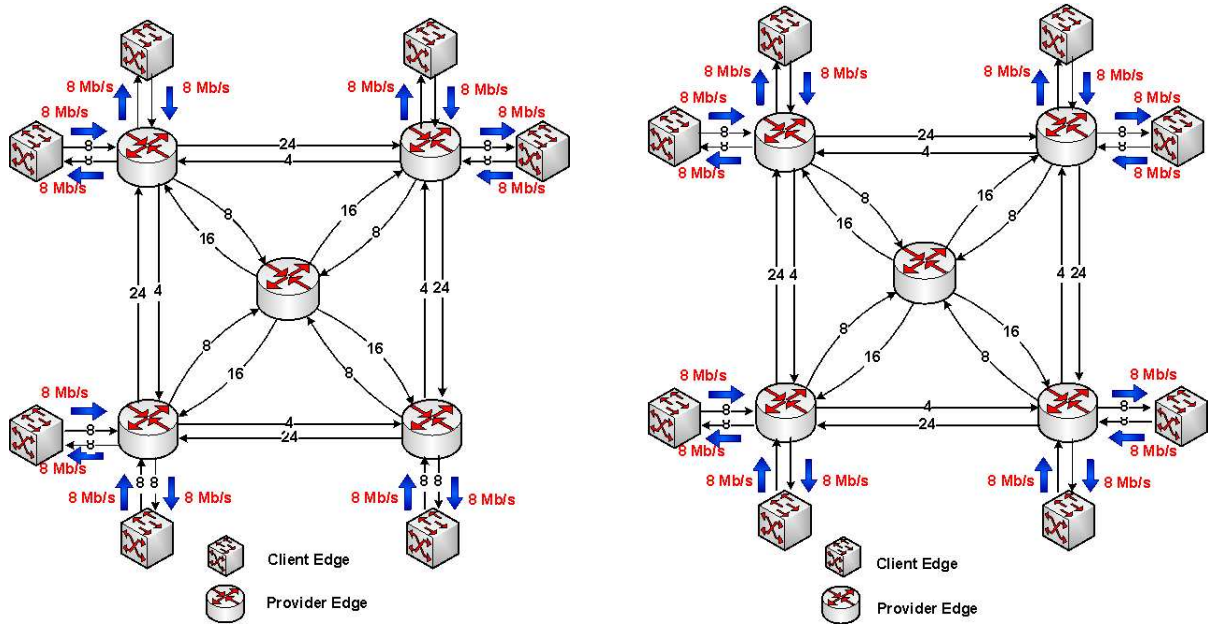
Les contraintes (2a) garantissent que le trafic sur chaque lien $e \in \mathcal{E}$ ne dépasse pas la capacité x_e réservée sur ce lien, et ce quelle que soit la matrice de trafic $D \in \mathcal{D}$. Les contraintes (2b) sont des contraintes de capacité. La formulation ci-dessus correspond à un problème d'optimisation qui comporte un nombre infini de contraintes puisque qu'il y a un nombre infini de matrices $\mathbf{D}$ dans $\mathcal{D}$. De manière classique, on peut contourner cette difficulté en dualisant la contrainte (2a), comme il a été fait dans [1, 7, 9]. La formulation du problème devient ainsi :

$$(3)\begin{cases} \min_{\mathbf{x},\mathbf{f}} & \sum_{e\in E} c_e x_e & & \\ \text{s.t.} & & & \\ & \sum_{v\in Q} b^+(v)\ .\ \pi_e(v) + b^-(v)\ .\ \lambda_e(v) \le x_e, & e\in\mathcal{E} & \text{(3a)} \\ & f^e_{s,t} \le \pi_e(s) + \lambda_e(t), & s,t\in Q, e\in\mathcal{E} & \text{(3b)} \\ & 0 \le x_e \le C_e, & e\in\mathcal{E} & \text{(3c)} \\ & \pi_e(u), \lambda_e(u) \ge 0, & u\in Q, e\in\mathcal{E} & \text{(3d)} \\ & \mathbf{x} \in \mathbb{R}^{|\mathcal{E}|} & & \text{(3e)} \\ & f \in \mathcal{F}, & & \text{(3f)} \end{cases}$$

Le problème ci-dessus peut être résolu en temps polynomial si l'on relaxe la contrainte (1d) en $0 \le f^e_{s,t} \le 1$ [13], autrement dit si on renonce au mono-routage pour accepter un partage de charge sur plusieurs chemins. Avec la contrainte de mono-routage, ce problème est connu pour être NP-difficile [3].

Non seulement les temps calcul augmentent exponentiellement avec la taille du problème, mais ils peuvent déjà être considérables pour des problèmes de petite taille comme l'illustre la figure 4. Dans cette figure, nous considérons deux réseaux de petites tailles. Le réseau représenté dans la figure 4(a) contient 7 noeuds terminaux (ou clients), alors que celui de la figure 4(b) contient 8 noeuds terminaux. Nous supposons dans les deux exemples que chaque terminal envoie et reçoit au maximum 8 Mb/s. On indique sur chaque lien physique sa capacité en Mb/s. Pour résoudre le problème (3), nous avons utilisé le solveur glpk disponible sur Internet [15]. Nous indiquons pour chaque exemple le temps de calcul mis par glpk pour trouver la solution optimale.

Il est toutefois interessant de noter qu'empiriquement on constate que dans de nombreux cas la solution du problème relaxé est une solution vérifiant la contrainte de monoroutage [1, 10]. Cela suggère d'utiliser la solution du problème relaxé comme point de départ pour rechercher la solution optimale du problème de monoroutage.



(a) Temps requis par glpk = 502 *sec*

(b) Temps requis par glpk = 9 523 *sec*

FIGURE 4 – Exemple d'instances générant des temps de calcul considérables.

3 Approximation par une méthode tabou

Pour résoudre le problème précédent, nous proposons d'utiliser un algorithme de recherche tabou.

3.1 Reformulation du problème

Pour décrire notre algorithme, il est utile de ré-écrire le problème suivant une formulation noeud-chemin. Notons $\Pi_{u,v}$ l'ensemble de tous les chemins entre les noeuds $u, v \in \mathcal{Q}$ dans $\mathcal{G}$ et $\Pi = \cup_{u,v \in \mathcal{Q}} \Pi_{u,v}$. Pour chaque chemin $\pi \in \Pi$, on notera $s(\pi)$ le noeud source et $t(\pi)$ le noeud destination. Introduisons les variables de décision suivantes :

$$f_\pi = \begin{cases} 1 & \text{si le trafic entre } s(\pi) \text{ et } t(\pi) \text{ est routé sur } \pi, \\ 0 & \text{sinon.} \end{cases}$$

Un plan de routage peut alors être défini de manière équivalente comme un vecteur $\mathbf{f} \in \mathcal{F}_\pi$, où,

$$\mathcal{F}_\pi = \left\{ \mathbf{f} \in \{0,1\}^{|\mathcal{Q}|\,(|\mathcal{Q}|-1)} \;:\; \sum_{u,v \in \mathcal{Q}} f_\pi = 1 \quad \forall u, v \in \mathcal{Q} \right\}$$

On remarquera que l'on obtient facilement la définition précédente d'un plan de routage avec la formule suivante :

$$f^e_{u,v} = \sum_{\pi \in \Pi_{u,v}} \gamma^e_\pi \, f_\pi \quad e \in \mathcal{E}, u, v \in \mathcal{Q}$$

où γ^e_π est égal à 1 si le chemin π passe par l'arc e, et 0 sinon.

Nous définissons une solution du problème comme un vecteur $\mathbf{f} \in \mathcal{F}_\pi$. Une solution est dite admissible si elle est telle que $m_e(\mathbf{f}) \leq C_e$ pour tout $e \in \mathcal{E}$, où $m_e(\mathbf{f})$ est la solution du programme linéaire suivant :

$$(4) \begin{cases} m_e(\mathbf{f}) = \max_{\mathbf{D}} & \sum_{\pi \in \Pi} \gamma^e_\pi \, f_\pi \; . \; d_{s(\pi),t(\pi)} & & \\ \text{s.t.} & & & \\ & \sum_{t \in Q} d_{s,t} \leq b^+(s), & s \in Q & \text{(4a)} \\ & \sum_{t \in Q} d_{t,s} \leq b^-(s), & s \in Q & \text{(4b)} \\ & d_{s,s} = 0, & s \in Q & \text{(4c)} \\ & d_{s,t} \geq 0, & s,t \in Q, s \neq t & \text{(4d)} \end{cases}$$

En fait, $m_e(\mathbf{f})$ représente le trafic maximal qui peut passer sur le lien e, c'est à dire la capacité à réserver. Le problème à résoudre s'écrit alors de la façon suivante :

$$(5) \begin{cases} \min_{\mathbf{f} \in \mathcal{F}_\pi} & \sum_{e \in \mathcal{E}} c_e \, m_e(\mathbf{f}) & & \\ \text{s.t.} & & & \\ & m_e(\mathbf{f}) \leq C_e, & e \in \mathcal{E} & \text{(5a)} \end{cases}$$

Il faut bien entendu noter que ce problème d'optimisation biniveau ne correspond au problème

original (2) que si l'ensemble Π contient *t*ous les chemins possibles entre chaque paire origine-destination. Ceci n'étant pas possible pour des graphes de grande dimension, on est amené à ne considèrer que K chemins pour chaque couple origine-destination. Ces chemins peuvent par exemple être obtenus avec un algorithme de K plus courts chemins [14]. Le nombre de solutions dans $\mathcal{F}_\pi$ est alors $K^{|\mathcal{Q}|\,(|\mathcal{Q}|-1)}$.

Le fait de limiter le nombre de chemins pour chaque couple origine-destination induit une approximation par rapport au problème de départ. Cette approximation peut toutefois être considérée comme négligeable si le nombre K de plus courts chemins considérés est suffisamment grand. En pratique, un choix de K entre 5 et 10 semble raisonnable.

Nous avons déjà signalé que la détermination d'une solution admissible pour le problème (5) est NP-difficile [3]. Cela constitue un obstacle majeur pour la mise en oeuvre d'une méthode de recherche locale, puisque la solution initiale doit satisfaire les contraintes (5a). Pour résoudre cette difficulté, nous proposons l'utilisation des méthodes de pénalité [11]. L'idée est de s'affranchir des contraintes (5a) pour pouvoir démarrer d'une solution arbitraire dans $\mathcal{F}_\pi$, pour ensuite guider la recherche vers une zone de l'espace des solutions contenant des solutions admissibles. Le problème que nous allons chercher à résoudre dans la suite, s'écrit de la façon suivante :

$$(6) \quad \min_{\mathbf{f}\in\mathcal{F}_\pi}\Phi(\mathbf{f}) = \sum_{e\in\mathcal{E}} c_e\, m_e(\mathbf{f}) \;+\; A\,.\,\sum_{e\in\mathcal{E}} max\left(0, [C_e - m_e(\mathbf{f})]^2\right)$$

où $A > 0$ est un paramètre ajustable.

3.2 Algorithme

La méthode que nous proposons est basée sur la recherche tabou. Son originalité principale est d'utiliser une borne supérieure $\Psi(\mathbf{f})$ sur le coût d'une solution $\mathbf{f}$. En effet, l'évaluation du coût $\Phi(\mathbf{f})$, tel que présenté dans le problème (6), nécessite le calcul de $m_e(\mathbf{f})$ pour chaque lien $e \in \mathcal{E}$, ce qui veut dire résoudre le problème (4) pour chaque lien. Pour explorer le voisinage d'une solution, c'est à dire évaluer le coût de tous ses voisins, il faut alors résoudre le problème (4) un très grand nombre de fois, ce qui est pénalisant en temps calcul. L'idée est alors d'utiliser une borne supérieure $\Psi(\mathbf{f})$ sur le coût $\Phi(\mathbf{f})$, à la fois très simple à calculer et proche du coût réel, pour déterminer si le voisinage contient une solution améliorante. Si c'est le cas, ce voisin sera choisi comme solution courante de la prochaine itération. Sinon, on refait une exploration du voisinage en utilisant cette fois le coût exact pour pouvoir garantir que le voisinage ne contient pas de solution améliorante. Les détails de l'algorithme sont décrits ci-dessous.

3.2.1 Solution Initiale

Nous avons déjà signalé que la solution du problème (2) quand on relaxe la contrainte $f^e_{s,t} \in \{0, 1\}$ en $0 \leq f^e_{s,t} \leq 1$ est souvent une solution entière vérifiant la contrainte de monoroutage. Ce problème peut alors être résolu en temps polynomial. Il apparaît alors intéressant d'utiliser la solution du problème relaxé comme point de départ pour rechercher la solution optimale du problème de monoroutage.

Deux cas de figure peuvent se produire. Dans le premier cas, la solution du problème relaxé est une solution entière. On a dans ce cas directement obtenu la solution optimale du problème original. Dans le deuxième cas, la solution est continue, et on l'utilise alors pour construire la solution initiale $\mathbf{f}_0$ de la recherche tabou. Pour cela, on choisit pour chaque couple origine-destination (u, v) le chemin ayant le plus grand poids dans la solution continue.

3.2.2 Structure du Voisinage

Soit $\mathbf{f}$ une solution. Une solution voisine de $\mathbf{f}$ est obtenue en routant un et un seul flux (u, v) (lien virtuel) sur un chemin différent de celui emprunté par ce flux dans $\mathbf{f}$. Autrement dit, le voisinage de la solution $\mathbf{f}$ est défini par

$$\mathcal{N}(\mathbf{f}) = \{\mathbf{f}' \in \mathcal{F}_\pi \;:\; \exists!\pi \in \Pi,\; f'_\pi = 1 \quad \text{et} \quad f_\pi = 0\}$$

La taille de ce voisinage est $|\mathcal{Q}|\,(|\mathcal{Q}| - 1)\,(K - 1)$.

3.2.3 Liste Tabou

Dans notre algorithme, nous avons utilisé un mécanisme de recherche tabou pour éviter les cycles. Une fois un minimum local atteint, nous donnons un statut tabou au chemin dont le changement mène au voisin de moindre coût pour éviter d'y revenir avant un certain temps. Nous interdisons d'utiliser ce chemin comme un chemin d'Ã©change pendant un certain nombred'itérations tiré aléatoirement.

3.2.4 Evaluation exacte du coût d'une solution

Le coût d'une solution $\mathbf{f} \in \mathcal{F}_\pi$ est donné par :

$$\Phi(\mathbf{f}) = \sum_{e \in \mathcal{E}} c_e \, m_e(\mathbf{f}) \; + \; A \, . \sum_{e \in \mathcal{E}} max \left(0, [C_e - m_e(\mathbf{f})]^2\right)$$

Le calcul de ce coût suppose de déterminer pour chaque lien e la valeur $m_e(\mathbf{f})$, qui représente le trafic maximal qui peut passer sur le lien e pour l'ensemble des matrices de trafic valides du polyèdre $\mathcal{D}$.

On a vu que, pour un lien e donné, $m_e(\mathbf{f})$ est en fait le coût optimal du problème (4). Remarquons que la solution de ce problème ne nous intéresse pas, puisque nous voulons seulement connaître le coût optimal. Le programme linéaire (4) peut être efficacement résolu en utilisant une technique générique tel que l'algorithme de simplexe. Néanmoins, il faut bien observer que pour calculer le coût d'une solution il faudra résoudre cela pour chaque lien $e \in \mathcal{E}$. Par conséquent, l'exploration du voisinage d'une solution va necéssiter un total de $|\mathcal{Q}| \, (|\mathcal{Q}| - 1) \, (K - 1) \, |\mathcal{E}|$ résolutions de ce problème.

Pour accélérer la procédure de calcul de $m(e)$, nous remarquons que le problème (4) est en fait un problème de transport. Comme illustré sur la figure 5, on construit un graphe biparti en plaçant à gauche les noeuds qui envoient du trafic sur un chemin passant par le lien e et à droite les noeuds qui reçoivent du trafic sur un de ces chemins. En d'autres termes, pour chaque chemin p passant par le lien e, on place à gauche le noeud $s(p)$ et à droite le noeud $t(p)$. Il peut bien sûr arriver qu'un même terminal apparaîsse à gauche et à droite. Chaque noeud u situé à gauche a une offre de $b^+(u)$, et chaque noeud v à droite une demande égale à $b^-(v)$. Il existe dans le graphe biparti un arc entre les sommets u et v si on peut trouver un chemin p passant par e tel que $u = s(p)$ et $v = t(p)$. Le poids de cet arc est alors fixé à -1 (pour se ramener à un problème de minimisation). La résolution du problème de transport peut alors se faire en utilisant l'algorithme de Stepping Stone [2].

3.2.5 Borne Supérieure sur le coût d'une solution

Considérons les deux problèmes linéaires suivants qui sont des relaxations du problème (4) obtenue en remplaçant respectivement les contraintes (4b) et (4a) par (7b) dans le problème (7) et par (8a) dans le problème (8).

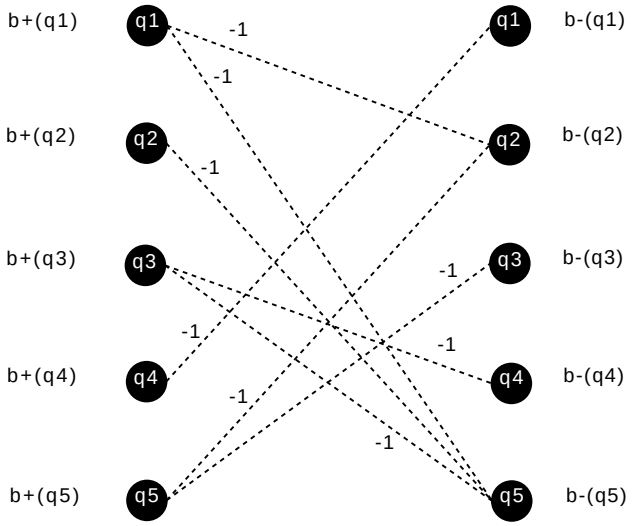


FIGURE 5 – Construction du réseau de transport relatif à l'arc $e \in E$ (seuls les arcs avec un coût non nul sont représentés).

$$
(7)\begin{cases}
\alpha_e = \max_{\mathbf{D}} & \sum_{\pi\in\Pi} \gamma_\pi^e f_\pi \ . \ d_{s(\pi),t(\pi)} & & \\
\text{s.t.} & & & \\
& \sum_{t\in Q} d_{s,t} \le b^+(s), & s \in Q & \text{(7a)}\\
& d_{t,s} \le b^-(s), & t,s \in Q & \text{(7b)}\\
& d_{s,s} = 0, & s \in Q & \text{(7c)}\\
& d_{s,t} \ge 0, & s,t \in Q, s \ne t & \text{(7d)}
\end{cases}
$$

$$
(8)\begin{cases}
\beta_e = \max_{\mathbf{D}} & \sum_{\pi\in\Pi} \gamma_\pi^e f_\pi \ . \ d_{s(\pi),t(\pi)} & & \\
\text{s.t.} & & & \\
& d_{s,t} \le b^+(s), & s,t \in Q & \text{(8a)}\\
& \sum_{t\in Q} d_{t,s} \le b^-(s), & s \in Q & \text{(8b)}\\
& d_{s,s} = 0, & s \in Q & \text{(8c)}\\
& d_{s,t} \ge 0, & s,t \in Q, s \ne t & \text{(8d)}
\end{cases}
$$

Il est clair que toute matrice $\mathbf{D} \in \mathcal{D}$ solution de (4) est aussi une solution de (7) et de (8). Il est clair aussi que les coûts optimaux α_e de (7) et β_e de (8) sont des bornes supérieures sur le coût optimal $m_e(\mathbf{f})$ de (4), c'est à dire sur la capacité à réserver sur le lien e pour la stratégie de routage $\mathbf{f}$. Dans ce qui suit, nous proposons un algorithme efficace pour calculer les valeurs de α_e et β_e .

Concentrons nous d'abord sur le problème (7). Vu que le routage est en mono-chemin, ce problème peut être écrit sous la forme suivante :

$$(9)\begin{cases} \alpha_e = \max \sum_{s\in Q}\sum_{t\in\Gamma_e^+(s)} d_{s,t} & \\ \text{s.t.} & \\ \quad \sum_{t\in\Gamma_e^+(s)} d_{s,t} \le b^+(s), & s\in Q \quad (9a)\\ \quad d_{s,t} \le b^-(t), & s\in Q, t\in\Gamma_e^+(s) \quad (9b)\\ \quad d_{s,t} \ge 0, & s\in Q, t\in\Gamma_e^+(s) \quad (9c)\\ \quad d_{s,t} = 0, & s\in Q, t\notin\Gamma_e^+(s) \quad (9d) \end{cases}$$

avec $\Gamma_e^+(s)_{s\in Q} = \{t(\pi) \ : \ \pi\in\Pi, s(\pi)=s \quad \text{et} \quad \gamma_\pi^e = 1\}$. Intuitivement, $\Gamma_e^+(s)$ est l'ensemble de noeuds situés à droite du graphe biparti qui sont directement reliés au noeud source s. Il est facile de vérifier qu'une solution du problème (9) est telle que chaque noeud source envoie le maximum possible sans pour autant dépasser les capacités des noeuds récepteurs. Formellement, ceci peut être écrit de la façon suivante :

$$\alpha_e = \sum_{u,\in Q} \alpha_e(u) \quad \text{où} \quad \alpha_e(u) = min\left(b^+(u), \sum_{v\in\Gamma_e^+(u)} b^-(v)\right).$$

De la même façon, le coût optimal β_e du problème (8) est donnée par la formule suivante :

$$\beta_e = \sum_{u,\in Q} \beta_e(u) \quad \text{où} \quad \beta_e(u) = min\left(b^-(u), \sum_{v\in\Gamma_e^-(u)} b^+(v)\right)$$

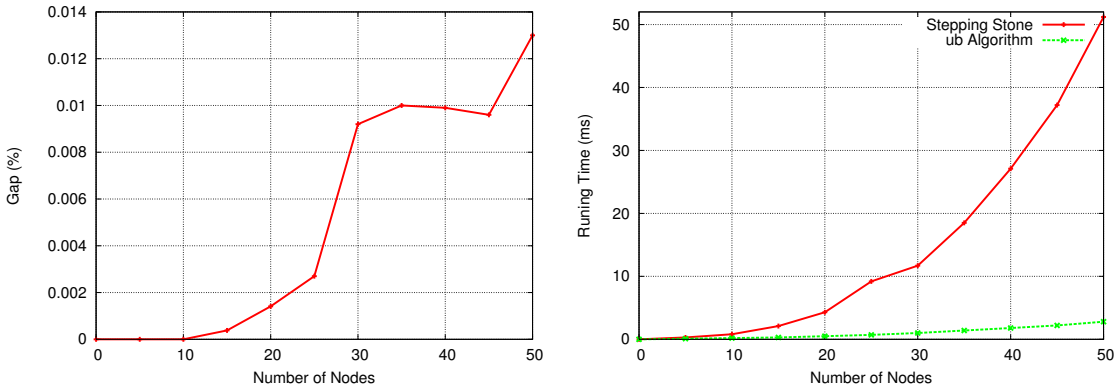
Finalement une borne supérieure $\bar{m}_e(\mathbf{f})$ sur $m_e(\mathbf{f})$ est donnée par $\bar{m}_e(\mathbf{f}) = min(\alpha_e, \beta_e)$. On obtient ainsi très facilement une borne supérieure sur la bande-passante à réserver sur un lien, connaissant la stratégie de routage.

Nous avons évalué la qualité de cette borne supérieure expérimentalement. Nous avons effectué 10000 tests sur des réseaux de transport générés aléatoirement (figure 6). Nous avons comparé l'écart entre le cout exact obtenu avec la méthode du Stepping Stone et les valeurs de la borne supérieure. Comme on peut le constater sur la figure 6(a), l'écart moyen ne dépasse pas 0.013%, valeur maximale obtenue pour 50 noeuds (c'est à dire 50 noeuds à droite et 50 noeuds à gauche dans le graphe biparti de transport). La comparaison entre les temps de calcul des deux méthodes est présentée sur la figure 6(b). Dans cette dernière figure, "ub Algorithm" représente l'agorithme de calcul de la borne supérieure. On peut voir que le temps de calcul nécessaire pour calculer la borne supérieure est négligeable par rapport à celui nécessaire au calcul du coût exact avec l'algorithme du Stepping Stone.

La borne supérieure $\bar{m}_e(\mathbf{f})$a sur la capacité à réserver sur chaque lien peut être utilisée pour en déduire une borne supérieure sur le coût d'une stratégie de routage $\mathbf{f}$ de la façon suivante :

$$\Psi(\mathbf{f}) = \sum_{e\in\mathcal{E}} c_e\, \bar{m}_e(\mathbf{f}) \ + \ A \,.\, \sum_{e\in\mathcal{E}} max\left(0, [C_e - \bar{m}_e(\mathbf{f})]^2\right)$$

La valeur $\Psi(\mathbf{f})$ peut être calculée très rapidement par rapport au coût exact $\Phi(\mathbf{f})$. De plus, $\Psi(\mathbf{f})$



(a) Ecart relatif entre les valeurs de $\bar{x}_e$ et x_e (b) Temps de Calcul

FIGURE 6 – Tests sur la qualité de la borne supérieure.

est en général une très bonne approximation de $\Phi(\mathbf{f})$, tout en étant toujours supérieur. Ainsi, cette borne supérieure permet de trouver rapidement une solution améliorante dans le voisinage de la solution courante $\mathbf{f}_k$. En effet, si l'on détermine une solution $\mathbf{f} \in \mathcal{N}(\mathbf{f}_k)$ telle que $\Psi(\mathbf{f}) < \Phi(\mathbf{f}_k)$, alors on peut affirmer que $\Phi(\mathbf{f}) < \Phi(\mathbf{f}_k)$, et donc que la transition de $\mathbf{f}_k$ vers $\mathbf{f}_{k+1} = \mathbf{f}$ va permettre de faire décroître le coût.

4 Résultats

Afin d'évaluer notre heuristique, nous avons effectué des tests sur des réseaux ayant des caractéristiques fréquemment observées dans la pratique. Le nombre de noeuds de ces réseaux varie entre 10 et 40, et pour chaque instance, cinq topologies de test sont générées aléatoirement. Les valeurs des trafics d'entrée et de sortie varient entre $1Mb/s$ et $50Mb/s$. Nous avons supposé que les coûts d'utilisation des liens sont unitaires ($c_e = 1,\ e \in E$). Nous n'avons considéré dans les résultats présentés dans cette section que les topologies pour lesquelles la solution du problème du routage en multi-chemin n'est pas une solution en mono-chemin.

Tous les tests ont été effectués sous un processeur Pentium Centrino 3 Ghz, fonctionnant sous un système d'exploitation Linux avec 2 Go de mémoire disponible. Notre algorithme de recherche tabou a été implémenté en C++ et nous avons utilisé le solveur glpk pour résoudre les problèmes linéaires ainsi que les problème linaire mixtes. Pour générer, dans le cadre de notre algorithme de recherche tabou, les K-plus courts chemins entre chaque paire d'origine-destination, nous avons utilisé la méthode HREA décrite dans [14]. Finalement, nous avons fixé un temps limite de calcul de 1 jour pour glpk ; au delà de cette limite, glpk a été interrompu.

Les résultats de calcul sont résumés dans les tableaux 1 et 2, qui incluent les informations suivantes :

– La taille des instances
– Ecart RL/BorneInf : L'écart relatif en % entre le coût obtenue avec notre heuristique et la borne inférieure sur le coût optimal obtenue en relaxant la contrainte de monoroutage. Rappelons que cette borne inférieure peut être obtenue en temps polynomial.
– Ecart Exact/BorneInf : L'écart relatif en % entre le coût optimal et la borne inférieure sur ce coût
– Les temps calcul en secondes pour calculer les valeurs de la borne inférieure, le cout de l'heuristique et le cout optimal (on met le symbole ∞ quand le seuil de temps est dépassé).

Scenario	$\|V\|$	$\|Q\|$	Ecart RL/BorneInf	Ecart Exact/Borne-Inf
1	10	7	13.76	12.68
2	15	10	24.67	22.61
3	20	12	24.31	-
4	25	16	5.4	-
5	30	20	1.87	-
5	40	24	14.58	-

TABLE 1 – Ecarts relatifs en % entre le coût de l'heuristique et la borne inférieure et entre le coût optimal et la borne inférieure

Scenario	$\|V\|$	$\|Q\|$	BorneInf	RL	Exact
1	10	7	1	61	5 623
2	15	10	2.4	156	8 778
3	20	12	9	375	∞
4	25	16	75.39	626	∞
5	30	20	488	1658	∞
5	35	22	687	3365	∞
6	40	24	915	5086	∞

TABLE 2 – Temps calcul en secondes.

On peut voir de ces tableaux que pour des réseaux de taille entre 10 et 15 noeuds, notre algorithme de recherche tabou génère des solutions situées à 1 ou 2 % de l'optimum. Pour ces réseaux le temps de calcul des solutions optimales est considérable.

On peut voir aussi que les temps de calcul, requis pour trouver les solutions optimales, augmentent exponentiellement avec la taille du problème. Au delà de 15 noeuds, le calcul du coût optimal par une méthode exacte prend un temps non raisonnable. L'utilisation de la méthode tabou proposée permet alors d'obtenir des solutions de bonne qualité en des temps calcul réduits.

5 Conclusion

Dans cet article, nous avons étudié le problème de la conception des VPN spécifiés suivant le modèle Hose avec un plan de routage en mono-chemin. Ce problème est connu pour être NP-difficile. Nous avons proposé une approximation basée sur la recherche tabou pour résoudre ce problème. L'originalité de cette méthode est d'utiliser une borne supérieure sur la bande-passante à réserver pour explorer le voisinage de la solution courante.

Nous avons démontré à l'aide de résultats expérimentaux l'efficacié de notre heuristique.

Ces résultats soulignent l'intérêt de poursuivre le travail sur la planifications des VPN Hose. Nous envisageons dans le futur d'étudier le problème d'optimisation des VPN Hose avec stratégie de routage à base de partage de charge équitable. C'est le cas des VPNs utilisant les protocoles de routage OSPF et IS-IS qui sont largement répandus.

Références

[1] A. Altin et al., *Provisioning Virtual Private Networks under traffic uncertainty*, Networks, Vol. 20, No. 1, January 2007, pp. 100-115.

[2] A. Charnes and W. W. Cooper , *The Stepping Stone Method of Explaining Linear Programming Calculations in Transportation Problems*, Mark. Sci. 1 (1954), 49-69.

[3] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi and B. Yener, *Provisioning a Virtual Private Network : A Network Design Problem for Multicommodity Flow*, in Proc. of the 33rd Annual ACM Symposium on Theory of Computing (STOC), 2001. pp. 389-398.

[4] A. Gupta, A. Kumar, and T. Roughgarden, *Simpler and better approximation algorithms for network design*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC), 2003, pp. 365-372.

[5] A. Juttner, I. Szabo, and A. Szentesi, *On bandwidth efficiency of the hose resource management model in virtual private networks*, in Proc. IEEE INFOCOM 2003, San Francisco, Apr. 2003, pp. 386-395.

[6] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, *Algorithms for provisioning virtual private networks in the hose model*, IEEE/ACM Transactions on Networking, vol. 10, no. 4, pp. 565-578, August 2002

[7] C. A. J. Hurkens, J. C. M. Keijsper and L. Stougie, *Virtual Private Network Design : A Proof of the Tree Routing Conjecture on Ring Networks*, SIAM Journal on Discrete Mathematics, Volume 21 , Issue 2, pp. 482-503, 2007

[8] C. Swamy and A. Kumar, *Primal-dual algorithms for connected facility location problems*, in Proceedings of the 5th International Workshopon Approximation Algorithms for Combinatorial Optimization (APPROX'02), ser. LNCS 2462, 2002, pp. 256-270.

[9] D. Applegate and E. Cohen, *Making intra-domain routing robust to changing and uncertain traffic demands : understanding fundamental tradeoffs*, in Proc. ACM SIGCOMM Karlsruhe, Germany, 2003, pp. 313 - 324

[10] G. S. Poo and H. Wang, *Multi-path routing versus tree routing for VPN bandwidth provisioning in the hose model*, Computer Networks Journal, Volume 51 , Issue 6, pp. 1725-1743, 2007

[11] L. Lasdon, *Optimization Theory for Large Systems*, Mac Millan Company Ed., 1970.

[12] N. G. Duffield et al. , *A flexible model for resource management in virtual private networks*, in Proc. ACM SIGCOMM, San Diego, CA, Aug. 1999, pp. 95-108.

[13] T. Erlebach, and M. Ruegg*Optimal bandwidth reservation in hose-model VPNs with multi-path routing*, in proceeding of INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies Vol. 4, 7-11 March 2004. pp.2275-2282.

[14] V. M. Jiménez and A. Marzal , *Computing the K Shortest Paths : A New Algorithm and an Experimental Comparison*, Springer-Verlag, Lecture Notes in Computer Science Series, Vol. 1688, pp.15-29, 1999

[15] http ://www.gnu.org/software/glpk/

Optimisation du développement de nouveaux produits dans l'industrie pharmaceutique par algorithme génétique multicritère

José Luis Pérez-Escobedo, Catherine Azzaro-Pantel, Luc Pibouleau, Serge Domenech

Université de Toulouse, Laboratoire de Génie Chimique, UMR 5503 CNRS, ENSIACET INPT, 4, allée Emile Monso,
BP 84234- 31432 Toulouse Cedex 4 - France
{joseluis.perezescobedo,catherine.azzaropantel,luc.pibouleau,serge.domenech}@ensiacet.fr

Résumé : *Le développement de nouveaux produits constitue une priorité stratégique de l'industrie pharmaceutique. Il s'agit d'un problème fortement combinatoire impliquant des processus relativement longs (de l'ordre d'une dizaine d'années), largement incertains et des investissements très importants. Dans ce contexte, une approche basée sur le couplage entre un simulateur à événements discrets pour représenter la dynamique du système et un algorithme d'optimisation multicritère (de type NSGA II) pour sélectionner les produits est proposée. Trois critères sont étudiés, la maximisation du bénéfice actualisé, la minimisation du risque et celle de l'horizon de temps. La méthodologie est illustrée à travers un exemple de portefeuille de 9 produits pour 3 maladies cibles. Les tendances obtenues sont spécifiques, mais l'approche fournit un cadre d'étude pour une aide à la décision qui pourra être adoptée lors du traitement de nouveaux cas.*

Mots-Clés : *Gestion du portefeuille de produits, Développement de nouveaux produits, Simulation par événements discrets, Optimisation, Algorithmes génétiques multicritères.*

1 Introduction

L'industrie pharmaceutique implique des processus de développement relativement longs et fortement régulés. A titre d'illustration, une société pharmaceutique peut traiter plus de 100 projets à des stades différents de leur développement (pré-clinique, clinique, ...). Le processus, qui peut durer jusqu'à plus de 12 ans pour les molécules qui atteindront le stade de produit approuvé, comporte de nombreuses incertitudes et s'avère coûteux en termes de ressources humaines, matérielles et temporelles. Notons, à ce stade, que ces ressources sont essentiellement prises en compte à travers une allocation de budget. Par ailleurs, le taux de mortalité des projets depuis la recherche jusqu'au produit commercialisé est très élevé. Le développement d'une méthodologie d'optimisation est ainsi particulièrement intéressant pour assister le décideur dans la tâche de sélection des produits et donner une vision claire à la fois sur le portefeuille global et les niveaux opérationnels.

La gestion du "pipeline" de nouveaux produits implique une série de compromis, visant à maximiser le bénéfice actualisé, minimiser le risque tout en maintenant une diversité dans les différents produits pour un niveau donné de ressources renouvelables et non renouvelables. Dans ce contexte, l'incertitude et la limitation des ressources contribuent à la complexité du problème.

Lorsqu'une molécule candidate est promue molécule d'intérêt, elle suit les étapes impliquées dans le diagramme de flux de la figure 1. Toutes les activités ou tâches sont décrites par des rectangles et les relations de précédence entre les tâches par des flèches. Une description plus précise des différentes activités est proposée dans [1]. Par exemple, on constate une vitesse d'attrition très élevée au fur et à mesure que les produits candidats se déplacent à travers le "pipeline". Ainsi, environ 20 % des produits sont abandonnés après la Phase I, tandis que 80 % des produits restants sont éliminés après le test d'efficacité sur l'homme en Phase II. Le portefeuille doit être sélectionné de telle sorte que la compétition parmi les produits candidats pour les ressources limitées ne conduisent pas à des temps de développement de produits excessivement longs, ou a des commercialisations tardives. En outre, il doit être suffisamment étendu pour compenser les échecs de produits sans mobiliser toutefois les ressources de façon excessive. Le problème est également rendu complexe par la présence de dépendances entre produits. Une caractéristique majeure du problème réside dans le fait que la génération des profits ne s'effectue qu'après la phase de commercialisation, comme l'illustre la figure 2.

Dans ce contexte, l'objectif de ce travail consiste à proposer une méthodologie d'aide à la décision basée sur un couplage d'un simulateur à événements discrets, reproduisant les différentes étapes impliquées dans le cycle de vie du produit et leurs interconnexions, avec un algorithme d'optimisation, adapté à la combinatoire du problème et à l'aspect multiobjectif précédemment évoqué. Compte tenu de ces caractéristiques, le choix s'est porté sur un algorithme génétique multicritère. Ce document est surtout focalisé sur la démarche d'optimisation. Néanmoins, dans une première partie, le système d'étude est décrit et un exemple significatif sert de support méthodologique. Dans un deuxième temps, les points clefs de la simulation sont abordés, en vue de l'évaluation des différents critères. Dans une troisième partie, sont détaillés les éléments essentiels pour le développement de la procédure d'optimisation. Des résultats significatifs sont ensuite analysés, en vue d'une aide à la décision.

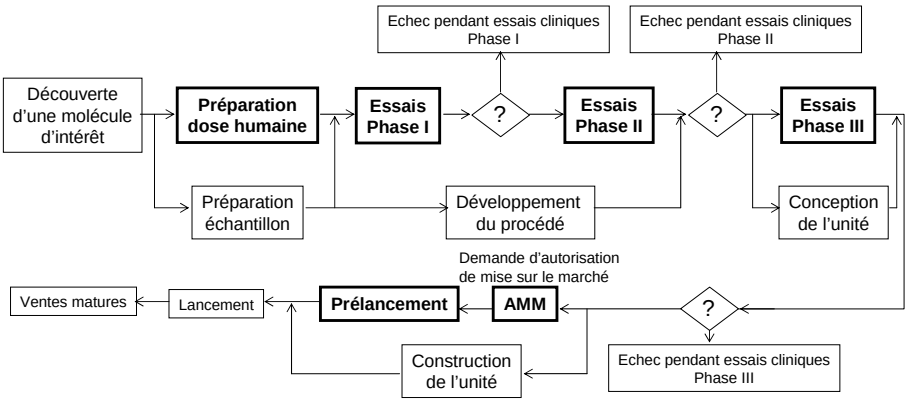


FIGURE 1 – Diagramme de flux des différentes activités impliquées dans le développement et la commercialisation d'un nouveau médicament.

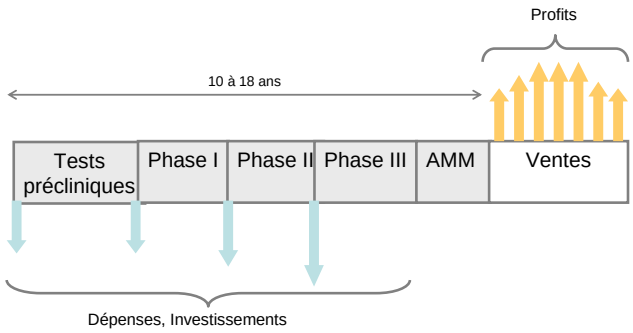


FIGURE 2 – Coûts et profits lors de la phase de développement de nouveaux produits.

2 Description du système d'étude

L'exemple retenu pour illustrer la procédure est celui présenté dans [1] : il met en jeu neuf produits pour trois types de maladies selon le processus de la figure 1. Chaque activité impliquée dans le processus est décrite par plusieurs paramètres :

- une durée (Valeurs minimale d_{min}, moyenne d_{moy}, et maximale d_{max}), exprimée en jours ;
- un coût (Valeurs minimale C_{min}, moyenne C_{moy}, et maximale C_{max}) exprimé en $MM\$$;
- un niveau de ressources disponibles exprimé en $MM\$$;

Chaque produit P_i est décrit de la façon suivante :

- Capacité à traiter un type de maladies (M_1,M_2,M_3) ;
- Probabilités de succès correspondant aux Phases I, II et III ;
- Un coût en investissement (Valeurs minimale I_{min}, moyenne I_{moy} et maximale I_{max}), exprimé en $MM\$$;
- Un niveau de ventes à maturité (Valeurs minimale V_{min}, moyenne V_{moy} et maximale V_{max}) exprimé en $MM\$$ correspondant au seul produit sur le marché pour traiter la maladie ;
- Un degré de difficulté, variant de 1 à 10.

Les valeurs numériques adoptées dans le cadre de ce travail sont celles de [1] et sont représentatives de données historiques typiques d'une société pharmaceutique. Le Degré de Difficulté (DdD) est un paramètre traduisant la liaison entre les temps d'activités et les coûts pour un médicament candidat. L'estimation subjective de ce paramètre résulte de l'expérience des chercheurs impliqués dans le processus de développement et reflète la difficulté de réalisation d'une étape (compte tenu par exemple de la complexité de synthèse de la molécule), à travers son coût associé. La valeur la plus probable de la distibution est proportionnelle au DdD. Ainsi, pour un degré de difficulté égal à 1 (respectivement à 10), la valeur la plus probable correspond à la valeur minimale (respectivement maximale) de la distribution triangulaire. Pour des valeurs intermédiaires du DdD, les valeurs les plus probables sont linéarisées à partir de la valeur médiane du DdD égale à 5. La figure 3 illustre cette procédure.

Plusieurs types de dépendances entre produits existent et ont été modélisés :

- *Dépendance concurrentielle :* cette dépendance traduit le fait que si plusieurs produits ciblant une même maladie arrivent avec succès jusqu'au stade final, les ventes relatives à chaque produit se voient diminuées par rapport à la valeur V d'un certain coefficient, reflétant ainsi la concurrence

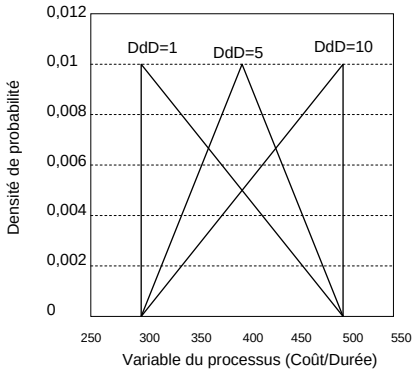


FIGURE 3 – Impact du degré de difficulté (DdD) sur les distributions triangulaires.

entre les produits ;
– *Dépendance financière :* pour une séquence de produits ciblant une maladie donnée, le premier produit se voit attribuer la totalité des besoins en capital nécessaire, le deuxième la moitié, le troisième le tiers etc.
– *Prise en compte de l'apprentissage :* on tient compte de la courbe d'apprentissage dans la mise au point d'un produit pour une maladie à travers une réduction du degré de difficulté pour les produits de la même maladie dans une séquence.

