# Recherche d'un chemin de meilleur compromis dans un graphe multicritère

L. Galand

Laboratoire d'Informatique de Paris 6 (LIP6)
8 rue du Capitaine Scott
75015 Paris
lucie.galand@lip6.fr

**Résumé** Dans cet article nous nous intéressons à la version multicritère du problème de plus court chemin dans un graphe. Après avoir rappelé la définition de meilleur compromis à l'aide du critère pondéré de Tchebycheff [16], nous proposons un algorithme pour résoudre le problème de recherche du chemin de meilleur compromis (problème NP-difficile). Notre approche repose sur une énumération efficace des plus courts chemins pour la somme pondérée des coûts en utilisant l'algorithme d'Eppstein [4]. Nous établissons un résultat de majoration permettant de stopper cette énumération lorsque l'optimalité de la solution trouvée est garantie. Des expérimentations numériques ont été menées et sont présentées dans cet article, montrant l'efficacité de notre approche.

**Mots-Clefs.** Optimisation multiobjectif ; Aide à la décision ; Optimisation combinatoire ; Graphes.

## 1 Introduction

De nombreux problèmes concrets peuvent se modéliser comme un problème de recherche de chemin optimal dans un graphe, dans le domaine des télécommunications ou des transports par exemple. Ces problèmes ont traditionnellement été étudiés en ne considérant qu'une seule fonction objectif à optimiser. Or il existe diverses situations dans lesquelles la prise en compte de plusieurs objectifs est nécessaire. Lors de la planification de trajets par exemple, on peut considérer le temps de trajet, la distance à parcourir, le coût du trajet et la praticabilité de la route. On est donc amené à optimiser plusieurs objectifs, qui n'ont pas forcément de liens entre eux et qui peuvent même s'avérer conflictuels. A cet effet, la théorie de la décision a fourni de nombreux modèles sophistiqués permettant de guider le choix d'une solution parmi un ensemble réduit en prenant en compte plusieurs critères. Néanmoins, l'adoption de ces modèles peut complexifier singulièrement la recherche d'une solution préférée lorsque l'espace des solutions est de nature combinatoire. Plus précisément ici, nous modélisons les préférences entre les chemins à l'aide de la norme pondérée de Tchebycheff [16]. Les préférences induites par ce modèle invalidant les approches usuelles de résolution pour le problème de plus court chemin, il est alors nécessaire de mettre en

oeuvre des stratégies algorithmiques nouvelles pour déterminer la solution préférée. C'est précisément ce que nous proposons dans cet article. Dans la première partie de l'article, nous présentons formellement le problème et les difficultés qu'il présente. Dans la deuxième partie, nous introduisons notre algorithme et nous décrivons l'algorithme des $k$ plus courts chemins d'Eppstein [4] que nous utilisons. Enfin, nous présentons dans la troisième partie nos expérimentations numériques et les résultats que nous avons obtenus.

## 2 Problème de graphe multicritère

### 2.1 Présentation du problème

En optimisation multicritère, il est souvent difficile de comparer les solutions d'un problème entre elles. En effet, les différents critères pris en compte peuvent s'avérer conflictuels et, de ce fait, une solution optimisant chacun des critères simultanément n'existe pas en général. Dans ce contexte, la notion de *non-dominance* est couramment utilisée, l'ensemble des solutions non dominées étant l'ensemble des solutions telles qu'il n'existe pas d'autre solution réalisant une meilleure performance sur un critère sans réaliser une plus mauvaise performance sur au moins un autre critère. De nombreux articles portent sur la recherche de l'ensemble des solutions non-dominées d'un problème de chemin multicritère (e.g. [7,2,6,10,15]). Cependant, la recherche d'un tel ensemble s'avère parfois trop délicate. En effet, Hansen [7] montre que le nombre de solutions non-dominées d'un problème multicritère de chemins croît exponentiellement avec le nombre de sommets du graphe sur certaines instances, ce qui rend irréalisable le calcul de l'ensemble de ces solutions pour ces instances. De plus, même lorsque la taille de l'ensemble des solutions non-dominées est raisonnable, une grande partie de ces solutions sont trop déséquilibrées pour être intéressantes en pratique pour le décideur.

Face à ces différentes difficultés, des méthodes interactives ont été proposées basées sur une interaction avec le décideur (e.g. [5,3]). L'interaction repose sur l'alternance d'une phase de calcul (génération d'une solution optimale à l'étape courante) et d'une phase de dialogue (récolte de l'information préférentielle et mise-à-jour du modèle). La rapidité de la phase de calcul étant cruciale pour l'efficacité d'une telle procédure, nous nous concentrons dans cet article sur la recherche directe de la solution la plus satisfaisante, à l'étape courante, pour le décideur, c'est-à-dire celle réalisant le meilleur compromis entre les critères (la notion de meilleur compromis étant révisable à chaque étape de l'interaction). L'algorithme que nous présentons détermine donc directement une telle solution sans chercher à engendrer l'ensemble des solutions efficaces.

### 2.2 Notations

Soit $V = \{1, \ldots, n\}$ un ensemble fini de noeuds et $E \subseteq V \times V$ un ensemble fini d'arcs, on se donne un graphe $G = (V, E)$ comportant une source $s \in V$

et un puits $t \in V$. A chaque arc $u \in E$, est associé un vecteur de fonctions coûts à valeurs dans $\mathbb{N}$ de la forme $c(u) = (c_1(u), \ldots, c_q(u))$ où $q$ est le nombre de critères du problème. Un chemin $P$ entre deux noeuds $s$ et $t$ de $V$ est une séquence de noeuds débutant par $s$ et se terminant par $t$, telle que deux noeuds $v_h$ et $v_{h+1}$ qui se suivent dans la séquence sont reliés par un arc de $E$. Tout chemin $P$ de $s$ à $t$ est évalué par le vecteur $c(P) = (c_1(P), \ldots, c_q(P))$ en posant $c_i(P) = \sum_{u \in P} c_i(u)$ pour $i = 1, \ldots, q$. L'image d'un chemin $P$ dans l'espace des critères est un vecteur $p = (p_1, \ldots, p_q) \in \mathbb{N}^q$ tel que $c(P) = p$. L'ensemble des chemins de $s$ à $t$ réalisables dans $G$ (appelés aussi solutions) est noté $\mathscr{P}$. On appelle $X$ l'ensemble des vecteurs représentants les coûts des chemins de $\mathscr{P}$ dans l'espace des critères, et $ND$ l'ensemble des vecteurs non-dominés de $X$ (i.e. $ND = \{p \in X, \nexists p' \in X, p' \succ p\}$ où $p' \succ p$ ssi $(\forall i = 1, \ldots, q\ p'_i \leq p_i)$ l'une des inégalités étant stricte).

