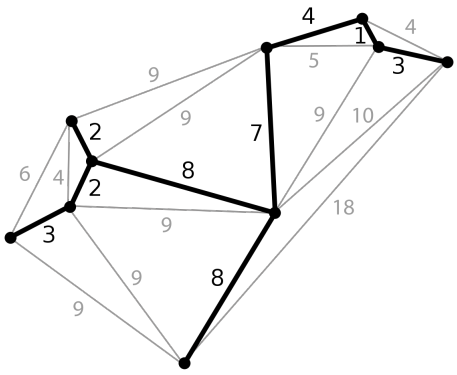
LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Nantes Université

Calcul paramétrique des solutions supportées extrêmes du problème bi-objectif d'arbre couvrant de poids minimum



Rapport de TER

Nacim CHAFAA

Encadré par : Anthony PRZYBYLSKI

Nantes Université - UFR des Sciences et Techniques
Master informatique parcours "optimisation en recherche opérationnelle" (ORO)
Année académique : 2022 - 2023

Mai 2023

Table des matières

1 Introduction

Le domaine de l'optimisation combinatoire et de la recherche opérationnelle offre des outils et des méthodes pour résoudre des problèmes complexes. Le problème de l'arbre couvrant de poids minimum en est un exemple, où l'objectif est de trouver un arbre qui relie tous les sommets d'un graphe tout en minimisant le coût total de cet arbre. Ce problème est bien connu et bien traité, mais il devient plus difficile dans le cas bi-objectif, où deux critères doivent être optimisés simultanément.

C'est dans ce contexte que s'inscrit le travail de recherche présenté dans ce mémoire de première année de master en informatique, parcours optimisation en recherche opérationnelle, encadré par Anthony PRZYBYLSKI de l'équipe de recherche MODELIS, au LS2N. L'objectif de ce travail est de présenter une nouvelle méthode de calcul paramétrique des solutions supportées extrêmes du problème bi-objectif d'arbre couvrant de poids minimum, en s'appuyant sur des sommes pondérées.

Dans les prochaines sections, nous allons décrire plus en détail le problème de l'arbre couvrant de poids minimum et ses difficultés de résolution dans le contexte bi-objectif, ainsi que des méthodes existantes pour résoudre ce problème. Nous présenterons ensuite notre approche, en expliquant les étapes de calcul et les avantages de notre méthode. Enfin, nous évaluerons les performances de notre méthode sur différentes instances et comparerons les résultats avec ceux d'une méthode existante.

2 Généralités

Afin d'aider à la compréhension de ce travail, il est nécessaire de revenir sur quelques définitions et propriétés élémentaires des arbres couvrants, ainsi que de l'optimisation multi-objectif [1].

2.1 Problème de l'arbre couvrant de poids minimum

Définition 1. *Soit un graphe simple non orienté $G = (V, E)$, avec $V = \{v_1, ..., v_n\}$ son ensemble de sommets, tel que $|V| = n$, et $E = \{e_1, ..., e_m\}$ son ensemble d'arêtes, tel que $|E| = m$.*

1. *G est dit connexe, si pour toutes paires de sommets $(v_i, v_j) \in V(G)$, il existe un chemin les reliant.*
2. *G est dit acyclique s'il ne possède aucun cycle. Dans le cas contraire, il est dit cyclique.*

Définition 2. *Un arbre est un graphe acyclique et connexe.*

Propriété 1. *Pour un arbre T à k sommets, il y a une équivalence entre les propriétés suivantes :*

(i) T est un arbre.

(ii) T est un graphe connexe qui possède exactement $k-1$ arêtes, et la suppression de toute arête le déconnecte.

(iii) T est un graphe acyclique à $k-1$ arêtes, et l'ajout de toute arête entre les sommets de T le rend cyclique, et la suppression de l'une des arêtes de ce cycle engendre un nouvel arbre.

Il est important de noter que la propriété 1 (*iii*) sera un élément fondamental de la méthode que nous proposons.

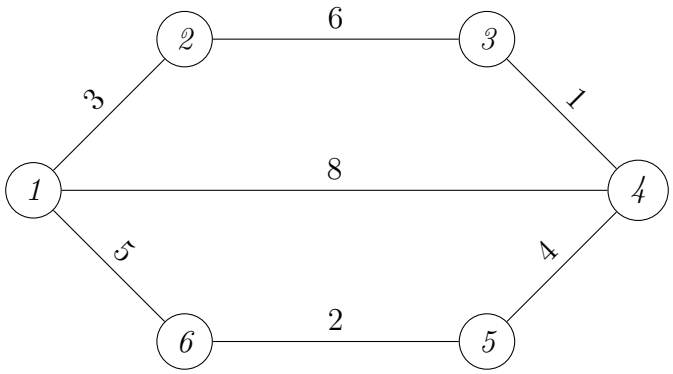
Définition 3. *Un graphe partiel $G'(V, E')$ de G est :*
- *Un graphe qui a les mêmes sommets que G.*
- *Un graphe dont l'ensemble des arêtes E' est inclus dans E.*

Définition 4. *Un arbre couvrant de G est un graphe partiel $T \subset G$ qui est connexe et sans cycle. Il possède donc exactement $n - 1$ arêtes.*

Définition 5. *Un graphe pondéré $G = (V,E,C)$, tel que $C : E \to \mathbb{R}$ est un graphe où un poids $c_i \in C$ est affecté à chaque arête $e_i \in E$, avec $i \in \{1, .., m\}$.*

Définition 6. *Le poids (ou coût) d'un graphe est la somme des poids des arêtes de ce graphe. On le note $\omega(G)$.*