Si tous les paramètres utilisés dans ce travail se réfèrent à [1] pour la validation de l'approche proposée, il faut noter qu'aucune dépendance technique n'a été prise en compte entre les produits d'une même séquence.

3 Développement du simulateur à événements discrets

Une approche par simulation à événements discrets a été retenue pour représenter le comportement du système : elle est basée sur l'extension d'un simulateur développé lors de travaux antérieurs en ordonnancement et conception d'ateliers [2]. Il s'agit d'une bibliothèque, implantée en C++, comportant quatre parties ou couches superposées : le moteur de simulation à événements discrets, les événements, les objets de l'atelier et le superviseur. Nous ne détaillerons pas ici la phase de réutilisation et d'extension de ce simulateur qui a fait l'objet d'une autre présentation [3].

Un certain nombre de similitudes existe avec le développement d'une approche d'ordonnancement de produits : dans le cadre de la gestion de projets, un projet relatif à un médicament est assimilé à un produit en ordonnancement ; de même, les recettes de produits correspondent ici à l'enchaînement des tâches au sein du processus global.

Soit une séquence de produits qui entrent dans le "pipeline" décrit sur la figure 1. Le simulateur reproduit le comportement du système pour des valeurs des paramètres générées aléatoirement à partir des lois de distribution présentées précédemment. Une contrainte importante concerne le niveau des ressources financières qui ne doit pas être violée. Le renouvellement des ressources intervient

de façon annuelle. Il en résulte une gestion des files d'attente (règle FIFO) qui ralentit le processus et freine le lancement des produits suivants. L'analyse des résultats s'effectue à partir d'un nombre d'échantillons représentatifs (300 au total), selon une méthode de type Monte-Carlo. L'analyse bibliographique montre que les critères essentiels sont liés au bénéfice actualisé et au risque financier. Compte tenu de la longueur des processus impliqués, il nous semble important de considérer également ce dernier objectif. Il faut aussi constater que les études précédentes [1, 4, 5] sont surtout abordées du point de vue monocritère (bénéfice actualisé), le risque étant considéré sous l'angle d'une contrainte. Dans cette étude, sont donc pris en compte simultanément les critères suivants :

– Le *bénéfice actualisé* global d'une séquence, calculé classiquement à partir de la moyenne des bénéfices actualisés des échantillons. Un taux d'actualisation de 15 % a été retenu.
– Le *risque*, apprécié à travers le nombre de fois où on constate une valeur négative du bénéfice actualisé par rapport au nombre total de prises d'échantillons ;
– La *durée*, également calculée à partir de la moyenne des durées des échantillons, déterminées de façon directe dans le simulateur.

4 Optimisation

Compte tenu de l'aspect combinatoire du problème et de sa nature multicritère, il a été retenu un algorithme génétique multicritère comme méthode d'optimisation. Parmi les variantes disponibles, notre choix s'est porté vers NSGAII [6], car il s'agit d'un algorithme particulièrement performant en termes de méthode de tri des individus non dominés et de mise en oeuvre d'une politique d'élitisme. Par ailleurs, il n'implique pas la spécification de paramètre dit de "sharing"' selon le terme anglo-saxon consacré qui s'avère délicate dans d'autre méthodes. Il est également particulièrement efficace pour la gestion des contraines.

Les phases clés du codage, des procédures de croisement et de mutation ainsi que la gestion des contraintes sont détaillées. Le calcul des fonctions d'évaluation est effectué par appel au simulateur à événements discrets (maximisation du bénéfice actualisé, minimisation du temps, minimisation du risque).

4.1 Codage, croisement et mutation

Une séquence est modélisée par l'intermédiaire de deux types de chromosomes comportant un nombre de gènes identique, égal au nombre de produits à considérer dans le portefeuille global. Chaque produit P_i se voit attribuer un numéro i et correspond à la position chromosomique i.

– Le premier chromosome $Chrom_1$ est lié à l'ordre de lancement des produits d'une séquence et se base sur une permutation des éléments. Les gènes sont donc des entiers, variant de la valeur 1 au nombre total de produits de la séquence. La valeur de chaque gène ne peut se retrouver qu'une seule fois dans le chromosome. Pour une position i d'un gène, sa valeur correspond à l'ordre du produit i de la séquence ;
– Le second chromosome $Chrom_2$ est uniquement composé de variables binaires, la valeur 1 d'un gène en position i (respectivement 0) correspondant à la présence (respectivement l'absence) du produit.

Le chromosome correspondant à chaque séquence s'obtient alors en multipliant chaque gène du $Chrom_1$ avec celui du $Chrom_2$, locus à locus. Il faut souligner que ce codage est non univoque, ce qui peut introduire un biais dans la recherche. Certes, une alternative plus séduisante consisterait à

coder directement les chromosomes représentant une séquence, de longueur variable selon le nombre de produits de la séquence. Toutefois, cette approche peut conduire lors de la phase de croisement à des individus non réalisables, de taille supérieure à celle correspondant au nombre effectif de produits de la séquence. L'efficacité de la première procédure ayant été testée à travers les exemples étudiés, elle a été retenue dans le cadre de ce travail.

La Figure 4 illustre le codage utilisé et transmis au simulateur pour l'évaluation d'une séquence : pour le nombre de produits impliqué et pour l'ordre de lancement retenu, 300 appels au simulateur sont effectués pour calculer les critères précédemment évoqués.

Le croisement et la mutation s'effectuent par des procédures spécifiques sur chacun des types de chromosome.

Les chromosomes représentant l'ordre de lancement des produits sont haploïdes, mais tous les gènes entiers le composant doivent être différents, variant de 1 au nombre total de produits de la séquence. Dans ce but, un opérateur de croisement respectant les contraintes de génotype sans génération de doublon dans le code génétique des enfants a été mis en oeuvre, croisement dit MPX (Maximal Preservative X) [7].

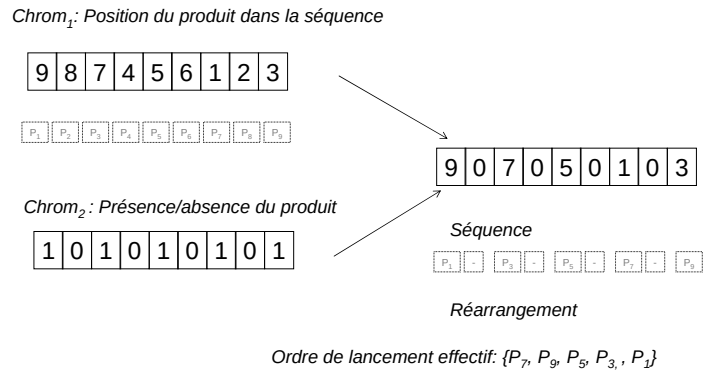


FIGURE 4 – Schéma du codage d'une séquence

Concernant la mutation, une procédure classique par inversion de gènes a été utilisée pour les deux chromosomes.

4.2 Contraintes

Les contraintes spécifiquement traitées par NSGA II sont relatives à l'existence d'au moins un médicament dans la séquence pour chaque maladie cible. Pour l'exemple retenu, trois maladies et neuf molécules sont à l'étude :

- Maladie M_1 : Produits P_2, P_3, P_6, P_7 ;
- Maladie M_2 : Produits P_4, P_5, P_8, P_9 ;

– Maladie M_3 : Produit P_1.

Ces contraintes se traduisent par la valeur d'au moins un gène égale à 1 pour le chromosome $Chrom_2$ dans les loci correspondant aux gènes des produits de la maladie concernée g_i, soit :

$$g_2 + g_3 + g_6 + g_7 \geq 1 \tag{1}$$

$$g_4 + g_5 + g_8 + g_9 \geq 1 \tag{2}$$

$$g_1 = 1 \tag{3}$$

5 Analyse des résultats

Les essais d'optimisation sont, dans un premier temps, menés selon une approche bicritère, puis tricritère. Les paramètres retenus sont les suivants : nombre d'individus égal à 80, 200 générations, taux de croisement fixé à 0.9 et taux de mutation de 0.1.

Pour tenir compte de la nature stochastique de l'algorithme génétique, chaque optimisation est répétée 5 fois. Le temps CPU de chaque optimisation est difficile à évaluer, compte tenu du double aspect stochastique (AG, Simulation avec approche-Monte-Carlo). Chaque optimisation, représentant le cumul des 5 essais, est de l'ordre de 25 heures CPU.

Les résultats relatifs à chaque optimisation bicritère présentent le front de Pareto obtenu à partir des solutions non-dominées des 5 essais.

5.1 Optimisation bicritère *Bénéfice Actualisé-Risque*

Pour la première optimisation bicritère *Bénéfice Actualisé-Risque* (voir Figure 5), on constate que la variation du risque s'échelonne entre 10% et 40% . Au-delà de 40% de risque, il n'existe plus de solution. Un résultat intéressant concerne le nombre de médicaments dans la séquence :

– Pour les valeurs du risque comprises entre 12% et 20%, le nombre de médicaments dans les séquences est de 4 et les produits présents sont systématiquement $[P_1 \wedge P_2 \wedge P_5 \wedge P_7]$, avec une valeur du bénéfice actualisé comprise entre 1400 et 1700 $MM\$$.

– Pour les valeurs du risque entre 27% et 39% le nombre de produits est de 6 et les produits présents sont systématiquement $[P_1 \wedge P_2 \wedge P_3 \wedge P_5 \wedge P_6 \wedge P_7]$ avec une valeur du bénéfice actualisé comprise entre 1800 et 2000 $MM\$$.

Pour ces deux comportements (4 ou 6 produits pour le front de Pareto), les solutions se distinguent par l'ordre de lancement de la séquence. Une analyse plus fine des résultats peut être obtenue à partir de l'identification des séquences (Tableaux 1 et 2). On remarque qu'une même séquence (par exemple 1 et 2 dans le tableau 1 peut conduire à plusieurs solutions dans l'espace objectif, en raison du double aspect stochastique déjà mentionné. On constate que les résultats relatifs à une même séquence vont toujours dans le sens où une augmentation du risque conduit à une augmentation du bénéfice actualisé, à séquence donnée. Les différences entre les valeurs des critères restent dans les marges d'erreur admissibles, par exemple 8% pour le bénéfice actualisé et 20% pour le risque pour les séquences 1 et 2, ce qui correspond aux variations maximales.

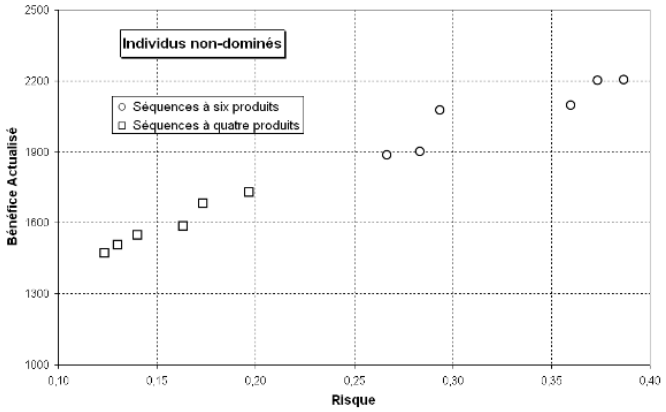


FIGURE 5 – Individus non-dominés pour les critères *Bénéfice Actualisé - Risque*

Séquence	Bénéfice Actualisé	Risque	Ordre de lancement			
1	1730.06	20	P2	P7	P5	P1
2	1586.83	16	P2	P7	P5	P1
3	1683.01	17	P7	P5	P2	P1
4	1507.49	13	P5	P7	P2	P1
5	1472.43	12	P5	P2	P7	P1
6	1549.20	14	P2	P7	P5	P1

TABLE 1 – Résultats de l'étude bicritère pour les séquences à 4 produits

Une optimisation complémentaire pour affiner la recherche consisterait à jouer uniquement sur ces deux paramètres, à nombre fixé de produits.

5.2 Optimisation bicritère *Bénéfice Actualisé-Durée*

Les résulats relatifs à ce deuxième couple de critères sont proposés sur la Figure 6. Les calculs mettent en évidence des valeurs négatives pour le bénéfice actualisé, qui correspondent à des valeurs trop faibles de l'horizon de temps, que le décideur éliminera bien évidemment. Sur cet exemple également, on trouve un nombre important de séquences à 3 ou 4 produits. Pour les séquences à 3 produits, les médicaments retenus sont $[P_1 \wedge (P_2 \vee P_6) \wedge (P_5 \vee P_8)]$. Les séquences à 4 produits contiennent les médicaments $[P_1 \wedge P_2 \wedge P_5 \wedge P_7]$.

Séquence	Bénéfice Actualisé	Risque	Ordre de lancement					
1	2206.62	39	P6	P1	P3	P2	P7	P5
2	2203.78	37	P6	P2	P1	P7	P5	P3
3	2097.14	36	P2	P6	P1	P7	P5	P3
4	1900.84	28	P7	P6	P3	P2	P5	P1
5	1887.21	27	P3	P6	P7	P2	P5	P1
6	2077.74	29	P3	P6	P7	P2	P5	P1

TABLE 2 – Résultats de l'étude bicritère pour les séquences à 6 produits

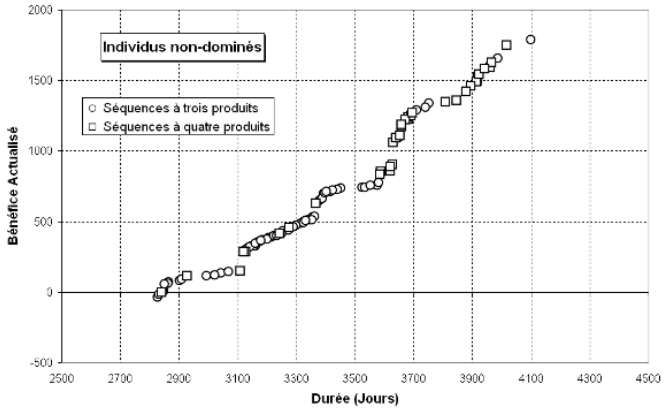


FIGURE 6 – Individus non-dominés pour les critères *Bénéfice Actualisé - Durée*

5.3 Optimisation bicritère *Durée-Risque*

La Figure 7 est relative à l'étude de la troisième paire de critères *durée - risque*.

Ces résultats permettent de quantifier l'impact drastique de la diminution de la durée du processus sur l'augmentation du risque. Le risque passe de 13% à près de 70% lorsqu'on passe de 11,5 ans à 7,7 ans. Les séquences dont le risque s'échelonne entre 13 et 18% comportent exclusivement 3 produits dont $[P_1 \wedge (P_2 \vee P_7) \wedge (P_5 \vee P_8)]$. Les séquences dont le risque varie de 0.21 à 0.73% possèdent 4 produits $[P_1 \wedge P_2 \wedge P_5 \wedge P_7]$. Il existe une petite zone de recouvrement de 0,18 à 0,21 % avec des types de séquences mixtes à 3 et 4 produits (les mêmes que précédemment). Le nombre plus élevé de séquences présenté sur le schéma par rapport à la stricte combinatoire du problème correspond à de légères variations au niveau de l'évaluation des critères pour des individus "doublons", en raison de l'aspect stochastique du problème.

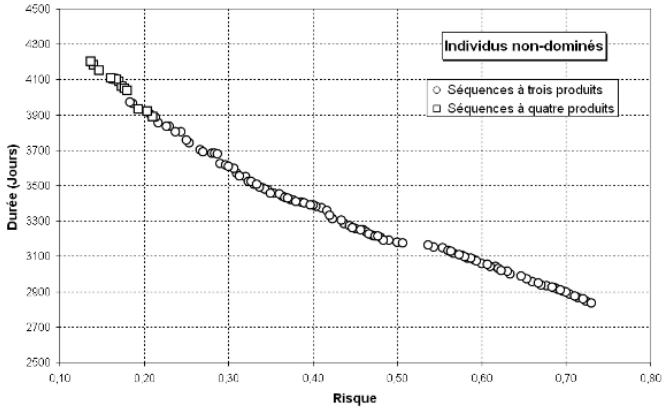


FIGURE 7 – Individus non-dominés pour les critères *Risque - Durée*

L'optimisation bicritère menée successivement sur les différentes paires de critères montre ainsi globalement que les longues séquences, qui atténuent certes l'impact des échecs des produits, ne sont pas privilégiées en raison de la pénalisation qu'elles induisent vis-à-vis de la mobilisation des ressources.

5.4 Optimisation tricritère *Bénéfice Actualisé-Durée-Risque*

Dans ce paragraphe, l'étude tricritère est menée afin de vérifier les tendances précédentes. Pour faciliter leur interprétation, les résulats sont visualisés à travers une projection sur le plan concerné (Figures 8, 9, 10).

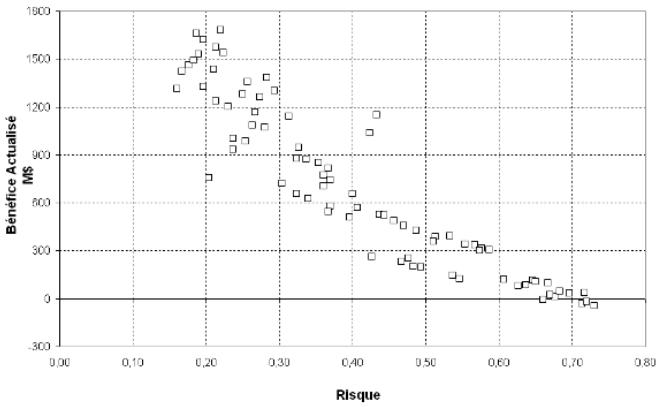


FIGURE 8 – Solutions tricritères en projection sur les axes *Bénéfice actualisé - Risque*

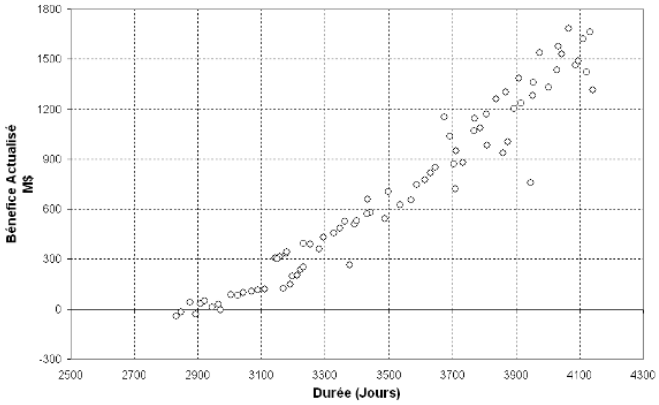


FIGURE 9 – Solutions tricritères en projection sur les axes *Bénéfice actualisé - Durée*

On retrouve globalement les mêmes tendances que dans l'approche bicritère : un faible nombre de produits permet les meilleurs compromis entre les différents critères. La Figure 8 nécessite un commentaire particulier, puisqu'on constate une évolution dans un sens contraire de celle observée dans le cas bicritère et illustré sur la Figure 5. Cela s'explique par une exploration du domaine liée au critère "'risque"' beaucoup plus large, au-delà de la zone de 40%, attribuable à l'influence simultanée du critère "'temps"' que l'on cherche aussi à minimiser. On remarque que dans la zone de risque inférieure à 30%, il est difficile de dégager une tendance entre les solutions.

Une analyse plus fine consisterait à analyser les produits en fonction des différents paramètres (coût, durée, probabilité de succès aux différentes phases ...). A partir de ces solutions de compromis, le décideur peut ensuite, à partir du niveau de risque qu'il se fixe (voir tableau 3), déterminer la séquence à retenir, le bénéfice actualisé et l'horizon de temps du "pipeline" .

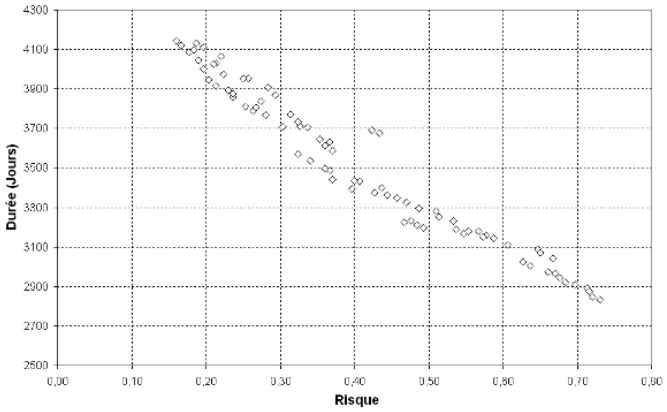


FIGURE 10 – Solutions tricritères en projection sur les axes *Risque* - *Durée*

Risque %	Bénéfice Actualisé en $MM\$$	Durée en jours
16	1316,29	4141,52
20	1620,7	4110,4
25	1280,99	3952,73
30	721,11	3709,74
35	850,35	3645,91

TABLE 3 – Valeurs des critères résultats de l'optimisation tricritère

6 Conclusions

Cette étude a montré l'intérêt du couplage entre un simulateur à événements discrets et un algorithme d'optimisation multicritère pour tenir compte des spécificités de la phase de développement de nouveaux produits dans l'industrie pharmaceutique. Les processus impliqués sont relativement longs (de l'ordre d'une dizaine d'années), largement incertains et mettent en jeu des investissements très importants. Dans ce contexte, le recours à l'optimisation est une voie séduisante pour guider le décideur dans ses choix. La méthodologie proposée a été illustrée à travers un exemple comportant 9 produits et 3 maladies cibles : certes, les tendances obtenues sont spécifiques, mais l'approche fournit un cadre méthodologique qui pourra être adopté pour le traitement de nouveaux cas.

Une perspective intéressante consisterait à améliorer la robustesse de la méthode : une première voie serait de travailler avec un codage univoque au niveau de la représentation d'une séquence au sein de l'algorithme génétique. On pourrait aussi envisager de pénaliser les séquences qui conduiraient à des solutions dispersées pour un ordre de lancement identique, nuisant ainsi à l'interprétation et l'aide à la décision.

Références

[1] G.E. Blau, J. E. Pekny, V.A. Varma and P. R. Bunch, Managing a portfolio of interdependent new products candidates in the pharmaceutical industry. *Journal of Product Innovation Management*, 21:227-245, 2004.

[2] F. Bérard, C. Azzaro-Pantel, L. Pibouleau, S. Domenech, D. Navarre and M. Pantel, Towards an incremental development of discrete-event simulators for batch plants : use of object-oriented

concepts *Comp. And Chem. Eng. Supplements*, :S565-S568, 1999.

[3] J.L. Pérez Escobedo, F.Morales Mendoza, A. Aguilar-Lasserre, C. Azzaro-Pantel, S.Domenech, L.Pibouleau, Analysis of new pharmaceutical product mix selection with discrete-event simulation, *SFGP 2009 - Marseille, Palais du Pharo, 14-16 octobre 2009*.

[4] M.J. Rogers, A. Gupta and C. D. Maranas, Real Options based Analysis of optimal pharmaceutical research and development portfolios. *Industrial Engineering Chemical*, 41:6607-6620, 2002.

[5] V. A. Varma, J. E. Pekny, G.E. Blau and G. V. Reklaitis, A framework for addressing stochastic and combinatorial aspects of scheduling and resource allocation in pharmaceutical R&D pipelines. *Computers & Chemical Engineering* , 32:1000-1015, 2008.

[6] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, A fast and elitist multiobjective genetic algorithm *NSGA-II. IEEE Trans. Evol. Comput.*, 6(2):182-197, 2002.

[7] P. Baudet, C. Azzaro-Pantel , S. Domenech and L. Pibouleau, Un couplage entre un algorithme génétique et un modèle de simulation pour l'ordonnancement à court terme d'un atelier discontinu de chimie fine. *RAIRO Operations Research*, 33:299-338, 1999.

Détection de régions génomiques homologues par un algorithme de flot avec coûts

Eric Audemard, Thomas Faraut, Thomas Schiex

INRA toulouse ; Chemin de Borde Rouge BP 52627, 31326 Castanet Tolosan cedex , France
{eric.audemard,thomas.faraut,thomas.schiex}@toulouse.inra.fr

Résumé : *L'identification de régions génomiques homologues, c'est à dire possédant une origine ancestrale commune, est centrale à l'étude des génomes. La difficulté résulte de l'érosion des similiarités dues à cette relation d'homologie. Le principe est alors de rechercher les régions d'homologues parmi un ensemble de courtes régions similaires. Nous présentons un formalisme modélisant toutes les solutions dans un graphe. Dans ce graphe, un chemin est une représentation de deux régions potentiellement homologues et notre but est de retrouver un ensemble cohérent de chemins, c'est à dire de régions homologues. Cet ensemble est reconstruit à l'aide d'une méthode d'optimisation globale basée sur la théorie des flots.*

Mots-Clés : *bioinformatique, évolution des génomes, génomique comparative, flots, réseaux de transport.*

1 Introduction

La recherche d'une relation de parenté, c'est à dire d'une origine ancestrale commune, appelée également relation d'homologie (voir FIG. 1), entre gènes d'espèces différentes, ou au sein d'une même espèce, est une étape importante de l'étude des génomes. La mise en évidence d'une origine ancestrale commune de deux gènes, ou plus généralement de deux régions porteuses d'une fonction biologique, fournit des arguments en faveur d'une fonction biologique commune. Cette recherche d'origine ancestrale commune est surtout centrale à l'étude de l'évolution des gènes, des génomes et des espèces. Avec l'arrivée des génomes entièrement séquencés, cette recherche de relations de parenté se transpose naturellement aux segments des chromosomes. L'origine commune des espèces implique que deux espèces quelconques partageaient la même organisation chromosomique chez leur plus proche parent. Pour ces deux espèces, les organisations actuelles sont le résultat d'une histoire évolutive propre, puisque indépendante depuis leur séparation (spéciation), faite de mutations chromosomiques de différentes natures (de la substitution de nucléotides à des réarrangements chromosomiques pouvant impliquer l'échange de grands fragments entre chromosomes). Une forte similitude entre deux régions suggère une conservation depuis leur plus proche parent. On parle

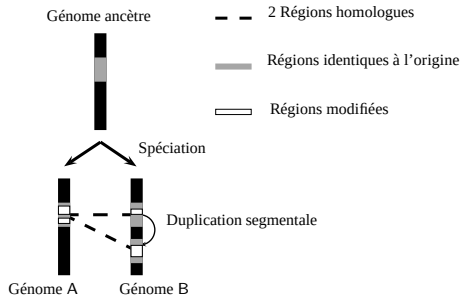


FIGURE 1 – Illustration d'une histoire évolutive de deux séquences avec un ancètre commun, ainsi que les mouvements de spéciation et de duplication segmentale.

alors de segments chromosomiques conservés, ou de segments homologues, et l'identification de cette conservation correspond à la mise en évidence d'une origine ancestrale commune pour ces segments et permet d'appliquer, à cette échelle, la démarche d'inférence de fonction et l'étude de l'évolution des génomes. La recherche de segments homologues au sein de génomes entièrement séquencés fait l'objet du présent travail.

Peu après la séparation de deux espèces, la conservation des génomes est telle qu'il est facile de les comparer (aligner) et d'identifier les longs segments chromosomiques conservés. Lorsque la spéciation est plus ancienne, l'identification des régions conservées est rendue difficile par la poursuite des différents mécanismes évolutifs (mutations ponctuelles, réarrangements...). Ces modifications peuvent mener à la disparition locale de la similarité, à des changement d'orientation ou d'ordre entre éléments inclus dans la région (gènes, régulateurs...). Les seuls indices aisément identifiables permettant de retrouver ces régions homologues sont de courtes régions suffisamment similaires pour pouvoir être alignées (aussi appelées ancres), traces potentielles de l'homologie entre les deux régions. La densité de ces ancres dans une région, la succession d'une quantité importante d'ancres dans un ordre et/ou une orientation identique sont autant d'éléments permettant de suggérer l'existence d'une région homologue.

L'identification de segments homologues entre deux génomes est une extension naturelle du problème d'alignement entre deux séquences. L'une des formalisations de ce problème d'alignement identifie l'alignement à un chemin dans un graphe [4]. Nous proposons de formaliser le problème de la recherche de segments homologues sous la forme d'une recherche d'un ensemble de chemins dans un graphe. Cette formalisation n'est pas nouvelle, elle a été utilisée implicitement ou explicitement par de nombreuses méthodes [9, 2, 8, 14, 15] dédiées à la détection de segments homologues. Une fois le graphe créé, elles utilisent une approche gloutonne, utilisant un algorithme de "plus court chemin", pour sélectionner les chemins un par un. D'autres méthodes [17, 12, 10, 13] de type *cluster* sont aussi développées, mais la sélection des chaînes reste gloutonne. La principale nouveauté de notre approche réside dans le choix de reconstruire simultanément un ensemble cohérent de chemins et dans l'utilisation d'un algorithme de "flot de coût mininimum" pour ce faire. Nous montrons également comment la prise en compte de contraintes supplémentaires permet d'adapter simplement l'algorithme à l'identification de segments homologues au sein d'un même génome.

L'article est organisé de la manière suivante. Nous commençons par rappeler brièvement le problème biologique et sa formalisation sous forme de la recherche de chemins dans un graphe. Nous montrons ensuite comment une adaptation de l'algorithme de "flot maximum coût mininimum" permet de résoudre le problème. Nous présenterons enfin une étude comparative des résultats de notre algorithme avec une approche gloutonne d'une part et avec des logiciels publiés et dédiés à ce problème d'autre part.

2 Modélisation du problème

2.1 Le problème biologique

L'évolution des chromosomes est donc un processus qui se déroule à différentes échelles. Aux modifications locales n'impliquant que quelques nucléotides, s'ajoutent des réarrangements microscopiques locaux pouvant impliquer de quelques milliers à quelques dizaines de milliers de nucléotides et des réarrangements de plus grande ampleur portant sur plusieurs millions de nucléotides voire des bras chromosomiques entiers. Ces deux types de réarrangements, microscopiques et macroscopiques, n'ont pas la même dynamique, les seconds étant bien moins fréquents que les premiers (un génome

de mammifère subit quelques dizaines de mutations par an et à peine un réarrangement de grande ampleur par million d'années). Pour illustrer l'évolution des génomes, on peut utiliser l'analogie d'un long texte organisé en chapitres. Les modifications microscopiques locales fréquentes porteraient sur l'orthographe des mots, leur substitution ou un réarrangement local du texte alors que les réarrangements macroscopiques déplaceraient de grandes sections de texte au sein d'un chapitre ou même entre chapitres. Notre problème consiste, à partir de l'observation de deux livres ayant évolués indépendamment, à identifier de longues portions de texte globalement conservées. La difficulté réside dans le fait que les modifications microscopiques locales brouillent le signal évolutif de conservation.

Le principe de détection de segments homologues consiste à exploiter des similitudes locales fortes, identifiées au préalable. Ces similitudes définissent des points d'ancrage sur chacun des deux génomes A et B qui seront exploitées pour identifier les segments homologues. Dans la suite du texte, ces similitudes qui établissent une correspondance locale entre les deux génomes seront appelées ancres. Comme l'illustre la figure 2, l'organisation interne de ces similitudes, au sein des segments homologues, est susceptible d'être modifiée au cours de l'évolution : modification de l'ordre ou de l'orientation [1]. La méthode de reconstruction exploitera le degré de conservation de l'organisation de ces ancres pour identifier les segments homologues.

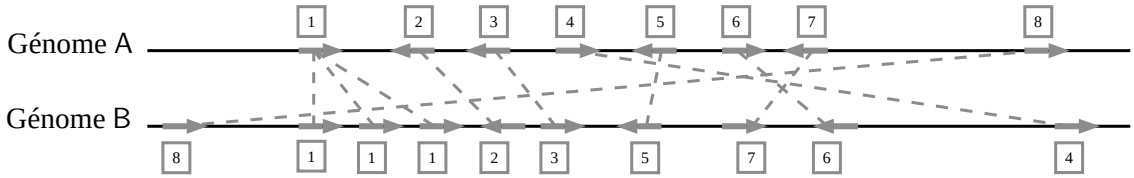


FIGURE 2 – Exemple de deux régions homologues entre le génome A et le génome B qui ont subi des évènements évolutifs au cours du temps. Une ancre est représentée par deux flèches de même numéro sur chacun des génomes. Le sens de la flèche représente le sens de lecture des gènes. Les ancres 3, 6 et 7 sont des ancres de polarités négatives. Les ancres 2, 4 et 5 sont des ancres de polarités positives. L'enchaînement des ancres 5, 6, 7 sur le génome A et 5, 7, 6 sur le génome B ; montre un changement d'ordre. Les ancres 3 et 5 se sont pas séparés par la même distance sur chacun des génomes.

2.2 Formalisme

Nous considérons que les génomes étudiés sont constitués chacun d'un seul chromosome, la méthode se généralisant facilement au cas multi-chromosomique (cf. infra remarque 2). Les génomes A et B sont donc modélisés par deux séquences de caractères, de tailles respectives n et m, sur un alphabet de quatre lettres $\{A, C, G, T\}$. Un segment d'un génome est entièrement déterminé par la donnée de l'index de la première lettre et de celui de la dernière lettre, il peut donc être noté par un intervalle $I = [deb, fin]$, avec $1 \leq deb < fin$ ($fin \leq n$ pour un intervalle de A et $fin \leq m$ pour un intervalle de B). On munit ces intervalles d'une relation d'ordre, on note $I < I'$ le fait que l'intervalle I' soit situé strictement après l'intervalle I. La distance entre 2 intervalles, définie comme la distance séparant ses éléments les plus proches, sera notée $dist(I, I')$.

Nous allons définir un graphe dont les sommets sont constitués des ancres et nous ramènerons le problème à la recherche de chemins dans ce graphe.

1. Il est important de noter que les éléments génétiques portés par l'ADN sont orientés, c'est en particulier le cas des gènes. L'orientation est définie par le sens de lecture de l'information génétique.

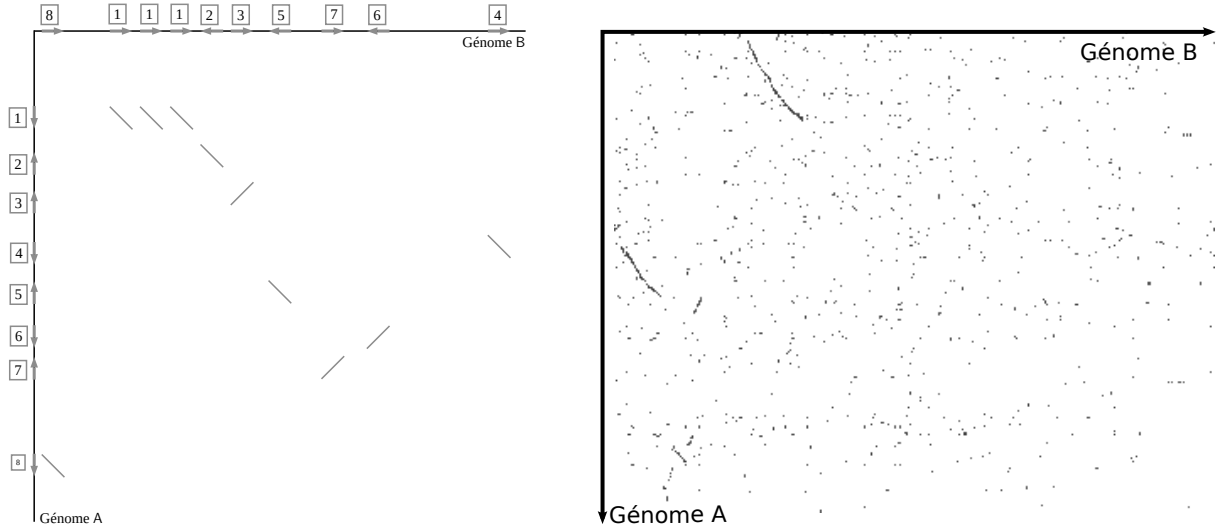


FIGURE 3 – A gauche le dotplot correspondant à la comparaison du génome A et B de la figure 2. A droite, une partie d'un dotplot d'une comparaison du génome d'*Arabidopsis thaliana* contre lui-même. Des régions homologues, formées d'un grand nombre d'ancres, apparaissent sous forme de diagonales sombres.

2.3 Les ancres comme sommets du graphe

La première étape de l'identification de segments homologues entre deux génomes A et B se fait par la recherche d'*ancres*, ou paires de régions génomiques dont la similarité est significative et qui s'appuie sur l'utilisation d'outils d'alignement locaux tels que Blast [1]. Dans tous les cas, l'entrée du processus de détection de segments homologues est formée d'un ensemble $\mathcal{A}$ d'ancres, une notion que nous précisons maintenant :

Une ancre a définit une correspondance entre un segment de A et un segment de B. On note a^{A} et a^{B} les intervalles correspondants. Une ancre est susceptible de relier des éléments d'orientations opposées, on dira alors qu'elle est de polarité négative, elle est de polarité positive sinon. Enfin, la qualité de la similarité observée entre les deux régions est capturée dans un score (fourni par les outils d'alignement locaux). Une ancre a_i est donc finalement déterminée par la donnée de quatre composantes : un intervalle a_i^{A} sur le génome A, un intervalle a_i^{B} sur le génome B, une polarité notée $a_i.pol$ et un score noté $a_i.s$.

On appelle *dotplot* la représentation graphique en deux dimensions de ces ancres (voir figure 3). Les deux axes du dotplot représentent les deux génomes, les intervalles $[1, n]$ et $[1, m]$, et chaque ancre a_i est représentée par un segment de droite dans cette représentation en deux dimensions dont les projections sur les deux axes correspondent exactement aux intervalles a_i^{A} et a_i^{B}. La pente de ce segment de droite est négative si a_i est de polarité positive, $a_i.pol = +$ (figure 3, ancres 6 et 7), et positive si a_i est de polarité négative, $a_i.pol = -$ (figure 3, ancres 2 et 3).

2.4 Le voisinage et la compatibilité des ancres

Un certain nombre de propriétés permettent de distinguer quand il est raisonnablement possible de supposer que deux ancres $a_i, a_j \in \mathcal{A}$ font partie d'un même segment ancestral.

Une première condition naturelle est d'exiger que les deux ancres ne soient pas trop éloignées phy-

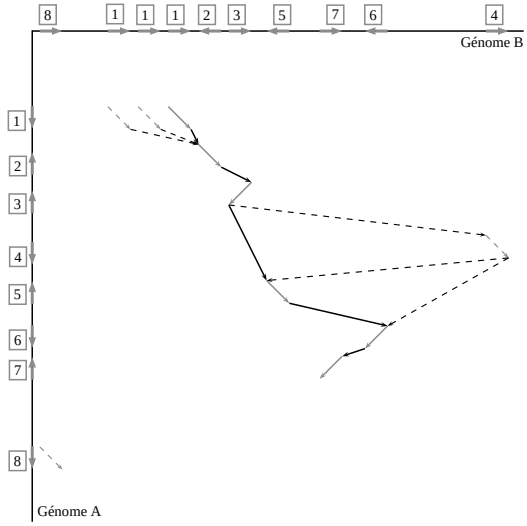


FIGURE 4 – Graphe de relation R construit à partir du dotplot à gauche de la figure 3. Les arcs et sommets non utilisés dans une chaîne sont en pointillés.

siquement l'une de l'autre sur les deux génomes, à une distance bornée notée $distMax^{\mathsf{g}}$ (dépendant du génome considéré).

$$dist(a_i^{\mathsf{g}}, a_j^{\mathsf{g}}) < distMax^{\mathsf{g}} \text{ pour } g = \mathsf{A} \text{ et } g = \mathsf{B} \tag{1}$$

Définition 1 (Ordre sur les ancres) *Soit deux ancres $a_i, a_j \in \mathcal{A}^2$. On dit que $a_i \prec a_j$ lorsque a_i et a_j respectent la condition (1) et que $a_i^{\mathsf{A}} < a_j^{\mathsf{A}}$, c'est à dire a_i précède a_j sur le génome A.*

Remarque 1 *$a_i \prec a_j$ n'implique pas $a_i^{\mathsf{B}} < a_j^{\mathsf{B}}$*

Remarque 2 *Pour généraliser au cas multi-chromosomique, il faut ajouter à la condition (1) : $a_i^{\mathsf{g}}.chrom = a_j^{\mathsf{g}}.chrom$ avec $a_i^{\mathsf{g}}.chrom$ le chromosome de a_i sur le génome g*

On peut noter que la relation $\prec$ définit un ordre strict partiel sur les ancres. Cet ordre partiel strict se capture naturellement sous la forme d'un graphe orienté, appelé *graphe de relation* :

Définition 2 (Graphe de relation) *Pour un ensemble d'ancres donné $\mathcal{A}$, le graphe de relation R associé à $\mathcal{A}$ est un graphe orienté dont l'ensemble des sommets est $\mathcal{A}$ et qui contient l'arc (a_i, a_j) si et seulement si $a_i \prec a_j$.*

Les arcs de ce graphe seront de plus pondérés et nous préciserons ces pondérations au paragraphe 2.8.

La relation $\prec$ étant un ordre, ce graphe est sans circuit. Tout chemin dans ce graphe représente une succession d'ancres qui respectent la condition (1) et qui sera appelé une chaîne par la suite. Toute chaîne définit donc une paire de segments homologues potentielle. Afin de différencier les chaînes, un score sera associé à chaque arc et sommet de R avec l'idée que plus le score est élevé, plus il est vraisemblable que l'élément appartienne à une paire de segments homologues.

Des contraintes supplémentaires peuvent être imposées pour les ancres qui participent à un segment ancestral : une contrainte de polarité (toutes les ancres doivent avoir la même polarité) ou une contrainte sur la conservation de l'ordre des ancres sur les deux génomes (l'ordre doit être conservé si la polarité est positive et totalement inversé le cas contraire). Un certain nombre d'outils existants imposent ces contraintes supplémentaires. Afin de nous placer dans le cas le plus général, nous ne les utiliserons pas mais l'ensemble de ce travail s'applique de façon immédiate si l'on souhaite exploiter ces propriétés supplémentaires et l'implémentation permet d'imposer ces conditions supplémentaires.

2.5 Les chaînes et les ancres

Une chaîne est un chemin dans le graphe de relation R. Par définition de la relation $\prec$, toute chaîne c traverse des ancres dont les intervalles se succèdent sur le génome A (mais pas nécessairement sur le génome B). A chaque chaîne c, et pour chaque génome $\mathsf{g} \in \{\mathsf{A}, \mathsf{B}\}$, on peut associer un ensemble d'informations : le début de la chaîne $c^{\mathsf{g}}.deb$, défini par la plus petite position d'une ancre dans c sur g, et sa fin $c^{\mathsf{g}}.fin$ (définie à partir de la plus grande position, on a donc $c^{\mathsf{g}}.deb < c^{\mathsf{g}}.fin_x$). Comme pour les ancres, une chaîne c définit donc deux intervalles sur chacun des génomes, notés c^{A} et c^{B}. Le nombre d'ancres dans la chaîne est noté $c.n$.