On s'intéresse ici à la recherche d'un chemin de meilleur compromis dans un graphe $G$. Nous allons voir dans la section suivante comment définir une fonction permettant d'évaluer l'aptitude d'une solution à réaliser un bon compromis entre les critères.

### 2.3 Solution de meilleur compromis

Il peut paraître naturel d'envisager de définir la fonction coût du problème comme la somme pondérée $s^1_\omega$ des performances des critères. On a ainsi :

$$\forall P \in \mathscr{P},\ s^1_\omega(p) = \sum_{i=1}^{q} \omega_i p_i$$

où $\omega_1, \ldots, \omega_q$ représentent les poids attribués aux $q$ critères. En remarquant que $\sum_{i=1}^q \omega_i p_i = \sum_{i=1}^q \omega_i \sum_{u \in P} c_i(u) = \sum_{i=1}^q \sum_{u \in P} \omega_i c_i(u) = \sum_{u \in P} \sum_{i=1}^q \omega_i c_i(u)$ et en posant $\forall u \in E$, $c^1_\omega(u) = \sum_{i=1}^q \omega_i c_i(u)$, la formule se réécrit :

$$s^1_\omega(p) = \sum_{u \in P} c^1_\omega(u)$$

Ainsi pour trouver la solution minimisant $s^1_\omega$ dans $G$, il suffit de chercher le plus court chemin dans $G$ scalarisé (i.e. valué par la fonction scalaire $c^1_\omega$), ce qui nous ramène à un problème monocritère que l'on sait résoudre en temps polynomial lorsque le graphe $G$ est sans circuit absorbant. Cependant, comme nous le rappelons dans l'exemple suivant, une telle approche n'est pas satisfaisante car elle ne permet pas d'atteindre toutes les solutions non-dominées, et donc parfois, celle de meilleur compromis.

**Exemple 1** *Considérons un ensemble $X$ composé des vecteurs $p$, $p'$ et $p''$ associés aux chemins non-dominés $P$, $P'$ et $P''$ tels que $p = (10, 4)$, $p' = (9, 6)$ et $p'' = (3, 11)$. Nous allons déterminer pour quels vecteurs $\omega = (\omega_1, \omega_2)$, $p$, $p'$ ou $p''$ minimise $s^1_\omega$. On a : $s^1_\omega(p) = 10\omega_1 + 4\omega_2$, $s^1_\omega(p') = 9\omega_1 + 6\omega_2$ et $s^1_\omega(p'') = 3\omega_1 + 11\omega_2$. Ainsi : $s^1_\omega(p) < s^1_\omega(p'') \Leftrightarrow \omega_1 < \omega_2$, $s^1_\omega(p) < s^1_\omega(p') \Leftrightarrow \omega_1 < 2\omega_2$ et*

$s^1_\omega(p'') < s^1_\omega(p') \Leftrightarrow \omega_2 < \frac{6}{5}\omega_1$. *Donc, dès que* $\omega_1 < \omega_2$, *le vecteur minimisant* $s^1_\omega$ *est* $p$, *dès que* $\omega_2 < \omega_1$, *le vecteur minimisant* $s^1_\omega$ *est* $p''$, *et* $p$ *et* $p''$ *minimisent tous deux* $s^1_\omega$ *lorsque* $\omega_1 = \omega_2$. *Le vecteur* $p'$ *n'est donc jamais trouvé, quel que soit le vecteur* $\omega \in \mathbb{R}^2$. *Pourtant, c'est celui qui représente le meilleur compromis sur les deux critères.*

Nous considérons donc un autre type de fonction d'évaluation $s^\infty_\omega$, définissant le chemin de meilleur compromis comme celui dont le vecteur de coûts est de distance minimum à un point de référence $r \in \mathbb{N}^q$ selon la norme pondérée de Tchebycheff [16] :

$$\forall P \in \mathscr{P},\ s^\infty_\omega(p) = \max_{i \in \{1,\cdots,q\}} \omega_i |p_i - r_i|$$

La norme pondérée de Tchebycheff est très souvent utilisée au sein des procédures interactives [12,14], car elle permet d'explorer l'ensemble des solutions non-dominées, supportées (i.e. atteignables par une somme pondérée) ou non, en faisant varier le vecteur de pondération. Le problème que nous traitons est donc la recherche d'un chemin de meilleur compromis selon cette norme :

$$(P_\omega) : \min_{P \in \mathscr{P}} \max_{i \in \{1,\cdots,q\}} \omega_i |p_i - r_i|$$

Ce modèle induit implicitement un préordre total sur l'ensemble des chemins de $\mathscr{P}$ puisqu'un chemin $P$ est préféré (i.e. jugé comme réalisant un meilleur compromis) à un chemin $P'$ ssi $s^\infty_\omega(p) \leq s^\infty_\omega(p')$

Dans la suite de l'article, nous utilisons comme point de référence $r$ le point idéal $id$ défini par : $\forall i = 1, \ldots, q,\ id_i = \min_{x \in ND} x_i$. Ce point constitue la référence dont on voudrait se rappeler car il représente une solution fictive qui optimise tous les critères simultanément. Notons que, pour tout chemin $P$ on a $p_i \geq id_i$ ce qui permet d'écrire $s^\infty_\omega(p) = \max_{i \in \{1,\cdots,q\}} \omega_i \{p_i - id_i\}$. Classiquement, les vecteurs de pondération sont définis par :

$$\omega_i = \frac{\alpha_i}{|nad_i - id_i|} \text{ avec } \forall i = 1, \ldots, q,\ nad_i = \max_{x \in ND} x_i$$

où $\alpha_i$ représente le poids du critère $i$ et $\frac{1}{|nad_i - id_i|}$ un facteur de normalisation. Ainsi, $s^\infty_\omega(p)$ représente ici la distance normalisée de la valeur du chemin $P$ au point idéal selon la norme pondérée de Tchebycheff. La Figure 1 illustre cette définition dans un espace bicritère. Les lignes en pointillés représentent les lignes de niveau et le point entouré représente la solution optimale $P^*$, c'est-à-dire celle dont la distance selon la norme pondérée de Tchebycheff au point idéal est minimale ($s^\infty_\omega(p^*) = \min_{p \in \mathscr{P}} s^\infty_\omega(p)$).