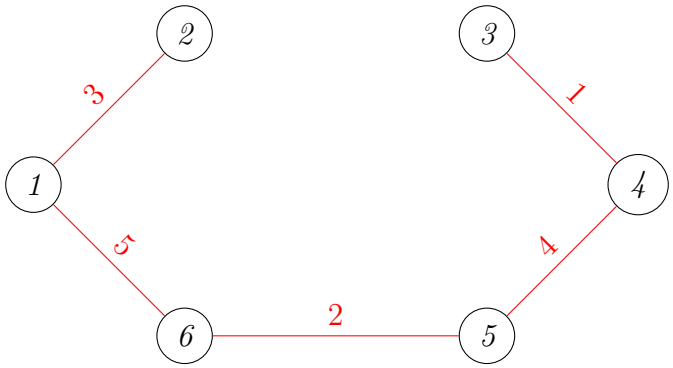
Exemple 1. *Soit $G = (V, E, C)$ le graphe pondéré suivant :*



$$\omega(G) = 26$$

Définition 7. *On appelle arbre couvrant de poids minimum de G, noté ACPM ou MST (minimum spanning tree), tout arbre couvrant dont la somme des poids des arêtes le constituant est minimale.*

Exemple 2. *Soit l'arbre couvrant de poids minimum du graphe G de l'exemple 1 :*



Proposition 1. *Un graphe admet un arbre couvrant si, et seulement si, il est connexe.*

Propriété 2. *Soit G un graphe pondéré. Il possède les proprietés suivantes :*

(i) Pour un même graphe pondéré, il peut exister plusieurs arbres couvrants de poids minimum.

(ii) Si toutes les arêtes de G ont le même poids, tous les arbres couvrants de G sont de poids minimum.

(iii) Soit un cycle $C \subset G$. S'il existe une arête $e_i \in C$ tel que pour toute arête $e_j \in C$ on a $c(e_i) > c(e_j)$, alors e_i ne peut faire partie d'un arbre couvrant de poids minimum.

(iv) Si chaque arête de G a un poids distinct, alors il n'existera qu'un seul et unique arbre couvrant de poids minimum pour G.

(v) S'il existe une arête $e \in E$ tel que pour tout $f \in E$ on a $c(e) < c(f)$, alors cette arête est incluse dans tous les arbres couvrants de poids minimum de G.

Théorème 1. *(Cayley, 1889) : Le nombre d'arbres couvrants d'un graphe complet à n sommets est n^{n-2}.*

Cela implique, avec la propriété 2.(ii), que tout graphe $G = (V, E, C)$ possède au plus n^{n-2} arbres couvrants de poids minimum.

Théorème 2. *Soient $T1$ et $T2$ deux arbres couvrants de poids minimum d'un même graphe G. Alors il existe une séquence de transpositions d'arêtes qui mène de T1 à T2, et inversement.*

2.2 Méthodes de résolution du problème mono-objectif

Il existe de nombreux algorithmes de constrution d'un arbre couvrant de poids minimum. Les plus connus étant l'algorithme de **Prim** et l'algorithme de **Kruskal**. Ces derniers sont tous deux des algorithmes gloutons qui se résolvent en temps polynomiaux. Dans la méthodologie qui va être présentée par la suite, nous nous baserons sur l'algorithme de Kruskal afin de construire un arbre couvrant de poids minimum en partant d'une séquence d'arêtes.

Algorithme de Kruskal : [2] :

- **Données :** un graphe pondéré $G = (V, E, C)$.
- **Initialisation :** initialisation de l'ensemble d'arêtes de l'arbre $T = \emptyset$ et tri des arêtes de G par ordre croissant de leurs poids.
- **Boucle principale :** on rajoute successivement les arêtes triées à T, et ce, sans créer de cycle, jusqu'à avoir $|T| = |V| - 1$.
- **Sortie :** T un arbre couvrant de poids minimum de G.

2.3 Optimisation multi-objectif

Dans cette partie, nous allons poser quelques notions, notations et propriétés des problèmes d'optimisation multi-objectif. Cependant, nous nous restreindrons dans le cadre de ce travail au cas bi-objectif uniquement.

Définition 8. *Un problème d'optimisation multi-objectif (MOP) consiste à optimiser (minimiser/maximiser) plusieurs fonctions objectifs $f^1, f^2, ..., f^p$, $p \geq 2$ sur un domaine (ensemble) de solutions admissibles, appelé région admissible, et que l'on notera X.*

Soit un problème bi-objectif à n variables et deux fonctions objectifs à optimiser f^1 et f^2.

Définition 9. *Un couple de valeurs pour les deux fonctions objectifs est appelé point, et si ce point correspond à l'image d'une solution admissible, on parle de point réalisable.*

Définition 10. *Soit un problème d'optimisation multi-objectif $f^1, ..., f^p$, avec $p \geq 2$.*

- *On appelle espace de décision l'espace dans lequel sont représentées les valeurs des solutions $x_i \in \mathbb{R}$.*
- *On appelle espace des critères l'espace dans lequel sont représentés les points $f(x) \in \mathbb{R}^p$, avec $x \in X$.*

Les comparaisons entre les différentes solutions se feront dans l'espace des critères.

Dans le cas d'un problème d'optimisation mono-objectif, la comparaison de deux solutions est triviale. En effet, on associe une valeur réelle à chaque solution. En d'autres termes il suffit de prendre la plus petite pour un problème de minimisation, et la plus grande pour un problème de maximisation. Cela diffère en situation de problème d'optimisation multi-objectif, car deux solutions peuvent être incomparables. En effet, si comparer deux valeurs réelles ne pose pas de question, dans le cas de vecteurs il est nécessaire de poser des opérateurs de comparaison entre ces solutions.