À partir de l'ensemble des positions de début et de fin de l'ensemble des ancres traversées par une chaîne, il est possible de calculer un coefficient de corrélation de Bravais-Pearson entre les coordonnées sur le génome A et sur le génome B. Ce coefficient, noté $c.corr$, renseigne sur le degré de dépendance linéaire entre les coordonnées et fournit une mesure de la conservation de l'organisation des ancres du segment sur les deux génomes. Le signe de ce coefficient définit la polarité $c.pol$ d'une chaîne c. Enfin, le score d'une chaîne $c.s$ est défini comme la somme des scores des éléments (sommets et arcs) qui la composent.

Tout comme pour les ancres, il est possible de définir un ensemble de propriétés destinées à capturer les chaînes qui sont les plus représentatives de paires de segments homologues. Ces conditions sont appelées par la suite conditions de *validité*. Une première série de conditions vise à éliminer les chaînes trop petites (contenant peu d'ancres) ou de trop mauvaises qualité (de score faible ou avec une coefficient de corrélation trop faible) :

Soit $minAncre$ le nombre minimum d'ancres d'une chaîne, $minScore$ le score minimum d'une chaîne et $minCorr$ le coefficient de corrélation minimum d'une chaîne, une chaîne valide doit respecter les conditions suivantes :

$$\begin{aligned} c_i.s &\geq minScore \\ c_i.n &\geq minAncre \\ c_i.corr &\geq minCorr \end{aligned} \tag{2}$$

Le problème de recherche de segments homologues entre deux génomes pourrait alors se ramener à la production d'un ensemble $\mathcal{C}$ de chaînes tirées de R, qui respectent les conditions précédentes et dont la somme des scores est maximum. Il est toutefois nécessaire de rajouter un ensemble de conditions liant entre elles les différentes chaînes apparaissant dans l'ensemble de chaînes $\mathcal{C}$.

Ainsi, une ancre ne peut pas appartenir à deux chaînes distinctes. Chaque ancre est en effet l'image d'un unique évènement de duplication (une ancre implique deux régions similaires seulement) et ne peut pas servir à en expliquer plusieurs.

$$\forall c, c' \in \mathcal{C}, \text{Si } a \in c \quad \text{alors } a \notin c' \tag{3}$$

De même, un couple de chaînes c_i, c_j de $\mathcal{C}$ connectées dans R ne peuvent pas se chevaucher à la

fois sur le génome A et sur le génome B. La duplication d'une région implique en effet la création d'une région qui n'est l'image que d'une unique origine.

$$(c_i^{\mathsf{A}} \cap c_j^{\mathsf{A}}) = \varnothing \quad \text{ou} \quad (c_i^{\mathsf{B}} \cap c_j^{\mathsf{B}}) = \varnothing \tag{4}$$

Ces deux conditions supplémentaires montrent que l'ensemble de chaînes qu'il faut produire n'est pas un ensemble de chaînes indépendantes mais qu'il forme un tout.

2.6 Le cas des duplications segmentales

Comparer un génome à lui-même pour rechercher des duplications segmentales, impose de vérifier qu'une région du génome ne participe pas plusieurs fois à une même chaîne : à la fois en tant que séquence du génome A et en tant que séquence du génome B. Ainsi des contraintes supplémentaires sont nécessaires.

La condition (4) précédente se spécialise. Une chaîne c qui met en correspondance des régions d'un même génome, ne peut pas se chevaucher elle-même car le résultat d'une duplication ne peut chevaucher son origine.

$$c_i^{\mathsf{A}} \cap c_i^{\mathsf{B}} = \varnothing \tag{5}$$

Cette condition empêche de fait la construction d'une chaîne unique qui correspondrait à une région dupliquée trois fois ou plus consécutivement (phénomène classique et appelé *duplications en tandem*). Plusieurs chaînes seront alors nécessaires.

Cette condition au niveau des chaînes permet de construire une condition nécessaire au niveau des ancres : deux ancres qui mettent en relation des régions ne peuvent pas appartenir à une même chaîne si la simple chaîne c formée de ces deux ancres définit deux régions qui se chevauchent.

Soit a_i et a_j deux ancres satisfaisant $a_i \prec a_j$ et soit c la chaîne formée des deux ancres a_i et a_j. Alors :

$$c^{\mathsf{A}} \cap c^{\mathsf{B}} = \varnothing. \tag{6}$$

Dans le cas d'analyse de duplications segmentales, cette condition sera donc directement mobilisée en supprimant dans R tout arc reliant deux ancres a_i et a_j violant cette propriété. Cette condition est nécessaire mais non suffisante.

2.7 Problème d'optimisation

Une fois le graphe R de relations construit, le problème de recherche de régions homologues entre A et B se ramène donc à la recherche d'un ensemble de chemins de score global maximum et vérifiant un ensemble de propriétés (au moins les propriétés (1), (2), (3) et (4)). La taille de cet ensemble est inconnue *a priori*.

La contrainte (1), qui exige une distance maximale entre ancres successives, est immédiate à prendre en compte dans le graphe de relation R et est donc satisfaite dans la majorité des outils existants, comme DAGchainer [8]. Les autres contraintes sont plus délicates à traiter. L'approche suivie par DAGchainer consiste à résoudre le problème de la construction d'un ensemble de chaînes de score maximum de façon gloutonne et en exploitant les contraintes restantes pendant l'algorithme, voire *a posteriori*. Pour cela, un premier chemin de score maximum est cherché, puis toutes les ancres du chemin trouvé sont retirées du graphe de relation R. La procédure est répétée tant que les chaînes

trouvées ont un score considéré comme suffisant. La suppression des ancres entre chaque étape permet de garantir que la contrainte (3) est satisfaite. Les autres contraintes sont ignorées ou font l'objet d'un filtrage *a posteriori* (en éliminant les chaînes qui les violent). Cette approche gloutonne ne garantit naturellement pas de construire un ensemble de chaînes dont le score total, sur toutes les chaînes, est maximum car les chaînes ne sont pas indépendantes les unes des autres et la maximisation du score d'une chaîne se fait au détriment du score des autres chaînes de l'ensemble, comme le montre un simple exemple :

Exemple 1 *Soient quatre chaînes c_1, c_2, c_3 et c_4 de score $S(c_1) = 100$, $S(c_2) = -98$, $S(c_3) = 99$ et $S(c_4) = 1$. Si c_1, c_2, c_3 se suivent, alors il existe une chaîne c de score $S(c) = 101$ construite à partir de ces trois chaînes. L'approche gloutonne qui maximise le score de la chaîne courante donnera : $\mathcal{C} = \{c, c_4\}$ avec $S(\mathcal{C}) = 102$. Il existe naturellement un ensemble de chaînes de meilleur score $\mathcal{C}' = \{c_1, c_3\}$ avec un score $S(\mathcal{C}') = 199 \approx 2 \times S(\mathcal{C})$. L'approche gloutonne a de plus tendance à construire des chaînes qui peuvent contenir des sous segments (ici c_2) de très mauvaise qualité.*

En pratique, tant que le graphe de relation est peu dense (avec un ratio d'arcs/sommets proche de 1), alors il y a sans doute peu de différences entre la maximisation gloutonne du score des chaînes et la maximisation du score de l'ensemble des chaînes (il y a peu de compétition pour les sommets/ancres). Cependant les choses sont différentes lorsque le graphe est dense. Notre graphe de relations, qui permet de capturer des mouvements évolutifs variés, crée un graphe plus dense que dans les précédents outils [9, 2, 8, 14, 15], et une optimisation plus globale du score de l'ensemble des chaînes prend alors de l'importance.

2.8 Fonctions de score

Pour différencier les ancres et les chemins les plus intéressants *a priori*, un score est associé à chaque sommet et à chaque arc. En ce qui concerne la fonction de score des ancres (sommets), nous nous appuyons sur le fait que ces ancres sont généralement détectées par des logiciels dédiés fournissant une e-value (l'espérance du nombre de régions similaires de cette qualité qui devraient exister du seul fait du hasard). Seules les ancres de e-value très faible (inférieure à 10^{-4}) sont habituellement conservées. Une ancre a_i de e-value $a_i.s$ se voit attribuée un score

$$S(a_i) = \min\{-log_{10}(a_i.s), 250\}$$

Le seuil de 250 permet de borner les scores extrêmes générés par des probabilités considérées comme nulles sur un flottant. On favorise donc la création de chaînes contenant des ancres solides.

En ce qui concerne la fonction de score des régions séparant les ancres (arcs), le score choisi favorise les paires d'ancres séparées par une distance courte. Pour favoriser les ancres dont les positions relatives sont cohérentes avec une duplication simultanée (sans insertion ou délétion supplémentaire), la distance choisie favorise les diagonales. Ainsi, pour deux ancres a_i et a_j séparées par $\Delta^{\mathsf{A}} = (a_j^{\mathsf{A}}.deb - a_i^{\mathsf{A}}.fin)$ et $\Delta^{\mathsf{B}} = (a_j^{\mathsf{B}}.deb - a_i^{\mathsf{B}}.fin)$, le score utilisé est :

$$S(a_i \prec a_j) = -\Big(\Delta^{\mathsf{A}} + \Delta^{\mathsf{B}} + |\Delta^{\mathsf{A}} - \Delta^{\mathsf{B}}|\Big)/2 \tag{7}$$

La distance corrrespondante est égale à la distance euclidienne pour les horizontales ou les verticales sur un dotplot mais divise la distance euclidienne par $\sqrt{2}$ pour les diagonales.

3 Algorithme

Nous cherchons un ensemble de n chemins dans R, ne partageant aucun sommet, et de score global maximum.

Pour résoudre efficacement ce problème fortement combinatoire (le simple nombre de chemins dans le graphe de relations peut croître de manière exponentielle avec la taille du graphe), nous exploitons une relation forte entre flots dans un réseau où toutes les capacités sont de 1 d'une part et ensembles de chemins reliant la source du réseau au puits d'un réseau d'autre part. Nous montrons ainsi que la recherche d'un ensemble de chemins ne partageant aucun sommet et de score maximum dans le graphe de relation peut se ramener à la recherche d'un flot de coût minimum dans un réseau dérivé du graphe de relations où les coûts des arcs seront égaux à l'opposé des scores du graphe d'origine.

Nous rappelons les définitions essentielles (réseau, flot, valeur d'une flot, coût d'un flot...).

3.1 Réseau et flot de coût minimum

Le problème de flot de coût minimum est défini sur un *réseau de transport* :

Définition 3 (Réseau de transport) *Un réseau de transport est un graphe orienté* $T = (S, A, C, W, \mathbf{s}, \mathbf{p})$ *dans lequel* S *et* A *représentent l'ensemble des sommets et arcs du graphe. Pour chaque arc* $a = (i, j) \in A$*, on dispose de la capacité de transport* $C(a) \in \mathbb{N}$*, notée aussi* c_{ij} *et du coût de transport* $W(a)$ *d'une unité de flot, aussi notée* w_{ij}*.* S *contient 2 sommets identifiés* $\mathbf{s}$ *(resp.* $\mathbf{p}$*) de degré entrant (resp. sortant) égal à zéro.*

Pour $u \in S$, on note $\Gamma^+(u)$ (resp. $\Gamma^-(u)$) l'ensemble des arcs ayant pour origine (resp. extrémité) u.

Définition 4 (Flot, valeur, coût) *Dans un réseau de transport* T*, un flot est une application* $f : A \to \mathbb{N}$ *(nous noterons* f_{ij} *la valeur du flot sur l'arc* (i, j)*) vérifiant :*

1. *Capacité :* $\forall (i, j) \in A, \quad 0 \leq f_{ij} \leq c_{ij}$
2. *Conservation du flot :* $\forall i \in S, i \notin \{\mathbf{s}, \mathbf{p}\} \quad \sum_{j \in \Gamma^-(i)} f_{ji} = \sum_{j \in \Gamma^+(i)} f_{ij}$
3. *Flot puits/source :* $\sum_{j \in \Gamma^+(\mathbf{s})} f_{\mathbf{s}j} = \sum_{j \in \Gamma^-(\mathbf{p})} x_{j\mathbf{p}} = V(f)$

$V(f)$ *est appelée la valeur du flot* f*. On définit le coût d'un flot* f *comme* $C(f) = \sum_{(i,j) \in A} w_{ij}.f_{ij}$.

Soit $\mathbf{F}$ l'ensemble des flots de valeur n, le problème de recherche d'un flot de valeur n de coût minimum consiste à trouver un flot f

$$f = arg \min_{f \in \mathbf{F}} C(f) \tag{8}$$

3.2 Représentation du graphe de relation sous forme de réseau

Dans un premier temps, nous transformons le graphe de relation R introduit dans la section précédente en un réseau de transport T_R. Avant toute chose, tous les scores du graphe R sont transformés en un coût de valeur opposée afin de ramener le problème de maximisation de score à un problème de minimisation de coût.

Les sommets du graphe de relation étant porteurs d'une information de coût, nous transformons, de façon classique, tout sommet a_i de R en une paire de sommets a_i et a'_i reliés par un arc allant de a_i à a'_i. Le coût $w_{ii'}$ associé à cet arc est égal à $-S(a_i)$.

Enfin, les sommets **s** et **p** sont ajoutés. **s** est relié à tous les sommets a_i de R par un arc de coût $w_{\mathbf{s}i} = 0$. De même, tous les sommets $a_{i'}$ ajoutés à l'étape précédente sont reliés à **p** par un arc de coût $w_{i'\mathbf{p}} = 0$. Toutes les capacités sont fixées à 1.

Propriété 1 *Dans T_R, tout flot entier f de valeur n et de coût $C(f)$ correspond à un ensemble de chemins ne partageant aucun sommet (propriété (3)) dans R et de score $-C(f)$.*

Preuve : Soit un flot entier de valeur n dans T_R, dont tous les arcs ont une capacité de 1. Chaque sommets a_i de R étant représenté par un arc dans T_R, chaque sommet de T_R (en dehors de **s** et **p**) a soit un degré entrant égal à 1, soit un degré sortant égal à 1. Il reçoit donc une unité de flot d'un seul de ses arcs entrants et la diffuse sur un seul des arcs sortants. Globalement, le flot de valeur n est donc formé d'un ensemble de n chemins $(\mathbf{s}, a_{i_1}, a_{i'_1}, \ldots, \mathbf{p})$ ne partageant aucun arc entre eux. Les sommets a_i de R étant représentés par un arc dans T_R, après suppression des sommets $a_{i'}$, ces n chemins définissent n chemins ne partageant aucun sommet entre eux dans R.

Réciproquement, pour tout chemin $(a_{i_1}, \ldots, a_{i_k})$ issu d'un ensemble donné de n chemins ne partageant pas de sommets dans R, on peut construire un chemin $(\mathbf{s}, a_{i_1}, a_{i'_1}, \ldots, a_{i_k}, a_{i'_k}, \mathbf{p})$ dans T_R. Les chemins obtenus ne partagent aucun arc et l'ensemble de ces arcs définit donc un flot de valeur n dans T_R.

Dans les deux cas, le score de l'ensemble des chemins, égal à la somme des scores des arcs et sommets traversés dans R est bien égal à l'opposé du coût du flot dans T_R (le flot étant d'une unité sur tout arc porteur de flot). □

On constate donc qu'il suffit de trouver un flot de coût minimum dans T_R pour trouver un ensemble de chemins de score optimal et ne partageant aucun sommet (ancre) dans R. L'algorithme de Busacker et Gowen [6, 5, 16, 7] permet de calculer un flot maximum de coût minimum par une procédure itérative. Dans notre cas, le flot maximum ne nous intéresse pas car il correspond à un flot de valeur $|A|$, traversant chaque ancre a_i indépendemment. Nous avons donc adapté l'algorithme de Busacker et Gowen. Dans notre cas, cet algorithme augmente la quantité de flot de 1 à chaque itération.

Propriété 2 *À chacune des itérations de l'algorithme, un flot de coût minimum est fourni par l'algorithme de Busacker et Gowen.*

A chaque itération, le flot est augmenté de 1 et cette propriété, démontrée par Gowen et Busacker [6] et Giroudeau [11], nous permet d'obtenir, à l'itération n, un flot de valeur n et de coût minimum. Chaque itération est en $O(sr)$ avec s le nombre de sommets et r le nombre d'arcs d'un graphe et l'algorithme s'exécute en $O(nsr)$. En pratique, le graphe de relation R n'étant pas connexe, nous appliquons la transformation précédente à chaque composante connexe et obtenons m *réseaux de transport*, où m est le nombre de composantes connexes de R. Ce qui permet d'exprimer la complexité de l'algorithme en $O(ns_{max}r_{max})$ avec r_{max} et s_{max} correspondant à la composante connexe la plus importante (voir algorithme 1). Cet algorithme définit la base de notre méthode de détection appelée ReD (**Re**gion **D**upliquées).

Algorithme 1 :

Données : $\mathcal{A} = \{a_1 \dots a_{|\mathcal{A}|}\}$ un ensemble d'ancres
Résultat : $C = \{c_1 \dots c_k\}$ un ensemble cohérent de chaînes de coût minimum, avec $k \leq n$
début
  Créer le graphe de relation R, [définition 2]
  Chercher les m composantes connexes R_i de R et créer les graphes de Transport T_{R_i}, [section 3.2]
  Rechercher le chemin augmentant de coût min, avec Bellman-Ford [11], sur chaque graphe T_{R_i}
  tant que *le critère d'arrêt n'est pas satisfait, [section 3.3]* **faire**
    Trouver le graphe T_{R_i} qui a le chemin augmentant de coût min
    Exécuter une itération de plus de Busacker et Gowen [6] sur T_{R_i}
    Rechercher le chemin augmentant de coût min, avec Bellman-Ford [11], sur le graphe T_{R_i}
  Reconstruire et filtrer les chaînes, [propriété 1] et [section 3.3]
fin

3.3 Les chaînes non valides

Contrairement à l'approche gloutonne, l'approche par flot de coût minimum présente l'avantage d'intégrer l'optimisation d'un score global, la prise en compte de la contrainte (1) de proximité des ancres (via le graphe R) mais aussi la condition (3) de non partage des ancres entre chaînes (via les capacités unitaires) et la condition (6) (dans le cas des duplications segmentales, en éclaircissant le graphe R).

Elle n'empêche cependant pas de produire des chaînes violant les conditions de non chevauchement (4) et (5) (dans le cas des duplications segmentales). Dans la pratique, certains outils, comme OSFinder [15], suppriment toutes les ancres intersectant le rectangle défini par la dernière chaîne créée dans un processus glouton (mais cette condition n'est cependant pas suffisante pour éviter totalement le problème : un arc peut encore chevaucher le rectangle). Dans DAGchainer [8], la violation de la contrainte (4) est négligée et les chaînes qui ne respectent pas la condition (5) sont déclarées en *tandem* et supprimées. DAGchainer viole aussi la contrainte (3) en exécutant deux passes de son algorithme, la première (resp. la deuxième) pour trouver les chaînes de polarités positives (resp. négatives).

Dans notre cas, les chaînes produites par l'algorithme et qui violent les conditions (4) et (5) sont classés en deux catégories. On distingue ainsi les chaînes dites "non valides temporaires" des autres. Ces chaînes sont non valides, mais contiennent au moins une sous-chaîne qui satisfait les conditions (4) et (5).

L'algorithme de Busaker et Gowen poursuit alors ses itérations (et la valeur du flot est incrémentée) tant que l'une des deux conditions suivantes est satisfaite : (1) les chemins produits par l'algorithme ont un coût supérieur à un paramètre *minScore* et (2) il existe dans les chemins représentés dans le flot courant un chemin *non valide temporaire*. Le paramètre *minScore* définit le score minimum attendu d'un chemin.

A la différence des algorithmes gloutons employés traditionnellement dans le domaine, notre algorithme produit donc de façon garantie un ensemble de chemins vérifiant les équations (3) (une ancre appartient à un seul chemin) et (4) (non chevauchement sur les deux génomes). La première propriété est garantie par l'algorithme de flot, et la seconde par la deuxième partie du critère d'arrêt utilisé (existence de chemins "non valides temporaires").

La seule propriété qu'il reste à vérifier est donc la propriété (2), exigeant que les chemins formés définissent des régions suffisamment "solides" (formées d'un nombre suffisant d'ancres, sans arc trop coûteux et raisonnablement linéaires). Notre implémentation se contente de filtrer toutes les chaînes violant cette propriété.

3.4 Estimation des paramètres

Afin de rendre l'algorithme plus robuste et facile à paramétrer, on normalise d'abord les scores des sommets et les scores des arcs indépendemment. Le score moyen des arcs est fixé arbitrairement à $-1\,000$ (pour rester entier) et celui des sommets à $1\,000.\alpha$ où α est un paramètre de contraste entre ancres (sommets) et arcs les reliant. Par la suite α est fixé à 5.

L'ensemble de chemins retournés par l'algorithme dépend alors encore de quatre paramètres : $DistMax_{\mathsf{A}}, DistMax_{\mathsf{B}}, minAncre, minScore$. Les paramètres $DistMax_{\mathsf{A}}, DistMax_{\mathsf{B}}$ sont estimés à l'aide de l'équation 9 ci-dessous, appliquée aux données fournies en entrée. Soit D_g la somme des longueurs entre deux ancres consécutives sur le génome g :

$$DistMax_g = \frac{D_g}{\sqrt{2|A|}} \tag{9}$$

$D_g/\sqrt{|A|}$ constitue une approximation de la distance euclidienne moyenne entre deux ancres distribués aléatoirement sur la région bidimensionelle d'un dotplot [14]. Le facteur $\sqrt{2}$ tient compte de la distance particulière utilisée dans notre cas.

Enfin, le score minimum d'un chemin $minScore$ est lié, via un paramètre β, au score moyen d'un plus petit chemin acceptable (formé de $minAncre$ ancres de score moyen $1000.\alpha$ et de $minAncre - 1$ arcs de score moyen 1000 les reliant) :

$$minScore = \beta.\Big(\alpha.1000.minAncre + 1000.(minAncre - 1)\Big) \tag{10}$$

Dans la suite β est fixé à 4, le nombre minimum d'ancres dans une chaîne $minAncre = 3$ et le coefficient de corrélation minimum $minCorr = 0,8$.

4 Évaluation expérimentale

Nous avons évalué notre algorithme dans le cas de la recherche de duplications segmentales sur le génome d'*Arabidopsis thaliana* et de régions homologues entre le génome de *Glycine max* et de *Medicago truncatula*. Sur chacun des cas, nous présentons d'abord une comparaison entre la méthode gloutonne et la méthode par flot de coût minimum, toutes choses étant égales par ailleurs (fonctions de scores, graphe de relation...). Dans un second temps, sur les même génomes, nous comparons l'outil ReD qui implémente la méthode par flot de coût minimum avec les logiciels DAGchainer [8] et OSfinder [15]. DAGchainer est le logiciel le plus cité utilisé dans les analyses génomiques. Nous y avons ajouté OSfinder en tant que logiciel très récent.

Pour alimenter ces outils, il faut détecter les ancres. Pour cela nous avons utilisé le logiciel BLASTP [1] en comparant les protéines du génomes A aux protéines de génomes B en ne conservant que les résultats ayant une *e-value* de 10^{-5} au plus. Pour réduire les lignes verticales et horizontales qui bruitent le dotplot, un filtre est ajouté pour supprimer toutes les paires de gènes qui possèdent au moins un gène présent dans plus de 50 (par défaut) paires de gènes.

Dans le cas de la recherche de duplications segmentales sur le génome d'*Arabidopsis thaliana* le graphe de relations comporte 165977 ancres et 487932 arcs, 12557 composantes connexes et la composante connexe la plus grande est composé de sommets 1328 et arcs 93078.

4.1 Comparaison avec la méthode gloutonne

L'approche par flot et gloutonne ont été comparées sur le problème de la recherche d'un nombre de chaînes fixées, en ignorant les contraintes de validité (2, 4 et 5). Dans notre approche le nombre de chaînes correspond exactement au nombre d'itérations. Dans tous les cas, le graphe de relation R et les fonctions de score sont identiques. Les performances sont comparées à l'aide de trois critères : le *score* de l'ensemble de chaînes produit ; le *coefficient de corrélation* évaluant la linéarité des chaînes produites et la *longueur* totale des chaînes (somme des longueurs des intervalles couvrant les deux séquences génomiques). Un score élevé, une forte linéarité et une large couverture des génomes est préférable sachant que ces critères sont souvent antagonistes.

Arabidopsis vs Arabidopsis		Score	Coef. de corrélation			Longueur en kb		
Méthode	Nombre de chaînes		min	moy	max	min	moy	max
Gloutonne	100	$21,2 \cdot 10^6$	0,15	0,93	1	231	2341	8968
	200	$27,7 \cdot 10^6$	0,02	0,82	1	61	1545	8968
	300	$32,6 \cdot 10^6$	0,02	0,84	1	41	1261	8968
Flot	100	$21,6 \cdot 10^6$	0,15	0,94	1	669	2540	8150
	200	$28,4 \cdot 10^6$	0,12	0,91	1	85	1741	8150
	300	$33,5 \cdot 10^6$	0,01	0,87	1	60	1435	8968

TABLE 1 – Différences au niveau du score, du cofficient de corrélation et de la longueur des chaînes entre la méthode *gloutonne* et la méthode de *flot*, dans le cas de la recherche de duplications segmentales sur le génome *Arabidopsis thaliana*.

Les résultats, présentés dans les tableaux 1 et 2 montrent que l'approche gloutonne fournit des scores inférieurs et que la différence entre les deux approches augmente avec le nombre de chaînes construites. L'approche par flot de coût minimum permet d'autre part de construire des chaînes plus longues (ajout d'ancres aux extrémités des chaînes) avec une meilleure linéarité : les choix locaux réalisés par l'approche gloutonne pénalisent la qualité globale des chaînes.

Glycine vs Medicago		Score	Coef. de corrélation			Longueur en kb		
Méthode	Nombre de chaînes		min	moy	max	min	moy	max
Gloutonne	200	$21,9 \cdot 10^6$	0,07	0,93	1	261	1951	16900
	400	$30,0 \cdot 10^6$	0,00	0,92	1	233	1357	16900
	600	$36,0 \cdot 10^6$	0,00	0,90	1	159	1092	16900
Flot	200	$21,9 \cdot 10^6$	0,08	0,95	1	255	2051	16900
	400	$30,2 \cdot 10^6$	0,10	0,93	1	242	1464	16900
	600	$36,3 \cdot 10^6$	0,00	0,92	1	159	1197	16900

TABLE 2 – Différences au niveau du score, du cofficient de corrélation et de la longueur des chaînes entre la méthode *gloutonne* et la méthode de *flot*, dans le cas de la recherche de régions homologues entre le génome de *Glycine max* et et le génome de *Medicago truncatula*.

4.2 Comparaison avec des logiciels existants

La comparaison avec des logiciels existants n'est pas évidente. Tout d'abord parce qu'il n'existe pas de jeux de données où la vérité est connue. Mais aussi car chaque logiciel intègre ses propres méthodes

de filtrage pour les données en entrées, des fonctions de score différentes voire même un graphe de relation différent. Pour évaluer les logiciels, en dehors des mesures de longueur et de corrélation déjà réalisées, nous nous sommes appuyés également sur le caractère cohérent des ensembles de chaînes produits. En effet, la succession d'évènements de spéciation et de duplication ne peuvent pas créer deux relations d'homologies qui se chevauchent sur les deux séquences génomiques (condition (4)). Les chaînes violant cette condition sont comptabilisées dans la colonne *Chevauche*. Dans le cas de la comparaison d'un génome contre lui-même, une relation d'homologie entre deux régions qui s'intersectent viole également la condition (5). Dans ce cas, les chaînes sont proches de la diagonale du dotplot et sont donc comptabilisées dans la colonne *Diag*.

Arabidopsis vs Arabidopsis	Nombres de chaînes		
Logiciel	Total	Diag	Chevauche
DAGchainer	306	65	63
OSfinder	286	5	126
ReD	281	0	0

Glycine vs Medicago	Nombres de chaînes	
Logiciel	Total	Chevauche
DAGchainer	708	116
OSfinder	7864	2143
ReD	1216	0

TABLE 3 – Ces tableaux montrent le nombres de chaînes crées et le nombres de chaînes incohérentes.

Le tableau 3 nous montrent que seul ReD crée un ensemble de chaînes cohérent alors que la proportion de chaînes incohérentes crées par DAGchainer et OSfinder est comprise entre 10% et 50%. Il est naturellement possible de filtrer ces chaînes, mais le nombre de chaînes diminue d'autant. Une alternative consiste à trouver un sous-ensemble de chaînes cohérent mais ce n'est pas un problème simple, les chaînes incompatibles s'excluant mutuellement. ReD règle se problème en amont, dans son formalisme, et dans l'algorithme de flot associé. De plus nous remarquons que le nombre de chaînes d'OSfinder est anormalement élevé dans le tableau de droite. Cette différence est sans doute due au fait qu'OSFinder a été développé pour détecter les régions homologues et testé sur des génomes qui possèdent moins de duplications que les génomes de plantes utilisés.

5 Conclusion

Via l'utilisation jointe d'une modélisation sous la forme d'un graphe orienté pondéré et d'un algorithme de recherche de flot de coût minimum dans un graphe de transport idoine, l'outil ReD permet donc d'identifier des régions homologues entre génomes avec des garanties en termes de score et de respect des contraintes qui dépassent celles des outils existants.

Le passage d'une méthode gloutonne à une méthode de recherche de flot, plus complexe, reste tout à fait confortable en termes de temps de calcul. Sur une machine récente l'implémentation actuelle de ReD, en C++, prend de l'ordre de 15 minutes pour s'exécuter. À la différence des outils existants, ReD permet d'activer ou non les contraintes optionnelles d'ordre et d'orientation présentées dans la section 2.4 et reste le seul outil prenant en compte les contraintes spécifiques de non chevauchement qui apparaissent dans la détection de duplications segmentales. Les sorties de ReD ont été intégrées à l'outil de visualisation Narcisse [3] qui permet de visualiser et d'analyser les résultats de l'analyse des duplications segmentales dans plusieurs génomes de plantes sous forme graphique.

Pour autant, ReD n'est encore qu'une étape sur la voie qui mène à l'outil idéal de recherche de régions homologues. La prise en compte des contraintes de non chevauchement dans le critère d'arrêt n'est pas toujours idéale et peut parfois mener à une fragmentation excessive des chaînes dans les génomes fortement dupliqués. Enfin et surtout, il reste à évaluer les performances de ReD dans un contexte plus difficile et encore peu exploré : l'exploitation d'ancres détectées directement au niveau

de l'ADN qui est largement plus bruitée qu'au niveau des protéines (utilisé traditionnellement pour détecter les ancres). ReD pourra alors être utilisé avant le processus d'annotation des génomes (détection des régions correspondant aux protéines) et ses sorties pourraient alors alimenter ce processus de prédiction.

Références

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. *Basic local alignment search tool.*, volume 215. National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD 20894., October 1990.

[2] S. B. Cannon, A. Kozik, B. Chan, R. Michelmore, and N. D Young. Diaghunter and genopix2d : programs for genomic comparisons, large-scale homology discovery and visualization. *Genome Biology*, pages 853–860, September 2003.

[3] E. Courcelle, Y. Beausse, S. Letort, O. Stahl, R. Fremez, C. Ngom-Bru, J. Gouzy, and T. Faraut. Narcisse : a mirror view of conserved syntenies. *Nucleic Acids Res*, 36(Database issue) :D485–90, 2008.

[4] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, July 1999.

[5] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2) :248–264, 1972.

[6] R. G.Busacker and P. J. Gowen. A procedure for determining minimal-cost network flow patterns. In *ORO Technical Report 15*, 1961.

[7] A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *J. ACM*, 36(4) :873–886, 1989.

[8] B. J. Haas, A. L. Delcher, J. R. Wortman, and S. L. Salzberg. Dagchainer : a tool for mining segmental genome duplications and synteny. *Bioinformatics*, 20(18) :3643–3646, December 2004.

[9] S. Chakravarty P. P. Calabrese and J. T. Vision. Fast identification and statitistical evaluation of segmental homologies in comparative maps. *Bioinformatics*, 19 :i74–i80, January 2003.

[10] P. Pevzner and G. Tesler. Genome rearrangements in mammalian evolution : Lessons from human and mouse genomes. *Genome Research*, 13(1) :37–45, January 2003.

[11] R. Giroudeau. *Cours Algorithmique/complexité/Calculabilité*. 2009. Cours de Master, Univ. de Montpellier II. Disponible sur http ://www.lirmm.fr/ rgirou/enseignement/pageenseignement.html.

[12] C. Simillion, K. Vandepoele, Y. Saeys, and Y. Van de Peer. Building genomic profiles for uncovering segmental homology in the twilight zone. *Genome research*, 14(6) :1095–1106, June 2004.

[13] A. U. Sinha and J. Meller. Cinteny : flexible analysis and visualization of synteny and genome rearrangements in multiple organisms. *BMC Bioinformatics*, 8 :82+, March 2007.

[14] C. Soderlund, W. Nelson, and A. Shoemaker et al. Symap : A system for discovering and viewing syntenic regions of fpc maps. *Genome research*, 16 :1159–1168, December 2006.

[15] K. Popendorf T. Hachiya, Y. Osana and Y. Sakakibara. Accurate identification of orthologous segments among multiple genomes. *Bioinformatics*, 25, February 2009.

[16] É. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3) :247–255, 1985.

[17] K. Vandepoele, Y. Saeys, C. Simillion, J. Raes, and Y. Van De Peer. The automatic detection of homologous regions (adhore) and its application to microcolinearity between arabidopsis and rice. *Genome Res*, 12(11) :1792–1801, November 2002.

Une approche de choix résolu au sens de Jaffray dans les arbres de décision munis de probabilités imprécises

Gildas Jeantet, Olivier Spanjaard

LIP6-CNRS ; UPMC ; 104, avenue du Président Kennedy, F-75016 Paris, France
{jeantet,spanjaard}@lip6.fr

Résumé : *Dans cet article, nous nous intéressons à l'optimisation du critère d'Hurwicz dans un arbre de décision muni de probabilités imprécises. Nous proposons une procédure de choix résolu au sens de Jaffray [5] afin de déterminer une stratégie dont toutes les sous-stratégies sont à la fois proches de l'optimum (au sens d'Hurwicz) et E-admissible (c'est-à-dire qu'il existe un jeu de probabilités sur le sous-arbre tel que la sous-stratégie maximise l'espérance d'utilité). Cette procédure fait appel à la résolution de programmes linéaires pour évaluer les sous-stratégies selon le critère d'Hurwicz, et pour tester leur E-admissibilité. Des résultats numériques sont fournis qui illustrent le caractère opérationnel de l'approche proposée.*

Mots-Clés : *décision séquentielle, probabilités imprécises, choix résolu, critère d'Hurwicz.*

1 Introduction

La décision dans l'incertain est un des domaines de recherche principaux en théorie de la décision, du fait de ses nombreuses applications (comme par exemple en diagnostic médical, système de contrôle robotique, décision stratégique, jeux...). La décision dans l'incertain traite des problèmes où les conséquences d'une décision dépendent d'événements incertains. Lorsque cette incertitude est probabilisée, on parle alors de décision dans le risque. Dans ce cadre, une décision peut alors être caractérisée par une *loterie* sur les conséquences possibles. Un critère de décision classique pour comparer les loteries (et donc les décisions) est l'espérance d'utilité (EU), introduite par von Neumann et Morgenstern [13]. Dans ce modèle, une *fonction d'utilité* u (propre à chaque décideur) assigne une valeur numérique à chaque conséquence. L'évaluation d'une loterie est alors réalisée via le calcul de son espérance d'utilité (à maximiser). Cependant, lorsque plusieurs experts ont des points de vue divergents ou quand certaines données empiriques manquent, il n'est pas évident d'éliciter des probabilités précises pour chacun des événements. Une manière naturelle de prendre en compte explicitement cette difficulté est d'utiliser des intervalles de probabilités en lieu et place des valeurs scalaires de probabilité. On parle alors de *probabilités imprécises*.

Comparer des décisions revient alors à comparer des loteries imprécises, c'est-à-dire des loteries où plusieurs distributions possibles de probabilité sont prises en compte. Un décideur pessimiste prendra la décision qui maximise la pire espérance d'utilité possible. Ceci est connu sous le nom de critère Γ-*maximin*. Inversement, un décideur optimiste prendra la décision qui maximise la meilleure espérance d'utilité possible. C'est le critère Γ-*maximax*. Entre ces deux attitudes extrêmes, Jaffray and Jeleva ont récemment proposé d'utiliser le critère d'Hurwicz qui permet de rendre compte d'attitudes plus

modérées en s'intéressant à une combinaison linéaire des deux critères précédents [2]. Remarquons qu'Hurwicz a initialement introduit ce critère dans un contexte de décision en situation d'ignorance complète (c'est-à-dire lorsqu'aucune information n'est disponible quant aux probabilités d'occurence des différents événements), mais Jaffray et Jeleva ont décidé de conserver la dénomination de "critère d'Hurwicz" car il s'étend naturellement au cas des probabilités imprécises.

A notre connaissance, les difficultés algorithmiques liées à l'utilisation du critère d'Hurwicz dans les problémes de décision séquentielle ont été peu étudiées jusqu'à aujourd'hui. Il est en effet fréquent de rencontrer des problèmes de *décision séquentielle* où l'on ne prend pas une unique décision mais où l'on suit plutôt une *stratégie* (c'est-à-dire une séquence de décisions conditionnées par des événements) susceptible de mener à différentes conséquences possibles. Plusieurs formalismes de représentation peuvent être utilisés dans les problèmes de décision séquentielle, tels que les arbres décision hasard [10], les diagrammes d'influence [12] ou encore les processus décisionnels Markoviens [9]. Un arbre de décision est une représentation explicite d'un problème de décision séquentielle, tandis que les diagrammes d'influence ou les processus décisionnels Markoviens sont des représentations compactes qui permettent de traiter des problèmes de décision de plus grande taille. Il est important de remarquer que, dans tous ces formalismes, l'ensemble des stratégies potentielles est combinatoire (c'est-à-dire que sa taille augmente exponentiellement avec la taille de l'instance). Le calcul d'une stratégie optimale pour une représentation donnée et un critère de décision donné est par conséquent un problème algorithmique en soi.

Dans un arbre de décision muni de probabilités scalaires, il est bien connu qu'une stratégie optimale pour EU peut être calculée en temps linéaire par programmation dynamique. Ce n'est plus le cas lorsqu'on utilise des probabilités imprécises et le critère d'Hurwicz. En effet, avec ce critère, une stratégie optimale vue de la racine de l'arbre de décision peut comporter des stratégies sous-optimales dans certains sous-arbres. Dans un précédent article [3], nous avons montré que la détermination d'une stratégie optimale pour le critère d'Hurwicz est un problème NP-difficile dans le cas général, et nous avons proposé un algorithme de résolution opérationnel pour une classe particulière d'arbre de décision. Contournant cette difficulté, dans le cas des critères Γ-maximin et Γ-maximax, Kikuti *et al.* [1] ont présenté des algorithmes qui retournent des stratégies calculées par induction arrière dans l'arbre de décision, en remontant le temps du futur vers le présent, et en prenant systématiquement la décision qui maximise le critère choisi étant donné les anticipations sur les actions futures. Ce type de stratégie sera suivie par un décideur *sophistiqué*, c'est-à-dire qui est capable d'anticiper sur ses actions futures, et qui adopte de plus un comportement *conséquentialiste*, c'est-à-dire que ses décisions ne dépendent ni du passé, ni d'événements contrefactuels (autrement dit, les événements qui auraient pu se passer). Il a néanmoins été montré que, lorsque les probabilités sont imprécises, l'adoption d'un tel comportement, à la fois sophistiqué et conséquentialiste, rend le décideur vulnérable aux manipulations et plus généralement susceptible de suivre une stratégie "dominée" en un certain sens [4]. Pour cette raison, à la suite de Jaffray [5], nous relâchons ici l'hypothèse de conséquentialisme pour nous intéresser à la détermination d'une stratégie représentant un compromis entre les différents egos d'un décideur sophistiqué. Les *egos* représentent le même décideur dans les diverses situations de choix susceptibles d'apparaître au cours du processus de décision. Chaque ego a ses propres préférences sur les meilleures stratégies à suivre de son point de vue dans son sous-arbre (ces préférences sont induites par l'application du critère de décision choisi). La stratégie de compromis est une stratégie dont chacune des sous-stratégies qui la composent figure parmi l'ensemble des stratégies préférentielles de l'ego correspondant. Contrairement à l'approche développée dans notre précédent travail [3], le choix d'une stratégie ne repose pas uniquement sur le point de vue de l'ego à la racine (qui agit comme une sorte de dictateur), mais tient compte de l'ensemble des points de vue des différents egos. Le décideur suit alors cette stratégie de façon résolue, sans s'autoriser à en dévier au cours du processus de décision. Cette approche a été introduite par McClennen sous le nom de *choix résolu* [8].

Le papier est organisé comme suit. Nous introduisons tout d'abord quelques définitions préliminaires sur les probabilités imprécises, le critère d'Hurwicz et les arbres décision hasard. Après avoir souligné la difficulté de comparer des stratégies au sens du critère d'Hurwicz, nous proposons une procédure de choix résolu. Cette procédure fait en particulier appel à la résolution de programmes linéaires afin de garantir de bonnes propriétés à la stratégie retournée. Enfin, nous présentons des expérimentations numériques illustrant le caractère opérationnel de cette approche.

2 Préliminaires

2.1 Probabilités imprécises et critères de décision

Plusieurs modèles mathématiques de probabilités imprécises ont été proposé dans la littérature [14, 15]. Dans ces différents modèles, un intervalle de probabilité $[P^-(E), P^+(E)]$ est souvent défini pour chaque événement E. A la suite de Jaffray et Jeleva [2], nous supposons qu'il existe une *vraie* probabilité P_0 telle que $P_0(E) \in [P^-(E), P^+(E)]$ pour tout événement E. Pour comparer des loteries imprécises (c'est-à-dire des loteries munies de probabilités imprécises), on doit donc considérer un ensemble $\mathcal{P}$ de distributions de probabilité possibles. Ceci peut être rapproché de ce qui se fait pour comparer des solutions réalisables en optimisation robuste avec données intervalles [6]. La principale différence vient de ce que l'ensemble des distributions de probabilité possibles n'est pas le produit cartésien des intervalles de probabilité des événements élémentaires. Une distribution de probabilité doit en effet respecter les axiomes de Kolmogorov ($P(E) \geq 0$, $P(\Omega) = 1$, $P(E_1 \cup E_2 \cup \ldots) = P(E_1) + P(E_2) + \ldots$ pour des événements E_i disjoints deux à deux).