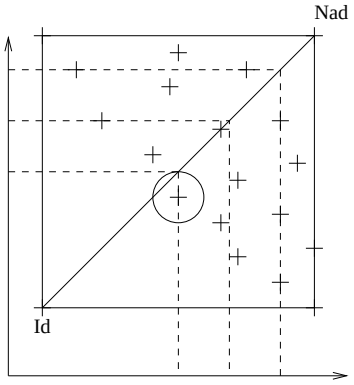


**Fig. 1.** Solution de meilleur compromis dans un espace bicritère

### 2.4 Complexité de la recherche d'une solution de meilleur compromis

Yu et Yang dans [17] montrent que le problème $(RDSP)$[1] qui peut s'écrire avec nos notations :

$$(RDSP)\ \min_{P\in\mathscr{P}} \max_{i\in\{1,\cdots,q\}} \{\sum_{u\in P} c_i(u) - r_i\}$$

est un problème NP-Difficile. Or $(RDSP)$ est un cas particulier de notre problème :

$$(P_\omega)\ \min_{P\in\mathscr{P}} \max_{i\in\{1,\cdots,q\}} \omega_i\{\sum_{u\in P} c_i(u) - r_i\}$$

En effet, en choisissant $\omega_i = \omega_j\ \forall i,j \in \{1,\ldots,q\}$ on se ramène au problème $(RDSP)$. $(P_\omega)$ est donc un problème NP-Difficile.

### 2.5 Violation du principe d'optimalité

La difficulté principale de la manipulation de la fonction $s^1_\omega$ pour définir la notion de meilleur compromis est qu'une approche par programmation dynamique ne permet pas forcément d'obtenir la solution minimisant $s^\infty_\omega$. En effet, le principe de Bellman [1] (tout sous-chemin d'un chemin optimal est optimal) n'est pas respecté, comme le prouve l'exemple suivant.

**Exemple 2** *On considère ici le graphe de la Figure 2, représentant une instance proposée par Hansen [7]. Dans un tel graphe, le nombre de sommets est* $n = 2k+1$ *et le nombre de chemins est* $2^k$*. Nous considérons dans cet exemple le graphe de Hansen pour 7 sommets (*$n = 7$ *dans la figure 2). Un chemin de meilleur compromis sur ce graphe est le chemin* $P^1 =<1, 3, 5, 6, 7>$ *de valeur* $(4,3)$*. Or, le sous-chemin* $<1, 3, 5>$ *de* $P^1$ *qui a pour valeur* $p = (0,3)$ *représente un moins bon compromis que le chemin* $<1, 2, 3, 5>$ *qui a pour valeur* $p' = (1,2)$*,*

[1] Robust Deviation Shortest Path

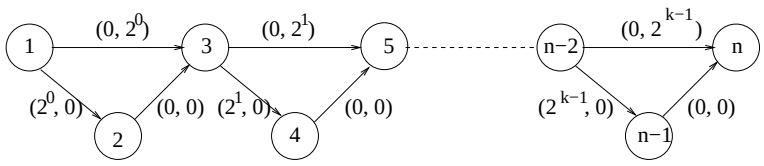


**Fig. 2.** Instance de Hansen[7]

*puisqu'en utilisant les vecteurs id, nad et $\omega$ tels qu'ils ont été définis dans le section 2.3 et en posant $\alpha = (1,1)$, on a $id = (0,0)$, $nad = (7,7)$ et $\omega = (\frac{1}{7}, \frac{1}{7})$, et donc $s^\infty_\omega(p) = \frac{3}{7} > \frac{2}{7} = s^\infty_\omega(p')$.*

On voit donc à travers cet exemple qu'un sous-chemin d'un chemin optimal selon $s^\infty_\omega$ n'est pas forcément optimal. La programmation dynamique semble donc ici peu appropriée pour la recherche directe du chemin de meilleur compromis. Dans la section suivante, nous proposons un autre type d'approche permettant de résoudre le problème $(P_\omega)$.

## 3 Algorithme proposé

### 3.1 Principe

Nous avons vu précédemment que, d'une part chercher à minimiser la somme pondérée des coûts dans le but de trouver un chemin de meilleur compromis n'est pas satisfaisant et d'autre part minimiser la norme pondérée de Tchebycheff est difficile, car le principe d'optimalité n'est pas vérifié. Néanmoins plutôt que minimiser $s^1_\omega$, on peut énumérer les chemins dans l'ordre croissant de leur évaluation $s^1_\omega$, ce qui permet d'atteindre tous les chemins réalisables (et donc tous les non-dominés) en exploitant un graphe scalaire (i.e. un graphe dont les arcs sont valués par un scalaire de $\mathbb{R}$). Dans cette optique, nous établissons une propriété permettant de stopper cette énumération à partir du moment où la garantie d'avoir engendré le chemin optimal de $(P_\omega)$ est obtenue :

**Propriété 1**

$$\forall x \in X,\ \forall y \in X,\ s^1_\omega(y) > s^1_\omega(\bar{x}) \Longrightarrow s^\infty_\omega(x) < s^\infty_\omega(y)$$

*où pour tout $x \in X$, $\bar{x}$ désigne le vecteur à valeurs dans $\mathbb{R}^+$ tel que :*

$$\forall i = 1, \ldots, q, \quad \bar{x_i} = id_i + \frac{1}{\omega_i} s^\infty_\omega(x) \tag{1}$$