Comparaison des vecteurs des solutions

Pour deux solutions x^1 et x^2 d'un problème de minimisation $min(f^1(x), f^2(x))$, et dont les points respectifs dans l'espace des critères sont y^1 et y^2, on dira que :

- y^1 **domine faiblement** y^2 et on notera $y^1 \leqq y^2$ si, et seulement si $y^1_k \leq y^2_k$, pour $k = 1, ..., p$.
- y^1 **domine strictement** y^2 et on notera $y^1 < y^2$ si, et seulement si $y^1_k < y^2_k$, pour $k = 1, ..., p$.
- y^1 **domine** y^2 et on notera $y^1 \leq y^2$ si, et seulement si $y^1_k \leq y^2_k$ et $y_1 \neq y_2$.

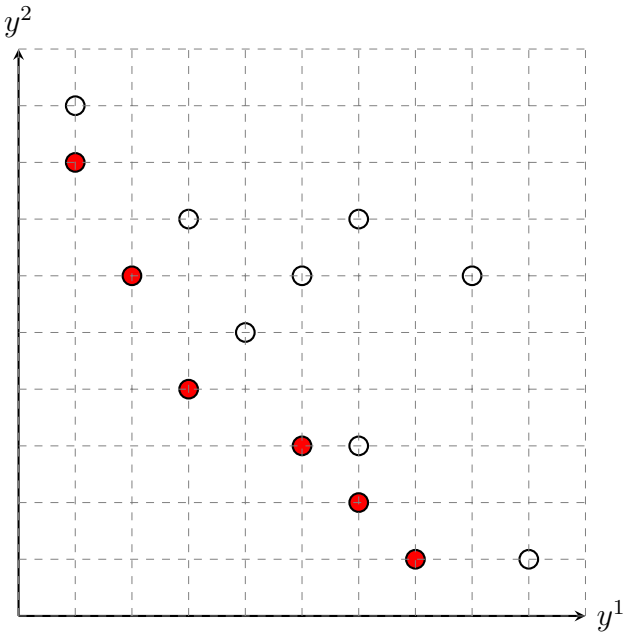


FIGURE 1 – Représentation d'un nuage de points réalisables sur l'espace des critères dans le cas bi-objectif. Les points non-dominés sont indiqués en rouge.

Dans la figure 1, on peut remarquer que le point (3,7) est strictement dominé par le point (2,6). Le point (6,3) n'est strictement dominé par aucun autre point réalisable. Par contre, il est dominé par (5,3) et (6,2). Indépendamment de l'importance des objectifs et de préférences que pourraient définir les décideurs à leur sujet, il ne sera jamais judicieux de leur fournir une solution correspondant à un point dominé.

Contrairement aux problèmes d'optimisation mono-objectif, il n'existe pas de point optimal unique pour un problème multi-objectif. Cela nous mène à considérer les notions de solutions efficaces et de points non-dominés.

Définition 11. *Les solutions admissibles correspondant aux points non-dominés sont appelées solutions efficaces. En effet, la notion de dominance est relative à l'espace des critères. Dans l'espace des objectifs, nous parlerons de notion d'efficacité :*

- *On dira qu'une solution $x^* \in X$ est **faiblement efficace** et le point $f(x^*)$ **faiblement non-dominé** s'il n'existe pas de $x \in X$ tel que : $f(x) < f(x^*)$.*
- *On dira que $x^* \in X$ est **efficace** et le point $f(x^*)$ **non-dominé** si n'existe pas de $x \in X$ tel que : $f(x) \leq f(x^*)$*

Par exemple, dans la figure 1, on peut voir que les points (1,8) et (3,4) sont non-dominés. C'est donc les images de solutions efficaces.

Nous faisons l'hypothèse qu'aucun objectif n'est plus important qu'un autre et que nous ne connaissons pas les préférences des décideurs au moment de la résolution. Dans ce contexte, la résolution exacte d'un problème d'optimisation bi-objectif consiste à calculer un ensemble complet, i.e. un ensemble contenant au moins une solution efficace pour chaque point non-dominé.

La question est maintenant de savoir comment calculer des solutions efficaces. Une méthode simple est la somme pondérée (Geoffrion, 1968), qui est utilisée pour transformer un problème d'optimisation bi-objectif en un problème mono-objectif en attribuant des poids $\lambda \in \mathbb{R}^2$ aux deux objectifs et en prenant leur somme pondérée.

Soit (P) un problème bi-objectif à optimiser tel que :

$$(P) = \begin{cases} \text{minimiser} & f(x) = (f^1(x), f^2(x)) \\ \text{s.t.} & x \in X. \end{cases}$$

La somme pondérée $P(\lambda)$ pour ce problème s'exprime alors comme :

$$(P(\lambda)) = \begin{cases} \text{minimiser} & f(x) = \lambda_1 f^1(x) + \lambda_2 f^2(x) \\ \text{s.t.} & x \in X. \end{cases}$$

Propriété 3. *Soit x^* une solution optimale pour une somme pondérée $P(\lambda)$:*

- *Si $\lambda_k > 0$ pour tout $k \in \{1, ..., p\}$, alors x^* est efficace.*
- *Si $\lambda_k \geq 0$ pour tout $k \in \{1, ..., p\}$, alors x^* est faiblement efficace.*

Remarque 1. *Il est possible que deux solutions admissibles correspondent à une même solution efficace.*

Propriété 4. *Il existe des solutions efficaces qui ne sont optimales pour aucun problème $P(\lambda)$.*

Par exemple, si on observe la figure 1, on peut voir que les points (5,3) et (6,2) sont les images de solutions efficaces qui ne sont optimales pour aucun problème $P(\lambda)$. Cela mène alors à une classification des solutions efficaces.