Différents critères de décision peuvent être utilisés pour comparer des stratégies en probabilités imprécises. Par exemple, considérons deux loteries f, g faisant intervenir trois événements E_1, E_2, E_3 disjoints deux à deux. Si E_1 (resp. E_2, E_3) se produit, f retourne -50 (resp. 0,100). Parallèlement, si E_1 (resp. E_2, E_3) se produit, g retourne -50 (resp. -30,130). Dans le modèle EU avec des probabilités précises, une loterie est évaluée selon son espérance d'utilité : pour f, celle-ci s'écrit $E(f) = P(E_1)u(-50)+P(E_2)u(0)+P(E_3)u(100)$. Supposons maintenant que les probabilités sont imprécises, avec $P_0(E_1) \in [0.2, 0.4]$, $P_0(E_2) \in [0.4, 0.6]$ et $P_0(E_3) \in [0.2, 0.3]$. L'ensemble $\mathcal{P}$ des distributions de probabilité possible est alors défini par $\mathcal{P} = \{P : P(E_i) \in [P^-(E_i), P^+(E_i)]\ \forall i,\ \text{et}\ \sum_i P(E_i) = 1\}$. Si le décideur cherche à se prémunir contre la pire espérance d'utilité possible, une loterie f est évaluée par la formule $\underline{\mathrm{E}}(f) = \min\{E(f, P) : P \in \mathcal{P}\}$, où $E(f, P)$ est l'espérance d'utilité de la loterie f pour la probabilité P. Ce critère est connu sous le nom de critère Γ-*maximin*. A titre d'illustration, comparons les loteries f et g selon ce critère. La distribution de probabilité P qui minimise $E(f, P)$ est donnée par $P(E_1) = 0.4$, $P(E_2) = 0.4$ et $P(E_3) = 0.2$ (cette distribution peut être calculée aisément par une procédure gloutonne [3]). Par conséquent, pour $u(x) = x$, on a $\underline{\mathrm{E}}(f) = 0.4 \times (-50) + 0.2 \times 100 = 0$. De façon similaire, on calcule la distribution P qui minimise $E(g, P)$: $P(E_1) = 0.2$, $P(E_2) = 0.5$ et $P(E_3) = 0.3$. On en déduit que $\underline{\mathrm{E}}(g) = -4$. La loterie f est donc préférée à g pour le critère Γ-maximin. Inversement, si le décideur cherche à maximiser la meilleure espérance d'utilité possible, une loterie f est évaluée selon la formule $\bar{E}(f) = \max\{E(f, P) : P \in \mathcal{P}\}$. Il s'agit du critère Γ-*maximax*. A titre d'illustration, considérons à nouveau les loteries f et g définies précédemment. La distribution de probabilité P qui maximise $E(f, P)$ est donnée par $P(E_1) = 0.2$, $P(E_2) = 0.5$ et $P(E_3) = 0.3$. Par conséquent, pour $u(x) = x$, on a $\bar{E}(f) = 20$. De façon similaire, on calcule la ditribution P qui maximise $E(g, P)$: $P(E_1) = 0.4$, $P(E_2) = 0.4$ et $P(E_3) = 0.2$. On en déduit que $\underline{\mathrm{E}}(g) = 30$. La loterie g est donc préférée à f pour le critère Γ-maximax. Ceci montre que les préférences sont bien évidemment très dépendantes du degré de pessimisme ou d'optimisme du décideur.

Pour cette raison, Jaffray and Jeleva [2] ont étendu le critère d'Hurwicz pour la décision en situation

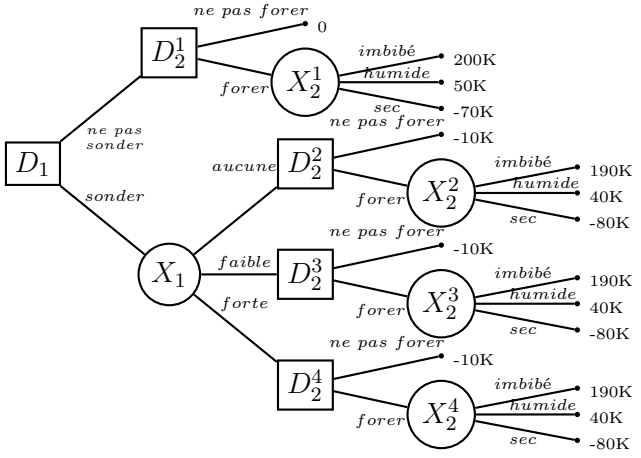
d'ignorance complète au cas des probabilités imprécises. Avec le critère d'Hurwicz, une loterie f est évaluée par la formule $\alpha\underline{E}(f)+(1-\alpha)\bar{E}(f)$. En d'autres termes, le décideur se focalise sur la pire et la meilleure espérance d'utilité possible et, selon son degré de pessimisme, accorde d'avantage de poids sur l'une ou l'autre. Ce critère se réduit à Γ-maximin pour $\alpha = 1$, et à Γ-maximax pour $\alpha = 0$. Quand on compare les loteries f et g précédentes selon le critère d'Hurwicz, f est préférée à g pour $\alpha > 5/7$, et g est préférée à f pour $\alpha < 5/7$. Etant donné un ensemble F de loteries définies sur un même espace d'événements élémentaires, notons que le critère d'Hurwicz garantit de sélectionner une loterie f^* non-dominée [2], c'est-à-dire qu'il ne peut exister une loterie $f \in F$ pour laquelle : $E(f, P) \geq E(f^*, P)$ pour toute distribution P et $E(f, P) > E(f^*, P)$ pour au moins une distribution P. Le sous-ensemble des loteries non-dominées dans F est bien évidemment particulièrement pertinent pour n'importe quel décideur rationel. Un autre sous-ensemble de loteries particulièrement intéressantes aux yeux d'un décideur est celui des loteries *E-admissible*. Cette notion a été introduite par Levi [7] et discutée par Seidenfeld [11]. Une loterie $f \in F$ est dite *E-admissible* s'il existe au moins une distribution de probabilité $P \in \mathcal{P}$ pour laquelle elle est maximale dans F au sens de l'espérance d'utilité. Cette dernière propriété garantie des propriétés désirables sur la qualité de la loterie dans l'ensemble F.

2.2 Arbres décision hasard munis de probabilités imprécises

On parle de problème de décision séquentielle lorsqu'on est confronté à un problème où l'on a à prendre une séquence de décision conditionnellement à des événements. Le formalisme des arbres décision hasard fournit une représentation simple et explicite des problèmes de décision séquentielle dans le risque. Il s'agit d'un arbre comportant trois types de nœuds : un ensemble $\mathcal{N}_D$ de nœuds de décision (représentés par des carrés), un ensemble $\mathcal{N}_C$ de nœuds de hasard (représentés par des cercles) et un ensemble $\mathcal{N}_U$ de nœuds d'utilité (les feuilles de l'arbre). A chaque nœud de décision (resp. nœud de hasard) est associé une variable de décision (resp. une variable aléatoire), dont le domaine correspond aux étiquettes des branches issues de ce nœud. Quand les probabilités sont imprécises, la probabilité précise qu'une variable aléatoire donnée prenne une valeur donnée est inconnue : on connaît seulement un intervalle de probabilité dans lequel elle est inclue. Les valeurs indiquées aux feuilles correspondent aux utilités des conséquences. Pour illustration, nous donnons ci-dessous un exemple de problème de décision séquentielle bien connu, et sa représentation sous forme d'un arbre de décision. Remarquons qu'on omet les orientations des arêtes quand on représente un arbre de décision.

Exemple 1 (problème du forage pétrolier [10]) *Une compagnie pétrolière doit décider de forer ou non un site donné. Dans ce but, elle doit d'abord décider si elle sonde ou non la structure géologique du site (decision D_1), ce qui coûte 10000€ et donne une meilleure estimation de la quantité de pétrole qui s'y trouve. Le résultat du sondage peut être vu comme une variable aléatoire X_1 qui peut prendre trois valeurs possibles :* aucune *si la sonde ne trouve aucune trace de pétrole,* faible *si la sonde trouve une faible trace de pétrole, ou* forte *si la sonde trouve une forte trace de pétrole. Ensuite, la compagnie doit décider si elle réalise un forage ou non (décision D_2), ce qui coûte 70000€. Finalement, si la compagnie décide de forer, le résultat du forage peut être vu comme une variable aléatoire X_2 (associée aux nœuds X_2^i pour $i = 1...4$) qui peut prendre trois valeurs possibles : le puit est* sec *(le revenu est nul),* humide *(120000€) ou* imbibé *(270000€). Ce problème peut être représenté par l'arbre de décision sur la gauche de la figure 1. Remarquons que la décision D_2 est dupliquée en plusieurs nœuds (nœuds D_2^1, D_2^2, D_2^3 et D_2^4) car elle peut être prise dans différents contextes (un sondage a été réalisé ou non, le résultat du sondage est encourageant ou non...).*

Avant de détailler les difficultés liées à l'utilisation de probabilités imprécises dans un arbre de décision $\mathcal{T}$, il nous faut introduire au préalable quelques notations qui nous seront utiles par la suite.



$P(X_2\|X_1)$	sec	humide	imbibé
aucune	[0.500,0.666]	[0.222,0.272]	[0.125,0.181]
faible	[0.222,0.333]	[0.363,0.444]	[0.250,0.363]
forte	[0.111,0.166]	[0.333,0.363]	[0.454,0.625]

X_1	aucune	faible	forte
$P(X_1)$	[0.181,0.222]	[0.333,0.363]	[0.444,0.454]

X_2	sec	humide	imbibé
$P(X_2)$	[0.214,0.344]	[0.309,0.386]	[0.307,0.456]

FIGURE 1 – Arbre de décision pour le problème du forage pétrolier.

Soit $X_1, \ldots, X_n$ l'ensemble des variables aléatoires figurant dans $\mathcal{T}$, et $X = \langle X_1, \ldots, X_n \rangle$ le vecteur aléatoire correspondant. Nous notons $\mathcal{D}(Y)$ le domaine d'une variable aléatoire (ou vecteur aléatoire) Y. De plus, étant donné un sous-ensemble $I \subseteq \{1, \ldots, n\}$ d'indices, nous désignons par X_I le vecteur aléatoire ayant pour composantes les X_i pour $i \in I$. En outre, étant donné un nœud de hasard N, nous désignons par $id(N)$ l'indice i de la variable aléatoire X_i associée au nœud N. De même, étant donné un noeud N, nous désignons par $\pi(N)$ l'ensemble des *indices* des variables aléatoires sur le chemin de la racine à N ($id(N)$ exclu si $N \in \mathcal{N}_C$). L'ensemble complémentaire de ce dernier est noté $\pi^c(N) = \{1, \ldots, n\} \setminus \pi(N)$. Enfin, nous désignons par $x(N)$ le vecteur des *valeurs affectées* aux variables aléatoires sur le chemin de la racine à N. Par exemple, sur l'arbre de la figure 1, $id(X_2^2) = 2$ car la variable aléatoire associée à X_2^2 est X_2, et $\pi(X_2^2) = \{1\}$ car seule la variable aléatoire X_1 figure sur le chemin menant de D_1 à X_2^2. Par conséquent, on a $X_{\pi(X_2^2)} = \langle X_1 \rangle$, et $x(X_2^2) = \langle aucune \rangle$ car il est nécessaire que X_1 prenne la valeur *aucune* pour arriver en X_2^2.

Quand les probabilités précises sont connues, chaque branche issue d'un nœud chance N est munie de la probabilité $P(X_{id(N)} = x | X_{\pi(N)} = x(N))$, où $x \in \mathcal{D}(X_{id(N)})$ représente la valeur correspondant à la branche. Remarquons que cette formule n'est valable que sous l'hypothèse que les décisions n'ont pas d'influence sur les distributions de probabilité des variables aléatoires (hypothèse faite dans ce papier). Par exemple, dans l'arbre de décision pour le problème du forage pétrolier, $P(X_2 = imbibé | D_1 = sonder, X_1 = aucune) = P(X_2 = imbibé | X_1 = aucune)$. Quand les probabilités sont imprécises, on suppose qu'une table de probabilités conditionnelles figure en chaque nœud de hasard de l'arbre de décision. Pour le problème du forage pétrolier, les tables de probabilités conditionnelles figurent à côté de l'arbre sur la figure 1.

Une *stratégie* consiste à fixer une valeur à chaque variable de décision conditionnellement à son passé. Nous représenterons une stratégies indifféremment comme un ensemble d'affectation de valeurs à des variables de décision ou comme l'ensemble des arcs de l'arbre correspondant à ces affectations. Par exemple, l'arbre de décision de la figure 1 comporte 10 stratégies réalisables, parmi lesquelles figure, entre autres, la stratégie $s = \{D_1 = sonder, D_2^2 = forer, D_2^3 = forer, D_2^4 = forer\}$ (remarquons que le nœud D_2^1 ne peut pas être atteint quand $D_1 = sonder$), qui peut aussi être notée $\{(D_1, X_1), (D_2^2, X_2^2), (D_2^3, X_2^3), (D_2^4, X_2^4)\}$. Dans notre cadre, une stratégie peut être associée à une loterie composée sur les utilités, où les probabilités des événements impliqués sont imprécises. Par exemple, la stratégie $s = \{D_1 = sonder, D_2^2 = ne\,pas\,forer, D_2^3 = forer, D_2^4 = forer\}$ correspond à la loterie composée retournant $-10K$ si $X_1 = aucune$, $190K$ (resp. $40K, -80K$) si $X_1 = faible$ ou $X_1 = forte$ et ensuite $X_2 = imbibé$ (resp. $humide, sec$). Comparer des stratégies se ramène donc à comparer des loteries composées. Etant donné un arbre de décision $\mathcal{T}$, l'évaluation d'une stratégie (plus précisément, de la loterie composée correspondante) selon le critère d'Hurwicz dépend de l'ensemble $\mathcal{P}_\mathcal{T}$ des distributions de probabilité possibles *sur l'arbre de décision* $\mathcal{T}$ (c'est-à-dire l'ensemble

des affectations de probabilités précises aux tables fournies avec $\mathcal{T}$). Cette évaluation est un problème combinatoire en soi du fait de la nature combinatoire de $\mathcal{P}_\mathcal{T}$.

3 Evaluation d'une stratégie selon le critère d'Hurwicz

Dans cette section, nous nous intéressons à l'évaluation d'une stratégie s selon le critère d'Hurwicz. Il est important de remarquer qu'une stratégie est caractérisée par l'ensemble $\mathcal{U} \subset \mathcal{N}_U$ des noeuds d'utilités qu'elle permet d'atteindre. L'évaluation $V(s)$ d'une stratégie s selon le critère d'Hurwicz peut s'écrire comme suit :

$$V(s) = \alpha \min_{P \in \mathcal{P}_\mathcal{T}} \sum_{N \in \mathcal{U}} P(X_{\pi(N)} = x(N)) \times u(N) + (1-\alpha) \max_{P \in \mathcal{P}_\mathcal{T}} \sum_{N \in \mathcal{U}} P(X_{\pi(N)} = x(N)) \times u(N)$$

où $\alpha \in [0,1]$ et $P(X_{\pi(N)} = x(N))$ représente (conformément à nos notations) le produit des probabilités assignées aux branches le long du chemin de la racine à N. Face à la nature combinatoire du problème, nous allons formuler un programme linéaire pour évaluer la valeur de s selon Γ-maximin (le terme de coefficient α), et un autre programme linéaire pour l'évaluer selon Γ-maximax (le terme de coefficient $1-\alpha$). Il suffira ensuite de combiner les valeurs obtenues selon le paramètre α pour obtenir l'évaluation selon le critère d'Hurwicz. La principale difficulté consiste alors à caractériser l'ensemble des distributions $P \in \mathcal{P}_\mathcal{T}$. En effet, il est important de remarquer que le fait que les probabilités somment à 1 en chaque nœud chance n'est pas suffisant pour garantir la cohérence globale de la distribution de probabilité sur $\mathcal{T}$. Pour illustration, considérons par exemple la distribution de probabilité partielle suivante sur l'arbre de la figure 1 : $P(X_2 = sec|X_1 = aucune) = 0.55$, $P(X_2 = sec|X_1 = faible) = 0.33$, $P(X_2 = sec|X_1 = forte) = 0.12$, $P(X_1 = aucune) = 0.20$, $P(X_1 = faible) = 0.35$, $P(X_1 = forte) = 0.45$, $P(X_2 = sec) = 0.22$. Cette distribution partielle peut être complétée de telle sorte que les probabilités somment à 1 en chaque nœud chance, mais est globalement incohérente puisque le théorème des probabilités totales n'est pas vérifié : $P(X_2 = sec|X_1 = aucune)P(X_1 = aucune) + P(X_2 = sec|X_1 = faible)P(X_1 = faible) + P(X_2 = sec|X_1 = forte)P(X_1 = forte) = 0.2795 \neq 0.22 = P(X_2 = sec)$. Afin de caractériser l'ensemble $\mathcal{P}_\mathcal{T}$ des probabilités "réalisables", nous allons donc définir un jeu de contraintes linéaires (que nous appellerons *contraintes de cohérence* par la suite) portant sur l'ensemble des variables $P(X = x)$ pour tout x. En effet, cet ensemble de variables suffit à caractériser une distribution de probabilité. Afin de garantir que les axiomes de Kolmogorov sont bien vérifiés, il est nécessaire de satisfaire la contrainte suivante :

$$\sum_{x \in \mathcal{D}(X)} P(X = x) = 1 \tag{1}$$

Par ailleurs, afin de lier les probabilités atomiques $P(X = x)$ aux probabilités $P(X_{\pi(N)} = x(N))$ il est nécessaire d'introduire les contraintes linéaires suivantes pour tout $N \in \mathcal{N}_C \cup \mathcal{N}_U$:

$$P(X_{\pi(N)} = x(N)) = \sum_{y \in \mathcal{D}(X_{\pi^c(N)})} P(X = (x(N), y)) \tag{2}$$

où $(x(N), y)$ désigne l'affectation du vecteur aléatoire X tel que $X_{\pi(N)} = x(N)$ et $X_{\pi^c(N)} = y$. Enfin, pour garantir que les intervalles de probabilités imprécises indiqués dans l'arbre sont bien respectés, il est nécessaire d'introduire les contraintes suivantes pour tout noeud de hasard N en utilisant les règles de Bayes :

$$P(X_{\pi(N)\cup\{id(N)\}} = (x(N), y)) \leq P(X_{\pi(N)} = x(N))P^{+}(X_{id(N)} = y | X_{\pi(N)} = x(N)) \quad (3)$$

$$P(X_{\pi(N)\cup\{id(N)\}} = (x(N), y)) \geq P(X_{\pi(N)} = x(N))P^{-}(X_{id(N)} = y | X_{\pi(N)} = x(N)) \quad (4)$$

Notons que la probabilité $P(X_{\pi(N)\cup\{id(N)\}} = (x(N), y))$ n'est autre que la variable $P(X_{\pi(N')} = (x(N), y))$ dès lors que N est le nœud de hasard précédant immédiatement N' et que l'affectation $X_{id(N)} = y$ figure sur le chemin menant à N'.

Exemple 2 *Considérons l'arbre de décision $\mathcal{T}$ de la figure 1. Nous allons tout d'abord définir l'ensemble des contraintes de cohérence permettant de caractériser $\mathcal{P}_{\mathcal{T}}$. L'ensemble des variables définissant les probabilités atomiques sont : $\{p_{as}, p_{ah}, p_{ai}, p_{fs}, p_{fh}, p_{fi}, p_{Fs}, p_{Fh}, p_{Fi}\}$, où p_{as} correspond à la probabilité $P(X_1 = \underline{a}ucune, X_2 = \underline{s}ec\}$ et ainsi de suite (il est à noter que le symbole f est utilisé pour faible, et F pour forte). La contrainte 1 est alors la suivante :*

$$p_{as} + p_{ah} + p_{ai} + p_{fs} + p_{fh} + p_{fi} + p_{Fs} + p_{Fh} + p_{Fi} = 1$$

Ensuite, en chaque noeud de hasard, l'ensemble des contraintes est défini comme suit :

Au noeud de hasard X_2^1 :
- $p_i = p_{ai} + p_{fi} + p_{Fi}$ *(contrainte 2)*
- $p_h = p_{ah} + p_{fh} + p_{Fh}$ *(contrainte 2)*
- $p_s = p_{as} + p_{fs} + p_{Fs}$ *(contrainte 2)*
- $0.307 \leq p_i \leq 0.456$ *(contraintes 3 et 4)*
- $0.309 \leq p_h \leq 0.386$ *(contraintes 3 et 4)*
- $0.214 \leq p_s \leq 0.344$ *(contraintes 3 et 4)*

Au noeud de hasard X_1 :
- $p_a = p_{ai} + p_{ah} + p_{as}$ *(contrainte 2)*
- $p_f = p_{fi} + p_{fh} + p_{fs}$ *(contrainte 2)*
- $p_F = p_{Fi} + p_{Fh} + p_{Fs}$ *(contrainte 2)*
- $0.181 \leq p_a \leq 0.222$ *(contraintes 3 et 4)*
- $0.333 \leq p_f \leq 0.363$ *(contraintes 3 et 4)*
- $0.444 \leq p_F \leq 0.454$ *(contraintes 3 et 4)*

Au noeud de hasard X_2^2 :
- $0.125p_a \leq p_{ai} \leq 0.181p_a$ *(contraintes 3 et 4)*
- $0.222p_a \leq p_{ah} \leq 0.272p_a$ *(contraintes 3 et 4)*
- $0.500p_a \leq p_{as} \leq 0.666p_a$ *(contraintes 3 et 4)*

Au noeud de hasard X_2^3 :
- $0.250p_f \leq p_{fi} \leq 0.363p_f$ *(contraintes 3 et 4)*
- $0.363p_f \leq p_{fh} \leq 0.444p_f$ *(contraintes 3 et 4)*
- $0.222p_f \leq p_{fs} \leq 0.333p_f$ *(contraintes 3 et 4)*

Au noeud de hasard X_2^4 :
- $0.454p_F \leq p_{Fi} \leq 0.625p_F$ *(contraintes 3 et 4)*
- $0.333p_F \leq p_{Fh} \leq 0.363p_F$ *(contraintes 3 et 4)*
- $0.111p_F \leq p_{Fs} \leq 0.166p_F$ *(contraintes 3 et 4)*

Une fois définies les contraintes de cohérences, l'évaluation d'une stratégie selon le critère d'Hurwicz peut être réalisée en définissant les deux programmes linéaires donnant les valeurs selon Γ-maximin et Γ-maximax. Considérons par exemple la stratégie $s = \{D_1 = sonder, D_2^2 = \text{ne pas forer}, D_2^3 = forer, D_2^4 = forer\}$. Les programmes linéaires se formulent comme suit :

Γ-$maximin(s) = \min \quad -10p_a + 190p_{fi} + 40p_{fh} - 80p_{fs} + 190p_{Fi} + 40p_{Fh} - 80p_{Fs}$
s.c : contraintes de cohérence

Γ-$maximax(s) = \max \quad -10p_a + 190p_{fi} + 40p_{fh} - 80p_{fs} + 190p_{Fi} + 40p_{Fh} - 80p_{Fs}$
s.c : contraintes de cohérence

Enfin la valeur de la stratégie s selon le critère d'Hurwicz est alors :

$$V(s) = \alpha\Gamma\text{-}maximin(s) + (1-\alpha)\Gamma\text{-}maximax(s)$$

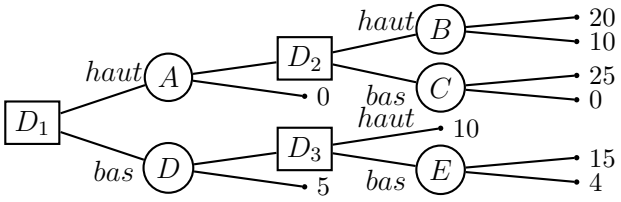


FIGURE 2 – Le principe d'optimalité n'est pas vérifié.

D_1	D_2	D_3	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
haut	haut	—	20	10	0
haut	bas	—	25	12.5	0
bas	—	haut	10	7.5	5
bas	—	bas	15	9.5	4

TABLE 1 – Les stratégies et leurs évaluations.

4 Recherche d'une stratégie optimale au sens d'Hurwicz

Lorsque l'on cherche à déterminer une stratégie optimale selon le critère d'Hurwicz dans un arbre de décision, il est important de remarquer que le principe d'optimalité n'est pas vérifié. Par exemple, considérons l'arbre de décision de la figure 2 où les variables aléatoires A, B, C, D, E sont mutuellement indépendantes, et supposons une complète ignorance sur les valeurs des probabilités (c'est-à-dire que tous les intervalles de probabilités sont $[0, 1]$). Posons $\alpha = 0.5$ et réalisons une induction arrière sur l'arbre avec $u(x) = x$. En D_2, le décideur préfère la décision *haut* à la décision *bas* (le critère d'Hurwicz vaut 15 pour $D_2 = haut$, contre 12.5 pour $D_2 = bas$). En D_3, il préfère également la décision *haut* à *bas* (une utilité certaine de 10, contre 9.5). Enfin, en D_1, le décideur a le choix entre une première loterie offrant un utilité minimum de 0 et une utilité maximum de 20 si il décide *haut*, et une seconde loterie offrant un minimum de 5 et un maximum de 10 si il décide *bas*. La meilleure décision selon le critère d'Hurwicz est *haut* (10 contre 7.5). La stratégie retournée par programmation dynamique est donc $\{D_1 = haut, D_2 = haut\}$ avec une valeur de 10. La table 1 indique les valeurs d'Hurwicz de toutes les stratégies pour différentes valeurs de α. Pour $\alpha = 0.5$, la stratégie $\{D_1 = haut, D_2 = bas\}$ est optimale avec une valeur de 12.5. Dans ce cas, on observe donc que la stratégie retournée par programmation dynamique est sous-optimale. Pour cette raison, un décideur utilisant le critère d'Hurwicz doit adopter un comportement de *choix résolu* [8], c'est-à-dire fixer initialement une stratégie puis ne jamais en dévier ensuite. Une manière de faire du choix résolu est de déterminer la stratégie optimale du point de vue de la racine et d'appliquer ensuite la stratégie sans jamais en dévier. Cependant, une critique faite à cette forme de choix de résolu est qu'il n'existe aucune assurance que le décideur ne va pas dévier de la stratégie. Ainsi la stratégie réellement appliquée par le décideur n'a plus aucune garantie de qualité. Face à ce constat, Jaffray [5] a proposé une approche différente de choix résolu. Dans son cadre, on considère chaque noeud de décision comme un *ego* indépendant des autres (différentes "incarnations" d'un même décideur dans plusieurs contextes décisionnels), chaque ego étant maître de la décision qu'il va prendre. Jaffray, dans un tel contexte, propose alors de construire une stratégie qui satisfasse tous les egos présents dans la stratégie. Ainsi, chaque ego, au moment de prendre sa décision, ne sera pas incité à dévier de la stratégie prédéfinie. L'idée est donc d'optimiser le critère de décision parmi l'ensemble des stratégies qui réalisent un compromis acceptable entre les egos. Dans ce travail nous nous intéressons à la détermination d'une telle stratégie.

Les stratégies représentant un compromis acceptable entre les egos sont celles dont toutes les sous-

Input : Arbre de décision $\mathcal{T}$, réel θ, entier k
Output : Une stratégie
pour chaque *noeud* N *de profondeur* p *à* 0 *dans* $\mathcal{T}$ **faire**
  si $N \in \mathcal{N}_D$ **alors**
    $\mathcal{S}_N \leftarrow \bigcup_{N' \in fils(N)} \left\{ \{(N, N')\} \cup s : s \in \mathcal{S}_{N'} \right\}$
  fin
  si $N \in \mathcal{N}_C$ **alors**
    $\mathcal{S}_N \leftarrow \prod_{N' \in fils(N)} \mathcal{S}_{N'}$
  fin
  pour chaque *stratégie* $s \in \mathcal{S}_N$ **faire**
    $V_s \leftarrow$ **evaluer**(s)
  fin
  $V_{max} \leftarrow \max_{s \in \mathcal{S}_N} \{V_s\}$
  pour chaque *stratégie* $s \in \mathcal{S}_N$ **faire**
    si *(*s *n'est pas* E-admissible*) OU (*$V_s < V_{max} - \theta$*)* **alors**
      $\mathcal{S}_N \leftarrow \mathcal{S}_N \setminus \{s\}$
    fin
  fin
  tant que $|\mathcal{S}_N| > k$ **faire**
    $\mathcal{S}_N \leftarrow \mathcal{S}_N \setminus \{\arg\min_{s \in \mathcal{S}_N} \{V_s\}\}$
  fin
fin
Retourner $\{\arg\max_{s \in \mathcal{S}_{racine}} \{V_s\}\}$

Algorithme 1 : Algorithme pour faire du choix résolu au sens de Jaffray

stratégies sont à la fois E-admissible et proches de l'optimum au sens d'Hurwicz. Nous présentons maintenant une approche procédurale visant précisément à déterminer des stratégies vérifiant ces propriétés. Cette approche est une spécification au cas du critère d'Hurwiczde l'algorithme proposé par Jaffray [5]. Le principe de l'algorithme est de remonter par programmation dynamique, en chaque noeud de décision N, un ensemble de sous-stratégies qui est le résultat d'un compromis entre l'ego associé à N et les egos présents dans le sous-arbre induit par N. Pour ce faire, chaque ego reçoit un ensemble de sous-stratégies estimées acceptables par l'ensemble des egos appartenant à son futur, et met à jour ces sous-stratégies en y préfixant les décisions dont il dispose. Une première condition nécessaire à l'adhésion d'un ego à une stratégie de compromis est l'E-admissibilité de la sous-stratégie induite. En effet, il est difficilement concevable qu'un ego puisse appliquer une stratégie qui n'est jamais optimale quelle que soit la vraie distribution de probabilité. C'est pourquoi on rejette toutes les sous-stratégies qui ne sont pas E-admissibles. Chaque sous-stratégie de l'ensemble est ensuite évaluée selon le critère d'Hurwicz. Soient V_{max} la meilleure évaluation de l'ensemble et θ la tolérance de perte d'utilité pour chaque ego (θ représente le degré de compromis acceptable par les egos afin de garantir qu'ils ne dévieront pas de la stratégie prédéfinie). Toutes les sous-stratégies dont l'évaluation est strictement inférieure à $V_{max} - \theta$ sont rejetées. Enfin, parmi les sous-stratégies E-admissibles dont la valeur est supérieure ou égale à $V_{max} - \theta$, on sélectionne les k meilleures afin de se préserver d'une explosion combinatoire de la taille des ensembles de sous-stratégies remontées en chaque noeud. Une fois l'induction arrière terminée, on sélectionne la stratégie de meilleure évaluation dans l'ensemble remonté à la racine de l'arbre et on la retourne. Notons que, dans certains cas, cet ensemble peut être vide et l'algorithme ne retourne alors aucune solution. Dans ces cas, il est nécessaire de réviser les valeurs des paramètres θ et/ou k si les ressources physiques de calcul le permettent. Ainsi, la stratégie retournée par la procédure comporte uniquement des sous-stratégies E-admissibles et proches de

l'optimum au sens du critère d'Hurwicz. Elle est bien sûr elle-même globalement E-admissible. La procédure est formalisée dans l'algorithme 1, où p désigne la profondeur de l'arbre, $\mathcal{S}_N$ (resp. $fils(N)$) l'ensemble des sous-stratégies remontées en un nœud N (resp. l'ensemble des fils du nœud N), et **evaluer**(s) retourne la valeur de s selon le critère d'Hurwicz en résolvant les programmes linéaires définis comme indiqués en section 3.

L'algorithme 1 parcourt chaque noeud N en partant des feuilles de l'arbre jusqu'à la racine. En chacun de ces noeuds, un ensemble $\mathcal{S}_N$ de stratégies est construit à partir des ensemble $\mathcal{S}_{N'}$ de ses fils. Ensuite, chaque stratégie est évaluée (avec la fonction **evaluer**()). Enfin, on ne conserve dans l'ensemble $\mathcal{S}_N$ que les k meilleures stratégies qui sont E-admissible, et dont l'évaluation est supérieure à $V_{max} - \theta$. Enfin, une fois l'ensemble de stratégies construit en la racine, on retourne celle d'évaluation maximale. Pour conclure cette partie algorithmique, il nous reste à expliquer comment déterminer si une stratégie est E-admissible ou pas.

Rappelons qu'une stratégie s est dite *E-admissible* s'il existe au moins une distribution de probabilité $P \in \mathcal{P}_\mathcal{T}$ pour laquelle la stratégie s est maximale au sens de l'espérance d'utilité. Nous montrons maintenant comment réduire le test d'E-admissibilité d'une stratégie à un problème de satisfaction de contraintes linéaires. Plus précisément, la stratégie s sera E-admissible si et seulement si le polyèdre défini par l'ensemble des contraintes est non-vide. L'ensemble des contraintes comporte bien évidemment les contraintes de cohérence (sur les probabilités) définies en section 3. Un autre jeu de contraintes caractérise l'ensemble des stratégies réalisables. Pour ce faire, on introduit une variable réelle y_N pour chaque nœud de décision N (cette variable est destinée à représenter l'espérance d'utilité maximale en N) et on définit les contraintes suivantes :

$$y_N = P\Big(X_{\pi(N)} = x(N)\Big) \times u(N) \qquad \forall N \in \mathcal{N}_U \tag{5}$$

$$y_N = \sum_{N' \in fils(N)} y_{N'} \qquad \forall N \in \mathcal{N}_C \tag{6}$$

Enfin, un dernier jeu de contraintes impose que la stratégie s soit maximale au sens de l'espérance d'utilité. Pour chaque arc (N, N') issus d'un nœud de décision N :

– si (N, N') appartient à la stratégie s, on ajoute la contrainte

$$y_N \leq y_{N'} \tag{7}$$

– si (N, N') n'appartient pas à la stratégie s, on ajoute la contrainte

$$y_N \geq y_{N'} \tag{8}$$

Ce dernier jeu de contraintes impose que y_N vaut au plus l'espérance d'utilité de la sous-stratégie induite par s dans le sous-arbre de racine N, et au moins l'espérance d'utilité maximum dans le sous-arbre de racine N. On en conclut que si une telle espérance existe en chaque noeud de décision, alors il existe une distribution de probabilité $P \in \mathcal{P}_\mathcal{T}$ (caractérisée par les variables $P(X = x)$) telle que l'espérance d'utilité de s soit de valeur maximale. Pour illustration, nous indiquons ci-dessous les contraintes obtenues pour une stratégie particulière dans l'exemple du forage pétrolier.

Exemple 3 *Considérons l'arbre de décision hasard de la figure 1 et la stratégie* $s = \{D_1 = sonder, D_2^2 = ne\ pas\ forer, D_2^3 = forer, D_2^4 = forer\}$. *Les contraintes de cohérence pour cet arbre sont données dans l'exemple 2. Par souci de brièveté, nous n'indiquons pas ici les contraintes et les variables associées aux noeuds d'utilité. Les contraintes en chacun des autres noeuds sont les suivantes :*

- *Au noeud* X_2^1 *:* $y_{X_2^1} = 200p_i + 50p_h - 70p_s$
- *Au noeud* D_2^1 *:* $y_{D_2^1} \geq 0$ *et* $y_{D_2^1} \geq y_{X_2^1}$
- *Au noeud* X_2^2 *:* $y_{X_2^2} = 200p_{ai} + 50p_{ah} - 70p_{as}$
- *Au noeud* D_2^2 *:* $y_{D_2^2} \leq -10p_a$ *et* $y_{D_2^2} \geq y_{X_2^2}$
- *Au noeud* X_2^3 *:* $y_{X_2^3} = 200p_{fi} + 50p_{fh} - 70p_{fs}$
- *Au noeud* D_2^3 *:* $y_{D_2^3} \geq -10p_f$ *et* $y_{D_2^3} \leq y_{X_2^3}$
- *Au noeud* X_2^4 *:* $y_{X_2^4} = 200p_{Fi} + 50p_{Fh} - 70p_{Fs}$
- *Au noeud* D_2^4 *:* $y_{D_2^4} \geq -10p_F$ *et* $y_{D_2^4} \leq y_{X_2^4}$
- *Au noeud* X_1 *:* $y_{X_1} = y_{D_2^2} + y_{D_2^3} + y_{D_2^4}$
- *Au noeud* D_1 *:* $y_{D_1} \geq y_{D_2^1}$ *et* $y_{D_1} \leq y_{X_1}$

5 Expérimentations numériques

Afin d'illustrer les performances de l'algorithme proposé ici, nous avons mené des expérimentations numériques sur des arbres décision hasard générés aléatoirement. Les arbres générés ont pour racine un noeud de décision et alternent noeud de décision et noeud de hasard sur toutes les branches pour s'achever par un noeud de hasard puis un noeud d'utilité. Notons également que les arbres générés sont des arbres binaires complets, c'est-à-dire que chaque noeud a exactement 2 fils (et 0 si le noeud est de profondeur max). Les utilités sont des réels tirés aléatoirement entre 0 et 500. Ensuite, à chaque noeud de hasard est associée une variable aléatoire. En ce qui concerne la génération des probabilités, nous générons d'abord l'ensemble des probabilité atomiques exactes de manière aléatoire afin de s'assurer qu'au moins une distribution de probabilité existe. A partir de cette distribution, en suivant les règles de Bayes, nous calculons les probabilités précises conditionnelles en chaque noeud de hasard. Enfin nous générons aléatoirement un intervalle autour de cette probabilité précise, qui constitue notre probabilité imprécise. Les expérimentations numériques ont été lancées sur un Pentium IV avec un processeur à $2,13GHz$. L'algorithme principal a été implémenté en C++ et les programmes linéaires ont été résolus à l'aide de CPLEX 11.

		$p=4$	$p=6$	$p=8$	$p=10$	$p=12$	$p=14$
$k=2$	$\theta=10$	<0.01	0.04	0.39	6.22	111.73	2518.95
$k=2$	$\theta=50$	<0.01	0.04	0.38	6.24	112.78	2487.56
$k=2$	$\theta=100$	<0.01	0.04	0.39	6.18	112.34	2660.13
$k=2$	$\theta=300$	<0.01	0.04	0.37	6.23	110.15	2532.40
$k=5$	$\theta=10$	<0.01	0.04	0.38	6.34	114.10	2711.24
$k=5$	$\theta=50$	<0.01	0.04	0.40	6.42	115.21	2689.83
$k=5$	$\theta=100$	<0.01	0.04	0.38	6.39	115.75	2802.88
$k=5$	$\theta=300$	<0.01	0.04	0.38	6.41	117.60	2765.12
$k=10$	$\theta=10$	<0.01	0.04	0.39	6.78	121.35	3398.91
$k=10$	$\theta=50$	<0.01	0.04	0.37	6.82	126.03	2991.33
$k=10$	$\theta=100$	<0.01	0.04	0.38	6.91	124.53	3132.48
$k=10$	$\theta=300$	<0.01	0.04	0.38	6.87	125.84	3255.04

TABLE 2 – Moyenne du temps d'exécution en seconde en fonction de la profondeur p de l'arbre et des paramètres k et θ.

Pour la première série d'expérimentations numériques, nous avons traité des arbres décision hasard où chaque niveau de noeud de hasard est associé à une même variable aléatoire. Remarquons que dans ce cadre, le nombre de probabilités atomiques ($\in O(2^v)$ si v est le nombre de variables aléatoires) est alors linéaire en la taille de l'instance puisque v est alors une valeur logarithmique du nombre n de noeuds dans l'arbre. Ceci implique une compléxité polynomiale de l'algorithme en fonction de k et n. Pour cette série d'expérimentations nous avons fait varier k, θ et p (la profondeur de l'arbre). Nous nous sommes intéressés aux temps d'exécution de l'algorithme en fonction de ces paramètres. Pour chaque jeu de paramètres où $p \leq 12$, 100 arbres ont été générés aléatoirement et nous avons indiqué

	$p=5$	$p=7$	$p=9$	$p=11$	$p=13$
$v=2$	<0.01	–	–	–	–
$v=3$	<0.01	0.04	–	–	–
$v=4$	<0.01	0.04	0.40	–	–
$v=5$	<0.01	0.05	0.66	6.42	–
$v=6$	<0.01	0.05	0.85	7.65	115.21
$v=7$	<0.01	0.08	1.18	10.87	135.28
$v=8$	<0.01	0.13	1.41	13.71	214.21
$v=9$	0.02	0.17	1.86	21.96	287.68
$v=10$	0.03	0.25	3.44	35.61	485.16
$v=11$	–	0.56	6.25	62.95	698.81
$v=12$	–	0.94	11.40	94.35	X
$v=13$	–	1.92	22.10	165.61	X
$v=14$	–	4.03	42.03	342.91	X
$v=15$	–	9.58	80.01	680.95	X
$v=16$	–	20.98	162.46	X	X
$v=17$	–	49.11	347.77	X	X
$v=18$	–	98.06	786.34	X	X

TABLE 3 – Moyenne du temps d'exécution en seconde en fonction de la profondeur p de l'arbre et du nombre de variable aléatoire v.

sur la table 2 la moyenne du temps d'exécution en secondes. Pour $p = 14$, seulement 5 instances ont été générées aléatoirement pour les différents paramètres. Les temps d'exécution dépendent évidement de k et θ puisqu'ils conditionnent le nombre de stratégies retenues en chaque noeud. Avec ce type d'instances, nous sommes capables de résoudre rapidement (en moins d'une heure) des instances jusqu'à la profondeur 14 (ce qui correspond à 65535 noeuds dans l'arbre).

Dans la seconde série d'expérimentations numériques, nous nous sommes intéressés aux arbres décision hasard où le nombre de variables aléatoires distinctes dans l'arbre de décision hasard est indépendant de la taille de l'instance. Pour cette série d'expérimentations nous avons fait varier p et v. Nous nous sommes intéressés à l'influence du nombre de variables sur le temps d'exécution. Nous avons donc fixé le paramètre k à 5 et le paramètre θ à 50. Pour chaque jeu de paramètres, nous avons généré 100 instances aléatoirement en attribuant de manière aléatoire les variables aléatoires aux noeuds de hasard. Les temps indiqués sur la table 3 représentent la moyenne du temps d'exécution en secondes pour chaque jeu de paramètres (le symbole "$-$" indique que l'instance est impossible et le symbole "X" correspond aux jeux de paramètres impliquant un temps de résolution supérieur à 1000 secondes). Nous constatons qu'à partir d'un certain nombre de variables aléatoires, l'ajout d'une variable aléatoire supplémentaire à l'instance double le temps d'exécution moyen de l'algorithme.