**Preuve.**

Supposons que l'on ait $y \in X$ et $x \in X$, tels que $s^1_\omega(y) > s^1_\omega(\bar{x})$

$$\text{i.e. } \sum_{i=1}^{q} \omega_i y_i > \sum_{i=1}^{q} \omega_i \bar{x_i} \text{ avec } \bar{x_i} = id_i + \frac{1}{\omega_i} s^\infty_\omega(x)$$

$$\text{On a donc } \sum_{i=1}^{q} \omega_i(y_i - id_i) > q s_\omega^\infty(x). \tag{2}$$

$$\text{Or } s_\omega^\infty(y) = \max_{i=1,\ldots,q} \omega_i\{y_i - id_i\}, \text{ donc } \sum_{i=1}^{q} s_\omega^\infty(y) \geq \sum_{i=1}^{q} \omega_i(y_i - id_i)$$

$$\text{ainsi, d'après (3), } q s_\omega^\infty(y) = \sum_{i=1}^{q} s_\omega^\infty(y) > q s_\omega^\infty(x)$$

$$\text{c'est-à-dire } s_\omega^\infty(y) > s_\omega^\infty(x)$$

□

Cette propriété nous assure que pour tous chemins $P, P' \in \mathscr{P}$ tels que $s_\omega^1(p') > s_\omega^1(\bar{p})$ alors $P'$ représente un moins bon compromis que $P$ au sens de $s_\omega^\infty$. Donc, une fois un tel chemin $P'$ engendré lors de l'énumération croissante (selon $s_\omega^1$) des chemins, on sait qu'il n'existe pas, parmi les chemins suivants, de chemin représentant un meilleur compromis que $P$ au sens de $s_\omega^\infty$.

L'algorithme que nous proposons consiste donc à énumérer les chemins réalisables du graphe $G$ dans l'ordre croissant selon $s_\omega^1$. Cette énumération s'arrête lorsqu'un chemin $P'$ de valuation $p'$ tel que $s_\omega^1(p') > s_\omega^1(\bar{p})$ est engendré (où $\bar{p}$ est la valuation définie par (1) à partir de la valuation $p$ du meilleur chemin courant $P$). En effet, lorsqu'un tel chemin $P'$ est engendré, $P$ est forcément le meilleur chemin selon $s_\omega^\infty$ sur l'ensemble des chemins de $\mathscr{P}$.

Dans [11], une approche similaire (énumération ordonnée des chemins selon la somme des coûts jusqu'à une certaine condition d'arrêt) a été proposée pour optimiser une autre fonction d'évaluation non linéaire, la norme euclidienne. Cependant, notre approche est plus appropriée à la recherche du chemin de meilleur compromis puisque la norme euclidienne, contrairement à la norme pondérée de Tchebycheff, ne permet pas d'atteindre toutes les solutions non-dominées [16].

L'algorithme 1 ci-dessous résume l'approche que nous proposons. Afin de déterminer notre $k$-ième chemin, nous utilisons l'algorithme d'Eppstein [4] car il permet de trouver les $k$ meilleurs chemins en $O(m + n \log n + k)$ dans un graphe scalaire où $m$ est le nombre d'arcs et $n$ le nombre de sommets. L'algorithme n'étant pas énoncé précisément dans [4], nous en présentons un énoncé formel dans la partie suivante. Les différentes structures et notations utilisées dans [4] sont aussi présentées. Néanmoins la justification de l'algorithme et l'utilisation précise des structures de données, détaillées dans [4], ne sont pas rappelées ici.

### 3.2 Algorithme des k plus courts chemins d'Eppstein

Le principe de cet algorithme est d'énumérer toutes les déviations du plus court chemin allant d'un noeud $v \in V$ au noeud $t$ pour tous les noeuds du graphe et de les stocker dans un tas (arbre binaire dans lequel chaque noeud a une évaluation inférieure ou égale à tous ses descendants) à l'aide desquels les chemins alternatifs au plus court chemin (déterminé au préalable) sont construits dans l'ordre croissant de leur coût.

**Algorithme 1** : Détermination du chemin de meilleur compromis

Déterminer le point idéal $id$ et le point nadir $nad$
$P^1 \longleftarrow arg\min_{P \in \mathscr{P}} s^1_\omega(p)$
$s^* \longleftarrow s^\infty_\omega(p^1)$
$b \longleftarrow s^1_\omega(\vec{p}^1)$
$P^* \longleftarrow P^1$
$k \longleftarrow 2$
**tant que** $s^1_\omega(p^{k-1}) \leq b$ **faire**
  Déterminer le $k$-ième meilleur chemin $P^k$ selon $s^1_\omega$
  **si** *il n'y a plus de chemins* **alors**
    Sortir de la boucle
  **finSi**
  **si** $s^\infty_\omega(p^k) < s^\infty_\omega(p^*)$ **alors**
    $b \longleftarrow s^1_\omega(\vec{p}^k)$
    $P^* \longleftarrow P^k$
    $s^* \longleftarrow s^\infty_\omega(p^*)$
  **finSi**
  $k \longleftarrow k + 1$
**finTq**
**Sorties** : $P^*$, chemin optimal au sens de $s^\infty_\omega$