Propriété 5. *Les solutions efficaces peuvent être divisées en deux ensembles :*

- *Les solutions **supportées**, dont les points correspondant à leurs images respectives se trouvent sur les bords de l'enveloppe convexe de la région admissible. Ces solutions peuvent être trouvées en résolvant des combinaisons linéaires des objectifs.*

- *Les solutions **non-supportées**, dont les points correspondant à leurs images respectives ne se trouvent pas sur les bords de l'enveloppe convexe mais ne sont pas dominés pour autant, ce qui les rend plus complexes à trouver que les solutions supportées.*

Il est à noter que dans ce travail, nous nous intéresserons à la recherche des solutions supportées seulement. Cet ensemble est lui même constitué de deux sous-ensembles : les solutions **supportées extrêmes** et les solutions **supportées non-extrêmes**. Si bien qu'elles sont toutes situées sur les bords de l'enveloppe convexe de la région admissible, les solutions supportées extrêmes sont situées sur les extrémités de cette dernière, là où les solutions supportées non-extrêmes ne le sont pas.

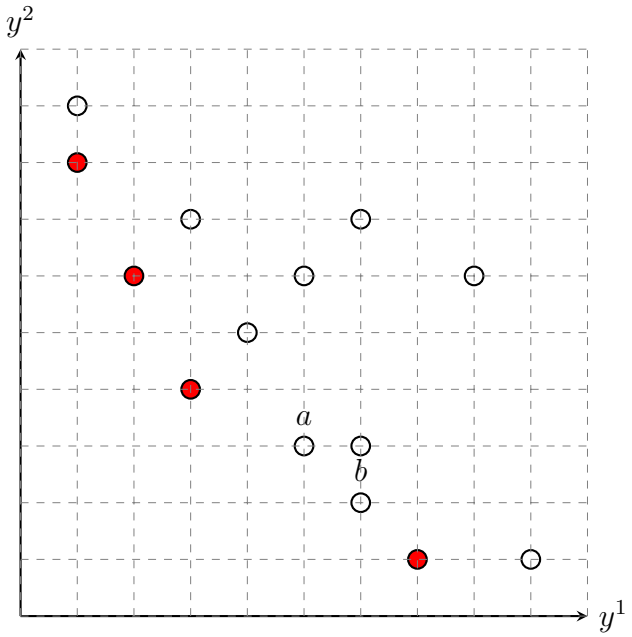


FIGURE 2 – Représentation d'un nuage de points réalisables. Les points non-dominés qui sont les images de solutions supportées sont coloriés en rouge, et les points non-dominés 'a' et 'b', qui sont les images de solutions non-supportées en sont déduits.

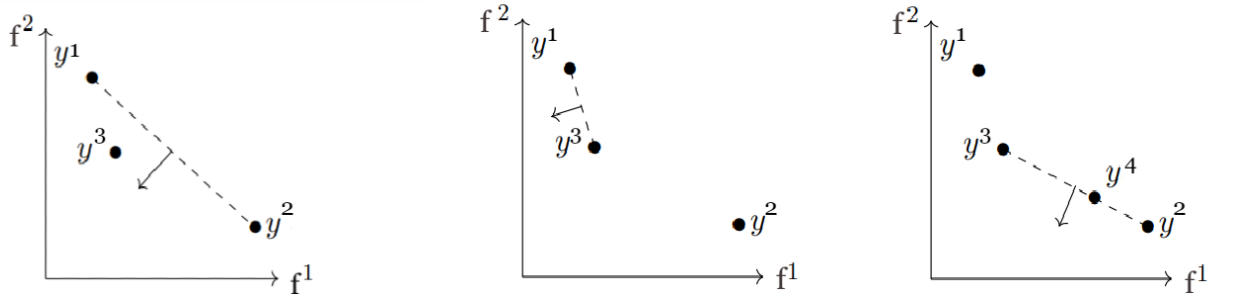
On peut voir sur la figure 2 que les points (1,8), (3,4) et (7,1) représentent les images de solutions supportées extrêmes, tandis que le point (2,6) est l'image d'une solution supportée non-extrême. Par ailleurs, les points a et b représentés sur la même figure sont des points non-dominés, mais ne sont pas les images de solutions supportées, puisqu'ils se situent à l'intérieur de l'enveloppe convexe de la région admissible.

Parmi les méthodes de résolution de problèmes d'optimisation bi-objectifs en utilisant la somme pondérée, se trouve la méthode dichotomique (Aneja et Nair, 1979). Cette méthode a été conçue dans le but de trouver l'ensemble des solutions supportées extrêmes et quelques solutions supportées non-extrêmes.

Méthode dichotomique [3] :

Soit un problème d'optimisation bi-objectif avec deux fonctions à minimiser f^1 et f^2. Comme l'illustre la figure 3, dans un premier temps, on calcule les points non-dominés qui minimisent f^1 et f^2. Ensuite, vient la phase recursive de la méthode, dans laquelle l'espace des critères est exploré entre chaque paire de points non-dominés consécutifs (y, y'). À chaque itération, un nouveau poids λ est trouvé en calculant la normale de la droite reliant les deux points y et y', et le problème pondéré est alors résolu comme un problème mono-objectif.