6 Conclusion

Dans cet article, nous avons présenté une procédure de choix résolu au sens de Jaffray pour calculer une stratégie dont toutes les sous-stratégies sont à la fois E-admissibles et proches de l'optimum au sens du critère d'Hurwicz. La principale difficulté vient de la nature combinatoire de l'ensemble des distributions de probabilité réalisables sur l'arbre de décision. Face à cette difficulté, nous avons

formulé des programmes linéaires qui permettent de tenir compte explicitement des contraintes sur les probabilités, que ce soit pour évaluer une stratégie au sens du critère d'Hurwicz ou pour tester l'E-admissibilité d'une stratégie. Les résultats sont assez encourageants, en effet, il est possible de traiter des instances d'arbres complets (arbres dont la combinatoire est la plus élevée) de plus de 65000 noeuds. Pour des travaux futurs, il serait intéressant de viser à garantir une propriété plus forte que l'E-admissibilité des sous-stratégies de la stratégie retournée. En effet, une sous-stratégie peut être E-admissible mais néanmoins dominée (au sens de la dominance définie en section 2.1). Par exemple, considérons un ensemble $\{f, g\}$ de loteries sur les événements E_1 et E_2. Supposons que $P_0(E_1) \in [0, 0.1]$ et $P_0(E_2) \in [0.9, 1]$. La loterie f retourne 100 si E_1 ou E_2 se produit. La loterie g quant à elle retourne 0 si E_1 se produit et 100 si E_2 se produit. Clairement, la loterie f domine g, bien que g soit E-admissible (pour $P(E_2) = 1$). C'est pourquoi nous pensons qu'il serait intéressant de proposer une méthode efficace pour tester qu'une stratégie est non-dominée, et de l'intégrer dans une procédure de choix résolu comme celle présentée ici (en remplacement du test d'E-admissibilité).

A la mémoire de Jean-Yves Jaffray, collègue et ami.

Références

[1] F. G. Cozman et C. P. de Campos D. Kikuti. Partially ordered preferences in decision trees : computing strategies with imprecision in probabilities. In *IJCAI Workshop on Advances in Preference Handling*, 2005.

[2] J.-Y. Jaffray et M. Jeleva. Information processing under imprecise risk with the Hurwicz criterion. In *5th International Symposium on Imprecise Probability : Theories and Applications*, pages 233–242, 2007.

[3] G. Jeantet et O. Spanjaard. Optimizing the Hurwicz criterion in decision trees with imprecise probabilities. In *International Conference on Algorithmic Decision Theory*, pages 340–352. Springer-Verlag, 2009.

[4] P. Hammond. Consequentialist foundations for expected utility. *Theory and Decision*, 25 :25–78, 1988.

[5] J.-Y. Jaffray. Rational decision making with imprecise probabilities. In *1st International Symposium on Imprecise Probability : Theories and Applications*, pages 183–188, 1999.

[6] A. Kasperski. *Discrete Optimization with Interval Data : Minmax Regret and Fuzzy Approach.* Studies in Fuzziness and Soft Computing. Springer, 2008.

[7] I. Levi. On indeterminate probabilities. *Journal of Philosophy*, 71(13) :391–418, 1974.

[8] E.F. McClennen. *Rationality and Dynamic choice : Foundational Explorations.* Cambridge University Press, 1990.

[9] M.L. Puterman. *Markov Decision Processes - Discrete Stochastic Dynamic Programming.* Wiley & Sons, 1994.

[10] H. Raiffa. *Decision Analysis : Introductory Lectures on Choices under Uncertainty.* Addison-Wesley, 1968.

[11] T. Seidenfeld. A contrast between two decision rules for use with (convex) sets of probabilities : G-maximin versus e-admissibility. *Synthèse*, 140 :69–88, 2004.

[12] R. Shachter. Evaluating influence diagrams. *Operations Research*, pages 34 :871–882, 1986.

[13] J. von Neuman et O. Morgenstern. *Theory of games and economic behaviour.* Princeton University Press, 1947.

[14] P. Walley. *Statistical reasoning with imprecise probabilities*, volume 91 of *Monographs on statistics and applied probability*. Chapman and Hall, 1991.

[15] K. Weichselberger. The theory of interval-probability as a unifying concept for uncertainty. In *1st International Symposium on Imprecise Probability : Theories and Applications*, pages 387–396, 1999.

Routage de guides d'onde dans un satellite de télécommunications

F. Bessaïh[1,3], B.Cabon[1], D. Feillet[2], Ph.Michelon[3]

[1] EADS ASTRIUM, 31 rue des Cosmonautes, Z.I. des Palays, 31402 Toulouse Cedex 4.
{fawzi.bessaih,bertrand.cabon}@astrium.eads.net
[2] École Nationale Supérieure des Mines de Saint-Étienne, CMP Georges Charpak, F-13541 Gardanne, France. feillet@emse.fr
[3] Université d'Avignon et des Pays de Vaucluse, Laboratoire Informatique d'Avignon (EA 931), F-84911 Avignon, France. {fawzi.bessaih,philippe.michelon}@univ-avignon.fr

Résumé : *Pour répondre à une demande sans cesse croissante en capacité et débit pour la télévision HD, la sécurité civile, les télécommunications haut débit etc..., les opérateurs satellitaires doivent accroître de façon importante la capacité de leurs nouveaux satellites. L'accroissement de cette capacité pose de nouvelles difficultés liées à la minimisation des pertes radio fréquentielles. Ces pertes sont liées à la dissipation au niveau des composants, pertes fixes et à la longueur des guides d'onde qui acheminent le signal, pertes proportionnelles à la longueur. Minimiser ces pertes revient à minimiser la longueur des guides d'onde et donc à effectuer le routage. Dans cet article, nous introduisons le Problème de Routage de Guides d'Onde (Wave Guide Routing Problem, WGRP). Après avoir formellement défini le WGRP, nous proposons une méthode de résolution basée sur la génération de colonnes, permettant de calculer la relaxation linéaire d'un modèle de couverture et donc d'obtenir une borne inférieure, et deux heuristiques fournissant une borne supérieure, l'une basée sur la génération de colonnes, l'autre sur un algorithme glouton associé à un mécanisme de réparation. Les deux heuristiques sont équivalentes en terme de coût et permettent en moyenne de borner l'optimum à 5,78%. En conclusion, de nouvelles perspectives sont envisagées pour réduire l'écart avec l'optimum.*

Mots-Clés : *validation de design, optimisation de l'aménagement d'équipements, génération de colonnes, plus court chemin, multi-routage, multiflot*

1 Introduction

La plupart des satellites de télécommunications ont une architecture assez similaire. Leur objectif principal est de recevoir des signaux en entrée, de les amplifier et de les réémettre vers la terre via l'antenne d'émission. Dans un premier temps, les signaux reçus sont dé-multiplexés et filtrés. Ils sont ensuite acheminés, via des câbles coax vers les tubes amplificateurs « TWTA », où ils subissent une forte amplification. Ils sont ensuite acheminés via des guides d'onde (waveguide en anglais) vers les canaux de sortie, où ils subissent un multiplexage pour être enfin renvoyer vers la terre via une antenne d'émission.

Pendant la phase de conception d'un satellite de télécommunication, le dimensionnement de ce dernier doit être effectué au plus juste, afin de limiter les coûts de production et en respectant le

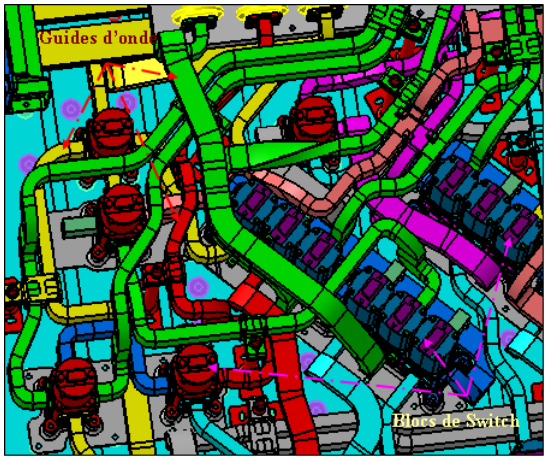


Figure 1 – Exemple de routage

cahier des charges (puissance d'amplification, robustesse aux pannes etc...). Ce dimensionnement est étroitement lié à la capacité d'amplification du satellite qui doit être garantie malgré les pertes radio fréquentielles subies par les signaux, durant la phase d'amplification : pertes fixes et durant leur acheminement via les guides d'onde : pertes proportionnelles à la longueur. Une limitation et une évaluation des pertes radio fréquentielles est alors nécessaire car une mauvaise évaluation entraînera l'augmentation de la puissance des « TWTA »(ou l'augmentation de leur nombre) et par la même occasion celle de la dissipation et de la consommation électrique, ce qui aura pour effet un redimensionnement du satellite et une augmentation du coût de production.

Ces pertes peuvent être limitées en minimisant la longueur des guides d'onde reliant les composants, mais aussi en optimisant l'aménagement de ces mêmes composants afin de réduire la distance qui les sépares. Or, pour un rendement optimal, l'ensemble des composants du satellite doivent être maintenus dans une gamme de température pour un fonctionnement optimum. C'est une problématique majeure au niveau de l'accommodation du satellite, notamment pour les éléments fortement dissipatifs tels que les tubes « TWTA »qui consomment à eux seuls 90% à 95% de la puissance du satellite dont 60% est dissipée sous forme de chaleur. Des caloducs, éléments conducteurs de chaleur sont positionnés en surface et à l'intérieur des murs du satellite, afin de créer trois zones de température, 2 zones froides et une zone chaude. Les composants y sont alors positionnés suivant leurs gammes de température. De ce fait, la minimisation des pertes radio fréquentielles est alors exclusivement liée à la minimisation de la longueurs des guides d'onde.

A l'heure actuelle, les concepteurs quantifient ces pertes en mesurant manuellement, à l'aide d'une règle la longueur des guides d'onde sur un diagramme logique. En parallèle, des équipes de CAO effectuent, sur une durée de 3 à 6 mois une modélisation en trois dimensions de l'aménagement des composants et des guides d'onde. Un exemple de routage en trois dimensions est illustré par la figure 1, où un certain nombre de guides d'onde connectant des blocs de switch y sont représentés. L'enchevêtrement de guides d'onde illustre la difficulté du routage. Une fois la modélisation effectuée, après 3 à 6 mois d'attente les concepteurs peuvent être amenés à reconsidérer le dimensionnement dans le cas d'une sous estimation des pertes radio fréquentielles. Avec l'accroissement de la complexité des satellites de télécommunication, il devient primordial de disposer rapidement d'estimations fines afin de faciliter et d'accélérer les échanges entre les différentes équipes de conception.

Cette problématique correspond à la recherche d'un ensemble de chemins disjoints de longueur totale minimale dans un graphe non orienté à capacité d'arc unitaire, entre des ensembles de sources et de destinations données. Il s'agit d'un cas particulier de problème de multiflot [2] entier avec demandes (dans notre cas, les demandes sont unitaires) à coût minimal. Un certain nombre d'applications du

problème de multiflot à coût minimal peuvent être trouvées dans [10, 11, 14]. Des problèmes similaires ont même été traités sous forme de problème de coloration de graphe (de chemin) par certaines équipes, [12, 1].

Dans cet article, nous introduirons le Problème de Routage de Guides d'Onde dans la section 2, par la présentation de notre approche pour modéliser l'espace de routage, suivie d'une description de la modélisation choisie et une présentation de la méthode permettant d'obtenir une borne inférieure, basée sur la génération de colonnes. Par la suite, nous décrirons dans la section 3 les deux heuristiques utilisées pour résoudre ce problème. Dans la section 4 nous présenterons enfin les expérimentations effectuées et les résultats obtenus et, avant de conclure les perspectives envisagées dans la section 5.

2 Le Problème de Routage de Guides d'Onde

Dans cette section, nous introduisons le Problème de Routage de Guides d'Onde (Wave Guide Routing Problem, WGRP). Après avoir formellement défini le WGRP, nous proposons une modélisation basée sur un modèle de couverture, où les variables de décision représentent la sélection de chemins pour les guides d'onde. Nous proposons alors une méthode de résolution basée sur la génération de colonnes qui permet de calculer la relaxation linéaire du modèle de couverture et donc obtenir une borne inférieure.

2.1 Définition du WGRP

La première étape dans la modélisation du problème abordé consiste à proposer une représentation de l'espace. Nous proposons de discrétiser l'espace de routage en une grille en 2D sur plusieurs niveaux, le passage d'un niveau à l'autre ne pouvant se faire qu'entre cases de mêmes coordonnées. Ce choix (plutôt qu'une modélisation en 3D par exemple) s'explique par les méthodes de conception actuelles des satellites, qui se font niveau par niveau. Pour chaque niveau, nous proposons un maillage en grille rectangulaire, choisi du fait de sa simplicité de mise en œuvre. Une case de la grille peut être

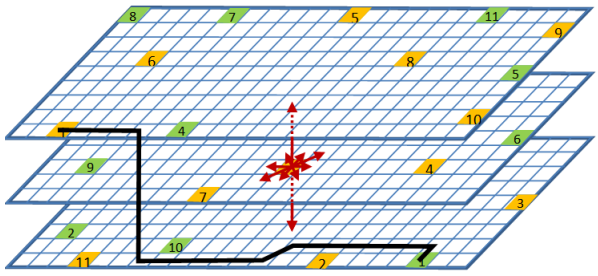


Figure 2 – Discrétisation en grille de la zone de routage

un espace libre, un obstacle (élément du satellite positionné à cet endroit), un point de départ ou un point d'arrivée de guide d'onde. Chaque case a au plus 10 cases adjacentes (8 cases adjacentes sur le plan – incluant les diagonales – et 2 sur les niveaux inférieurs et supérieurs respectivement). Avec cette modélisation, la manière de positionner un guide d'onde dans l'espace sera alors représentée par un chemin reliant une case d'origine à une case de destination par une suite de cases adjacentes (voir figure 2).

Soit $G = (V, A)$ le graphe non orienté construit de la manière suivante. L'ensemble des sommets V est obtenu en introduisant un sommet par case de la grille à l'exception des cases occupées par un

obstacle. A est l'ensemble des arcs, où l'arc $(v_i, v_j) \in A$ si les sommets $v_i \in V$ et $v_j \in V$ représentent des cases adjacentes. Soit W l'ensemble des guides d'onde à router. Un guide d'onde $w \in W$ est défini par un sommet source $s_w \in V$ et un sommet destination $t_w \in V$. $V_s = \{s_w, w \in W\} \subset V$ est l'ensemble des sommets origines, $V_t = \{t_w, w \in W\} \subset V$ l'ensemble des sommets destinations. Le positionnement d'un guide d'onde $w \in W$ sera alors représenté par un chemin dans le graphe G reliant son sommet d'origine s_w à son sommet destination t_w. Enfin, à chaque arc $(v_i, v_j) \in A$ est associé un coût positif c_{ij} indiquant sa longueur. Ici, ces longueurs ont la particularité de ne prendre que trois valeurs, correspondant respectivement à des déplacements non-diagonaux dans le plan, diagonaux dans le plan ou entre niveaux. La matrice des longueurs est supposée satisfaire l'inégalité triangulaire, ce qui revient à s'assurer que le coût d'un déplacement diagonal ne dépasse pas le coût de deux déplacements non-diagonaux.

En dehors des limites de cette modélisation liée à la discrétisation de l'espace, une difficulté provient de l'impossibilité physique pour deux guides d'onde de se chevaucher. Interdire le passage de deux guides d'onde par un même sommet du graphe n'est pas suffisant pour gérer cette contrainte du fait de la possibilité que deux chemins se croisent en diagonal sur la grille (voir figure 3). Afin de prendre en compte cette contrainte, nous introduisons l'ensemble D des points $d \in D$ de croisements de la grille. Notons alors $\delta_{ij}^d = 1$ si l'arc $(v_i, v_j) \in A$ représente deux cases adjacentes de la grille situées en diagonale et traversant le point de croisement $d \in D$, $\delta_{ij}^d = 0$ sinon.

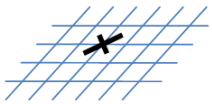


Figure 3 – Croisement non autorisé de guides d'onde

Le WGRP consiste à déterminer pour chaque guide d'onde $w \in W$ un chemin dans le graphe G reliant s_w à t_w, tel que la somme des longueurs des chemins soit minimale, les chemins soient disjoints deux à deux, et aucun point de croisement $d \in D$ ne soit emprunté par plus d'un chemin.

2.2 Modèle de couverture

Pour modéliser sous forme de programme mathématique le WGRP et proposer un schéma de calcul de borne inférieure, nous proposons le modèle de couverture suivant.

Soit $\Omega = \{r_1, r_2, ..., r_{|\Omega|}\}$ l'ensemble des chemins r_k (aussi appelés routes) réalisables pour les guides d'onde $w \in W$. $\Omega^w \subset \Omega$ est le sous-ensemble de Ω correspondant aux chemins réalisables de w, c'est-à-dire reliant s_w à t_w. La longueur c_k de la route $r_k \in W$ est égale à la somme des longueurs des arcs empruntés ; $a_{ik} = 1$ si la route r_k utilise le sommet $v_i \in V$, 0 sinon ; $b_{ij}^k = 1$ si la route r_k utilise l'arc $(v_i, v_j) \in A$, 0 sinon ; $\delta_{dk} = 1$ si la route $r_k \in \Omega$ utilise le point de croisement $d \in D$, 0 sinon.

Le modèle de couverture est alors

$$\text{minimiser} \sum_{r_k \in \Omega} c_k \theta_k + M \sum_{w \in W} z_w \tag{1}$$

sujet à

$$\sum_{r_k \in \Omega} a_{ik} \theta_k \quad \leq \quad 1 \quad (v_i \in V), \tag{2}$$

$$\sum_{r_k \in \Omega} \delta_{dk}\theta_k \leq 1 \quad (d \in D), \tag{3}$$

$$\sum_{r_k \in \Omega^w} \theta_k + z_w \geq 1 \quad (w \in W), \tag{4}$$

$$\theta_k \in \{0,1\} \quad (r_k \in \Omega), \tag{5}$$

$$z_w \in \{0,1\} \quad (w \in W). \tag{6}$$

où $\theta_k = 1$ si la route r_k est sélectionnée dans la solution, 0 sinon, $z_w = 1$ si aucune route n'a été attribuée au guide d'onde $w \in W$, 0 sinon, et M est un grand nombre assurant que n'importe quelle solution réalisable pour le WGRP sera meilleure qu'une solution ne routant pas un ou plusieurs guides d'onde. L'introduction de la variable auxiliaire z_w et le poids M se justifie par la nécessité d'identifier les guides d'onde à l'origine de la non réalisabilité d'une instance de routage.

Les contraintes (2) et (3) impliquent respectivement qu'une case ou qu'un point de croisement peut être occupé par au plus un guide d'onde. La contrainte (4) assure pour tout guide d'onde $w \in W$ que, soit le guide d'onde est bien routé, soit la variable z_w vaut 1.

2.3 Schéma de génération de colonnes

L'inconvénient du modèle (1)-(6) réside dans le nombre très important de variables. La génération de colonnes [8, 7] permet cependant de calculer sa relaxation linéaire en considérant de manière implicite ces variables. Dans la suite de l'article, nous appellerons problème maitre MP (pour Master Problem) la relaxation linéaire de (1)-(6) et problème maitre restreint MP(Ω_1) la restriction du problème maitre au sous ensemble $\Omega_1 \subset \Omega$. La génération de colonnes est une méthode itérative résolvant MP(Ω_1) puis enrichissant Ω_1 si besoin à l'aide d'un sous-problème. L'objectif du sous-problème est de détecter s'il existe dans $\Omega \setminus \Omega_1$ une ou des variables de coût réduit négatif, c'est-à-dire susceptibles d'améliorer la solution courante. En ce cas, une ou plusieurs de ces variables sont insérées dans Ω_1 et le processus de résolution continue. Dans le cas contraire, l'optimum de MP(Ω_1) est garanti optimal pour MP. Le déroulement de la génération de colonne est décrit par la figure 4.

Générer un ensemble initial de colonnes Ω_1
Faire
    Résoudre MP(Ω_1)
    $\Gamma \leftarrow$ *colonne*(*s*) retournée(s) par le sous problème
    $\Omega_1 \leftarrow \Omega_1 \cup \Gamma$
Tant que $\Gamma \neq \emptyset$

Figure 4 – Algorithme de génération de colonnes

Dans le cas du modèle (1)-(6), la condition indiquant si le coût réduit de la route $r_k \in \Omega \setminus \Omega_1$ est négatif est :

$$c_k + \sum_{v_i \in V} a_{ik}\lambda_i + \sum_{d \in D} \delta_{dk}\mu_d - \sigma_w < 0, \tag{7}$$

où λ_i, μ_d et σ_w sont les variables duales positives respectivement associées aux contraintes (2), (3) et (4), et $w \in W$ est le guide d'onde associé à r_k ($r_k \in \Omega^w$).

De manière équivalente, et pour mettre en évidence la nature du sous-problème, la condition (7) peut également s'écrire :

$$\sum_{(v_i,v_j) \in A} b_{ij}^k (c_{ij} + \lambda_i) + \sum_{(v_i,v_j) \in A} b_{ij}^k \sum_{d \in D} \delta_{ij}^d \mu_d < \sigma_w - \lambda_{t_w} \tag{8}$$

Le sous-problème revient ainsi à déterminer, pour chaque guide d'onde $w \in W$ s'il existe un chemin reliant s_w à t_w de coût strictement inférieur à $\sigma_w - \lambda_{t_w}$, où le coût de l'arc (v_i, v_j) est fixé à $\tilde{c}_{ij} = c_{ij} + \lambda_i + \sum_{d \in D} \delta_{ij}^d \mu_d$. En notant que les coûts définis sur les arcs sont positifs, il s'agit donc simplement, pour chaque guide d'onde, d'un problème de plus court chemin classique.

2.4 Résolution du sous-problème

Pour résoudre le sous-problème, nous proposons d'utiliser l'algorithme A* [9, 3]. Afin de générer dans la mesure du possible plusieurs chemins par itération, nous proposons une phase de post-traitement présentée dans la sous-section 2.4.2.

2.4.1 A*

L'algorithme A* peut être vu comme une extension de l'algorithme de Dijkstra [6] pour laquelle, au lieu de traiter les sommets dans l'ordre de leur distance par rapport au point de départ, le critère de sélection du prochain sommet prend en compte une évaluation (optimiste) de la distance jusqu'au point de destination.

Plus précisément, lorsque l'algorithme est lancé pour un guide d'onde $w \in W$ donné, le critère de sélection combine deux coûts : le coût $g(i)$ correspondant au coût du plus court chemin permettant d'atteindre le sommet i et le coût $h_w(i)$ évaluant la distance qui reste à parcourir pour atteindre $t_w \in V$. Notons $f(i) := g(i) + h_w(i)$ la fonction d'évaluation du sommet v_i. L'optimalité de l'algorithme A^* [5] est garantie sous condition de monotonie de h_w, à savoir $h_w(i) \leq \tilde{c}_{ij} + h_w(j)$ pour tout arc (v_i, v_j).

Afin de proposer une fonction h_w donnant une évaluation aussi précise que possible de la distance restant à parcourir, nous définissons $h_w(i)$ comme étant égal au coût du plus court chemin reliant v_i à la destination t_w avec les distances initiales (coûts c_{ij}). Du fait que $c_{ij} \leq \tilde{c}_{ij}$ et du fait que la matrice des coûts c respecte l'inégalité triangulaire, la condition de monotonie est vérifiée. Pour tout guide d'onde $w \in W$, la valeur de la fonction h_w est calculée en pré-traitement. Pour cela, l'algorithme de Dijkstra est utilisé en prenant comme source le sommet t_w.

2.4.2 Post-traitement

Le post-traitement proposé exploite le fait qu'à la fin de l'algorithme A*, il est possible d'avoir $f(i) = g(i) + h_w(i) < \sigma_w - \lambda_{t_w}$ pour un certain nombre de sommets $v_i \in V$. Les valeurs $g(i)$ et $h_w(i)$ sont de plus respectivement associés aux chemins reliant s_w à v_i et v_i à t_w, dont la concaténation pourrait ainsi permettre d'obtenir un chemin reliant s_w à t_w de coût réduit négatif. Le principe du post-traitement est de parcourir l'ensemble de ces chemins et de calculer leur coût vis-à-vis de la fonction de coût $\tilde{c}$. Précisons que seuls les chemins reliant les sommets v_i à t_w ont en fait besoin d'être parcourus, puisque leurs coûts $h(i)$ ont été calculés avec la matrice de coût initiale. Lorsqu'un chemin de coût réduit négatif est trouvé, il est inséré dans Ω_1.

Afin de limiter les temps de calcul et de générer, dans la mesure du possible des chemins assez différents, un filtrage est effectué. Premièrement, les sommets adjacents au plus court chemin fourni par l'algorithme A* sont exclus. Deuxièmement, les sommets v_i tels que le plus court chemin permettant d'atteindre v_i se dirige dans une direction opposée à celle du vecteur reliant s_w à t_w sont aussi supprimés.

3 Résolution heuristiques du WGRP

Dans cette section, nous présentons deux approches heuristiques pour la résolution du WGRP toutes deux basées sur des ajustements de chemins valides. L'une met en œuvre un algorithme de type glouton associé à un mécanisme de réparation local, l'autre quant à elle est basée sur la génération de colonnes.

3.1 Heuristique gloutonne avec réparation

Un certain nombre de définitions doivent être données avant d'exposer plus en détail le fonctionnement de l'heuristique. Deux chemins qui se chevauchent sont dit en situation de conflit. La zone de chevauchement est appelée zone de conflit. Un chemin fixe est considéré comme un obstacle par les guides d'onde non routés. Le routage d'un guide d'onde est dit indépendant si il est effectué sans prendre en compte les guides d'onde précédemment routés, la longueur obtenue est une borne inférieure et est appelée distance A^*.

Notre heuristique fonctionne selon le mécanisme « glouton », la solution est construite étape par étape, une étape correspondant au routage d'un seul guide d'onde. À l'étape « i », l'heuristique dans un premier temps route le guide $w \in W$ en autorisant les chevauchements avec les guides d'onde précédemment routés. Si aucune zone de conflit n'est détectée ou que le routage est impossible on passe à l'étape « i+1 ». Dans le cas contraire, la ou les zones de conflit sont successivement réparées. En cas de succès on passe à l'étape « i+1 », sinon un second routage est effectué mais cette fois-ci l'entrée en conflit n'est plus autorisée, les guides d'onde précédemment routés sont fixés. En cas d'échec du second routage, on passe à l'itération « i+1 ». Le fonctionnement de l'approche est décrit par l'algorithme 1.

L'heuristique peut rencontrer une situation de blocage caractérisée par l'impossibilité de router un ou plusieurs guides d'onde (échec du routage initial ou de la réparation). Cette situation ne peut pour l'instant pas être contournée, mais néanmoins elle peut être limitée en effectuant le routage des guides d'onde dans l'ordre croissant des distances A^* estimées (routage indépendant). C'est-à-dire que les guides d'onde les plus courts sont routés d'abord. Nous allons maintenant décrire les différents mécanismes mis en oeuvre par l'heuristique.

3.1.1 La réparation

Notre heuristique repose sur deux phases, l'entrée en conflit durant le routage initial d'un guide d'onde et la réparation de ou des zones de conflit potentielles. Tout d'abord, nous allons aborder la phase de réparation. Prenons l'exemple de deux guides d'onde, le guide n°1 précédemment routé et le guide n°2 de l'itération courante, qui se trouve être en conflit avec le n°1 comme illustré par la figure 5. Soit $S1$, $G1$, $S2$, $G2$ les points de départ et d'arrivée du guide d'onde n°1 et du n°2 respectivement. La zone de conflit entre les deux chemins est représentée par l'ensemble des cases marquées de la lettre C. Afin de réparer cette situation de conflit, des points de départ et d'arrivée secondaires, $s1$, $g1$, $s2$, $g2$ sont positionnés aux deux extrémités de cette zone, comme illustré par la figure 5. La réparation consiste alors à determiner les plus courts chemins reliant les points de départ et d'arrivée secondaires, tout en interdisant l'entrée en conflit. Le résultat est illustré par la figure 6.

Afin de garantir le succès de la réparation, deux règles doivent être impérativement respectées. La première est liée à la taille de la zone de conflit. Cette taille doit être limitée au maximum. C'est à dire qu'un chemin associé à un guide d'onde ne rentrera en conflit avec un autre chemin qu'en

Algorithme 1 Glouton + réparation

Trier W, dans le sens croissant des distances A^*.
pour tout $w \in W$ **faire**
  $r_w = A^*(w, 1)$ { Retourne un chemin r_w si il existe, null sinon. L'entrée en conflit est autorisée (1).}
  si $r_w \neq null$ **alors**
    si r_w est en conflit avec un ou plusieurs guides d'onde **alors**
      $r'_w = reparer(r_w)$ {Retourne un chemin r'_w en cas de succès de la réparation, null sinon.}
      si $r'_w \neq null$ **alors**
        r'_w est associé à w.
      sinon
        $r_w = A^*(w, 0)$ { Retourne un chemin r_w si il existe, null sinon. L'entrée en conflit n'est plus autorisée (0).}
        r_w est associé à w.
      finsi
    sinon
      r_w est associé à w.
    finsi
  finsi
fin pour

dernier recours. La figure 7 illustre le cas où cette taille n'est pas limitée. Une zone de conflit trop grande empêche toute tentative de réparation car les chemins reliant les points secondaires $s1$ à $g1$ et $s2$ à $g2$ se bloqueront mutuellement. La limitation de cette zone de conflit est effectuée grâce à l'ajout d'un coût supplémentaire sur les cases occupées par un guide d'onde précédemment routé, le n°1 dans notre exemple. Ce coût d'occupation s'ajoute au coût $g(i)$ de l'algorithme A^*.

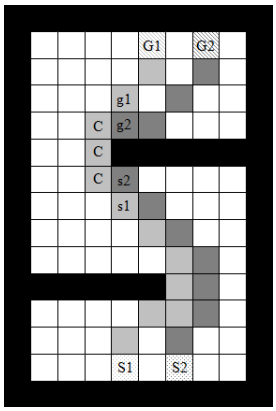


Figure 5 – Deux guides d'onde en conflit

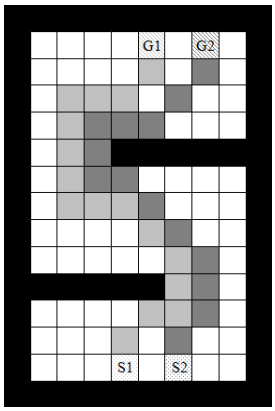


Figure 6 – Résultat de la réparation

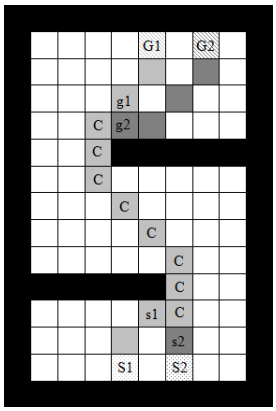


Figure 7 – Limitation de la taille de la zone de conflit

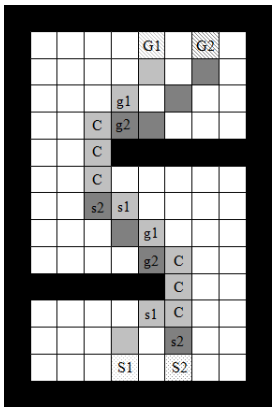


Figure 8 – Interdiction du croisement

La seconde règle concerne le croisement entre les chemins associés aux guides d'onde. Sur notre exemple, lors du routage du guide d'onde n°2, il est impératif d'interdire le croisement avec le n°1. La figure 8 illustre le cas où cette règle n'est pas respectée. On y observe deux zones de conflit et un croisement entre les deux chemins, ce qui empêche toutes tentatives de réparation. L'interdiction de croisement se fait très simplement grâce aux produits vectoriels entre $\overrightarrow{s1C_1} \cdot \overrightarrow{s2C_1}$ et $\overrightarrow{C_2g1} \cdot \overrightarrow{C_2g2}$, voir figure 9. La case $g2$ est la case de sortie de conflit avec le chemin associé au guide d'onde n°1. Le test de croisement est effectué à chaque sortie d'une situation de conflit.

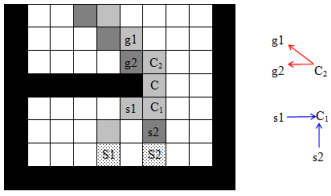


Figure 9 – Produits Vectoriels

Malheureusement, l'interdiction de croisement rompt l'optimalité de l'algorithme A*. Prenons l'exemple de la figure (10-gauche) qui illustre deux guides d'onde, le guide d'onde n°1 précédemment routé et le n°2 qui est entrain de l'être. L'algorithme A^* effectue une premiere extension du chemin associé au guide d'onde n°2, représentée par les cases en damier. Durant cette extension, il entre en conflit avec le guide d'onde n°1 par le coté droit. Une situation de conflit matérialisée par les cases marquées d'une lettre C de couleur blanche. L'algorithme A^* étend une deuxième extension par le coté gauche, représentée par les cases en brique, cas illustré par la figure (10-centre). À son tour, la seconde extension entre en conflit avec le guide d'onde n°1 au niveau de la case marquée d'une lettre C noir, une case déjà occupée par la premiere extension mais qui va être affectée à la seconde extension à la faveur d'un coût plus faible. Cette situation aurait été impossible si l'on avait uniquement considéré les coûts de déplacement, or le coût d'occupation servant à limiter la zone de conflit est pris en compte, ce qui a pour effet de favoriser l'extension du coté gauche. Une fois cette domination effective, l'extension du coté droit est perdue et il ne reste alors plus que celle du coté gauche qui ne peut croiser le chemin associé au guide d'onde n°1 afin d'atteindre sa destination. Le routage devient alors impossible, comme le montre la figure (10-droite) où toutes les cases ont été explorées en vain.

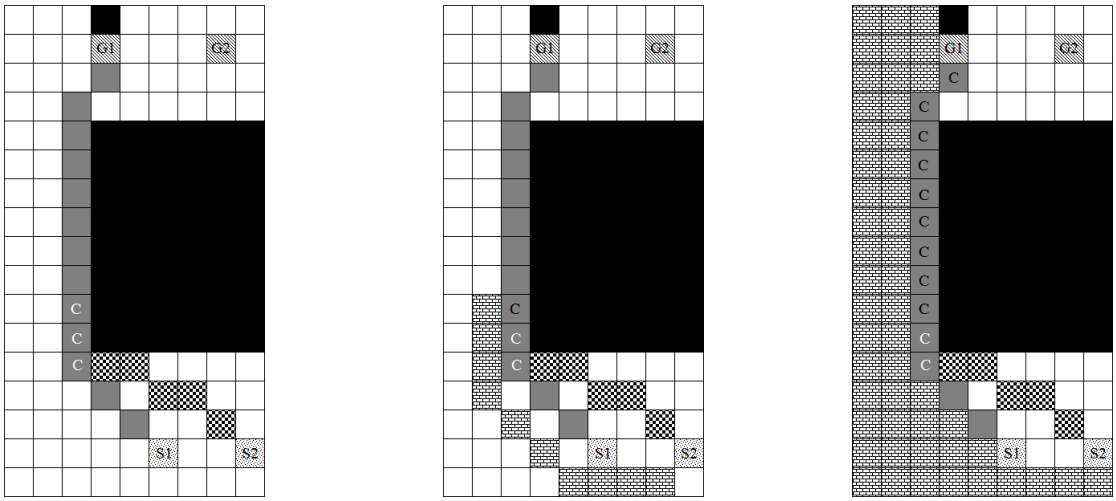


Figure 10 – Multi-labels

La solution adoptée pour rétablir l'optimalité de l'algorithme A* est de créer un label gauche et un label droit en plus du label utilisé jusqu'à présent, le label neutre. Lors de l'expansion, si une situation de conflit est détectée, le label correspondant au coté d'entrée en conflit, gauche ou droit est étendu. Quant au label neutre, ce dernier est utilisé si aucun conflit n'est détecté. Les règles de dominance adoptées sont les suivantes : un label neutre ne peut dominer qu'un label neutre. Les labels droit et gauche ne peuvent dominer qu'un label du même type ou un label neutre. Lors de l'expansion, tous les labels d'une case doivent être étendus. L'algorithme de recherche de plus court

chemin A* devient alors multi-label.

Dans l'optique d'améliorer l'efficacité de l'algorithme A^*, l'extension d'un label n'est pas effectuée pour toutes les directions réalisables. En effet, une analyse simple montre qu'un déplacement dans le plan provoquant un changement d'angle supérieur à l'angle droit ramène sur une case adjacente à la case visitée précédemment, ce qui à un coût nécessairement supérieur à un déplacement direct. Ainsi, par exemple, le chemin reliant successivement les cases de coordonnées $(0, 0, 0)$, $(1, 0, 0)$ et $(0, 1, 0)$, formant ainsi un angle de 135°, sera dominé par le chemin reliant directement $(0, 0, 0)$ à $(0, 1, 0)$.

La solution proposée jusqu'à present permet de réparer une situation de conflit entre deux guides d'onde. Pour un nombre plus important de guides d'onde, une nouvelle notion doit être introduite. Prenant l'exemple de trois guides d'onde, avec les n°1 et n°2 précédemment routés (figures 5-6) et le guide d'onde n°3 qui est en conflit avec le n°2, comme illustré par la figure 11. Actuellement, seul le routage des guides d'onde n°2 et n°3 est remis en cause, or celui du guide n°1 doit l'être aussi afin de libérer la place nécessaire au passage des trois guides d'onde. Entre alors en jeu la notion de boite englobante qui permet de détecter les guides d'onde proches de la zone de conflit afin de les remettre en question, le guide d'onde n°1 dans notre exemple.

La boite englobante fonctionne de la façon suivante : un rectangle est placé autour de la zone de conflit afin de l'inclure entièrement, ensuite il est agrandie selon un paramètre, le « radius ». La figure 12 illustre cette opération, avec un radius égale à une case. Une fois positionnée, la boite englobante est agrandie et les cases qui y sont incluses sont examinées dans l'optique de détecter l'ordre d'apparition des guides d'onde. Ce dernier détermine l'ordre de routage des portions de guides d'onde à réparer. Afin d'effectuer le routage, des points de départ et d'arrivée secondaires sont positionnés à la frontière de la boite englobante. Le résultat de la réparation est donné par la figure 13.

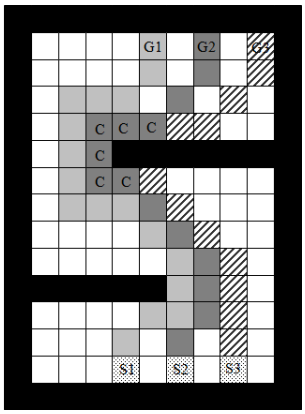


Figure 11 – Trois guides d'onde

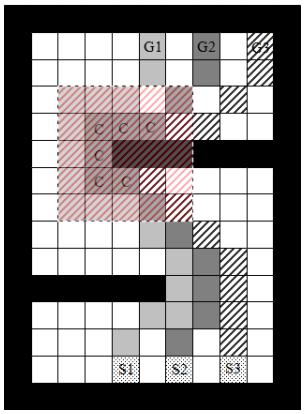


Figure 12 – Boite englobante

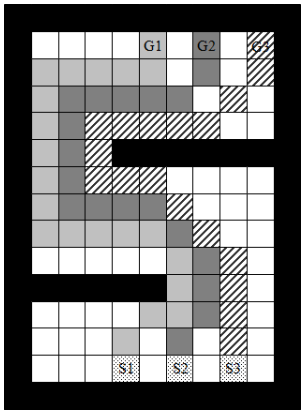


Figure 13 – Trois guides d'onde routés

3.2 Heuristique basée sur la génération de colonnes

Cette heuristique tente de construire de manière assez simpliste une solution entière à partir de la borne inférieure présentée en section 2 et obtenue par génération de colonnes. Si la borne inférieure correspond à une solution réalisable (c'est-à-dire que les variables ont des valeurs égale à 1), l'heuristique retourne cette solution, qui s'avère d'ailleurs être optimale. Dans le cas contraire, en plus des variables fixées par la génération de colonnes, une variable est sélectionnée puis fixée à 1. L'algorithme de génération de colonnes est alors relancé, en considérant les cases occupées par le ou les guides d'onde fixés comme des obstacles et en ne générant plus de colonnes pour ce ou ces guides

d'onde. Les deux phases de génération de colonnes et de fixation d'une ou des variables sont répétées jusqu'à obtention d'une solution entière ou détection d'une infaisabilité. La sélection de la variable à fixer à chaque itération se fait en deux temps. Tout d'abord, un guide d'onde est sélectionné selon deux critères évalués de manière hiérarchique :

1. Plus courte distance estimée séparant la source et la destination du guide d'onde. Permet de favoriser les guides d'onde les plus courts.
2. Le moins de variables fractionnaires, afin de favoriser les guides d'onde qui ont rencontrés le moins de conflit.

Une fois le guide d'onde sélectionné, la variable choisie est celle dont la valeur est maximale. En cas d'égalité, la variable de coût minimal est privilégiée. Pour rappel, une variable est associée à un chemin $r_k \in \Omega_w$, $w \in W$ étant le guide d'onde sélectionné.

Algorithme 2 Heuristique Génération de Colonnes

tantque La solution est fractionnaire **faire**
  GenerationDeColonnes(Ω_1) {Une solution est générée.}
  Fixer les variables de la solution dont la valeur est égale à 1.
  Sélectionner et fixer une variable. {Sélection selon les critères cités précédemment.}
fin tantque

4 Expérimentations

4.1 Description des instances de test

Les expérimentations sont effectuées sur des instances de test représentatives des différents cas de figure qui peuvent être rencontrés lors du routage des guides d'onde, que ce soit des instances crées de toutes pièces ou des instances provenant de véritables satellites : le croisement de deux guides d'onde sur le plan (figure 14) ; le goulot d'étranglement qui oblige les guides d'onde à passer dans un espace restreint (figure 15) ; le virage en S qui est une série de virages où un guide d'onde est successivement à l'intérieur et à l'extérieur de ces derniers (figure 16). Des instances supplémentaires composées des cas de figure précédemment cités sont aussi testées.

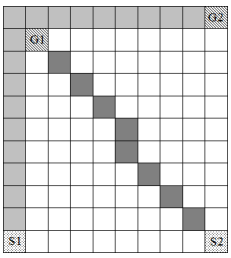


Figure 14 – Croisement

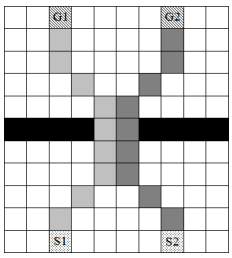


Figure 15 – Goulot d'étranglement

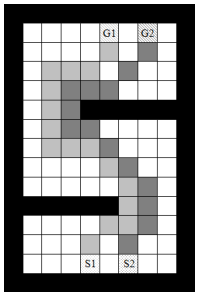


Figure 16 – Virage en S

Une instance est définie par :

(X, Y, Z) la taille de la grille, où X est le nombre de ligne, Y le nombre de colonne et le nombre de niveaux autorisés,

Nwg le nombre de guide d'onde à router,

Occ le taux d'occupation égale au ratio entre le nombre de cases occupées et le nombre total de cases.