Pour cela, on commence par déterminer le plus court chemin de n'importe quel sommet du graphe $G$ au sommet final $t$, à l'aide de l'algorithme de Dijkstra par exemple. On obtient ainsi l'arborescence des plus courts chemins $T$. On construit ensuite le graphe des regrets $R$, c'est-à-dire le graphe dont chaque arc issu d'un noeud $v \in V$ exprime le regret $\delta$ de ne pas prendre le plus court chemin de $v$ à $t$. Supposons par exemple que pour aller d'un sommet $u$ à un sommet $v$ de $G$, il existe deux chemins. Le premier chemin $<u, v>$ a un coût de 5, et le second $<u, w, v>$ a un coût de 4. Un seul arc est issu de $u$ dans le graphe des regrets qui va vers $v$ et a comme valeur $\delta = 1$ (= 5 - 4), qui exprime le regret de ne pas prendre le plus court chemin. Prendre le chemin $<u, v>$ au lieu du plus court ($<u, w, v>$) à partir de $u$ nous coûtera donc 1 de plus que prendre le plus court chemin. Le graphe des regrets permet ainsi d'exprimer tous les coûts supplémentaires que l'on a lorsqu'on ne prend pas le chemin le plus court à partir de n'importe quel sommet. Un exemple de graphe des regrets est donné sur la Figure 3. Le graphe de droite correspond au graphe des regrets déterminé à partir du graphe de gauche. Les arcs en trait plein du graphe de droite définissent le graphe des regrets $R$, et ceux en pointillés définissent l'arborescence des plus courts chemins $T$. Une fois $R$ construit, l'idée est d'exploiter ces regrets en ordre croissant afin d'engendrer les chemins dans l'ordre croissant des coûts. On construit alors un tas $H_{out}(v)$ dont les noeuds sont les arcs issus du sommet $v$ dans le graphe des regrets, pour chacun des sommets de $V$. La valeur associée à chacun de ces noeuds est celle du regret $\delta$. On construit ensuite un tas général $H_G(v)$ pour chacun des sommets $v$, dans lequel la racine est $H_{out}(v)$ et le noeud suivant est $H_{out}(w)$ où $w$ est le sommet suivant $v$ dans le plus court chemin allant de $v$ à $t$, autrement dit la tête de l'arc issu de $v$ dans $T$ est $w$ (ce que l'on note $w = Suivant_T(v)$). L'ensemble

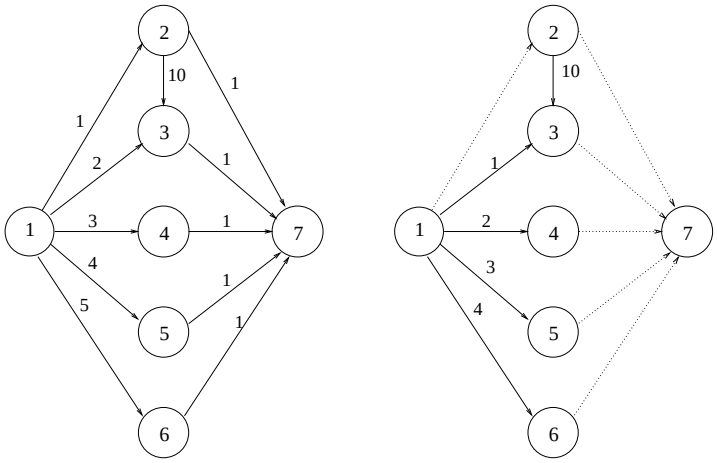


**Fig. 3.** Graphe G et son graphe des regrets

des tas $H_G(v)$ pour tout $v \in E$ définit la structure $D(G)$. Ainsi, tous les noeuds de $D(G)$ sont les arcs de $E - T$ que l'on appelle déviations. $D(G)$ représente l'ensemble des chemins de $s$ à $t$ qui diffèrent de $T$ par seulement un arc. C'est à partir de la structure $D(G)$ que l'on peut construire les $k$ plus courts chemins dans l'ordre des coûts croissants en utilisant une file de priorité $Q$.

L'algorithme 2 présente formellement l'algorithme des $k$ plus courts chemins d'Eppstein. Dans cet énoncé, $L(S^k) = \sum_{(i,j)\in S^k} \delta(i,j)$, $h(v)$ représente la déviation la moins coûteuse du plus court chemin de $v$ à $t$ et $S$ est un ensemble de déviations (i.e. d'arcs de $R$). Pour construire $P^i$ à partir de $S^i$ dans cet algorithme il suffit de compléter les arcs de $S^i$ par ceux du chemin de $v$ à $t$ dans $T$ où $v$ est la tête du dernier arc de $S^i$. L'étape 1 de cet algorithme permet de cumuler les déviations possibles. L'étape 2 permet de traiter les autres déviations à partir du sous-chemin allant de $s$ à la queue de $e$, c'est-à-dire celles qui conduiront à un plus mauvais chemin que celui qui vient d'être trouvé mais à partir du même sous-chemin.

La version de l'algorithme des $k$ meilleurs chemins que nous utilisons est la version améliorée [9] de l'algorithme d'Eppstein. Dans cette version, les tas $H_G(v)$ sont construits au fur et à mesure de leur utilité (c'est-à-dire lorsque le noeud $v$ intervient dans une déviation), ce qui permet de limiter les calculs et la taille des informations à stocker.

### 3.3 Exemple

Dans cette partie, nous illustrons le déroulement de notre algorithme de recherche d'un chemin de meilleur compromis à l'aide d'un exemple. Le graphe sur lequel nous travaillons est celui de gauche dans la Figure 4. Il s'agit de déterminer le chemin de meilleur compromis du sommet $a$ au sommet $g$. La première étape de l'algorithme est la détermination des points idéal et nadir. Dans cet exemple le point idéal est le point $id = (8, 11)$ et le point nadir est le

**Algorithme 2** : Détermination des $k$ plus courts chemins du graphe scalaire $G$

Déterminer l'arborescence des plus courts chemins $T$ et le graphe des regrets $R$
En déduire $P^1$
*Construction de $D(G)$ :*
**pour chaque** $v \in V$ **faire**
  Construire $H_{out}(v)$
  Insérer $H_{out}(v)$ dans $H_G(Suivant_T(v))$ pour former $H_G(v)$
**finPrCh**
$Q \longleftarrow h(s)$ (plus petite déviation possible à partir de $P^1$)
**pour** $k$ *de 2 à* $K$ **faire**
  Extraire $S^k$ l'ensemble de déviations de $Q$ ayant la plus petite évaluation
  En déduire $P^k$
  *Etape 1 :*
  $e \longleftarrow$ dernière déviation de $S^k$
  $f \longleftarrow h(tête(e))$
  Insérer $S^k \bigcup \{f\}$ dans $Q$ avec l'évaluation $L(S^k) + \delta(f)$
  *Etape 2 :*
  **pour chaque** *déviation* $S^\infty_\omega$ *tel que* $(e, f)$ *est un arc dans* $D(G)$ **faire**
    Insérer $S^k \backslash e \bigcup \{f\}$ dans $Q$ avec l'évaluation $L(S^k) - \delta(e) + \delta(f)$
  **finPrCh**
**finPr**
**Sorties** : $P^1, P^2, \ldots, P^k$