Dans le cas où la solution $\hat{x}$ du problème pondéré a un coût inférieur à celui de x et de x', dont les images correspondantes dans l'espace des critères sont y et y' respectivement, alors une nouvelle solution supportée est trouvée (comme c'est le cas pour la solution associée au point y^3 dans la figure 3.2). Dans l'autre cas, où $\hat{x}$ a un coût égal à x et x', alors son point-non dominé associé est égal à y ou y', ou est l'image d'une solution suportée non-extrême (c'est le cas du point y^4 dans la figure 3.3). Il s'agit par ailleurs de la condition d'arrêt de l'algorithme.



(1) Itération 1 : recherche dans la direction définie par y^1 et y^2 pour obtenir y^3.

(2) Itération 2 : pas de nouveau point trouvé dans la direction

(3) Itération 3 : recherche dans la direction définie par y^2 et y^3

FIGURE 3 – Méthode dichotomique pour un problème d'optimisation bi-objectif.

C'est à la méthode dichotomique que nous allons comparer la nouvelle méthode qui sera proposée à l'issue de ce travail.

2.4 Problème de l'arbre couvrant de poids minimum bi-objectif

Soit un graphe simple non-orienté $G = (V, E, C)$, où V est l'ensemble des n sommets, E l'ensemble des m arêtes, et $C : E \to \mathbb{R}^2$ les poids des arêtes sur les deux objectifs. Soit également τ, l'ensemble des arbres couvrants de poids minimum du graphe G (la région admissible des solutions du problème).

— On notera f^1 et f^2 la première et la seconde fonction objectif respectivement.
— On notera c_i^1 et c_i^2 les coûts d'une arête $i \in E$ par rapport à f^1 et f^2 respectivement.

Définition 12. *Le problème bi-objectif d'arbre couvrant de poids minimum est défini comme suit :*

$$\begin{cases} minimiser & f(T) = (f^1(T), f^2(T)) \\ s.t. & T \in \tau. \end{cases}$$

La complexité du problème bi-objectif d'arbre couvrant de poids minimum reste ouverte dans le cas où l'on cherche un ensemble complet. Cependant, nous connaissons certaines propriétés liées à ce problème.

Théorème 3. *(Ruzika et Hamacher, 2009). Le nombre de solutions efficaces et le nombre de points non-dominés du problème d'arbre couvrant de poids minimum bi-objectif peut être exponentiellement grand par rapport au nombre de sommets.[4]*

Preuve. Considérons le graphe complet $G = K_n$ qui possède n sommets et $m = \frac{n(n-1)}{2}$ arêtes. On sait d'après le théorème 1 que G possède n^{n-2} arbres couvrants. On notera l'ensemble des solutions efficaces Y_N et l'esemble des arbres couvrants de poids minimum τ.

On sait que chaque $e_i \in E$ a deux coûts définis par :

$$c^1(e_i) = 2^{i-1}$$
$$c^2(e_i) = 2^m - 2^{i-1} = 2^m - c^1(e_i).$$

On a donc $c^1(e_i) + c^2(e_i) = 2^m$ pour tout $i \in 1, ..., m$ et $c^1(T) + c^2(T) = (n-1)2^m$ pour tout $T \in \tau$. Cela veut dire que toutes les solutions efficaces sont optimales pour un problème pondéré. Les points non-dominés correspondants à ces solutions sont tous situés sur une droite dans l'espace des critères. Elles sont donc toutes supportées. Par conséquent, le nombre de points non-dominés supportés peut être exponentiellement grand par rapport au nombre de sommets. Remarquons tout de même qu'ici on a deux poibts supportés extrêmes, tous les autres sont non-extrêmes.

Théorème 4. *Le nombre de points supportés extrêmes est borné polynomialement, et leur nombre maximum est m^2.[4]*

Remarque 2. *Nous allons proposer une preuve au théorème 4 lors de la présentation de la méthode.*

3 Méthodologie proposée

La méthode que nous allons présenter dans ce qui suit permet de calculer un sous-ensemble de points non-dominés supportés extrêmes qui sont les images de solutions optimales pour une somme pondérée des deux objectifs (comme le montre la figure 2 pour une optimisation bi-objectif). Ceci est réalisable avec la méthode de Anja et Nair, mais demande de résoudre de manière répétée des problèmes d'optimisation mono-objectif. Notre but sera ici d'exploiter les propriétés structurelles du problème d'arbre couvrant de poids minimum afin d'éviter d'effectuer ces résolutions. Nous chercherons à procéder par réoptimisation pour passer d'une solution supportée à une autre. Suite à cela, nous comparerons les résultats obtenus avec ceux de la méthode dichotomique.

Remarque 3. *Nous utiliserons dans ce qui suit le terme "coût" pour désigner les poids sur les arêtes. En effet, le terme poids correspondra désormais à la somme pondérée.*

3.1 Notion de poids critiques

Nous allons utiliser la somme pondérée pour calculer les solutions supportées extrêmes d'un problème d'arbre couvrant de poids minimum bi-objectif.

Soient deux fonctions objectifs f^1 et f^2, nous considérons alors des sommes pondérées du type :

$$\lambda_1 f^1 + \lambda_2 f^2 \quad o\grave{u} \quad \lambda_1, \lambda_2 \geq 0 \tag{1}$$

De manière équivalente, on a :

$$\lambda f^1 + (1-\lambda) f^2 \quad o\grave{u} \quad \lambda \in [0,1] \tag{2}$$

En effet, il est possible de diviser $\lambda_1 f^1 + \lambda_2 f^2$ par $\lambda_1 + \lambda_2$ en conservant un problème équivalent. Nous avons alors $\frac{\lambda_1}{\lambda_1+\lambda_2} + \frac{\lambda_2}{\lambda_1+\lambda_2} = 1$. En posant $\lambda = \frac{\lambda_1}{\lambda_1+\lambda_2}$, nous obtenons $1-\lambda = \frac{\lambda_2}{\lambda_1+\lambda_2}$ avec $\lambda \in [0,1]$. Cette seconde solution aura l'avantage de ne faire apparaitre qu'un seul paramètre.