Dans la suite de l'article, nous notons, **H** la méthode heuristique gloutonne, **HGC** l'heuristique basée sur la génération de colonnes, **GC**, **A*** et **Eucl** respectivement les bornes inférieures donnée par la génération de colonnes, l'A* où l'on effectue le routage de chaque guide d'onde sans considerer les routages précédents et la distance euclidienne où l'on calcule la distance euclidienne entre s_w et t_w pour chaque guide d'onde.

4.2 Résultats

Les tableaux 1 et 3 regroupent les résultats des expérimentations des deux heuristiques. À chaque instance correspond un coût de la solution par heuristique, ainsi qu'un écart entre les deux heuristiques exprimé en pourcentage. Une instance non résolut (solution incomplete) a un coût associé égale à « - ». Parmi les 63 instances seulement 12 sont dans ce cas, avec au plus deux guides d'onde non routés : deux pour les instance (30 30 6 1 9), (25 25 5 1 8) et (50 50 11 1 7), et un seul guide d'onde pour les 9 restantes. De plus, la meilleure valeur obtenue est notée en gras. Les différents cas de figure cités précédemment sont représentés dans le tableau 1, alors que le tableau 3 regroupe des instances réelles. Notez que toutes les instances non résolut appartiennent au tableau 1.

Sur les 51 instances résolues par les deux heuristiques, on constate un écart moyen de 0,01% et au plus 4,73 % en faveur de la génération de colonnes. Plus particulièrement, on constate un écart moyen très faible, de 0,27% et au plus 1,98% pour les instances réelles. Au regard de ces premiers résultats, on peut affirmer que les deux heuristiques sont très proches et cela est confirmé par la proportion de meilleurs valeurs obtenues. Les heuristiques gloutonne et génération de colonnes donnent la meilleure valeur respectivement pour 24 et 26 instances, sont à égalité pour 11 instances. Seulement deux instances n'ont pas été résolues par chaque heuristique. On peut néanmoins remarquer que l'heuristique génération de colonne est meilleure pour 13 instances parmi les 20 instances réelles alors que l'heuristique gloutonne l'est pour 6, avec une seule égalité. Il faut tout de même relativiser ce dernier résultat au vu des écarts constatés entre les deux heuristique. Par contre, l'heuristique génération de colonne est en grand difficulté pour traiter les instances de type croisement, une seule instance résolue sur 4 contre la totalité pour l'heuristique gloutonne.

La méthode de résolution basée sur la génération de colonnes, permet de calculer la relaxation linéaire du modèle de couverture et donc d'obtenir une borne inférieure de meilleur qualité que celles fournie par l'A* et la distance euclidienne, mais dont il reste à évaluer la qualité face à la borne supérieure fournie par les deux heuristiques. Les tableaux 2 et 4 regroupent les écarts, en pourcentage de la meilleur borne supérieure trouvée (heuristiques gloutonne et génération de colonnes) et les trois bornes inférieures. On constate que pour la majorité des instances à un seul niveau, la borne supérieure est égale à la borne inférieure de la génération de colonnes, 25 instances parmi 33. On atteint donc la valeur optimale. Par contre, l'écart est en moyenne de 15,81% pour les instances de croisement. De plus, dès que le nombre de niveaux augmente, l'écart devient plus important, avec au plus 25,05% pour une instance à deux niveaux. On remarque aussi que l'écart est en moyenne de 7,06% et au plus 12,27% pour les instances réelles du tableau 2. Enfin, sur la totalité des instances on obtient une moyenne de 5,78% avec au plus un écart de 25,05%.

(X,Y,Z)	Nwg	Occ	H	HGC	Écart
			valeur	valeur	(%)
Croisement					
(5,5,1)	2	0	**10,83**	**10,83**	0,01
(10,10,1)	2	0	**29,31**	-	-
(30,30,1)	2	0	**97,60**	-	-
(50,50,1)	2	0	**165,88**	-	-
Goulet d'étranglement					
(15,15,1)	4	4	**65,11**	**65,11**	0,00
(15,15,1)	4	9	**66,53**	**66,53**	0,00
(15,15,1)	5	22	**85,67**	**85,67**	0,00
(30,30,1)	5	8	169,41	**163,95**	3,22
(30,30,1)	6	9	-	-	-
(30,30,1)	7	5	-	**225,21**	-
(30,30,1)	7	7	218,58	**217,17**	0,65
(50,50,1)	11	3	596,78	**593,26**	0,59
(50,50,1)	11	7	**623,00**	-	-
(50,50,2)	16	16	**390,75**	408,75	-4,61
Virage en S					
(15,20,1)	2	6	53,77	**53,18**	1,09
(15,20,1)	3	6	**77,08**	**77,08**	0,0
(15,20,1)	4	6	99,40	**98,23**	1,18
(30,30,1)	2	3	**85,98**	**85,98**	0,0
(30,30,1)	3	3	**124,37**	-	-
(30,30,1)	4	3	156,65	**156,07**	0,37
(30,30,1)	5	3	-	**210,25**	-
(50,50,1)	2	3	**189,62**	193,62	-2,11
(50,50,1)	3	3	**267,78**	**279,78**	-4,48
(50,50,1)	4	3	**344,37**	-	-
(50,50,1)	5	3	-	-	-
Croisement / Goulet d'étranglement / Virage en S					
(10,10,1)	2	56	**27,90**	**27,90**	0,0
(10,10,1)	3	18	**27,31**	**27,31**	0,0
(10,10,1)	4	22	**36,97**	**36,97**	0,0
(10,10,2)	6	42	**46,83**	**46,83**	0,0
(25,25,1)	5	11	**141,81**	142,40	-0,42
(25,25,1)	6	14	-	**206,02**	-
(25,25,1)	5	8	-	**256,92**	-
(25,25,1)	6	17	**175,54**	177,88	-1,33
(30,30,2)	15	19	241,81	**240,98**	0,34
(30,30,2)	23	24	**494,94**	497,87	-0,59
(30,30,2)	28	22	**342,98**	343,81	-0,24
(50,50,2)	14	34	505,28	**502,84**	0,48
(50,50,2)	31	20	**677,43**	690,16	-1,88
(50,50,2)	38	24	1030,50	**981,77**	4,73
(200,120,1)	42	28	**1388,50**	1394,74	-0,45
(200,120,1)	42	30	**1422,07**	1426,56	-0,32
(100,100,1)	23	14	**836,74**	844,99	-0,99
(100,100,1)	27	18	**1028,85**	1032,13	-0,32

Table 1 – Résultats d'expérimentation

(Eucl,A*,GC) vs Best(H,HGC) : écart en %					
(X,Y,Z)	Nwg	Occ	Eucl	A*	GC
Croisement					
(5,5,1)	2	0	20,50	16,25	13,54
(10,10,1)	2	0	17,98	17,98	15,98
(30,30,1)	2	0	17,41	17,41	16,81
(50,50,1)	2	0	17,30	17,30	16,95
Goulet d'étranglement					
(15,15,1)	4	4	13,88	2,53	0,00
(15,15,1)	4	9	15,72	1,25	0,00
(15,15,1)	5	22	23,68	7,28	0,00
(30,30,1)	5	8	17,57	4,31	0,00
(30,30,1)	6	9	-	-	-
(30,30,1)	7	5	15,99	2,94	0,00
(30,30,1)	7	7	12,88	2,29	0,00
(50,50,1)	11	3	9,11	1,26	0,00
(50,50,1)	11	7	13,45	2,71	0,79
(50,50,2)	16	16	21,58	11,69	10,45
Virage en S					
(15,20,1)	2	6	25,60	3,11	0,00
(15,20,1)	3	6	26,05	6,58	0,00
(15,20,1)	4	6	22,63	3,38	0,00
(30,30,1)	2	3	26,99	2,61	0,00
(30,30,1)	3	3	28,04	5,03	0,00
(30,30,1)	4	3	23,54	5,16	0,00
(30,30,1)	5	3	27,79	8,93	0,00
(50,50,1)	2	3	49,38	5,17	2,07
(50,50,1)	3	3	45,84	6,07	0,00
(50,50,1)	4	3	43,66	8,92	0,00
(50,50,1)	5	3	-	-	-
Croisement / Goulet d'étranglement / Virage en S					
(10,10,1)	2	56	38,39	2,97	0,00
(10,10,1)	3	18	15,42	0,88	0,00
(10,10,1)	4	22	12,44	2,25	0,00
(10,10,2)	6	42	44,22	42,73	24,69
(25,25,1)	5	11	15,19	4,33	0,00
(25,25,1)	6	14	30,02	5,78	0,00
(25,25,1)	5	8	53,19	16,65	0,00
(25,25,1)	6	17	17,87	0,95	0,00
(30,30,2)	15	19	30,68	21,24	18,22
(30,30,2)	23	24	36,32	29,39	25,05
(30,30,2)	28	22	30,43	23,92	19,70
(50,50,2)	14	34	33,79	2,36	2,36
(50,50,2)	31	20	29,23	21,71	20,27
(50,50,2)	38	24	32,98	26,34	23,63
(200,120,1)	42	28	15,81	1,47	0,27
(200,120,1)	42	30	17,79	1,72	0,39
(100,100,1)	23	14	11,21	1,04	0,00
(100,100,1)	27	18	14,44	2,83	0,36

Table 2 – Écart entre les bornes inférieures et la borne supérieure

(X,Y,Z)	Nwg	Occ	H	HGC	Écart
			valeur	valeur	(%)
(215,140,3)	80	24	2151,44	**2125,10**	1,22
(215,140,3)	37	21	867,59	**865,83**	0,20
(215,140,3)	138	34	4169,94	**4118,97**	1,22
(215,140,3)	112	31	2510,94	**2490,41**	0,82
(215,140,3)	140	34	3704,94	**3667,47**	1,01
(215,140,3)	96	27	2677,40	**2662,34**	0,56
(250,230,2)	13	7	**1084,15**	**1084,15**	0,0
(250,230,2)	48	2	**3173,21**	3183,22	-0,32
(250,230,3)	91	23	6060,94	**5977,23**	1,38
(300,200,3)	37	18	1201,11	**1190,08**	0,92
(300,200,3)	74	22	**1903,91**	1911,87	-0,42
(300,200,3)	74	23	2032,78	**2031,41**	0,07
(300,200,3)	80	21	**2939,40**	2953,79	-0,49
(300,200,3)	96	25	**3814,88**	3845,19	-0,79
(300,200,3)	112	28	**3390,69**	3390,94	-0,01
(300,200,3)	123	27	**5329,88**	5435,38	-1,98
(300,200,3)	138	32	5754,57	**5744,47**	0,18
(300,200,3)	140	32	5038,29	**4993,47**	0,89
(300,200,3)	148	30	6477,95	**6469,64**	0,13
(300,200,3)	168	35	7452,66	**7386,49**	0,89

Table 3 – Résultats d'expérimentation : instances réelles

(Eucl,A*,GC) vs Best(H,HGC) : écart en %					
(X,Y,Z)	Nwg	Occ	Eucl	A*	GC
(215,140,3)	80	24	29,58	12,42	9,98
(215,140,3)	37	21	28,79	4,98	4,06
(215,140,3)	138	34	25,05	10,88	8,64
(215,140,3)	112	31	31,16	12,39	9,34
(215,140,3)	140	34	29,79	14,67	12,27
(215,140,3)	96	27	20,06	8,41	7,78
(250,230,2)	13	7	10,36	4,85	4,85
(250,230,2)	48	2	7,59	3,16	3,07
(250,230,3)	91	23	15,03	6,13	5,79
(300,200,3)	37	18	22,37	3,53	3,05
(300,200,3)	74	22	20,22	7,21	6,92
(300,200,3)	74	23	21,17	7,94	7,44
(300,200,3)	80	21	22,83	8,92	8,26
(300,200,3)	96	25	16,07	6,28	5,93
(300,200,3)	112	28	24,10	8,83	8,07
(300,200,3)	123	27	19,84	7,48	2,27
(300,200,3)	138	32	19,19	7,50	6,99
(300,200,3)	140	32	22,40	9,81	9,20
(300,200,3)	148	30	21,73	9,80	9,43
(300,200,3)	168	35	21,69	8,35	7,89

Table 4 – Écart entre les bornes inférieures et la borne supérieure : instances réelles

5 Conclusion

Dans cet article, nous avons proposé deux méthodes heuristiques fournissant une borne supérieure. Deux méthodes équivalentes en terme de valeur, avec un écart moyen de 0,01% mais avec un net avantage pour l'heuristique gloutonne en ce qui concerne le temps d'execution et la capacité à résoudre les instances de type croisement. Afin d'évaluer la borne supérieure, une méthode de résolution fournissant une borne inférieure de bonne qualité à aussi été proposée, ce qui nous permet en moyenne de borner l'optimum à 5,78%.

Au vue de la simplicité de l'heuristique basée sur la génération de colonnes, on peut raisonnablement penser que des améliorations nous permettraient de réduire l'écart avec l'optimum. En ce qui concerne l'heuristique gloutonne, une nouvelle fonction de réparation globale (et non locale) donnerait de meilleurs résultats.

De plus, un algorithme A* grand voisinage permettrait d'affiner les trajectoires et d'obtenir ainsi une évaluation des longueurs plus précise. Il sera en tous points identique à l'algorithme A*, excepté pour la phase d'expansion où le grand voisinage lui permettra d'atteindre des cases non adjacente à la case à l'origine de l'expansion. Un mécanisme inspiré par les travaux de Watanabe et Sugiyama [13], qui ont proposé une adaptation de l'algorithme de Lee, avec comme paramètre supplémentaire la taille de la vague d'expansion. A chaque pas, les cases présentes sur le front d'expansion s'étendent de D_x cases.

Plusieurs autres perspectives s'offrent à nous. Tout d'abord, améliorer les heuristiques et le modèle de couverture afin de prendre en compte des contraintes supplémentaires, tel que l'épaisseur et de pouvoir ainsi confronter nos résultats à de véritables mesures de longueur. Ensuite, améliorer la borne donnée par la relaxation linéaire du modèle (intégration d'inégalités valides...). Et enfin, mettre en

place une politique de branchement (Branch and Price [4]) pour la convergence de la génération de colonne vers une solution entière optimale.

Références

[1] A. Aggarwal, A. Bar-Noy, D. Coppersmith, R. Ramaswami, B. Schieber, and M. Sudan. Efficient routing and scheduling algorithms for optical networks. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 412–423, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows : Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[3] Amit. Amit's a* pages. http ://theory.stanford.edu/ amitp/GameProgramming/, 2003.

[4] C. Barnhart, C. Hane, and P. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Oper. Res.*, 48(2) :318–326, 2000.

[5] R. Dechter and J. Pearl. Generalized best-first search strategies and the optimality of A*. *J. ACM*, 32(3) :505–536, 1985.

[6] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[7] Soumis F. Decomposition and column generation. Technical report, Les Cahiers du GERAD, 1997.

[8] L. R. Ford, Jr. and D. R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Manage. Sci.*, 50 :97–101, 1958.

[9] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4 :100–107, 1968.

[10] A. Hertz M.-C. Costa and M. Mittaz. Bounds and heuristics for the shortest capacitated paths problem. *Journal of Heuristics*, 8(4) :449–465, juillet 2002.

[11] F.-R Monclar et M. Zrikem M.-C. Costa. Variable neighborhood search for the optimization of cable layout. *Journal of Intelligent Manufacturing*, 13 :21–27, 2002.

[12] Y. Rabani. Path coloring on the mesh. *FOCS '96 : Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, page 400, 1996.

[13] T. Watanabe and Y. Sugiyama. A new routing algorithm and its hardware implementation. In *Proc. 23rd ACM/IEEE conference on Design Automation*, pages 574–580, Piscataway, NJ, USA, 1986. IEEE Press.

[14] M. Zrikem. *Optimisation du routage de câbles dans les installations de production d'électricité.* PhD thesis, CEDRIC-CNAM, 2001.

Optimisation de l'inférence dans les Réseaux Bayésiens Dynamiques

Morgan Chopin, Pierre-Henri Wuillemin

LIP6 ; Université de Paris 6 ; 104, avenue du Président Kennedy, 75016 Paris, France
{morgan.chopin, pierre-henri.wuillemin}@decision.lip6.fr

Résumé : *Nous proposons une méthode d'inférence dans les réseaux bayésiens dynamiques (dBNs) basée sur la résolution d'un problème classique de recherche opérationnelle : la coupe minimale dans un graphe. Dans cette approche, nous introduisons la notion d'interface (ensemble de nœuds séparant le passé du futur) permettant de faire le lien avec la coupe d'un graphe. Nous montrons alors que la recherche d'une interface minimale se réduit à un problème de coupe minimale, et on en déduit l'algorithme* `Min-Interface` *de recherche d'une interface de taille minimale. Cette interface nous permet de déduire une méthode de calcul réduisant la complexité de l'inférence, dont les résultats obtenus sur divers dBNs montrent l'efficacité.*

Mots-Clés : *réseau bayésien ; réseau bayésien dynamique ; inférence ; arbre de jonction ; coupe minimale ; réduction polynomiale.*

1 Introduction

Suivre l'évolution dans le temps d'un système complexe est un problème largement étudié : localiser un robot dans son environnement[7], analyser les réactions physico-chimiques se produisant lors du processus de fabrication d'un fromage [1], faire de la prédiction et du diagnostique de pannes [14], ou encore estimer la progression d'une maladie au sein d'une population. En général, chacun de ces problèmes peut être modélisé à l'aide d'un processus stochastique et, en particulier, à l'aide d'un *Réseau Bayésien Dynamique* (dBN) [6] (section 3). Les dBNs sont une extension des Réseaux Bayésiens (BN) [12, 13] (section 2) dans le sens où ils permettent de modéliser des systèmes dynamiques, au même titre que les Modèles de Markov Cachés (HMM) ou les filtres de Kalman (KF) ; la particularité des dBNs étant de pouvoir prendre en considération les interactions locales au sein d'un état pour représenter de manière compacte le système.

Une requête dans un BN (ou un dBN) s'effectue grâce à une inférence : il s'agit de calculer la probabilité d'un état en connaissance de certains faits ou observations. L'existence de différents types de requêtes amène à différents types d'inférences (voir Fig.6). Nous nous intéressons ici au problème de l'inférence *hors-ligne* ; c'est-à-dire, déterminer la probabilité que le système se trouve dans un certain état au temps $t \in [0, T]$ connaissant l'ensemble des observations sur toute la durée T de la simulation

Parmi les algorithmes qui permettent de réaliser cette tâche [15], certains font appel à un algorithme classique d'inférence dans les BNs : l'algorithme de l'arbre de jonction (JT) [8] (section 2). Dans une première phase, la partie "dure" de l'algorithme revient à trouver un bon ordre d'élimination des variables. Cet ordre conditionne directement la complexité des calculs de la seconde phase (phase de propagation) où l'on réalise l'inférence proprement dite. Il faut noter qu'effectuer une inférence dans un réseau bayésien quelconque est un problème NP-difficile [4]. Dans le cadre des dBNs, la tâche est

d'autant plus difficile que le réseau, qui permet de suivre des variables dans le temps, peut avoir une taille arbitrairement grande. De plus, les algorithmes standards qui déterminent l'ordre d'élimination reposent sur une heuristique gloutone qui n'exploite pas la structure particulière des dBNs et peuvent facilement détériorer le temps de calcul de la phase de propagation.

Dans les dBNs, les méthodes d'inférence qui semblent donner les meilleurs résultats sont celles qui utilisent cette structure pour contraindre l'ordre d'élimination [9, 10, 5, 2]. Autrement dit, au lieu de considérer l'ensemble des ordres d'élimination possibles, on place des contraintes pour n'en considérer qu'une partie ; ce qui permet de limiter l'espace de recherche, sans pour autant limiter la qualité des solutions [5]. En particulier, pour un dBN donné, un ordre d'élimination contraint permet de borner supérieurement et inférieurement l'exposant dans la complexité exponentielle de l'algorithme JT, limitant ainsi l'explosion combinatoire. Tout le travail consiste alors à réduire ces bornes dont la taille dépend d'un sous-ensemble de nœuds particulier que l'on appelle *interface*.

Nous proposons ici un algorithme qui se situe dans la continuation de ces approches. Il repose sur une méthode permettant de minimiser la taille de l'interface , et d'en déduire un ordre d'élimination. Après avoir formalisé ce problème, nous montrons qu'il est équivalent au problème de la recherche de la coupe minimale dans un graphe. De là, on en déduit un algorithme polynomial de recherche d'une interface optimale. On utilise alors cette interface pour déterminer un ordre d'élimination efficace pour l'algorithme JT. Pour finir, nous montrons quelques résultats encourageant obtenus sur divers réseaux bayésiens dynamiques.

La section 2 présente les réseaux bayésiens et l'algorithme de l'arbre de jonction dont il est question dans l'article. La section 3 introduit les réseaux bayésiens dynamiques et montre comment utiliser l'algorithme JT sur ces derniers. La section 4 décrit l'algorithme d'inférence par coupe minimale. La section 5 présente les résultats obtenus, et nous concluerons dans la section 6.

2 Réseaux bayésiens et inférences

2.1 le modèle BN

Un réseau bayésien permet de modéliser les intéractions entre les différents états d'un système à l'aide de *variables aléatoires* et d'un *graphe orienté*, qui représente la manière dont s'influencent les variables [12, 13]. Une requête dans un réseau bayésien consiste alors à calculer les *probabilités marginales* de variables étant donné de l'information sur certaines autres. Plus formellement, un réseau bayésien (voir FIG.1) représente une loi jointe de n variables aléatoires par un *graphe orienté sans circuit* G (DAG) et une probabilité conditionnelle associée à chaque nœud [1]. Les arcs définissent des dépendances entre les variables du réseau. Le graphe encode ainsi un sous-ensemble de l'ensemble des indépendances probabiliste de la loi jointe de la manière suivante : chaque variable est conditionnellement indépendante de ses non-descendants sachant ses parents [2]. La *table de probabilité conditionnelle* (CPT) associée à un nœud X est $P(X|pa(X))$ où $pa(X)$ désigne les parents de X dans le graphe G. Le graphe G ainsi que les CPT associées à chaque nœud permettent de représenter la loi jointe complète en utilisant l'équation suivante :

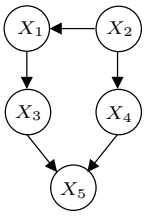
$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i))$$

Parmi les algorithmes qui permettent le calcul de requêtes dans un BN, on s'intéresse plus parti-

1. Dans l'article, nous noterons $X_1, X_2, ...$ des variables aléatoires discrètes, et $x_1, x_2, ...$ les instanciations de ces variables. Une observation (*evidence*) sur une variable X sera notée e_X. Par $X_{a:b}$, on désigne les variables $X_a, ..., X_b$. On identifiera les variables aléatoires et les nœuds. Un chemin passant successivement par les sommets $X_1, ..., X_n$, sera noté $< X_1, ..., X_n >$. Soit $\mathbf{X}$ un ensemble de variables, sa taille, notée $size(\mathbf{X})$, est le produit des cardinalités de chaque variable dans $\mathbf{X}$, sa dimension, notée $|\mathbf{X}|$, est le nombre de variables dans $\mathbf{X}$. Enfin, la réduction polynomiale d'un problème Q à un problème R se notera $Q \prec_P R$.

2. Propriété de Markov local

culièrement dans cet article à l'algorithme de l'arbre de jonction.



$P(X_1, X_2, X_3, X_4, X_5) = P(X_1|X_2)P(X_2)P(X_3|X_1)P(X_4|X_2)P(X_5|X_3, X_4)$.
En supposant que les variables soient toutes binaires, cette factorisation permet de réduire le nombre de valeurs de $2^5 = 32$ à $4 + 2 + 4 + 4 + 8 = 22$.

FIGURE 1 – Exemple de réseau bayésien et factorisation induite.

2.2 Algorithme de l'arbre de jonction (JT)

L'algorithme JT [8] consiste, d'abord, à transformer le réseau bayésien en une structure seconde, l'arbre de jonction qui est une arborescence dont les nœuds sont des *clusters* (sous-ensembles de variables du BN). Cette structure possède les propriétés nécessaires pour réaliser la phase de propagation [13], c'est à dire le calcul d'inférence à proprement parler (voir FIG.4). L'algorithme se décompose donc en :

• Une *phase de construction*, qui nécessite de passer par les trois sous-étapes suivantes : la moralisation du réseau bayésien, la triangulation du graphe moral et la construction de l'arbre de jonction. Cette phase est essentielle, car la qualité de l'arbre de jonction engendré influe directement sur la complexité des calculs lors de la phase de propagation.

• Une *phase de propagation* : c'est lors de cette étape que le calcul d'inférence s'effectue. Les *clusters* s'échangent des messages de manière à mettre à jour leurs potentiels (fonctions non-négatives des variables associées aux *clusters*) jusqu'à atteindre un état d'équilibre (voir FIG.2). Lorsque cet état est atteint, le potentiel de chaque *cluster* correspond alors à la probabilité jointe *a posteriori* des variables de ce *cluster*. Nous ne détaillerons pas plus ce processus car sa compréhension n'est pas nécessaire pour la suite de l'article. Par contre nous en donnerons la complexité, et détaillerons la phase de construction car celle-ci est critique pour la phase de propagation.

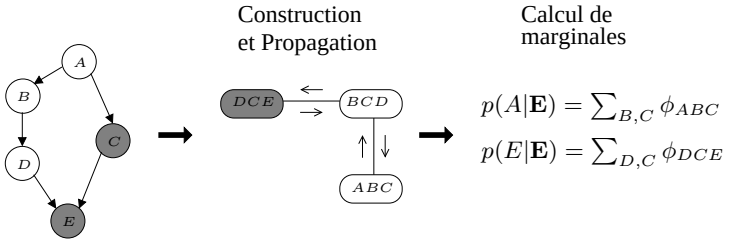


FIGURE 2 – Illustration de l'algorithme JT où les nœuds/clusters grisés comprennent une information, $\phi_{\mathcal{C}}$ désigne le potentiel du *cluster* $\mathcal{C}$, et $\mathbf{E} = \{e_C, e_E\}$ sont les observations.

2.2.1 Moralisation

La première étape de transformation du réseau bayésien est la moralisation du *DAG* noté G (voir FIG.3). Elle consiste à relier deux à deux, par une arête, les parents de chaque variable. Il suffit ensuite de remplacer tous les arcs restant par des arêtes. On obtient alors le graphe moral G^m qui est un graphe non-orienté.

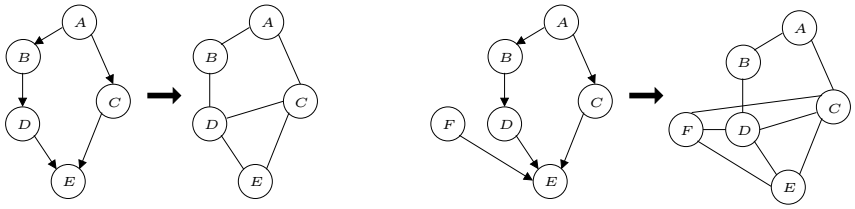


FIGURE 3 – Exemples de moralisation de réseaux bayésiens avec des nœuds d'au plus trois parents (à gauche), et d'au plus deux parents (à droite).

Cette opération de moralisation du graphe permet de révéler toutes les propriétés de dépendance impliquées par la factorisation de la distribution jointe. Deux nœuds sont reliés par une arête s'ils peuvent être amenés à être dépendants l'un de l'autre. Le graphe moral peut ainsi être vu comme une simplification engendrant la perte de certaines propriétés d'indépendance conditionnelle. Cependant, c'est une étape préliminaire nécessaire à la triangulation.

2.2.2 Triangulation

La deuxième étape consiste à trianguler le graphe moral G^m et à en extraire ses cliques qui formeront les futurs nœuds de l'arbre de jonction (voir FIG.4). La triangulation est nécessaire à la construction d'un arbre de jonction [13]. L'opération de triangulation consiste à ordonner les nœuds, puis à les éliminer selon cet ordre *i.e.* créer une clique, en rajoutant des arêtes, entre le nœud éliminé et ses voisins. Un graphe est alors triangulé si, et seulement si, il existe un ordre d'élimination tel que les nœuds peuvent être éliminés sans rajouter une seule arête. Les cliques du graphe triangulé forment les *clusters* de l'arbre de jonction, et la phase de propagation dépend directement de la taille des cliques. Or trouver un ordre d'élimination de sorte à obtenir des cliques de tailles minimales est un problème NP-difficile. On a donc recours à des heuristiques, plus ou moins sophistiquées, pour déterminer l'ordre d'élimination.

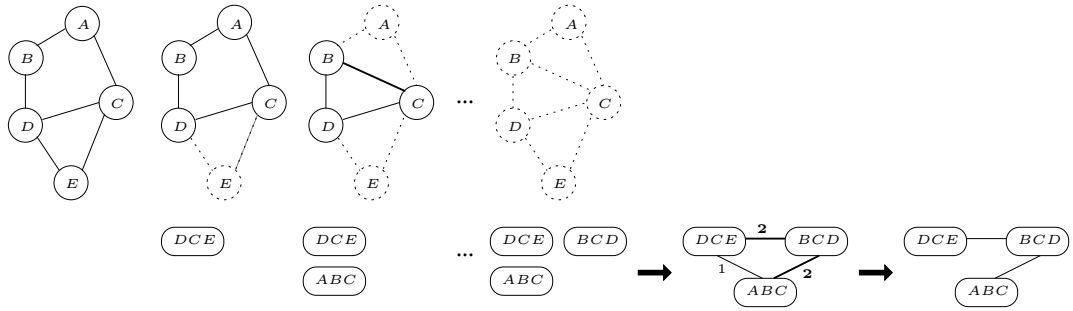


FIGURE 4 – Exemple de triangulation utilisant l'ordre d'élimination : E, A, C, D, B, puis de construction de l'arbre de jonction à l'aide de l'algorithme de Kruskal. Les cliques obtenues au fur et à mesure de l'élimination des variables sont également représentées.

2.2.3 Construction de l'arbre de jonction

Étant donné un graphe triangulé, on sait qu'il est possible de construire un arbre de jonction à partir de celui-ci (voir FIG.4). Pour l'obtenir, il suffit d'appliquer l'algorithme de Kruskal pour trouver l'arbre couvrant de poids maximal sur le graphe défini comme suit : soit $C_1, ..., C_n$ l'ensemble

des cliques du graphe moralisé et triangulé. On ajoute un nœud pour chaque clique C_i, et une arête (C_i, C_j), dont le poid est $|C_i \cap C_j|$, si $C_i \cap C_j \neq \emptyset$. L'algorithme de Kruskal appliqué à ce graphe donnera alors un arbre couvrant de poids maximal qui est un arbre de jonction.

2.2.4 Complexité du calcul d'inférence

Si $m(\sigma)$ désigne la taille de la clique maximale, *i.e.* $m(\sigma) = \max_i size(C_i)$, obtenue en utilisant l'ordre d'élimination σ, alors la complexité de la phase de propagation est en $O(m(\sigma))$. Il est important de noter que si $\mathbf{X}$ est un ensemble de variables, alors $size(\mathbf{X})$ est une valeur qui est exponentielle en fonction du nombre de variables de $\mathbf{X}$. Autrement dit, l'algorithme JT a une complexité exponentielle en fonction du nombre de variables. Pour réduire cette complexité, il est nécessaire de construire des cliques ayant un petit nombre de variables (et si possible, avec des variables ayant un petit nombre d'états). Autrement dit, il faut minimiser $m(\sigma)$. Ceci ne peut se faire que lors de l'étape de triangulation, et on a vu que trouver un ordre d'élimination tel que la taille des cliques formées soit la plus petite possible est un problème NP-difficile. Il s'ensuit que minimiser $m(\sigma)$ est un problème NP-difficile. L'étape de triangulation est donc l'étape critique de l'algorithme de l'arbre de jonction. Dans le cadre des réseaux bayésiens dynamiques, nous verrons que cette tâche est d'autant plus difficile.

3 Réseaux bayésiens dynamiques et inférences

3.1 Systèmes dynamiques

Nous nous intéressons ici aux systèmes qui évoluent dans le temps. L'évolution de tels systèmes peut-être modèlisée à l'aide d'un ensemble de variables discrètes : $\{\mathbf{X}_t : t \in \{1, ..., T\}\}$, qui représentent l'état du système à chaque instant t, et où chaque variable $\mathbf{X}_t$ est décomposée en un n-uplet $\mathbf{X}_t = (X_t^0, ..., X_t^n)$. Dans le cas de la propagation d'une maladie, l'état pourrait être composé du nombre de personnes atteintes, de la température, de la disponibilité des soins, etc...

En raison de la nature non-déterministe des systèmes étudiés, on utilise des processus stochastiques pour les modéliser. On définit donc une distribution de probabilité jointe $P(\mathbf{X}_1, ..., \mathbf{X}_T)$ sur l'ensemble des états possibles qu'a pu prendre le système au cours du temps. Du fait du nombre exponentiel d'états, et donc du nombre exponentiel de paramètres pour représenter cette loi, il est nécessaire de faire certaines hypothèses pour qu'elle soit manipulable en pratique. En faisant les hypothèses que le *processus est Markovien* et *invariant dans le temps*, il est possible de représenter la loi jointe de manière compacte à l'aide d'une *loi a priori* $P(\mathbf{X}_0)$, qui représente la distribution sur l'état initial du système $\mathbf{X}_0$; et un *modèle de transition* $P(\mathbf{X}_{t+1}|\mathbf{X}_t)$ qui donne la probabilité que le système passe dans l'état $\mathbf{X}_{t+1}$ sachant qu'il était dans l'état $\mathbf{X}_t$ au pas précédent.

Bien que l'on ait considérablement réduit la taille de la représentation de la loi jointe, comme les états $\mathbf{X}_i$ sont des n-uples, on risque de rester avec une loi ayant un nombre exponentiel de paramètres. Cependant, les indépendances locales dans un état permettent de factoriser le modèle de transition et la loi *a priori* à l'aide d'un réseau bayésien dynamique.

3.2 Le modèle dBN

Les réseaux bayésiens dynamiques peuvent être vus comme une extension temporelle des réseaux bayésiens. Comme précisé précédemment, on suppose que le processus étudié est *Markovien* et homogène, c'est à dire que la probabilité d'être dans un certain état au temps $t + 1$, ne dépend que de

l'état dans lequel on est au temps t et ne varie pas au cours du temps, d'où :

$$P(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = P(\mathbf{X}_t|\mathbf{X}_{t-1}) = P(\mathbf{X}_1|\mathbf{X}_0) \tag{1}$$

À partir de ces hypothèses, et en exploitant les indépendances au sein d'un état et entre les états, on peut utiliser un réseau bayésien pour représenter le processus de manière compacte :

Définition 1 *Un 2-TBN (Two time-slice temporal bayesian network) est un réseau bayésien défini comme suit : les sommets du graphe sont partitionnés en $\mathbf{X}_0$ et $\mathbf{X}_1$ appelés slices. Le 2-TBN représente alors les modèles de transition $P(\mathbf{X}_t|\mathbf{X}_{t-1})$ pour tout t, par $P(\mathbf{X}_1|\mathbf{X}_0)$ tel que :*

$$P(\mathbf{X}_1|\mathbf{X}_0) = \prod_{i=0}^{n-1} P(X_1^i|pa(X_1^i)) \tag{2}$$

et la loi a priori par :

$$P(\mathbf{X}_0) = \prod_{i=0}^{n-1} P(X_0^i|pa(X_0^i)) \tag{3}$$

On peut alors définir un dBN (voir FIG.5) :

Définition 2 *Un dBN d'horizon T est un réseau bayésien que l'on obtient en déroulant T fois le 2-TBN associé. Chaque état $\mathbf{X}_t$ du dBN est également appelé slice au temps t.*

D'après cette définition, il est donc possible d'appliquer l'algorithme JT directement sur la version déroulée du dBN en le considérant comme un BN. Cependant, cette méthode est, la plupart du temps, moins efficace que celle que nous allons présenter plus loin (section 3.3), et qui nécessite l'introduction de la notion d'*interface* (définition 3) : c'est un sous-ensemble de nœuds tel que si ils étaient supprimés, les informations du passé n'auraient plus aucune incidence sur le futur. Cette définition, que nous proposons, permet de réunir les différentes notions d'interfaces définies dans [9, 10, 5, 2] (proposition 1). De plus, c'est en fonction de la taille de l'interface que s'exprime la complexité des algorithmes que nous allons voir.

Définition 3 *Un sous-ensemble $\mathbf{I}$ de nœuds d'un dBN d'horizon T est appellé interface, si tout chemin d'un nœud de $\mathbf{X}_0$ à un nœud $\mathbf{X}_T$ passe par un sommet de $\mathbf{I}$ dans le dBN moralisé.*

Une interface correspond à une séparation des nœuds du passé et du futur par $\mathbf{I}$ dans le dBN moralisé. Cette définition généralise les notions d'interface avant et arrière suivantes [5] :

Définition 4 *Soit un dBN d'horizon T. L'interface avant $\mathbf{I}_t^{\rightarrow}$ est l'ensemble des nœuds au temps $t < T$ qui ont au moins un enfant au temps $t+1$. L'interface arrière $\mathbf{I}_t^{\leftarrow}$ est l'ensemble des nœuds au temps $t > 0$ d'un dBN qui ont, eux ou leurs enfants, un parent au temps $t-1$ (voir Fig.5).*

Une interface n'est pas nécessairement incluse entièrement dans un *slice*, contrairement aux interfaces avant et arrière, nous autorisons donc les interfaces à s'étaler sur plusieurs *slices*.

Comme la topologie du dBN possède une structure repétitive (voir FIG.5), il est possible, étant donnée une interface, d'en obtenir plusieurs autres par translation de celle-ci, à gauche ou à droite, dans le dBN. Pour la même raison, on pose $size(\mathbf{I}_t^{\rightarrow}) = i^{\rightarrow}$, $size(\mathbf{I}_t^{\leftarrow}) = i^{\leftarrow}$ pour tout t, et $size(\mathbf{X}_t) = s_{\text{slice}}$ pour tout $t > 0$.

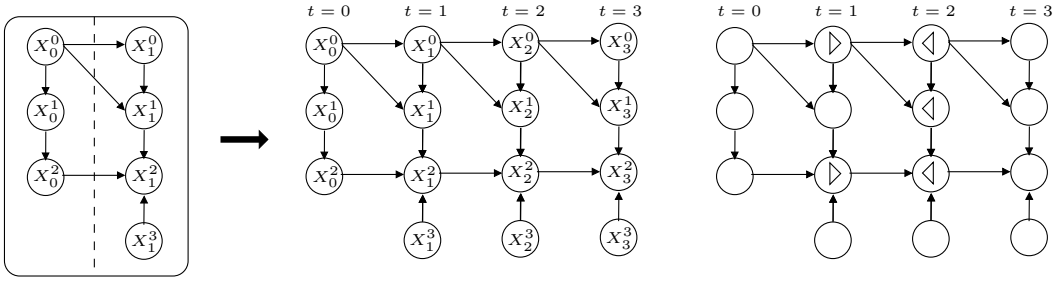


FIGURE 5 – Un exemple de réseau bayésien dynamique sous sa forme 2-TBN (à gauche) puis déroulée trois fois (au centre). Sur la figure à droite sont représentées : l'interface avant à $t = 1$ par $\triangleright$, et l'interface arrière à $t = 2$ par $\triangleleft$.

Proposition 1 *Soit un dBN d'horizon T. Alors $\mathbf{I}_t^{\rightarrow}$ pour $t < T$, et $\mathbf{I}_{t'}^{\leftarrow}$ pour $t' > 0$ sont des interfaces.*

Preuve : On démontre d'abord que $\mathbf{I}_t^{\rightarrow}$ est une interface. Soit $\mathbf{N}_t^{\rightarrow}$ les nœuds au temps t n'appartenant pas à $\mathbf{I}_t^{\rightarrow}$. Supposons qu'il existe un chemin d'un nœud de $\mathbf{X}_0$ à un nœud de $\mathbf{X}_T$ qui ne passe pas par un nœud de $\mathbf{I}_t^{\rightarrow}$. Il doit donc exister une arête (v_1, v_2) dans le dBN moralisé, avec $v_1 \in \mathbf{N}_t^{\rightarrow}$ et $v_2 \in \mathbf{X}_{t+1}$. Si (v_1, v_2) était un arc dans la version non-moralisé, alors v_1 serait dans $\mathbf{I}_t^{\rightarrow}$ ce qui est impossible. Donc, (v_1, v_2) doit être une arête rajoutée lors de la moralisation. Pour que cela soit possible, il faut que v_1 et v_2 aient un fils en commun dans le dBN non-moralisé. Ce fils est donc soit dans $\mathbf{X}_t$, soit dans $\mathbf{X}_{t+1}$. S'il est dans $\mathbf{X}_{t+1}$, alors v_1 possède un fils au temps $t+1$, et donc v_1 fait partie de l'interface $\mathbf{I}_t^{\rightarrow}$ ce qui est impossible. Enfin, si le fils appartient à $\mathbf{X}_t$, alors il existe un arc de v_2 vers ce dernier, ce qui n'est pas permis dans un dBN (pas d'arc du futur vers le passé). Par conséquent, l'arête (v_1, v_2) ne peut pas exister, et tous les chemins d'un nœud de $\mathbf{X}_0$ à un nœud de $\mathbf{X}_T$ passent donc par un nœud de $\mathbf{I}_t^{\rightarrow}$. La preuve que $\mathbf{I}_t^{\leftarrow}$ est une interface se démontre selon le même schéma de raisonnement. □

La définition 3 permet de faire le lien avec le problème de la coupe minimale dans un graphe. Le résultat précédent montre ainsi que les interfaces avant et arrière sont des coupes particulières, mais non minimales. Par conséquent, l'objectif sera d'utiliser un algorithme de recherche de la coupe minimale afin de trouver une interface de plus petite taille. Nous allons d'abord montrer, dans la section suivante, comment la taille de l'interface peut intervenir dans la complexité du calcul d'inférence dans un dBN.

3.3 Algorithme JT pour les dBNs

Une fois le réseau baysésien dynamique spécifié ou, plus précisèment, sa représentation 2-TBN, on est capable de l'utiliser pour effectuer plusieurs types d'inférences (voir FIG.6) qui peuvent être :

Filtrage : Calculer $P(\mathbf{X}_t|\mathbf{e}_{1:t})$.

Prédiction : Calculer $P(\mathbf{X}_{t+h}|\mathbf{e}_{1:t})$ avec $h > 0$.

Lissage : Calculer $P(\mathbf{X}_{t-l}|\mathbf{e}_{1:t})$ avec $0 \leq l \leq t$.