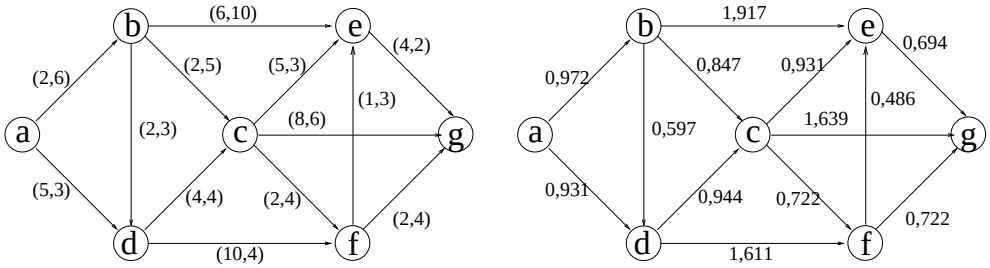


**Fig. 4.** Graphe bicritère et graphe scalaire correspondant

point $nad = (17, 19)$, obtenus à partir des chemins $<a, b, c, f, g>$ de coût $(8, 19)$ et $<a, d, f, g>$ de coût $(17, 11)$. Le vecteur $\omega$ vaut alors $(\frac{1}{9}, \frac{1}{8})$. Une fois ces valeurs déterminées, les arcs du graphe sont valués par la somme pondérée des coûts de leurs critères. Le graphe scalaire obtenu est indiqué sur la droite de la Figure 4. Le plus court chemin dans ce graphe scalaire est le chemin $<a, b, c, f, g>$ dont le vecteur coût est $(8, 19)$. C'est la solution $P^1$ avec $s^\infty_\omega(p^1) = 1$. On a alors $\bar{p}^1 = (8 + (9 \times 1), 11 + (8 \times 1)) = (17, 19)$. Ceci nous permet d'initialiser la borne $b$ à $\frac{17}{9} + \frac{19}{8} = 4,264$. On sait donc que tout chemin $P$ tel que $s^1_\omega(p) > 4,264$ ne peut être optimal. On détermine ensuite le deuxième meilleur chemin $P^2$. C'est le chemin $<a, d, f, g>$ de coût $(17, 11)$ avec $s^1_\omega(p^2) = 3,264 \leq b$, la génération des solutions continue donc. $b$ n'est pas modifié puisque $s^\infty_\omega(p^2) = 1 = s^\infty_\omega(p^1)$. Les étapes de l'algorithme sont récapitulées dans la Table 1. Les

| Chemin $P$ | Chemin | Coût $p$ | $s^1_\omega(p)$ | $s^\infty_\omega(p)$ | $s^1_\omega(\bar{p})$ | b |
|---|---|---|---|---|---|---|
| $P^1$ | $<a, b, c, f, g>$ | (8,19) | 3,264 | 1 | 4,264 | 4,264 |
| $P^2$ | $<a, d, f, g>$ | (17,11) | 3,264 | 1 | 4,264 | 4,264 |
| $P^3$ | $<a, d, c, f, g>$ | (13,15) | 3,319 | 0,556 | 3,375 | 3,375 |
| $P^4$ | $<a, b, c, e, g>$ | (13,16) | 3,444 | 0,625 | 3,514 | 3,375 |

**Tab. 1.** Solutions engendrées

chemins sont engendrés dans l'ordre croissant de $s^1_\omega$ et numérotés dans l'ordre de leur génération.

Comme $s^1_\omega(p^4) > b$, aucune autre solution ne pourra améliorer la meilleure solution courante ($P^3$) d'après la Propriété 1, et l'algorithme peut donc s'arrêter. En conséquence, la solution optimale est cette meilleure solution courante $P^3$. Ainsi, en construisant uniquement 4 chemins (dont 3 non-dominés), le chemin de meilleur compromis a été trouvé, alors que ce graphe contient 17 chemins différents.

Cependant, soulignons que dans le pire des cas, notre algorithme engendrera l'ensemble des chemins du graphe. Sur des graphes comme ceux proposées par Hansen [7] (dont une instance est représentée dans la Figure 2) par exemple, les chemins ont tous la même somme pondérée des coûts. L'énumération ne s'arrêtera donc qu'une fois tous les chemins engendrés. Le nombre de chemins dans ce type de graphes est de $2^{\frac{n-1}{2}}$ où $n$ est le nombre de sommets. Ainsi, l'énumération de tous les chemins avec l'algorithme d'Eppstein se fait en $O(m + n \log n + 2^{\frac{n}{2}})$. Il existe donc des instances pour lesquelles le temps de détermination du chemin de meilleur compromis croît exponentiellement avec le nombre de sommets du graphe. Notre algorithme se révèle toutefois très performant en moyenne comme nous le montre la section suivante.

## 4 Résultats expérimentaux

L'algorithme que nous proposons a été implémenté en C++ et des tests consistant à engendrer différents graphes et observer la performance de calcul ont été effectués sur un processeur Pentium 4 à 2.5GHz.

### 4.1 Génération des graphes

Les tests ont été effectués sur des graphes engendrés à l'aide de trois générateurs :

- Générateur 1 : génération aléatoire des graphes de la manière suivante : un sommet $i$ est relié à un sommet $j$ par un arc $(i, j)$ pour $i < j$ avec une probabilité de $0,5$. De plus, les sommets $i$ et $j$ ne peuvent pas être reliés par un arc si $j - i > \frac{n}{2}$ où $n$ est le nombre de sommets. Cela permet d'éviter d'avoir des chemins avec très peu d'arcs puisqu'ainsi les sommets "proches" de $s$ dans le graphe ne sont pas reliés par un arc aux sommets "proches" de $t$ dans le graphe.