Si $\lambda = 1$, on a f^1, et si $\lambda = 0$, on aura f^2. Le but sera de partir de $\lambda = 1$ et de décroitre progressivement la valeur de λ jusqu'à 0.

On sait que pour un poids λ, on a une séquence d'arêtes triées associée. Dans la méthode que nous proposons, nous allons nous intéresser à l'évolution de cette séquence quand nous modifions progréssivement les poids. Autrement dit, on cherche donc à savoir quand est-ce que deux arêtes changent de place.

On notera dans ce qui suit $c_i = (c_i^1, c_i^2)$ et $c_j = (c_j^1, c_j^2)$ les coûts de deux arêtes i et j respectivement par rapport aux deux objectifs f^1 et f^2.

Nous allons maintenant chercher des poids dits **critiques** pour lesquels des sommes pondérées des paires d'arêtes (i,j) sont égales pour les deux objectifs f^1 et f^2. Nous cherchons alors les $\lambda(i,j)$ tel que :

$$\lambda c_i^1 + (1-\lambda) c_i^2 = \lambda c_j^1 + (1-\lambda) c_j^2 \tag{3}$$

Ceci est équivalent à :

$$\lambda(c_i^1 - c_i^2 - c_j^1 + c_j^2) = c_j^2 - c_i^2 \tag{4}$$

On obtient alors :

$$\lambda(i,j) = \frac{c_j^2 - c_i^2}{c_i^1 - c_i^2 - c_j^1 + c_j^2} \tag{5}$$

Cette équation a une solution si $c_i^1 - c_i^2 - c_j^1 + c_j^2 \neq 0$

Pour une valeur λ' inférieure à $\lambda(i,j)$, nous avons $\lambda' c_i^1 + (1-\lambda')c_i^2 > \lambda' c_j^1 + (1-\lambda')c_j^2$ ou le contraire, et pour une valeur λ' supérieure à $\lambda(i,j)$, l'inégalité changera de sens.

Deux cas sont à remarquer lors du calcul des poids critiques :

— Nous n'avons pas forcément $\lambda(i,j) \in]0,1[$. Si c'est le cas, cela implique que nous n'avons un changement de sens pour les inégalités ci-dessus que pour des vecteurs de poids qui incluent des valeurs négatives ou nulles. De tels vecteurs de nous intéressent pas dans la suite.

— $\lambda(i,j)$ n'existe pas forcément. Ce cas arrive si on a toujours l'égalité $\lambda c_1^i + (1-\lambda)c_2^i = c_1^j + (1-\lambda)c_2^j$ indépendamment de la valeur de λ. On peut aussi toujours avoir l'inégalité $\lambda c_1^i + (1-\lambda)c_2^i < \lambda c_1^j + (1-\lambda)c_2^j$ (ou l'inégalité dans l'autre sens) indépendamment de la valeur de λ.

Puisqu'un poids critique est calculé pour chaque paire d'arêtes, on a au plus $m(m-1)$ poids critiques. Dans le pire des cas, ces poids critiques sont tous inclus dans l'intervalle]0,1[. Cela veut dire qu'on a au plus m(m-1)+2 arbres couvrants de poids minimum, chacun associé à un poids critique différent (les deux solutions ajoutées sont celles correspondant aux poids $\lambda = 1$ et $\lambda = 0$ respectivement).

Nous revenons alors au théorème 4 et nous récapitulons ce qu'on a obtenu.

3.2 Déscription de la méthode

Pour commencer, on effectue un tri lexicographique des arêtes par rapport à leurs poids sur le premier puis le second objectif. On obtient alors la séquence d'arêtes initiale que l'on notera S_{init}. On construit ensuite l'arbre couvrant de poids minimum associé à cette séquence. Suite à cela, on calcule le poids critique de chaque paire d'arêtes. On obtient alors une liste de poids critiques.

Proposition 2. *À chaque séquence d'arêtes S, on peut associer un arbre couvrant de poids minimum.*

Dans un premier temps, nous ferons l'hypothèse simplificatrice que tous les poids critiques sont distincts. Nous allons parcourir ces derniers de manière décroissante. Cela nous ménera à effectuer des transpositions des arêtes successives. En effet, à chaque itération, on cherche à partir d'une séquence d'arêtes S dans laquelle l'arête i précéde l'arête j. Le but sera d'effectuer une transposition des deux arêtes dans S afin d'obtenir une nouvelle séquence S'. T' est l'arbre couvrant associé à S'.

Proposition 3. *S et S' sont identiques, sauf pour les arêtes i, j qui sont transposées comme suit :*

$$S = (..., i, j, ...)$$
$$S' = (..., j, i, ...)$$

Proposition 4. *Soit T un arbre associé à la séquence d'arêtes S et T' un arbre associé à la séquence d'arêtes S'. On a alors quatre cas possibles :*

(i) Si i, $j \notin T$, alors $T' = T$.

(ii) Si i, $j \in T$, alors $T' = T$.

(iii) Si $i \notin T$, $j \in T$, alors $T' = T$.

(iv) Si $i \in T$, $j \notin T$, alors on a deux situations possibles à considérer :

(a) Si l'ajout de j dans T engendre un cycle C, et si $i \in C$. Alors on a $T' = (T \cup j) \backslash i$.