Lissage hors-ligne : Calculer $P(\mathbf{X}|\mathbf{e}_{1:T}), \forall \mathbf{X} \subseteq \prod_{t=0}^{T} \mathbf{X}_t$.

C'est ce dernier type d'inférence qui nous intéresse, et il existe à ce titre plusieurs méthodes de calcul pour mener à bien cette tâche. Cependant, nous allons nous intéresser à celles utilisant l'algorithme de l'arbre de jonction (JT).

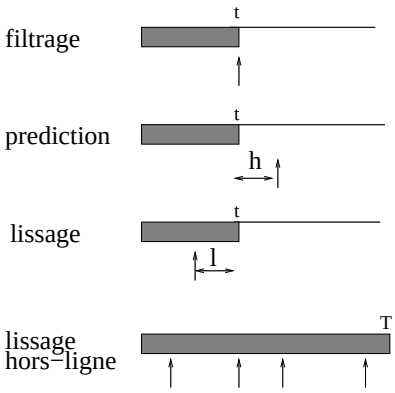


FIGURE 6 – Illustration des différents types d'inférence possibles dans un dBN [10]. La région grisée correspond à l'intervalle où l'on a des observations sur un certain nombre de variable. La flèche indique l'endroit où l'on souhaite effectuer l'inference, la requête.

Une des manières d'utiliser l'algorithme JT sur un dBN consiste à dérouler ce dernier, et à appliquer directement l'algorithme en considérant le dBN comme un réseau bayésien. Lors de la phase de triangulation, on peut décider d'utiliser soit l'ordre d'élimination produit par une heuristique, et que l'on appelle ordre d'élimination non-contraint, soit utiliser un ordre d'élimination contraint, c'est à dire trouver un ordre d'élimination qui soit compatible avec un ordre partiel que l'on aura préalablement choisi. Nous allons voir l'implication de l'un et l'autre de ces choix dans le cas des dBNs.

Ordre d'élimination non-contraint – Que se passe-t-il si on ne contraint pas l'ordre d'élimination ? *A priori*, on pourrait penser obtenir un meilleur arbre de jonction. Cependant, ce n'est pas le cas en pratique (la taille des cliques peut être fonction de l'horizon). Rappelons que trouver un ordre d'élimination optimal est un problème NP-difficile. De ce fait, les ordres d'élimination sont déterminés à l'aide d'un algorithme de type glouton. Dans une telle situation, il est souvent intéressant de limiter l'espace de recherche de ce type d'algorithme afin d'obtenir un optimum local de bonne qualité.

Un autre inconvénient de cette approche est que l'on est obligé de re-trianguler entièrement le dBN dès lors que l'on change l'horizon de celui-ci. C'est typiquement ce que l'on voudrait éviter de faire étant donné la lenteur du processus, et le fait qu'un dBN est, par nature, d'horizon non-fixé *a priori*. L'idée serait alors de créer un « petit »arbre de jonction à partir d'une fraction fixe du dBN de telle sorte que, si l'horizon du dBN change, il suffirait de rajouter à l'arbre de jonction le bon nombre de ces « petits »arbres de jonction. La difficulté vient du fait qu'il faut s'assurer qu'un tel arbre de jonction puisse exister et être modifié de cette manière. Clairement, il faut qu'un tel arbre de jonction posséde, comme le dBN, une structure répétitive ; le seul moyen d'obtenir une telle propriété est d'utiliser un ordre d'élimination contraint.

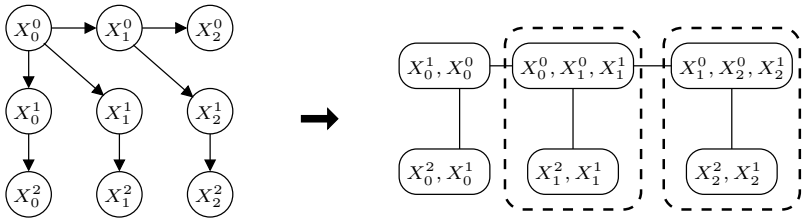


FIGURE 7 – Exemple de construction d'un arbre de jonction (à droite) en utilisant un ordre d'élimination SS-PF sur le dBN à gauche. La structure qui se répète est entourée en pointillé.

Ordre d'élimination contraint – Les ordres d'élimination contraints qui semblent les plus naturels sont les deux suivants : l'élimination *slice* par *slice* du futur vers le passé (SS-FP) qui consiste à enlever les nœuds du *slice* t avant ceux au *slice* $t-1$, et l'élimination *slice* par *slice* du passé vers le futur (SS-PF) qui consiste à enlever les nœuds du *slice* $t-1$ avant ceux au *slice* t [5, 10]. Afin de voir les implications de ces ordres d'éliminations, nous allons considérer la figure FIG.7. Dans cet exemple, on voit clairement une répétition de la structure apparaître dans l'arbre de jonction. Par conséquent, si le dBN augmente son horizon de 2, il suffit d'ajouter deux fois la structure entourée en pointillé afin d'obtenir le nouvel arbre de jonction. De plus, on peut noter que les interfaces arrières se retrouvent systématiquement dans une clique. C'est la conséquence du résultat suivant :

Propriété 1 *(Rose, 1976) Soit $X_1, ..., X_n$ un ordre d'élimination pour trianguler un graphe G, et soient X_i et X_j deux nœuds non-adjacents. Alors l'ordre d'élimination implique l'ajout de l'arête (X_i, X_j) si, et seulement si, il existe un chemin de X_i à X_j tel que tous les nœuds sur ce chemin soient éliminés avant X_i et X_j.*

Si $m(\sigma)$ désigne la taille de la clique maximale obtenue en utilisant l'ordre d'élimination σ, alors, d'une manière générale, on ne pourra obtenir une valeur m(SS-PF) (resp. de m(SS-FP)) plus petite que la taille de l'interface arrière (resp. avant). C'est la contre-partie à l'utilisation de tels ordres d'élimination. D'un autre côté, il a été montré que [5] :

Proposition 2 *Soit un dBN d'horizon T, alors m(SS-PF) $\leq i^{\leftarrow}.s_{slice}$, m(SS-FP) $\leq i^{\rightarrow}.s_{slice}$ et les bornes sont fines.*

De ces deux propositions, on en déduit que la complexité de l'inférence dans un dBN utilisant l'algorithme JT est en $O(i^{\leftarrow}.s_{\text{slice}})$ (resp. $O(i^{\rightarrow}.s_{\text{slice}})$) avec un ordre d'élimination SS-PF (resp. SS-FP) [3]. De plus, c'est sur de tels ordres d'élimination que l'algorithme d'interface [10] se base pour construire l'arbre de jonction uniquement en utilisant le 2-TBN, et sans le dérouler. Nous allons voir maintenant comment améliorer cette complexité en cherchant d'autres ordres d'élimination contraints.

4 Interface minimale et coupe minimale

Afin d'améliorer la complexité de l'algorithme JT utilisant, par exemple, un ordre d'élimination contraint SS-PF, il faudrait pouvoir trouver une interface dont la taille est plus petite que $i^{\leftarrow}$. Pour ce faire, nous proposons de formaliser le problème de recherche d'une interface de taille minimale dans un dBN, puis de montrer qu'il se réduit à un problème de coupe minimale. Ensuite, nous montrerons comment utiliser cette interface pour réduire la complexité de l'algorithme JT dans un dBN. Enfin, nous donnerons la complexité en temps de notre méthode.

4.1 Vers la recherche d'une interface optimale

Comme on peut le constater sur la figure 9, on peut trouver une interface dont la taille est strictement inférieure à celle de l'interface avant et arrière, et qui s'étend sur deux *slices*. Le problème MIN-INTERFACE de rechercher une interface minimale pouvant s'étaler sur plusieurs *slices* à donc un sens et s'énonce ainsi :

Définition 5 MIN-INTERFACE : *Étant donné un dBN d'horizon fixé, trouver une interface I de taille minimale.*

3. Ce sont bien des complexité exponentielles, voir paragraphe 2.2.4.

On rappelle également les problèmes MIN-CUT et MIN-VERTEX-CUT que l'on utilisera pour montrer notre réduction :

Définition 6 MIN-CUT : *Étant donné un graphe orienté pondéré $G = (V, E)$, et deux sommets s et t, trouver un ensemble d'arcs de poids minimum tel que tout chemin de s à t passe par un arc de la coupe.*

Définition 7 MIN-VERTEX-CUT : *Étant donné un graphe non-orienté $G = (V, E)$ avec des poids sur les sommets, et deux sommets s et t, trouver un ensemble de sommets de poids minimum $C \subseteq V - \{s, t\}$ tel que tout chemin de s à t passe par un sommet de C.*

Nous avons :

Proposition 3 MIN-INTERFACE $\prec_P$ MIN-VERTEX-CUT.

Preuve : Soit la transformation polynomiale qui transforme une instance $G = (\mathbf{X}_0 \cup ... \cup \mathbf{X}_T, E)$ de MIN-INTERFACE, en une instance $G' = (\{s, t\} \cup \mathbf{X}_0 \cup ... \cup \mathbf{X}_T, E')$ de MIN-CUT suivante :

- Moraliser G.
- Ajouter deux nœuds s et t.
- Ajouter une arête (s, X_0^i) pour chaque $X_0^i \in \mathbf{X}_0$, et une arête (X_T^j, t) pour chaque $X_T^j \in \mathbf{X}_T$.
- Le poids d'un nœud est égale à la log-cardinalité de la variable représentée par ce nœud.

Soit $\mathbf{I}$ une interface de taille minimale dans G. Soit $P' =< s, X_0^i, ..., X_T^j, t >$, un chemin de s à t dans G'. Par construction, le sous-chemin $P =< X_0^i, ..., X_T^j >$ de P' existe dans G moralisé, et passe par un sommet de $\mathbf{I}$. Par conséquent, tout chemin de s à t dans G' passe par un sommet de $\mathbf{I}$. Comme l'attribution des poids est la même dans G et G', alors $\mathbf{I}$ est une coupe-sommet minimale dans G'. La réciproque se démontre selon le même schema de raisonnement. □

Proposition 4 MIN-VERTEX-CUT $\prec_P$ MIN-CUT.

Preuve : Soit la transformation polynomiale qui transforme une instance $G = (V, E)$ de MIN-VERTEX-CUT, en une instance $G' = (V', E')$ de MIN-CUT suivante :

- Pour chaque sommet $v \in V - \{s, t\}$, ajouter deux sommets v_i (sommet *in*) et v_o (sommet *out*) dans V'. Ajouter également un arc de v_i à v_o et que l'on désignera par *arc-sommet*.
- Pour chaque arête $(s, v) \in E$ (resp.$(v, t) \in E$) avec $v \in V$, ajouter un arc (s, v) (resp. (v, t)) dans E'.
- Pour chaque arête $(u, v) \in E$, ajouter deux arcs $(u_o, v_i) \in E'$ et $(v_o, u_i) \in E'$.
- Orienter chaque arête (s, v) et (v, t).
- Pour chaque poids d'un sommet $v \in V$, ajouter le même poids à l'arc $(v_i, v_o) \in E'$.
- Pour tous les arcs de E' n'ayant pas reçu de poids, affecter le poids $+\infty$.

Par construction, il existe pour chaque sommet $v \in V - \{s, t\}$ un unique arc $(v_i, v_o) \in E'$ de même poids, et inversement.

Supposons qu'il existe dans G une coupe-sommet de poids minimale C. Soient C' l'ensemble des arc-sommets dans G' dont le sommet correspondant dans G est dans C, et P' un chemin de s à t dans G'. Alors, par construction, ce chemin emprunte nécessairement un arc-sommet tel que le sommet correspondant dans G soit dans la coupe C. Il s'ensuit que C' est une coupe de G'. De plus, comme l'affectation des poids sur les sommets de G et les arc-sommets de G' est la même, alors C' est une coupe minimale de G'.

Réciproquement, soient C' une coupe minimale dans G',et C l'ensemble des sommets dans G dont l'arc-sommet correspondant dans G' est dans C'. Comme les arc-sommets sont les seuls arcs ayant des poids finis, alors la coupe C' ne contient que des arc-sommets. En utilisant le même raisonnement que précédemment, on en déduit que C est une coupe minimale de G'. □

Grâce aux deux résultats précédents, on en déduit que :

Théorème 1 Min-Interface $\prec_P$ Min-Cut.

À horizon fixé, la recherche d'une interface optimale est donc un problème de coupe minimale résoluble en temps polynomial, et on notera $\mathbf{I}^{(t)}$ l'interface obtenue en résolvant Min-Interface à l'horizon $T = t$. Il reste maintenant à déterminer l'horizon t^* pour lequel la résolution Min-Interface donnera l'interface de taille la plus petite. La réponse est donnée par la proposition suivante :

Proposition 5 *Si* $t^* = \min_t\{t \in \mathbb{N} : size(\mathbf{I}^{(t)}) = size(\mathbf{I}^{(t+1)})\}$*, alors* $size(\mathbf{I}^{(t^*)}) \leq size(\mathbf{I}^{(t)})$ *pour tout* t*.*

Preuve : Soit $t^* = \min_t\{t \in \mathbb{N} : size(\mathbf{I}^{(t)}) = size(\mathbf{I}^{(t+1)})\}$. L'interface $\mathbf{I}^{(t^*)}$ est donc solution optimale pour le dBN aux horizons t^* et $t^* + 1$. Supposons que $size(\mathbf{I}^{(t^*)}) > size(\mathbf{I}^{(t^*+2)})$. L'interface $\mathbf{I}^{(t^*+2)}$ a donc soit au moins un nœud, $v \in \mathbf{I}^{(t^*)}$, supprimé, soit au moins un nœud remplacé par un autre de poids plus petit. Considérons le premier cas et un chemin d'un nœud de $\mathbf{X}_0$ à un nœud de $\mathbf{X}_{t^*+1}$ passant par un seul nœud de $\mathbf{I}^{(t^*)}$ qui est v. Par conséquent ce chemin ne passe pas par $\mathbf{I}^{(t^*+2)}$. Comme la topologie entre, et dans les *slices* t^* et $t^* + 1$, puis $t^* + 1$ et $t^* + 2$ est identique, et que ce chemin va d'un nœud de $\mathbf{X}_{t^*}$ à un nœud de $\mathbf{X}_{t^*+1}$, il peut alors se prolonger d'un nœud de $\mathbf{X}_{t^*+1}$ jusqu'à un nœud de $\mathbf{X}_{t^*+2}$. Il s'ensuit que $\mathbf{I}^{(t^*+2)}$ n'est donc pas une interface. Considérons le second cas, si le nœud de remplacement se trouve dans l'un des *slices* de 0 à $t^* + 1$. Comme la topologie des *slices* $t^* + 1$ et $t^* + 2$ est identique, on aurait dû trouver ces nœuds à l'itération précédente et donc $\mathbf{I}^{(t^*)}$ ne serait pas optimal ce qui est contradictoire. Si le nœud de remplacement se trouve dans le *slice* $t^* + 2$, alors, pour la même raison, ce nœud aurait pu être remplacé à l'étape précédente. Par conséquent $size(\mathbf{I}^{(t^*)}) \leq size(\mathbf{I}^{(t^*+2)})$. En appliquant ce raisonnement récursivement, on en déduit le résultat. □

Nous avons également les propriétés suivantes :

Propriété 2 *Pour tout* $t \in \mathbb{N}$*, on a :*

(i) $size(\mathbf{I}^{(t+1)}) \leq size(\mathbf{I}^{(t)})$
(ii) $size(\mathbf{I}^{(t)}) \leq size(\mathbf{X}_0)$

Preuve :
(i) Pour un dBN d'horizon t, tout chemin d'un nœud de $\mathbf{X}_0$ à un nœud de $\mathbf{X}_t$ passe, par définition, par un nœud de $\mathbf{I}^{(t)}$. De plus, à l'horizon $t + 1$, tout chemin d'un nœud de $\mathbf{X}_0$ à un nœud de $\mathbf{X}_{t+1}$ passe nécessairement par un nœud de $\mathbf{X}_t$ et donc par un nœud de $\mathbf{I}^{(t)}$. Il s'ensuit que $\mathbf{I}^{(t)}$ est une interface possible pour le dBN d'horizon $t + 1$. Par conséquent, on a $size(\mathbf{I}^{(t+1)}) \leq size(\mathbf{I}^{(t)})$.
(ii) Pour un dBN d'horizon 0, la seule interface possible est le premier *slice*, *i.e.* $\mathbf{I}^{(0)} = \mathbf{X}_0$. D'après (i), on a $size(\mathbf{I}^{(t)}) \leq size(\mathbf{I}^{(t-1)}) \leq ... \leq size(\mathbf{I}^{(0)}) = size(\mathbf{X}_0)$.
□

Pour trouver l'interface minimale, il suffit donc de résoudre Min-Interface en partant de $T = 1$ puis en incrémentant de un la valeur de T. On arrête dès que la taille de l'interface ne diminue

plus. On en déduit l'algorithme `Min-Interface` (voir FIG.8). C'est bien un algorithme polynomial puisque le nombre d'itérations dans la boucle est borné par $O(h)$, où $h = |\mathbf{X}_0|$, et les opérations dans la boucle s'effectuent en temps polynomial. En effet, d'après la propriété 2 (i), la taille de l'interface, avec initialement h nœuds, ne peut que diminuer ou rester égale. Si elle reste égale l'algorithme s'arrête, sinon, au pire des cas, à chaque itération et alternativement, on remplace un nœud par un autre de valeur plus faible, puis on en élimine un à l'itération suivante. On répéte ces deux opérations jusqu'à ce qu'il n'y ai plus qu'un seul nœud dans l'interface. Il s'ensuit que le nombre de passages dans la boucle est borné par $O(2h) = O(h)$.

Entrées: Un 2-TBN.
Sorties: Une interface
  min $\leftarrow +\infty$
  $\mathbf{I} \leftarrow$ Ensemble des nœuds du premier *slice*.
  $t \leftarrow 1$
  tantque $size(\mathbf{I}) <$ min **faire**
    min $\leftarrow size(\mathbf{I})$
    $\mathbf{I}_{\text{opt}} \leftarrow \mathbf{I}$
    $G \leftarrow$ Dérouler t fois le 2-TBN.
    Transformer G en une instance $G^{'}$ de MIN-VERTEX-CUT.
    Transformer $G^{'}$ en une instance $G^{''}$ de MIN-CUT.
    Résoudre MIN-CUT avec $G^{''}$ pour obtenir $\mathbf{I}$.
    $t \leftarrow t + 1$
  fin tantque
  Retourner $\mathbf{I}_{\text{opt}}$.

FIGURE 8 – Pseudo-code de l'algorithme `Min-Interface`.

4.2 Exploitation de l'interface

Étant donnée l'interface notée $\mathbf{I}_0$ trouvée à l'aide de l'algorithme `Min-Interface`, la prochaine étape consiste à trouver un ordre d'élimination contraint en prenant en compte cette interface. Pour ce faire, on considère les interfaces[4] $\mathbf{I}_i = \{X^k_{t+i} : X^k_t \in \mathbf{I}_0\}$ pour $i = 1, ..., n$ d'un dBN d'horizon T. Chaque $\mathbf{I}_i$ peut-être vue comme une « *translation* »de i de $\mathbf{I}_0$. De plus, pour chaque interface, il existe une partition des sommets du dBN en $\mathbf{P}_i$ (nœuds antérieurs à l'interface), $\mathbf{I}_i$ et $\mathbf{F}_i$ (nœuds postérieurs à l'interface) telle que tout chemin d'un nœud de $\mathbf{P}_i$ à un nœud de $\mathbf{F}_i$ passe par un nœud de $\mathbf{I}_i$. On définit alors l'ordre d'élimination MIN-ELIM de la façon suivante : Eliminer $\mathbf{P}_0$, puis, successivement, les nœuds $\mathbf{P}_i - \mathbf{P}_{i-1}$ pour $i = 1, ..., n-1$, où n est le nombre d'interface. Enfin, éliminer $\mathbf{I}_{n-1} \cup \mathbf{F}_{n-1}$ (voir FIG.9). L'algorithme final consiste alors à utiliser l'algorithme JT avec cet ordre d'élimination. On peut noter que rien n'est précisé sur l'ordre dans lequel éliminer les nœuds dans $\mathbf{P}_0$ et $\mathbf{P}_i - \mathbf{P}_{i-1}$. Dans nos expérimentations nous avons utilisé l'heuristique *min-fill*, mais d'autres approches peuvent être envisagées.

4.3 Complexité

Avant de donner la complexité de l'algorithme, nous devons prouver la propriété suivante :

Propriété 3 $size(\mathbf{P}_i - \mathbf{P}_{i-1}) = s_{slice}$ *pour* $i > 0$.

Preuve : Comme $\mathbf{I}_i$ peut être obtenue en translatant $\mathbf{I}_{i-1}$, il en est de même pour $\mathbf{P}_i$ et $\mathbf{P}_{i-1}$. En particulier, pour obtenir $\mathbf{P}_i$, il suffit d'ajouter dans $\mathbf{P}_{i-1}$ les nœuds X^k_{t+1} tel que $X^k_t \in \mathbf{P}_{i-1}$. On a donc $\mathbf{P}_i - \mathbf{P}_{i-1} = \{X^0_{t_1}, ..., X^n_{t_n}\}$, d'où $size(\mathbf{P}_i - \mathbf{P}_{i-1}) = size(\{X^0_{t_1}, ..., X^n_{t_n}\}) = s_{\text{slice}}$.

4. On pourra vérifier facilement que se sont bien des interfaces.

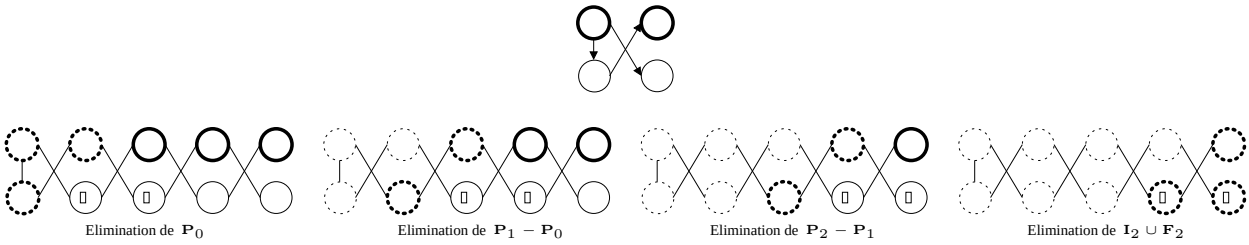


FIGURE 9 – Illustration de l'ordre d'élimination MIN-ELIM sur un dBN moralisé d'horizon $T = 4$. Les nœuds en trait épais indiquent des variables de cardinalité élevée. Enfin, les nœuds avec un rectangle à l'intérieur sont dans l'interface trouvée par `Min-Interface`.

□

Proposition 6 *En ne considérant pas les cliques de l'arbre de jonction créées en éliminant* $\mathbf{S}_0$ *et* $\mathbf{I}_{n-1} \cup \mathbf{T}_{n-1}$*, la complexité de l'algorithme est en* $O(size(\mathbf{I}).s_{slice})$ *avec* $size(\mathbf{I}) \leq min(i^{\rightarrow}, i^{\leftarrow})$*, où* $\mathbf{I}$ *est l'interface renvoyée par* `Min-Interface`*.*

Preuve : Comme on a montré que le calcul de l'interface $\mathbf{I}$ se fait en temps polynomial, la complexité de l'algorithme est dominée par la complexité du calcul dans l'arbre de jonction. Pour obtenir l'arbre de jonction, on utilise l'ordre d'élimination MIN-ELIM. Or, lorsque l'on élimine $\mathbf{P}_i - \mathbf{P}_{i-1}$, l'ensemble des nœuds impliqués est $(\mathbf{P}_i - \mathbf{P}_{i-1}) \cup \mathbf{I}_{i+1}$ donc la taille de la clique créée est inférieure à $size(\mathbf{P}_i - \mathbf{P}_{i-1}).size(\mathbf{I}) = size(\mathbf{I}).s_{\text{slice}}$. Enfin, comme $\mathbf{I}$ est l'interface de taille minimale, et $\mathbf{I}^{\rightarrow}$ et $\mathbf{I}^{\leftarrow}$ sont des interfaces particulières, alors nécessairement la taille de $\mathbf{I}$ est inférieure ou égale à celle de $\mathbf{I}^{\rightarrow}$ et $\mathbf{I}^{\leftarrow}$. □

La complexité exprimée ci-dessus ne prend pas en compte la taille des cliques créées lorsque l'on élimine $\mathbf{S}_0$ et $\mathbf{I}_{n-1} \cup \mathbf{T}_{n-1}$, puisque ces dernières, même si elles peuvent avoir une taille supérieure à $size(\mathbf{I}).s_{\text{slice}}$, ne sont présentes qu'en début et fin de l'arbre de jonction. Ainsi, si le dBN est suffisamment long, la taille moyenne des cliques deviendra proche de $size(\mathbf{I}).s_{\text{slice}}$.

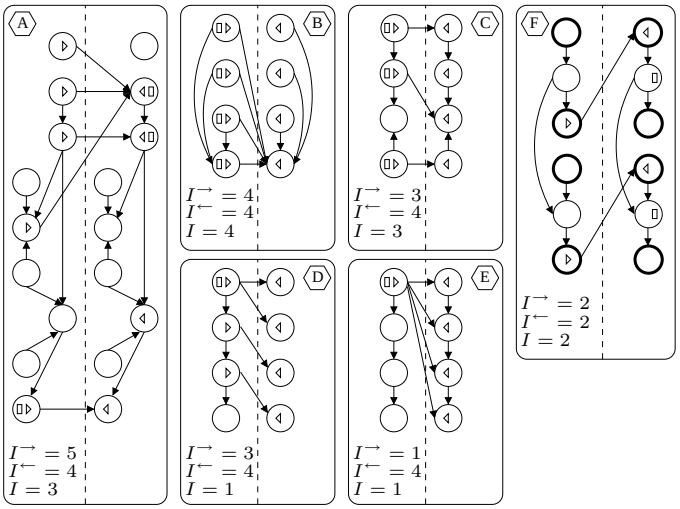
5 Résultats expérimentaux

Dans cette partie, nous présentons les premiers résultats obtenus sur des réseaux bayésiens dynamiques connus (voir FIG.10) [5], et générés aléatoirement (voir FIG.11). Pour ce faire, nous avons utilisé la librairie `C++` aGrUM dévéloppée au sein du LIP6, et utiliser les ordres d'élimination vus précédemment.

Pour la figure 10, nous donnons, pour chaque dBN et chaque ordre d'élimination employé, la taille moyenne des cliques dans l'arbre de jonction obtenu, mais également la plus grande et la plus petite clique en nombre de nœuds et en taille. Enfin, on donne le nombre total de nœuds dans l'arbre. La taille d'une clique est le log du produit des cardinalités des variables de cette clique. Comme on peut le constater, parmi tous les ordres d'élimination, l'ordre MIN-ELIM donne toujours une taille moyenne des cliques qui est minimale. Toutefois, à part pour un dBN, au moins une autre méthode donne le même minimum. Pour nous assurer que ce cas n'est pas rare, nous avons également étudier le comportement de notre algorithme dans le cas de génération aléatoires de dBNs.

La figure 11 donne la taille moyenne des cliques en fonction de l'horizon sur des dBN générés aléatoirement. À un nombre de nœud par *slice* (5 puis 10) et horizon fixés, nous avons généré une trentaine de dBN pour appliquer, sur chacun d'eux, les quatre méthodes d'élimination, puis nous

5. Les tailles sont obtenues comme somme des *log* des poids des nœeuds.



dBNs	Mesures	LIBRE	SS-FP	SS-PF	MIN-ELIM
MILDEW (A)	Taille moy.	3.28917	3.16743	**3.03157**	**3.03098**
	Min-max noeuds	2–7	1–6	1–5	1–5
	Min-max taille	0.69–4.85	0.69–3.46	0.69–3.46	0.69–3.46
	Cliques	3592	4004	4004	4005
WORST (B)	Taille moy.	**5.54518**	**5.54518**	**5.54518**	**5.54518**
	Min-max noeuds	8–8	8–8	8–8	8–8
	Min-max taille	5.54–5.54	5.54–5.54	5.54–5.54	5.54–5.54
	Cliques	500	500	500	500
GIMPEL (C)	Taille moy.	**2.77120**	**2.77224**	3.00210	**2.77155**
	Min-max noeuds	3–4	3–4	2–5	3–4
	Min-max taille	2.08–2.77	2.08–2.77	1.39–3.46	2.08–2.77
	Cliques	2001	2001	1501	2001
TREE (D)	Taille moy.	**1.38629**	**1.38629**	3.00256	**1.38629**
	Min-max noeuds	2–2	2–2	2–5	2–2
	Min-max taille	1.39–1.39	1.39–1.39	1.39–3.46	1.39–1.39
	Cliques	2003	2003	1501	2003
CASCADE (E)	Taille moy.	**2.07809**	**2.07806**	3.45883	**2.07806**
	Min-max noeuds	2–3	2–3	2–5	2–3
	Min-max taille	1.39–2.08	1.39–2.08	1.39–3.46	1.39–2.08
	Cliques	1543	1503	502	1503
BILMES (F)	Taille moy.	3.17167	3.46458	3.23338	**3.00264**
	Min-max noeuds	2–5	2–3	2–3	2–3
	Min-max taille	2.08–3.46	2.08–4.16	2.08–3.46	2.08–3.46
	Cliques	2647	3004	3004	3004

FIGURE 10 – Figure de gauche : Les dBNs utilisés dans les expériences. Les interfaces avant et arrière sont représentées selon la convention donnée à la figure 5. Les nœuds avec un rectangle sont dans l'interface trouvée par `Min-Interface`. Dans F, les nœuds en trait épais indiquent des variables de cardinalité élevée. Enfin, les chiffres donnent le nombre de nœuds dans l'interface de couleur correspondante. Le réseau A vient de [3], B, D et E sont de [10], et F vient de [2]. Le dBN B représente le pire des cas du point de vu de la complexité, puisque l'on est obligé de créer une clique qui contienne tous les nœuds. Figure de droite : Résultats obtenus sur ces dBNs à l'horizon $T = 500$.

avons aggrégé les résultats en faisant une moyenne. Comme on peut le constater, la taille moyenne des cliques est minimale en utilisant l'ordre d'élimination MIN-ELIM. Ce résultat valide notre approche en montrant que pour un nombre non négligeable de structures, MIN-ELIM donne de meilleurs résultats que toutes les autres approches.

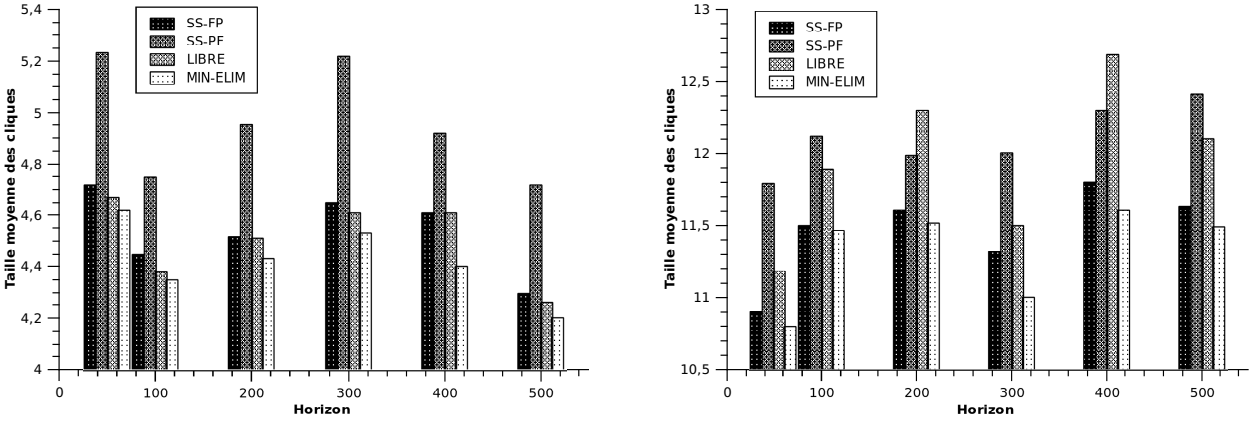


FIGURE 11 – Histogramme représentant la taille moyenne des cliques en fonction de l'horizon obtenue sur une trentaine de dBN générés aléatoirement avec cinq variables par *slices* (à gauche) puis dix variables par *slices* (à droite).

6 Conclusion

Dans ce travail, nous nous sommes intéressés à la recherche d'ordres d'éliminations contraints efficaces pour la triangulation d'un dBN. En définissant formellement le problème de la recherche d'une interface de taille minimale dans un dBN, nous avons pu faire le lien avec un problème classique issu de la recherche opérationnelle. Ce rapprochement a alors permis de déduire un algorithme poly-

nomial pour trouver une interface optimale, et en déduire un ordre d'élimination contraint efficace. Les résultats montrent que notre approche permet, par un prétraitement polynomial, d'optimiser effectivement les calculs d'inférence.

Les suites à donner à ce travail nous semblent être de deux types : dans un premier temps, il nous semble que l'interface optimale pourrait être utilisée efficacement dans d'autres algorithmes. En particulier, les inférences approchées (*Loopy Belief Propagation*, algorihtme de la frontière factorisée [11], etc...) utilisent le même genre de concept et pourraient donc être améliorées de la même façon. Dans un second temps, nous avons proposé un ordre d'élimination contraint basé sur la notion d'interface et nous prétendons avoir trouvé une méthode pour trouver l'interface optimale. Néanmoins, d'autres ordres d'éliminations contraints pourraient exister, basés sur d'autres principes de construction. Il nous apparait important de continuer notre investigation dans cette direction.

Remerciements

Cette étude a été financée par le projet ANR SKOOB (`http://skoob.lip6.fr`).

Références

[1] Cédric Baudrit, Mariette Sicard, Pierre-Henri Wuillemin, and Nathalie Perrot. Dynamic bayesian networks for modelling food processing : Application to the cheese ripening process. In *8th World Congress of Chemichal Engineering 09*, Montréal (Canada), 2009.

[2] J. Bilmes and C. Bartels. On triangulating dynamic graphical models. In *UAI*, pages 47–56, Acapulco, Mexico, 2003. Morgan Kaufmann Publishers.

[3] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proc. Fourteenth Annual Conference on Uncertainty in AI (UAI)*, pages 33–42, 1998.

[4] Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artif. Intell.*, 42(2-3) :393–405, 1990.

[5] Adnan Darwiche. Constant-space reasoning in dynamic bayesian networks. *International Journal of Approximate Reasoning*, 26 :161–178, 2001.

[6] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Comput. Intell.*, 5(3) :142–150, 1990.

[7] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11 :391–427, 1999.

[8] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quaterly*, 4 :269–282, 1990.

[9] Uffe Kjaerulff. dhugin : A computational system for dynamic time-sliced bayesian networks. *International Journal of Forecasting*, 11 :89–111, 1994.

[10] Kevin Murphy. *Dynamic Bayesian Networks : Representation, Inference and Learning*. PhD thesis, University of California, 2002.

[11] Kevin Murphy and Yair Weiss. The factored frontier algorithm for approximate inference in dbns. In *UAI*, pages 378–385, 2001.

[12] Patrick Naïm, Pierre-Henri Wuillemin, Philippe Leray, Olivier Pourret, and Anna Becker. *Réseaux bayésiens*. Eyrolles, troisième edition, 2007.

[13] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems :Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[14] Philippe Weber. Dynamic bayesian networks model to estimate process availability. In *8th International Conference Quality, Reliability, Maintenance*, Sinaia (Romania), 2002.

[15] G. Zweig. A forward-backward algorithm for inference in Bayesian networks and an empirical comparison with HMMs. Master's thesis, Dept. Comp. Sci., U.C. Berkeley, 1996.

Approches réactives robustifiées pour l'ordonnancement des trains sur une voie unique en présence d'aléas

Mohand Ait Alamara[1,2,3], Francis Sourd[1], Mohamed Ali Aloulou[2,3]

[1] Direction de l'Innovation et de la Recherche SNCF ; 45 Rue de Londres, 75379 Paris CEDEX 08
{francis.sourd,mohand.ait-alamara}@sncf.fr
[2] Université Paris Dauphine, LAMSADE, F-75775 Paris, France
[3] CNRS, FRE 3234, F-75016 Paris, France
aloulou@lamsade.dauphine.fr

Résumé : *Dans cet article, nous nous intéressons à la gestion de l'aléa dans le calcul d'un plan de circulation ferroviaire lorsqu'un incident se produit sur une voie. L'incertitude porte sur les durées de traversée des trains. Ce problème s'apparente à un problème d'ordonnancement à une machine dont l'objectif est de minimiser la somme des retards. Nous proposons deux approches d'optimisation robuste. La première approche s'inspire de la méthode Berstimas&Sim, elle paramètre la robustesse par un nombre limite de trains dont la durée est à son extrême. Dans la deuxième approche, la robustesse est paramétrée par un budget de retard à affecter aux trains. Ces méthodes robustes permettent de pallier au conservatisme des méthodes du type minmax. Nous introduisons des algorithmes réactifs pour « coller plus à la réalité ». Et pour finir, nous proposons des algorithmes alliant algorithmes réactifs et robustes et nous montrons, à travers des expérimentations, la supériorité d'une telle approche par rapport à une méthode totalement réactive.*

Mots-Clés : *Gestion de l'aléa, robustesse, ordonnancement dans l'incertain, algorithmes réactifs.*

1 Introduction

Le Réseau Ferré Français est l'un des plus importants au monde. Il connaît une activité dense, et intervient comme acteur économique et social de premier plan. Il est aussi complexe et nécessite une organisation rigoureuse et un entretien continu. Pour cette raison, un dysfonctionnement même minime, peut engendrer des désagréments importants. Les incidents sur les voies en font partie. Leurs conséquences peuvent varier d'un simple petit retard à un blocage complet du réseau.
Un des problèmes récurrents est celui où une des deux voies d'un tronçon est indisponible. Le plan de circulation doit alors être reconstruit en prenant en compte cette nouvelle contrainte et en se fixant comme objectif de réduire la gêne occasionnée. Des algorithmes d'optimisation du plan de circulation existent. Ils fournissent des solutions qui permettent de réduire le retard global de 30 % par rapport aux procédures opérationnelles utilisées auparavant [8]. Ces solutions peuvent être sensibles à des variations légères des paramètres d'entrée. Il est alors nécessaire de tenir compte de l'incertitude qui pèse sur ces paramètres et ainsi de construire des plans de circulation avec des garanties de performances compte tenu des variations que peuvent subir les paramètres d'entrée.
Le papier est structuré de la façon suivante : la section 2 est consacrée à la définition générale du

problème et à sa modélisation formelle. Dans la section 3, nous rappelons la méthode de résolution par programmation dynamique décrite dans [8] appliquée au problème sans prise en compte de l'aléa. Deux approches robustes, inspirées de la méthode de Bertsimas & Sim [2], sont ensuite introduites dans la section 4. Dans la section 5, nous proposons des approches purement réactives et des approches réactives robustifiées basées sur les algorithmes robustes décrits dans les sections 3 et 4. Pour finir, nous montrons à travers des expérimentations la supériorité des approches réactives robustifiées par rapport à des approches purement réactives.

2 Contexte de l'étude

2.1 Définition du problème

Sur le Réseau Ferré Français, les lignes principales sont en double voie, ce qui permet le croisement des trains y circulant. Toutefois, il arrive qu'un tronçon (section de voie) soit mis en voie unique :

- soit en raison d'un incident sur une voie en cours d'exploitation.
- soit en raison de travaux.

Les origines des incidents sont multiples : un arrachement de caténaire, un signal d'alarme activé, une rupture de rail, une panne de signalisation... La durée de l'incident est généralement inconnue. Dans ce cas, les trains utilisant la voie accidentée doivent se partager le tronçon de voie unique sous la contrainte principale que deux trains circulant dans des sens contraires ne peuvent utiliser cette ressource simultanément. En revanche, deux trains circulant dans le même sens peuvent, à condition que la voie soit équipée en signaux nécessaires, rouler sur ce même tronçon en respectant une vitesse limitée et un écart minimal.

On se place ici dans le cas d'un tronçon traversé par deux voies en sens inverses, entre deux points remarquables (points d'où il est possible d'effectuer des bifurcations, typiquement des gares). Suite à un incident, un tronçon doit être momentanément fermé. Dans le cas où une seule des deux voies est indisponible, on utilisera les aiguillages de bifurcation pour faire circuler les trains des deux sens en alternance sur la voie disponible.

Pour des raisons de sécurité, un écart est intercalé entre les trains. Nous distinguons deux cas :

- **les deux trains sont dans le même sens :** Comme les trains peuvent avoir des vitesses différentes, l'écart entre deux trains dépendra de leurs durées de traversée. Cet **espacement** sera modélisé de manière à avoir, entre les deux trains, un écart temporel minimal dépendant de l'infrastructure.
- **les deux trains sont dans deux sens différents :** Deux trains circulant dans des directions opposées ne peuvent se trouver sur un même tronçon de voie. Un train ne peut s'engager dans une voie que si le train précédent ne s'y trouve plus. De plus, un temps de **séparation** est intercalé entre les deux trains. Cet écart correspond globalement au temps nécessaire au premier train pour sortir de la déviation.

L'objectif est d'écouler les trains pendant la durée de l'incident en minimisant la somme des retards. Nous considérons qu'aucun train n'est *a priori* en avance. Enfin, certains trains étant plus prioritaires que d'autres (ex. train voyageurs *vs* train fret...), on pondère les trains selon leur importance. On obtient alors comme critère la minimisation de la somme pondérée des retards des trains. Nous pouvons aussi imaginer la possibilité de supprimer un train. Dans certaines situations, annuler un train permet de réduire considérablement le retard global (ex. un train lent). Cette suppression a bien sur un coût (remboursement des usagers, image de marque...). On a là un critère supplémentaire, celui qui consiste à minimiser le coût global des trains annulés. Un autre critère intéressant à considérer est le nombre de trains en retard et/ou supprimés. Ces critères peuvent être considérés simultanément (approches multi-critères). Dans ce papier, nous nous limiterons au critère somme pondérée des retards.