– Générateur 2 : générateur Rudy [8]
– Générateur 3 : génération des graphes proposés par Hansen [7]

Pour déterminer les coûts des arcs (Générateur 1 et 2), nous avons spécifié une fonction linéaire telle que la performance du critère $i$ de tout arc $u \in E$ est $c_i(u) = a_i z_i(u) + d_i$ où $z_i(u)$ est tirée aléatoirement entre 1 et 20, $a_i$ entre 1 et 300 et $d_i$ entre 1 et 10000. Ainsi, on peut espérer avoir des critères dont le coût peut varier de quelques dizaines et d'autres dont le coût peut varier de quelques milliers.

### 4.2 Performances obtenues

Nous comparons ici les performances de notre algorithme à celles d'une approche en deux phases consistant dans un premier temps à engendrer l'ensemble des solutions non-dominées à l'aide de l'algorithme MOA* [13], puis dans un second temps à rechercher dans cet ensemble celle qui minimise la norme pondérée de Tchebycheff. Cette approche est désignée par "I" (approche Indirecte) tandis que notre algorithme est désigné par "D" (approche Directe) puisqu'il détermine directement la solution de meilleur compromis.

La Table 2 présente les résultats obtenus sur les graphes engendrés par le Générateur 1 lorsque l'on fait varier le nombre de sommets et de critères. Le temps obtenu correspond au temps moyen réalisé par les algorithmes exécutés sur une centaine d'instances aléatoires. Le temps d'exécution comprend le temps de calcul du point idéal et celui de recherche du meilleur compromis. Le symbole "–" signifie que la solution n'était toujours pas déterminée au bout d'une heure.

| #sommets | 100 | 500 | 800 | 1000 | 1200 | 1500 | 1800 | 2000 | 2500 |
|---|---|---|---|---|---|---|---|---|---|
| I (5 critères) | 34ms | 11s | 26s | 80s | 103s | 211s | 354s | 406s | 478s |
| D (5 critères) | 3ms | 59ms | 153ms | 235ms | 336ms | 627ms | 769ms | 948ms | 1,4s |
| I (10 critères) | 1,72s | 1397s | 4808s | - | - | - | - | - | - |
| D (10 critères) | 7ms | 73ms | 205ms | 327ms | 532ms | 1s | 1,2s | 1,8s | 2,7s |
| I (20 critères) | 64,6s | - | - | - | - | - | - | - | - |
| D (20 critères) | 24ms | 146ms | 362ms | 612ms | 948ms | 1,33s | 2,30s | 3,12s | 4,30s |
| I (40 critères) | 925s | - | - | - | - | - | - | - | - |
| D (40 critères) | 79ms | 847ms | 565ms | 666ms | 1,07s | 1,78s | 2,91s | 3,72s | 6,64s |

**Tab. 2.** Résultats obtenus (Générateur 1)

La Table 3 présente les résultats obtenus sur des graphes à 5 critères engendrés par le Générateur 2 (Rudy) lorsque l'on fait varier le nombre de sommets et la densité $d$ ($d = \frac{2m}{n(n-1)}$ où $n$ est le nombre de sommets et $m$ le nombre d'arcs). Le temps obtenu correspond au temps moyen réalisé par les algorithmes exécutés sur une vingtaine d'instances aléatoires. Ce temps comprend la détermination du point idéal. Ces différents résultats expérimentaux montrent bien la rapidité de notre approche face à celle consistant à engendrer au préalable l'ensemble des solutions non-dominées, et ce, quel que soit le nombre de critères. En effet, les

| #sommets | 100 | 500 | 800 | 1000 | 1200 | 1500 | 1800 | 2000 | 2500 |
|---|---|---|---|---|---|---|---|---|---|
| I ($d = 30\%$) | 13ms | 3,5s | 17s | 42,6s | 132s | 223s | 238s | 380s | 539s |
| D ($d = 30\%$) | 1ms | 31ms | 75ms | 104ms | 154ms | 218ms | 321ms | 399ms | 620ms |
| I ($d = 60\%$) | 57ms | 27s | 118s | 193s | 525s | 666s | - | - | - |
| D ($d = 60\%$) | 5ms | 56ms | 134ms | 184ms | 266ms | 386ms | 642ms | 803ms | 1,3s |
| I ($d = 90\%$) | 135ms | 148s | 326s | 570s | 647s | - | - | - | - |
| D ($d = 9\ 0\%$) | 8ms | 60ms | 210ms | 310ms | 426ms | 600ms | 1,1s | 1,3s | 2,3s |

**Tab. 3.** Résultats obtenus (Générateur 2)

Tables 2 et 3 montrent que le temps d'exécution de notre approche augmente beaucoup moins rapidement que ceux de l'approche "I" lorsque l'on augmente le nombre de sommets du graphe, sa densité ou le nombre de critères. L'approche "D" fournit par exemple la solution de meilleur compromis d'un graphe de 2 000 sommets valué par 5 critères en moins d'une seconde alors que "I" fournit cette même solution en plus de 6 minutes. Lorsque le nombre de critères augmente, "I" dépasse très vite une heure d'exécution alors que "D" fournit une solution en quelques secondes.

Le nombre de solutions construites pour chacune des approches pour les graphes à 5 critères est précisé dans la Table 4. Pour "I", ce nombre représente le nombre total de solution non-dominées du problème puisqu'on les génère toutes avant de rechercher celle de meilleur compromis. La notation $x(y)$ sur la ligne "D" signifie que $x$ solutions ont été construites et que $y$ de ces $x$ solutions sont non-dominées.

| #sommets | 100 | 500 | 800 | 1000 | 1200 | 1500 | 1800 | 2000 | 2500 |
|---|---|---|---|---|---|---|---|---|---|
| I | 69 | 369 | 458 | 603 | 613 | 615 | 651 | 675 | 762 |
| D | 80(34) | 74(53) | 142(81) | 107(80) | 127(93) | 102(82) | 115(93) | 121(91) | 78(70) |

**Tab. 4.** Nombres de chemins construits (Générateur 1)

Ces résultats montrent d'une part que "D" n'engendre pas la totalité des solutions non-dominées, d'autre part que le nombre de solutions engendrées (dominées ou non) par "D" n'augmente pas avec le nombre de sommets aussi rapidement que le nombre de solutions non-dominées. Ceci montre clairement l'intérêt de notre approche puisque le nombre de solutions engendrées ne croît pas forcément avec le nombre de sommets.