(b) Dans le cas contraire, on a $T' = T$.

Preuve.

(i) La sous-séquence précédent i et j dans S et S' est identique, les arêtes i et j ne peuvent donc pas faire partie de T', comme c'est le cas pour T.

(ii) La sous-séquence précédent i et j dans S et S' est identique, les arêtes font donc partie de T', tout comme elles faisaient partie de T.

(iii) Les arêtes insérées dans T' lorsqu'on arrive à j sont les mêmes que dans T, c'est pourquoi on insert j dans T'. Quant à i, elle ne faisait pas partie de T, et puisque les arêtes insérées dans T' lorsqu'on arrive à i sont les mêmes elle n'est donc pas dans T' non plus.

(iv) (a) Dans la construction de T, j n'a pas été inséré du fait que i était présent. Avec la séquence S', on inserera j et à l'inverse, on ne pourra pas insérer i car cela engendrera à nouveau le cycle C. Il reste maintenant à prouver que la suite de la construction de l'arbre T' est identique à l'arbre T. Pour cela, associons des coûts aux arêtes. On suppose que les arêtes jusqu'à i et j inclus ont un coût nul. Les k arêtes suivantes ont des coûts strictement croissants donnés par $2^0, 2^1, ..., 2^{k-1}$. Supposons que les arêtes insérées dans T et T' après i et j ne soient pas identiques. Dans ce cas, le coût des deux arbres est différent. Supposons que le coût de T' soit inférieur à celui de T. Dans ce cas, l'arbre T'' défini par $(T' \cup i) \backslash j$ a le même coût que T' et respecte bien le début de la séquence associée à T. Ce qui entre en contradiction avec le résultat qu'on obtiendrait avec l'algorithme de Kruskal.

(b) La sous-séquence précédant i et j dans S et S' est identique, et puisque l'ajout de l'arête j à T n'engendre pas de cycle contenant i, cela implique que j ne peut pas être insérée dans T'.

On effectue alors les transpositions des arêtes successives deux à deux et on ajoute chaque nouvel arbre trouvé à la liste des solutions. À chaque transposition des arêtes, une nouvelle solution associée à la nouvelle séquence d'arêtes ne sera obtenue que dans le cas $iv.a$ présenté dans la proposition 4.

Il est important de noter qu'il n'y aura un changement d'ordre pour la séquence d'arêtes triées que lorsqu'un poids critique sera atteint (soit un des $\lambda(i, j)$). Par conséquent, il sera juste nécessaire de parcourir les $\lambda(i, j)$ dans l'ordre dans lequel ils ont été triés, et déduire les conséquences sur la solution optimale de la somme pondérée.

3.3 Exemple d'application

La méthode sera illustrée à l'aide d'un exemple didactique représenté par le graphe pondéré $G = (V, E, C)$ de la figure 4. À chaque arête $i \in E$ sont associés les poids (c_i^1, c_i^2) représentent les coûts de l'arête par rapport à la première et la seconde fonction objectif respectivement.

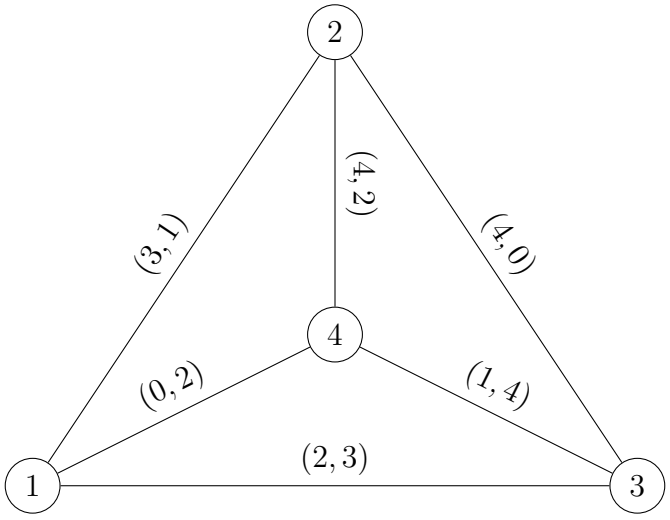


FIGURE 4 – Graphe simple non-orienté pondéré G

Cet exemple servira de base dans la présentation de l'idée. Le calcul des poids critiques nous obtenons la liste des poids critiques suivante :

Paire d'arêtes	**Coûts**	**Poids critique**
(1,2) - (1,4)	(3,1) - (0,2)	$\frac{1}{4}$
(1,2) - (1,3)	(3,1) - (2,3)	$\frac{2}{3}$
(1,2) - (2,3)	(3,1) - (4,0)	$\frac{1}{2}$
(1,2) - (2,4)	(3,1) - (4,2)	Pas de poids critique
(1,2) - (3,4)	(3,1) - (1,4)	$\frac{3}{5}$
(1,4) - (1,3)	(0,2) - (2,3)	-1
(1,4) - (2,3)	(0,2) - (4,0)	$\frac{1}{3}$
(1,4) - (2,4)	(0,2) - (4,2)	0
(1,4) - (3,4)	(0,2) - (1,4)	2
(1,3) - (2,3)	(2,3) - (4,0)	$\frac{3}{5}$
(1,3) - (2,4)	(2,3) - (4,2)	$\frac{1}{3}$
(1,3) - (3,4)	(2,3) - (1,4)	$\frac{1}{2}$
(2,3) - (2,4)	(4,0) - (4,2)	1
(2,3) - (3,4)	(4,0) - (1,4)	$\frac{4}{7}$
(2,4) - (3,4)	(4,2) - (1,4)	$\frac{2}{5}$

TABLEAU 1 – Liste des poids critiques sur l'instance séléctionnée

Un premier tri des arêtes est effectué par rapport au premier puis au second objectif. La séquence d'arêtes triées que l'on va appeler S_{init} est donc la suivante :

$$S_{init} = (1,4), (3,4), (1,3), (1,2), (2,3), (2,4)$$

On construit l'arbre couvrant de poids minimum T_1 associé à la séquence S_{init}.