2.2 Modélisation formelle du problème

2.2.1 Formalisation en problème d'ordonnancement

La plupart des problèmes d'optimisation de la circulation des trains sur un réseau sont très complexes et sont modélisés par des programmes linéaires ou résolus de manière approchée (voir par exemple [6]). Dans le cas du tronçon de voie unique décrit ci-dessus, le problème peut être modélisé comme un problème d'ordonnancement classique à une machine.
Tout d'abord, considérons le cas où un seul train peut emprunter le tronçon de voie unique, ce qui correspond dans la pratique au cas où la voie n'est pas équipée de signalisation de contre-sens. Comme la ressource "voie unique" est bloquée pendant tout le passage du train, le problème est typiquement un problème à une machine où chaque parcours du tronçon par un train est représenté par une tâche. Lorsque deux trains circulant dans le même sens peuvent emprunter simultanément le tronçon de voie unique, les contraintes de circulation peuvent être modélisées en ajoutant des temps de *setup* au modèle à une machine. Plus précisément, les tâches ont cette fois une durée nulle. Les temps de *setup* correspondent aux écarts décrits dans la section précédente, ils dépendent de la séquence et se décomposent en deux types :

1. **espacement** : écart entre deux trains i, i' dans le même sens (i' précède i). Cet écart est égal à $esp(i, i') = \text{ESP} + \max(0, p_i - p_{i'})$, avec p_i durée de traversée du train i et ESP constante dépendant de l'infrastructure.
2. **séparation** : écart entre deux trains qui sont dans deux sens opposés, il égale : $sep(i, i') = \text{SEP} + p_{i'}$, avec SEP constante dépendant de l'infrastructure.

Il y a de plus la contrainte que les trains circulant dans un même sens ne peuvent pas se doubler. Nous avons donc deux chaînes de précédences entre les tâches. Selon la notation $\alpha|\beta|\gamma$ proposée par Graham et al.[3] et étendue pour les processus avec *setup* par Allahverdi *et al.* [1], le problème considéré est :

$$1|ST_{sd}, chains, r_i| \sum w_i L_i$$

avec :
- ST_{sd} : les *setup* dépendent de la séquence.
- $chains$: contraintes de précédences entres les trains.
- r_i : date de disponibilité du train i.
- $\sum w_i L_i$: somme pondérée des retards des trains.

2.2.2 Prise en compte de l'incertitude

Modélisation de l'aléa sur la durée de traversée Le temps de traversée p_i des trains entre deux points remarquables peut varier de façon importante. Cette variation peut avoir pour cause une défaillance matérielle, un aléa météorologique, mais le plus souvent cela dépend des conducteurs. La multiplicité et l'imprévisibilité des origines des variations rendent toute modélisation à base de probabilités très difficile. Pour simplifier, on supposera que l'ensemble de valeurs de la durée de traversée du train i est l'intervalle discret $[\underline{p}_i, \overline{p}_i]$. En considérant que les trains ne sont pas en avance, on posera $\underline{p}_i = p_i$ la valeur déterministe nominale. Notre modélisation de l'incertain nous conduit donc à concevoir une solution du type "optimisation discrète robuste".

Critère de robustesse : le critère qui peut paraitre le plus adapté est celui du pire cas (*minmax*) tel que défini dans Kouvelis and Yu [7], parce que couvrant un maximum de risques. Cette approche ne convient pas à ce que nous cherchons, car cela revient à considérer un cas faiblement probable,

celui où la durée de traversée de chaque train i est à sa valeur maximale $\overline{p}_i$. Dans la section 4, nous allons pallier au conservatisme de l'approche *minmax* en considérant deux approches où la robustesse est paramétrée.

3 Résolution du cas déterministe

Dans cette partie, nous supposons que les données du problème sont connues avec certitude. La Direction de l'Innovation et de la Recherche de la SNCF a développé une application (SIOUCS [1]) qui s'appuie sur la programmation dynamique et qui résout notre problème dans des temps appréciables (cf F. Sourd et Ch. Weber [8]).
La programmation dynamique s'adapte bien au problème défini précédemment. En effet, décider, à un instant donné t, lequel des trains i (sens direct [2]) ou j (sens inverse) faire passer revient à calculer l'ordonnancement optimal des sous-problèmes restreints suivants :
- $i-1$ trains dans le sens direct et j trains dans le sens inverse
- i trains dans le sens direct et $j-1$ trains dans le sens inverse

On choisira la solution minimisant le retard global.

3.1 Formulation en programmation dynamique

Définitions des variables de la programmation dynamique

Un état $\mathcal{E}(i,j,s,t)$ de la programmation dynamique est défini par la donnée de 4 variables décrites dans le tableau 1. On associe à chaque état $\mathcal{E}(i,j,s,t)$ une fonction de coût $\mathcal{F}(i,j,s,t)$ qui représente

Table 1 – Variables de la programmation dynamique

variable	définition	domaine
i	le nombre de trains passés dans le sens direct	$[1,d]$
j	le nombre de trains passés dans le sens inverse	$[1,r]$
s	sens du dernier trains	$\{direct, reverse\}$
t	date de fin de passage du dernier train passé	[0,T]

la somme pondérée des retards, à l'instant t, du passage de i trains dans le sens direct, j trains dans le sens inverse, tout en sachant que le dernier train est passé dans le sens s. Notons le dernier train passé dans le sens s par $tr^s_{\mathcal{E}}$.

Conditions initiales

- aucun train ne circule

$$\mathcal{F}(0,0,s,t) = 0,\ \forall s \in \{direct, reverse\}, \forall t \in [0,T] \tag{1}$$

- le train $tr^s_{\mathcal{E}}$ n'est pas disponible à la date $t - p_{tr^s_{\mathcal{E}}}$

$$\mathcal{F}(i,j,s,t) = +\infty,\ \forall i,j,s,t : t < r_{tr^s_{\mathcal{E}}} + p_{tr^s_{\mathcal{E}}} \tag{2}$$

1. SIllons Optimisés pour l'Utilisation du ContreSens
2. Le réseau français est en étoile et est centré sur Paris. Il est convenu de nommer par *direct* le sens des trains qui se dirigent vers Paris et par *inverse* l'autre sens.

Équations de Bellman

A l'étape précédente, deux états sont possibles :
– $\mathcal{E}^s_{prec}$: le sens de passage du dernier train est le même que s. Dans ce cas, les deux trains $tr^s_{\mathcal{E}}$ et $tr^s_{\mathcal{E}_{prec}}$ sont séparés par un espacement

$$esp^s(tr^s_{\mathcal{E}}, tr^s_{\mathcal{E}_{prec}}) = esp^s + max(0, p_{tr^s_{\mathcal{E}_{prec}}} - p_{tr^s_{\mathcal{E}}})$$

avec esp^s l'espacement minimal entre deux trains qui se suivent dans le sens s.
– $\mathcal{E}^{\bar{s}}_{prec}$: le sens du passage du dernier train est l'opposé de s. Les trains $tr^s_{\mathcal{E}}$ et $tr^{\bar{s}}_{\mathcal{E}_{prec}}$ sont séparés par un temps de séparation $sep^{\bar{s}} + p_{tr^s_{\mathcal{E}}}$
Le coût $\mathcal{F}(i, j, s, t)$ est alors le minimum des coûts de $\mathcal{E}^s_{prec}$ et de $\mathcal{E}^{\bar{s}}_{prec}$ additionné au coût du passage du train $tr^s_{\mathcal{E}}$ à la date t calculé par

$$cout(tr^s_{\mathcal{E}}, t) = w_{tr^s_{\mathcal{E}}}(t - p_{tr^s_{\mathcal{E}}} - r_{tr^s_{\mathcal{E}}})$$

Nous obtenons les équations de récurrences suivantes :

$$\begin{aligned}
\mathcal{F}(i, j, direct, t) &= \min \left\{ \begin{array}{l} \mathcal{F}(i-1, j, direct, t - esp^s(i, i-1) + cout(i, t)) \\ \mathcal{F}(i-1, j, reverse, t - p_i - sep^r) + cout(i, t) \end{array} \right. \\
\mathcal{F}(i, j, reverse, t) &= \min \left\{ \begin{array}{l} \mathcal{F}(i, j-1, direct, t - p_j - sep^d) + cout(j, t) \\ \mathcal{F}(i, j-1, reverse, t - esp^r(j, j-1)) + cout(j, t) \end{array} \right.
\end{aligned} \tag{3}$$

La solution du problème est donnée par :

$$\text{SOLUTION} = \min \left\{ \begin{array}{l} \mathcal{F}(d, r, direct, T) \\ \mathcal{F}(d, r, reverse, T) \end{array} \right. \tag{4}$$

3.2 Complexité

La méthode par programmation dynamique nécessite la construction et le remplissage d'une table à 4 dimensions qui correspondent aux 4 variables

$$\text{TABLE}[1..d][1..r][direct, reverse][0..T]$$

qui est donc de taille $d \times r \times 2 \times T$. Le calcul de chaque case (un min) se fait en $O(1)$. Lors de la phase de *backtracking*, on remonte la table de la programmation dynamique dans le chemin inverse de celui qui a déterminé la solution optimale et ce jusqu'à arriver à $\text{TABLE}[0][0][s][0]$. Cette remontée se fait en $d + r$ pas, chaque pas en $O(1)$. Le coût de cette dernière phase étant négligeable devant le coût de construction de la table de la programmation dynamique, la complexité temporelle de l'algorithme est en $O(n^2T)$. C'est donc un algorithme pseudo-polynomial.

4 Approches robustes

Jusqu'à présent, nous avons considéré des modèles où les paramètres étaient connus de façon déterministe. Ces paramètres sont : la durée de traversée, la date de disponibilité (d'arrivée), la durée de l'incident, les temps d'espacement et de séparation. Dans la réalité, les valeurs de ces paramètres ne sont pas toujours connues ou bien connues de manière imprécise, et une perturbation de l'un d'eux peut avoir des conséquences sur l'optimalité (voir la faisabilité) de la solution fournie par le modèle

sans aléa.
Vu la complexité du problème, prendre en compte tous ces paramètres comme incertains nous conduit à des modèles, a priori, trop difficiles à résoudre. L'étude s'est donc portée sur le seul paramètre durée de traversée.
Dans la suite, nous proposons deux approches robustes. La première s'inspire de la méthode Berstimas&Sim, elle paramètre la robustesse par un nombre limite de trains dont la durée est à son extrême. Dans la deuxième approche, la robustesse est paramétrée par un budget de retard à affecter aux trains.

4.1 Robustesse paramétrée par un nombre Γ de trains dont le temps de traversée est au pire cas (approche de Bertsimas & Sim)

L'approche de Bertsimas & Sim [2] part du principe qu'il est très peu probable que tous les paramètres incertains dévient simultanément dans le mauvais sens. Le but est de se protéger de tous les cas où au plus Γ trains ont leur temps de traversée au pire cas.

4.1.1 Formulation par programmation dynamique

Nous étendons la formulation proposée dans la section 3. Un état est représenté par un tuple $\mathcal{E}(i,j,s,t,\gamma)$ où i (resp. j) est le nombre de trains déjà passés dans le sens direct (resp. inverse), s le sens du dernier passage, t la date du dernier passage et γ le nombre de trains qui ont leur temps de traversée au maximum. On associe à chaque état $\mathcal{E}(i,j,s,t,\gamma)$ une fonction de coût $\mathcal{F}(i,j,s,t,\gamma)$ qui représente la somme pondérée des retards. La formulation du problème par programmation dynamique se fait de la manière suivante :

Conditions initiales
- aucun train ne circule : $\mathcal{F}(0,0,s,t,0) = 0,\ \forall s \in \{direct, reverse\}, \forall t \in [0,T]$
- condition de disponibilité du train $tr^s_{\mathcal{E}}$: $\mathcal{F}(i,j,s,t,\gamma) = +\infty,\ \forall i,j,s,\gamma,t : t < r_{tr^s_{\mathcal{E}}} + p_{tr^s_{\mathcal{E}}}$
- conditions de cohérence : $\mathcal{F}(i,j,s,t,\gamma) = +\infty, \forall t, \forall i,j,\gamma : i + j < \gamma$

Équations de Bellman

$$\mathcal{F}(i,j,direct,t,\gamma) = \min \left\{ \begin{array}{l} \max \left\{ \begin{array}{l} \mathcal{F}(i-1,j,direct,t-\widetilde{esp}^d_i(\overline{p_i}),\gamma-1) + cout(i,t) \\ \mathcal{F}(i-1,j,direct,t-\widetilde{esp}^d_i(\underline{p_i}),\gamma) + cout(i,t) \end{array} \right. \\ \max \left\{ \begin{array}{l} F(i-1,j,reverse,t-\overline{p_i}-sep^r,\gamma-1) + cout(i,t) \\ F(i-1,j,reverse,t-\underline{p_i}-sep^r,\gamma) + cout(i,t) \end{array} \right. \end{array} \right.$$

$$\mathcal{F}(i,j,reverse,t,\gamma) = \min \left\{ \begin{array}{l} \max \left\{ \begin{array}{l} \mathcal{F}(i,j-1,reverse,t-\widetilde{esp}^r_j(\overline{p_j}),\gamma-1) + cout(j,t) \\ \mathcal{F}(i,j-1,reverse,t-\widetilde{esp}_j(\underline{p_j}),\gamma) + cout(j,t) \end{array} \right. \\ \max \left\{ \begin{array}{l} F(i,j-1,direct,t-\overline{p_j}-sep^d,\gamma-1) + cout(i,t) \\ F(i,j-1,direct,t-\underline{p_j}-sep^d,\gamma) + cout(i,t) \end{array} \right. \end{array} \right. \quad (5)$$

Calcul de l'espacement : on suppose (sans perte de généralité) que $tr^s_{\mathcal{E}} = i$.
L'*espacement* dans le cas déterministe entre deux trains i et $i-1$ se calcule par $esp_i = esp^d + \max(0, p_i - p_{i-1})$. Dans l'approche robuste, les durées de traversée des trains sont incertaines, ainsi

dans l'équation [5], $esp_i^d(\hat{p}_i)$ avec $\hat{p}_i \in \{\overline{p}_i, \underline{p}_i\}$ indique l'écart entre le train i dont la durée est $\hat{p}_i$ et le train $i-1$ dont on ne connait pas encore *a priori* la durée de traversée dans $\{\overline{p}_{i-1}, \underline{p}_{i-1}\}$. Pour garantir un écart minimal de esp^d, il faudrait prendre la valeur minimale de p_{i-1}, c'est-à-dire $\underline{p}_{i-1}$, ainsi $\widetilde{esp}_i^d(\hat{p}_i) = esp^d + \max(0, \hat{p}_i - \underline{p}_{i-1})$.

La solution du problème est donnée par :

$$\text{SOLUTION} = \min \begin{cases} \mathcal{F}(d, r, direct, T, \Gamma) \\ \mathcal{F}(d, r, reverse, T, \Gamma) \end{cases} \tag{6}$$

Complexité : la complexité (temporelle et spatiale) de la programmation dynamique est due à la construction de la table. Dans notre cas, la table a 5 variables. Sa taille est $d \times r \times 2 \times T \times \Gamma$. La complexité temporelle de l'algorithme robuste est en $O(n^3T)$. C'est un algorithme pseudo-polynomial.

Cette formulation conduit notre modélisation à ne plus être optimale. En effet, considérer la valeur minimale pour p_{i-1} est très contraignant pour l'ordonnancement. Cela peut induire un décalage sur i qui peut se répercuter sur le reste de l'ordonnancement. C'est là une des limites de cette modélisation. Il est possible de dépasser cette limite en ajoutant une dimension $late \in \{0, 1\}$ à l'état de la programmation dynamique qui devient $\mathcal{E}(i, i, s, t, \gamma, late)$. $late = 1$ indique que la durée du dernier train passé est à son extrême. Il sera alors plus facile de calculer les espacements. La complexité temporelle de l'algorithme sera toujours en $O(n^3T)$ mais il nécessitera plus de mémoire. Cela limitera considérablement son exécution sur des machines standards.

4.2 Robustesse paramétrée par un budget de retard

Dans cette nouvelle approche, l'incertitude est paramétrée non pas par un nombre de trains au maximum de leur retard, mais par un budget B de minutes de retard à affecter aux durées de traversée. Cette approche est une généralisation du pire cas et de l'approche déterministe.
La modélisation par programmation dynamique dans le cas déterministe s'étend facilement. Un état $\mathcal{E}$ est représenté par le tuple (i, j, s, t, β). i, j, s et t ont la même signification que dans l'approche précédente. À l'instant t, le dernier train à être passé dans le sens s est le train $tr_{\mathcal{E}}^s$. β indique le nombre de minutes supplémentaires affectées aux durées de traversée des trains de $1...i$,$1..j$.
De la même manière, on associe à chaque état $\mathcal{E}(i, j, s, t, \beta)$ une fonction de coût $\mathcal{F}(i, j, s, t, \beta)$ qui représente la somme pondérée des retards. La formulation du problème par programmation dynamique se fait de la manière suivante :

conditions initiales
- aucun train ne circule : $\mathcal{F}(0, 0, s, t, 0) = 0, \forall t, \forall s$
- condition de disponibilité du train $tr_{\mathcal{E}}^s$: $\mathcal{F}(i, j, s, t, \beta) = +\infty,\ \forall i, j, s, \beta, t : t < r_{tr_{\mathcal{E}}^s} + \underline{p}_{tr_{\mathcal{E}}^s}$
- conditions de cohérence

$$\begin{aligned} \mathcal{F}(i, j, s, t, \beta) &= +\infty \quad , \forall i, j, s; \forall t, \beta : t < \beta \\ \mathcal{F}(i, j, s, t, \beta) &= +\infty \quad , \forall i, j, s, \beta : \sum_{k=0..i, 0..j} \overline{p_k} - \underline{p_k} < \beta \end{aligned}$$

Équations de Bellman

$$F(i,j,direct,t,\beta) = \min \begin{cases} \max_{b\in\{0..\min(\beta,\overline{p_i}-\underline{p_i})\}} F(i-1,j,direct,t-esp_i^d(\underline{p_i}+b),\beta-b)+cout(i,t) \\ \max_{b\in\{0..\min(\beta,\overline{p_i}-\underline{p_i})\}} F(i-1,j,reverse,t-sep^d-(\underline{p_i}+b),\beta-b)+cout(i,t) \end{cases}$$

$$F(i,j,reverse,t,\beta) = \min \begin{cases} \max_{b\in\{0..\min(\beta,\overline{p_j}-\underline{p_j})\}} F(i,j-1,reverse,t-esp_j^r(\underline{p_j}+b),\beta-b)+cout(j,t) \\ \max_{b\in\{0..\min(\beta,\overline{p_j}-\underline{p_j})\}} F(i,j-1,direct,t-sep^r-(\underline{p_j}+b),\beta-b)+cout(j,t) \end{cases} \quad (7)$$

La solution du problème est donnée par :

$$\text{SOLUTION} = \min \begin{cases} \mathcal{F}(d,r,direct,T,B) \\ \mathcal{F}(d,r,reverse,T,B) \end{cases} \quad (8)$$

Notre algorithme n'est plus optimal. En effet, considérer la valeur minimale pour p_{i-1} est très contraignant pour l'ordonnancement. Si l'on note par p^*_{i-1} la valeur de p_{i-1} dans la solution optimale, le décalage induit sur i sera de p^*_{i-1} ; celui-ci se répercutera sur le reste de l'ordonnancement.
De la même manière que pour l'approche B&S proposée précédemment, ajouter une dimension b (qui représente le nombre de minutes supplémentaires "consommées" par le dernier train à être passé) à la programmation dynamique résoudra ce problème. Mais la taille du nouveau modèle grandirait considérablement et affecterait les performances de la méthode et rendrait ainsi difficile l'intégration de la méthode dans un outil d'aide à la décision.

Complexité : comme indiqué dans la section précédent 3.2, la complexité (temporelle et spatiale) de la programmation dynamique est due à la construction de la table. Dans notre cas, la table a 5 dimensions. Sa taille est $d \times r \times 2 \times T \times B$. Pour chaque cellule, une comparaison entre $2B$ autres cellules est nécessaire. La complexité temporelle de l'algorithme robuste est en $O(n^2TB^2)$. Cet algorithme est aussi pseudo-polynomial.

4.3 Limites des approches robustes

Les approches robustes paramétrées ont pour objectif de cerner l'aléa, et ce faisant, d'essayer de fournir des solutions où l'impact de l'incertain est amoindri. Néanmoins, cette démarche a des limites :
- les solutions fournies dépendent d'un paramétrage qui est souvent arbitraire. Dans notre cas, la taille des intervalles, les paramètres Γ et B sont laissés à l'appréciation de l'utilisateur.
- les tailles et complexités des problèmes deviennent importantes. Il devient nécessaire d'opérer des simplifications qui peuvent rendre le modèle peu réaliste et/ou sous-optimal.
- enfin, calculer un ordonnancement en début de période ne prend pas en compte l'évolution du système qui peut rendre la solution calculée caduque. Une ré-optimisation dans le temps devient nécessaire.

5 Approches réactives

Dans les approches robustes proposées, l'objectif était de planifier à l'avance l'ordre de passage des trains en spéculant sur leurs durées de traversée. L'inconvénient majeur était que, une fois un ordonnancement robuste établi en fonction d'un paramétrage donné, il se peut que la réalité contrevienne aux hypothèses de départ. Par exemple, la durée de traversée réelle d'un train dépasse la borne supérieure de son intervalle de variation initial, le nombre de trains au pire cas dépasse le

paramètre Γ, la durée de l'incident est plus grande que prévue... Dans ces cas, l'ordonnancement planifié devient caduque. Il faut alors réagir et s'adapter à la nouvelle donne. C'est l'objectif des modèles réactifs.
Les algorithmes réactifs que nous proposons sont inspirés des modèles On-line [5]. Ils ont pour but de résoudre des problèmes dont les données ne sont pas entièrement connues lors de la résolution. C'est le cas d'une grande partie des problèmes réels.
Les modèles de ce genre de problèmes sont basés sur deux hypothèses :

1. l'instance du problème est révélée pas à pas
2. le décideur fait son choix une fois qu'une partie de l'instance est révélée. Ce choix est définitif et irrévocable.

De ces hypothèses, on peut définir différents modèles qui dépendront de la manière dont est révélée l'instance ainsi que des droits du décideur sur la construction de la solution finale. Pour notre problème d'ordonnancement, le modèle réactif correspondant est décrit ainsi : À un instant t donné, deux cas se présentent

1. un train direct ou inverse se présente.
2. deux trains (sens direct et inverse) sont présents.

Le cas trivial où il n'y a pas de train à faire passer n'appelle aucune action.
L'objectif est de déterminer le prochain train à ordonnancer. Une fois cette décision prise, elle est de fait irrévocable.
En posant R l'ensemble des trains à venir, S l'ensemble des trains déjà ordonnancés, on définit un algorithme réactif générique de la façon suivante :
Algorithme générique :
`TANT QUE` $R \neq \emptyset$ `FAIRE`
    $s_t \leftarrow$ chooseNextTrain(S, R)
    $R \leftarrow R \backslash \{s_t\}$
    $S \leftarrow S \cup \{s_t\}$
    increment(t, s_t)
`FIN TQ`
avec

- chooseNextTrain(S, R) : choix du prochain train parmi R, sachant que les trains de S ont été ordonnancés.
- increment(t, s_t) : Choix du prochain instant de décision.

L'instanciation de l'algorithme générique se fait en instanciant les fonctions chooseNextTrain(S, R) et increment(t, s_t).

5.1 Algorithmes purement réactifs

Nous expérimentons deux algorithmes qui définissent les fonctions précédentes.
– **réactif glouton** : chooseNextTrain(S, R) choisit, dans le cas où on a deux trains de sens opposés se présentent, celui qui a le plus grand retard.
– **Réactif SameDirectionFirst** : chooseNextTrain(S, R) choisit le train de R qui est dans le même sens que le dernier train ordonnancé de S.

Complexité : les algorithmes réactifs précédents se basent sur la réaction à un événement qui survient (arrivée d'un train). Le traitement de ce train se fait en temps constant $O(1)$. La complexité de l'algorithme est alors en $O(n)$.
L'avantage de cette modélisation réactive est d'avoir des informations mises à jour et ainsi de coller

à la réalité. Mais cela a un coût. En effet, récupérer des informations sur les trains en circulation nécessite des équipements et des investissements coûteux. Coûts qui ne sont pas pris en compte dans la modélisation. Autre inconvénient majeur, la myopie de l'approche : elle recherche un optimum local et a une vue trés limitée sur le long terme.
On pourrait alors penser à corriger cette myopie en "robustifiant" la méthode, c'est-à-dire en choisissant le prochain train de manière robuste. C'est ce qui va être abordé dans la partie suivante.

5.2 Modèles réactifs robustifiés

Les versions On-line de problèmes d'optimisation combinatoire stochastique ont été étudiées de façon approfondie dans l'ouvrage de Pascal Van Hentenryck et Russel Bent [4]. Dans leur analyse, ils supposent que les distributions des futures requêtes, ou des approximations de celles-ci, sont disponibles et facilement échantillonnables (*sampling*). Ils proposent différents algorithmes pour les versions Réactives de problèmes tels que le routage de véhicules, les réservations stochastiques, l'ordonnancement de paquets IP... Ils présentent des garanties de performance en fonction du nombre de scénarios (taille de l'échantillon) et de la qualité de la relaxation de certains problèmes difficiles.
Nous allons nous intéresser à "l'Algorithme de prise de décision immédiate" énoncé dans l'ouvrage, et nous allons l'adapter à notre problématique. Dans cet algorithme, le temps entre deux décisions est mis à profit pour pré-calculer le maximum de scénarios et ainsi obtenir une solution robuste. Pour notre étude, nous n'avons pas de lois de distribution pour les paramètres incertains. Mais nous pouvons adapter cet algorithme de la façon suivante : Á l'instant t, nous aurons à décider entre deux alternatives. Faire passer le train i dans le sens *direct* ou bien le train j sens $reverse$. Pour savoir quel train faire passer, nous appliquons l'approche robuste sur deux sous-problèmes : celui où le train i est ordonnancé en premier, et celui où c'est j qui passe en premier. On choisira la solution de moindre coût.
Dans ce paragraphe, nous adaptons "l'Algorithme de prise de décision immédiate" décrit plus haut à notre problématique. Nous utilisons les variables et le fonctions suivantes :

Définition des variables

- unscheduledTrains : liste des trains non encore traités
- currentTr : train déjà planifié -en cours de traversée-
- nextTr : prochain train à ordonnancer
- nextDecisionTime : date limite pour décider quel train sera ordonnancé à l'itération suivante

Définition des fonctions

- chooseNextTrain(t,currentTr,unscheduledTrains) :
  décide à l'instant t quel train parmi unscheduledTrains sera ordonnancé après currentTr
- computeNextDecisionTime(currentTr,nextTr) :
  calcule la date de départ du prochain train ordonnancé (nextTr) en fonction de celui en cours de traversée (currentTr)
- updateData(unscheduledTrains) : met à jour les informations (date d'arrivée, durée de traversée) des trains à traiter (unscheduledTrains)
- processScheduling(currentTr) : simule le passage du train currentTr et récupère sa durée de traversée

Algorithme de prise de décision immédiate

```
1   t ← startTime ;
2   currentTr← NULL ;
3   unscheduledTrains ← AllTrains ;
4   WHILE (unscheduledTrains ≠ ∅) DO
5   |   nextTr           ←  chooseNextTrain(t,currentTr,unscheduledTrains) ;
6   |   t                ←  computeNextDecisionTime(currentTr,nextTr) ;
7   |   unscheduledTrains ←  unscheduledTrains \ {currentTr} ;
8   |   currentTr        ←  nextTr ;
9   |  processScheduling(currentTr) ;
10  |  updateData(unscheduledTrains) ;
11  END-WHILE
```

Dans les lignes 1, 2 et 3, les variables sont initialisées. Le choix du prochain train à ordonnancer se fait en considérant le train $currentTr$ comme déjà traité, et en minimisant le retard cumulé de l'ordonnancement du reste des trains (ligne 5). Cette décision est prise au plus tard à la date de départ du train en cours de traitement (ligne 6). S'en suit alors une temporisation correspondant à la traversée du train déjà planifié ($currentTr$). Période qu'on met à profit pour mettre à jour les informations sur les trains à venir (annulations, limitations de vitesses...) (lignes 9 à 10). Il est important de noter que le choix du prochain train à traiter $nextTr$ est définitivement fixé à partir de la date de départ du train courant.
Dans ce papier, la fonction computeNextDecisionTime($currentTr, nextTr$) consiste à déterminer la date de départ du train courant $currentTr$. D'autres fonctions peuvent être envisagées. Par exemple , on pourrait choisir un pas de temps constant pour la date de décision, ou bien choisir le(s) prochain(s) trains à ordonnancer tous les x trains...
L'instanciation de cet algorithme se fait en instanciant la fonction
chooseNextTrain($t, currentTr, unscheduledTrains$) et

Complexité

L'exécution des fonctions computeNextDecisionTime($currentTr, nextTr$) se fait en temps constant $O(1)$. Celle de la fonction updateData($unscheduledTrains$) est en $O(|unscheduledTrains|)$. La complexité de l'algorithme dépend alors de celle de la fonction
chooseNextTrain($t, currentTr, unscheduledTrains$)

5.3 Algorithmes réactifs robustifiés

L'objectif est de bien choisir le prochain train à ordonnancer. On utilisera pour cela les différentes approches présentées en section 3 et 4.

5.3.1 Réactif/déterministe

Le prochain train à ordonnancer est calculé par la programmation dynamique déterministe. Rappelons que $currentTr$ est le train choisi à l'étape précédente pour être ordonnancé à l'instant t. Pour

déterminer le prochain train de façon optimale et en se basant sur des paramètres mis à jour, nous appelons la procédure d'optimisation déterministe sur les trains $\{currentTr\} \cup unscheduledTrains$, et ce en pondérant $currentTr$ par un poids suffisamment important pour qu'il soit choisi en premier. Le résultat de CHOOSENEXTTRAIN est alors le suivant de $currentTr$. La complexité de l'algorithme déterministe est $O(n^2T)$. Celle de l'algorithme réactif/déterministe est alors en $O(n^3T)$.
Même si les informations sur les trains sont mises à jour au cours de l'exécution, il reste que l'incertitude, même réduite, demeure. Les approches robustes étudiées dans le chapitre précédent peuvent, alors, servir à déterminer le prochain train.

5.3.2 Réactifs/B&S

De la même façon que pour l'algorithme réactif/déterministe, on pondérera $currentTr$ pour qu'il soit choisi en premier. Le choix du paramétrage est laissé au décideur. Nous avons considéré deux options suivantes :

- soit il détermine en début d'ordonnancement un nombre Γ de trains aux durées max qu'il réduit au fur et à mesure que certains trains atteignent leurs max lors de l'exécution.
- soit il prend comme paramètre un pourcentage des trains non encore ordonnancés.

L'algorithme de l'approche B&S est en $O(n^3T)$. La complexité de l'algorithme réactif/B&S est alors de $O(n^4T)$.

5.3.3 Réactif/Budget de temps

De façon analogue, et en prenant les mêmes précautions pour $currentTr$, on peut paramétrer le choix du prochain train à ordonnancer par un budget de temps B. Là encore, un choix s'offre au décideur quand au paramètre de la méthode :

- fixer un budget en début d'ordonnancement et lui défalquer les minutes supplémentaires prises par les trains lors de leur passage ;
- fixer un budget en fonction du nombre de trains qui restent à ordonnancer ;
- tirer aléatoirement (en suivant une loi normale par exemple) un budget.

L'algorithme de l'approche par budget de temps étant en $O(n^2TB^2)$, la complexité de Réactif/budget est donc en $O(n^3TB^2)$

6 Expérimentations et résultats numériques

Dans cette partie, nous analysons les résultats d'expérimentations des différentes approches et algorithmes.

Instance

- 25 trains dans le sens *direct* et 25 dans le sens *reverse*
- temps espacement minimal de 3 minutes, temps de séparation de 5 minutes (pour les deux sens).
- le sens *direct* est accidenté
- les intervalles des durées de traversée du sens non accidenté *reverse* sont $[p_k, 2p_k]$
- les intervalles des durées de traversée du sens accidenté *direct* sont $[10, 15]$

6.1 Comparaison des approches réactives robustifiées

Nous reportons dans ce qui suit les résultats obtenus pour chacune des méthodes réactives robustifiées, en comparant le retard cumulé de celles-ci à la moyenne des retards cumulés de l'algorithme purement réactif *sameDirectionFirst* et ce sur 100 scénarios générés aléatoirement. Dans les figures qui suivent, l'algorithme *sameDirectionFirst* est dénommé "Réactif Heuristique".

Réactif/Budget

La méthode est paramétrée par un budget B de minutes supplémentaires. Deux choix sont possibles :

– on déduit du budget initial les rallongements des durées de traversée constatés lors des passages des trains (Fig 2)

– le budget est fonction du nombre de trains à ordonnancer (Fig 1)

$budget = budgetInitial * |unschedulledTrains|/|allTrains|$.

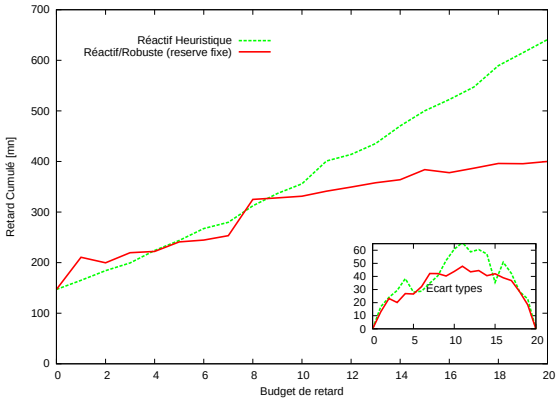


FIGURE 1 – Comparaison entre les algorithmes reactif/Budget et *sameDirectFirst* avec un budget fixe

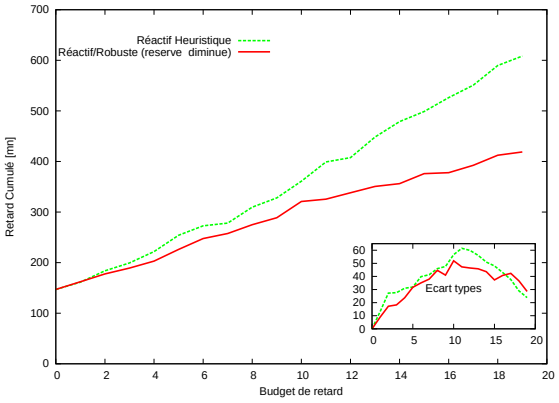


FIGURE 2 – Comparaison entre les algorithmes reactif/Budget et *sameDirectFirst* avec un budget décroissant

La figure (Fig 2) montre le gain en terme de retard de la méthode réactif/Budget par rapport à l'algorithme *sameDirectionFirst*, gain d'autant plus important que le budget des minutes supplémentaires est grand (*i.e* dans le cas de fortes perturbations).

Réactif/B&S

Ici le choix du prochain train à ordonnancer se fait par l'appel à l'algorithme robuste B&S. Un choix a été fait quant au paramétrage (choix du Γ). A l'exécution, les durées de passage des trains n'ont pas nécessairement leur valeurs maximales, il n'est donc pas pertinent de faire décroitre Γ au passage de trains ayant atteint leur max. Γ n'est pas une donnée métier mais un paramètre du modèle. Nous avons choisi de définir Γ comme un pourcentage τ des trains non encore ordonnancés :

$$\Gamma = \tau(\%) \times |unscheduledTrains|$$

Ce taux est laissé à l'appréciation de l'utilisateur.

La figure (Fig 3) montre bien le gain en terme de retard de la méthode réactif/B&S, gain d'autant

plus important que le nombre de trains dont la durée de passage atteint son maximum est grand.

Réactif/déterministe

Cette méthode ignore l'aléa et s'appuie sur l'algorithme déterministe. La figure (Fig 4) compare les moyennes des retards de l'algorithme réactif/déterministe avec l'heuristique réactive *sameDirectFirst*, et ce sur 100 scénarios. Nous pouvons remarquer que le gain de la méthode réactif/déterministe est faible. Cela parce qu'il n'intègre pas l'aléa.

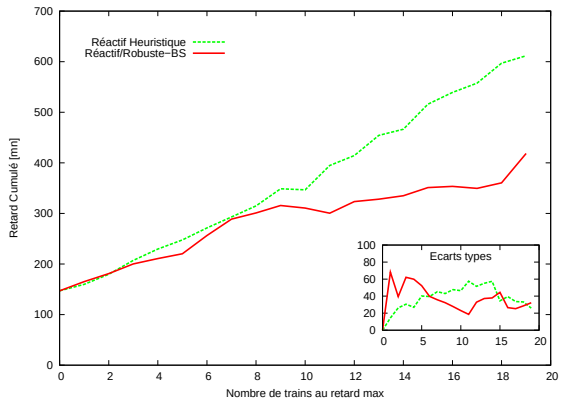


FIGURE 3 – Comparaison entre les algorithmes reactif/B&S et *sameDirectFirst* avec $\tau = 0.33$

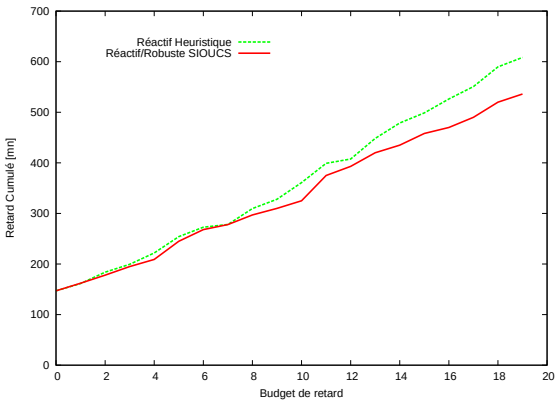


FIGURE 4 – Comparaison entre les algorithme reactif/déterministe et *sameDirectFirst*

Enfin, nous comparons les approches réactif/Budget avec budget fixe et budget décroissant et l'approche réactif/déterministe (Fig 5). Nous pouvons remarquer qu'il n'y a pas de grandes différences entre les deux variantes de réactif/budget. Mais, on note un gain non négligeable par rapport à l'algorithme Réactif/déterministe.

6.2 Temps de calculs

Les différents algorithmes étudiés ainsi que les expérimentations ont été implémentés en Java sur une machine dotée d'un processeur AMD Athlon™ d'une fréquence de 2,19GHz et d'une mémoire vive de 2,00 Go. Les tests ont été réalisés sur l'instance décrite plus haut avec des données réelles. Les résultats (Fig 6 et Fig 7) montrent que le temps d'exécution de la fonction CHOOSENEXTTRAIN reste relativement bas par rapport l'échelle du temps de la décision qui est de l'ordre de la minute. Cela est valable pour les deux approches robustes (Budget et B&S).

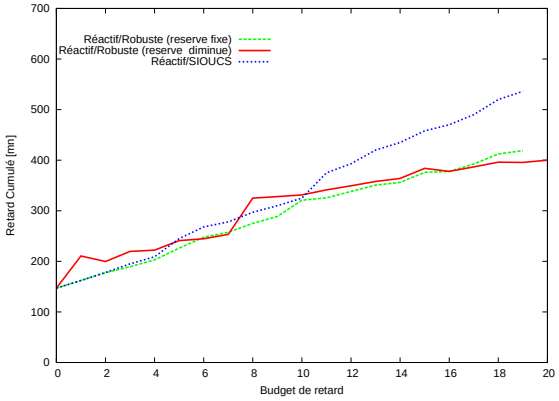


FIGURE 5 – Comparaison entre les variantes réactif/Budget (budget fixe ou décroissant) et réactif/déterministe

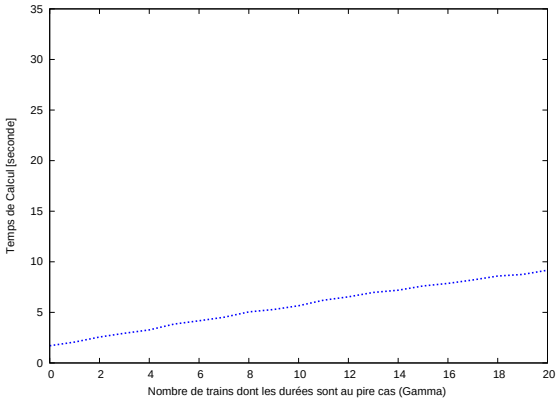


FIGURE 6 – Temps de calcul de la fonction CHOOSENEXTTRAIN/Budget en fonction de Γ

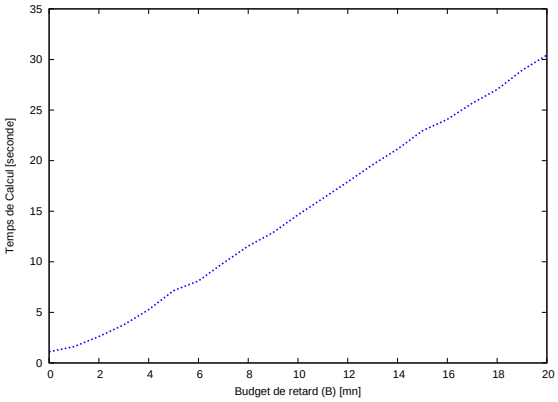


FIGURE 7 – Temps de calcul de la fonction CHOOSENEXTTRAIN/B&S en fonction de B

7 Conclusion

La gestion des aléas dans l'ordonnancement de trains sur une voie unique lors d'un incident n'est pas une tâche aisée. Après avoir défini un modèle pour l'incertitude sous forme d'intervalles, deux approches robustes paramétrées ont été mises en œuvre. La première se fonde sur l'approche Bertsimas & Sim : le modèle est paramétré par un nombre maximum de trains ayant leur durée de traversée au maximum de sa valeur. La seconde approche consiste à paramétrer le modèle par un budget de retard. Ces deux méthodes ont été implémentées par programmation dynamique, en étendant le modèle développé pour le cas déterministe.
Une toute autre approche, plus réactive, a été développée. Le choix du prochain train à ordonnancer est dynamique. Nous y avons associé la robustesse telle que définie précédemment pour obtenir une approche réactive robustifiée. Les résultats expérimentaux de cette dernière montrent un gain important en terme de retard cumulé par rapport à une planification purement robuste ou une démarche réactive d'optimisation locale. D'une manière générale, il n'est pas toujours aisé d'appliquer des modèles théoriques à des problématiques industrielles complexes. Le concours de plusieurs méthodes et

algorithmes est souvent nécessaire.

Remerciements

Le travail du troisième auteur a bénéficié en partie d'une aide de l'Agence Nationale de la Recherche portant la référence ANR-08-BLAN-0331-01 (projet ROBOCOOP).

Références

[1] A. Allahverdi, C.T. Ng, T.C.E. Cheng, and M. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187 :985–1032, 2008.

[2] D. Bertsimas and M. Sim M. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1) :49–71, September 2003.

[3] R.L. Graham. Optimization and approximation in deterministic sequencing and scheduling : A survey. *Annals of Discrete Mathematics*, 1979.

[4] P. Van Hentenryck and B. Russell. *Online Stochastic Combinatorial Optimization*. The MIT Press, 2006.

[5] D.S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[6] D. Huisman. Operations research in passenger railway transportation. *Statistica Neerlandica*, 59(4) :467–497, 2005.

[7] P. Kouvelis and G. Yu. Robust discrete optimization and its applications, 1997.

[8] F. Sourd and Ch. Weber. Ordonnancement des trains sur une voie unique. *Livre des Résumés, 10ème Congrès de la ROADEF*, page 294, 2009.