La Table 5 présente les résultats obtenus sur les graphes bicritères engendrés par le Générateur 3 (graphes proposés par Hansen [7]). Sur de tels graphes, "D" comme "I" construit la totalité des chemins puisqu'ils sont tous non-dominés et que la somme pondérée de leur vecteur de coûts est identique pour chacun d'eux. Ces résultats montrent que même dans le cas d'instance pathologiques comme celles d'Hansen [7], la génération des solutions est plus efficace avec "D" qu'avec "I". En effet pour un graphe de ce type à 35 sommets, "I" ne parvient pas à énumérer les 131 072 chemins en une heure alors que "D" les construit en 2s.

| #sommets | 21 | 23 | 25 | 27 | 29 | 31 | 33 | 35 | 37 | 39 | 41 | 43 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | 60ms | 340ms | 2,6s | 13,2s | 55,4s | 229s | 926,4s | - | - | - | - | - |
| D | 0ms | 10ms | 30ms | 80ms | 170ms | 380ms | 780ms | 2s | 4,5s | 13s | 71s | 322s |

**Tab. 5.** Résultats obtenus (Générateur 3)

Ces différents résultats expérimentaux montrent bien l'efficacité de notre approche face à une approche indirecte et mettent en évidence les deux avantages qu'elle présente : elle ne nécessite pas la génération de l'ensemble des solutions non-dominées et les solutions engendrées (dominées ou non) le sont rapidement grâce à l'utilisation d'un graphe scalarisé.

## 5 Conclusion

Dans cet article, nous avons présenté une nouvelle approche pour traiter les problèmes combinatoires multicritères de chemins. Cette approche consiste à utiliser l'algorithme des $k$ plus courts chemins d'Eppstein [4] sur le graphe scalarisé afin de déterminer rapidement la solution de meilleur compromis entre tous les critères. Ceci étant réalisable grâce à la Propriété 1 que nous avons établie. Les tests numériques que nous avons menés montre l'efficacité de notre approche en moyenne, notamment en comparaison avec celle en deux phases consistant à engendrer l'ensemble des solutions non-dominées puis à rechercher la solution de meilleur compromis dans cet ensemble. Un tel algorithme peut sembler intéressant dans un cadre interactif, où la phase de calcul se doit d'être rapide pour ne pas lasser le décideur. Notons par ailleurs que cette approche est assez générique car elle peut s'appliquer à tout problème pour lequel il existe un algorithme efficace d'énumération des $k$ meilleures solutions (la Propriété 1 permettant l'arrêt de l'algorithme est indépendante de la structure de l'espace des solutions). C'est pourquoi nous pensons que notre approche devrait pouvoir s'adapter efficacement par exemple au problème multicritère de sac-à-dos, pour lequel un algorithme performant de recherche des $k$ meilleurs a déjà été proposé [4].

## Remerciements

Je remercie Louis-Xavier Storme, Olivier Spanjaard et Patrice Perny pour les échanges qui ont permis d'aboutir à cet article et les relecteurs anonymes pour leurs suggestions pertinentes.

## Références

1. Bellman, R. : Dynamic Programming, Princeton University Press, 1957.
2. Climaco, J.N. et Martins E.Q. : A bicriterion shortest path algorithm, European Journal of Operational Research, vol. 11, 399-404, 1982.

3. Coutinho-Rodrigues, J.M., Climaco, J.C.N., Current, J.R. : An interactive bi-objective shortest path approach : searching for unsupported nondominated solutions, Computer and Operations Research, vol. 26, 789-798, 1999.
4. Eppstein, D. : Finding the $k$ shortest paths. SIAM Journal on Computing, vol. 28(2), 652-673, 1998.
5. Granat, J. et Guerriero, F. : The interactive analysis of the multicriteria shortest path problem by the reference point method, European Journal of Operational Research, vol. 151, 103-118, 2003.
6. Guerriero, F. et Musmanno, R. : Label correcting methods to solve multicriteria shortest path problems, Journal of Optimization Theory and Applications vol. 111, 589-613, 2001.
7. Hansen, P. : Bicriterion Path Problems. Multiple Criteria Decision Making : Methods and Applications, Springer-Verlag, 109-127, 1980.
8. Helmberg, c. et Rendl, F. : A Spectral Bundle Method for Semidefinite Programming, SIAM Journal on Optimization, vol. 10(3), 673-696, 1999.
9. Jiménez, V.M. et Marzal, A. : A lazy version of Eppstein's k shortest path algorithm. Proceedings of. 2nd International Workshop on Experimental and Efficient Algorithms (WEA 2003), Lecture Notes in Computer Science, vol. 2647, 179-190, 2003.
10. Martins, E.Q. : On a multicriteria shortest path problem, European Journal of Operational Research, vol. 16, 236-245, 1986.
11. Paixão, J.M.P., Martins, E.Q.V., Rosa, M.S. et Santos, J.L.E. : The determination of the path with minimum-cost norm value, Networks, vol. 41(4), 184-196, 2003.
12. Steuer, R.E. et Choo, E.U. : An interactive weighted tchebycheff procedure for multiple objective programming, Mathematical Programming, vol. 26, 326-344, 1983.
13. Stewart, B.S. et White, C.C : Multiobjective A*. Journal of the Association for Computing Machinery, vol. 38, 775-814, 1991.
14. Vanderpooten, D. et Vincke, P. : Description and analysis of some representative interactive multicriteria procedures, Mathematical and Computer Modelling, vol. 12, 1221-1238, 1989.
15. Vincke, P. : Problèmes multicritères, Cahier du Centre d'Etudes Recherche Opérationnelle, vol. 16, 425-439, 1974.
16. Wierzbicki, A.P. : On the completeness and constructiveness of parametric characterizations to vector optimization problems, OR Spectrum, Vol. 8(2), 73-87, 1986.
17. Yu, G. et Yang, J. : On the robust shortest path problem, Computers and Operations Research, vol. 25(6), 457-468, 1998.