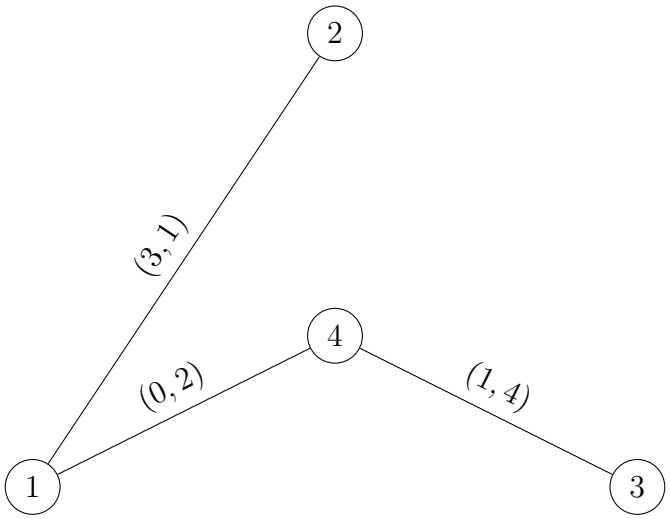


FIGURE 5 – Première solution - T_1

Nous allons maintenant parcourir la liste des poids critiques par ordre décroissant. L'ordre reste donc inchangé jusqu'à $\lambda = \frac{2}{3}$.

1. $\lambda = \frac{2}{3}$: On transpose les arêtes (1,3) et (1,2) pour obtenir la séquence d'arêtes S_1 tel que :

   $$S_1 = (1,4), (3,4), (1,2), (1,3), (2,3), (2,4)$$

   On remarque que l'arête $(1,2) \in T$ et l'arête $(1,3) \notin T_1$, ce qui correspond au cas (iii) de la proposition 4. Aucun nouvel arbre n'est donc engendré. L'ordre reste alors inchangé jusqu'à $\lambda = \frac{3}{5}$.

2. $\lambda = \frac{3}{5}$: On remarque que deux paires d'arêtes donnent ce poids critique, à savoir la paire (1,2) et (3,4), et la paire (1,3) et (2,3) respectivement. Les deux paires étant indépendante (n'ont pas d'arête en commun), l'échange s'effectuera en même temps sur chacune d'entre elles. Cela engendre une nouvelle séquence d'arêtes S_2 tel que :

   $$S_2 = (1,4), (1,2), (3,4), (2,3), (1,3), (2,4)$$

   On voit que les arêtes $(1,2)$ et $(3,4) \in T_1$, ce qui correspond au cas (ii) de la proposition 4, tandis que les arêtes $(1,3)$ et $(2,3) \notin T_1$, ce qui correspond au cas (i) de la proposition 4. Aucun nouvel arbre couvrant n'est donc engendré.

3. $\lambda = \frac{4}{7}$: On transpose les arêtes (3,4) et (2,3). On obtient donc la séquence d'arêtes S_3 tel que :

   $$S_3 = (1,4), (1,2), (2,3), (3,4), (1,3), (2,4)$$

L'arête $(3,4) \in T_1$, mais pas l'arête $(2,3)$, et l'ajout de cette dernière à T_1 créerait un cycle qui contient l'arête $(3,4)$, ce qui correspond au cas (iv) de la proposition 4. Une nouvelle solution T_2 est donc trouvée.

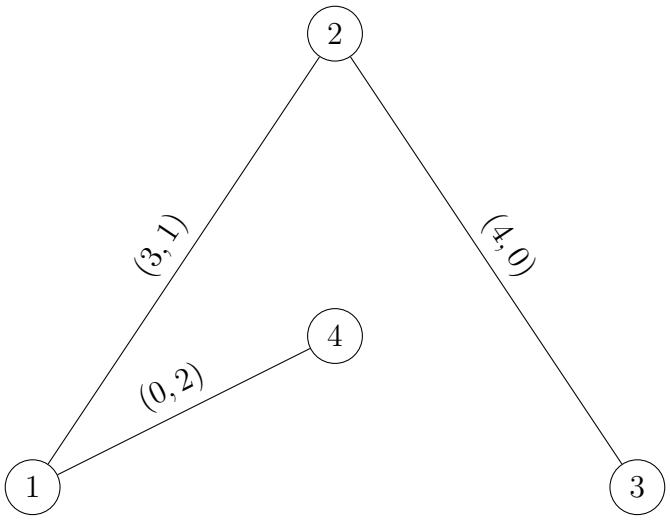


FIGURE 6 – Seconde solution - T_2

4. $\lambda = \frac{1}{2}$: Comme pour $\lambda = \frac{3}{5}$, il existe deux couples d'arêtes indépendants qui donnent ce poids critique, à savoir la paire (1,2) et (2,3) ainsi que la paire (1,3) et (3,4).
   Après les transpositions, on obtient la séquence d'arêtes S_4 tel que :
   $$S_4 = (1,4),(2,3),(1,2),(1,3),(3,4),(2,4)$$
   Les arêtes (1,2) et (2,3) $\in T_2$, ce qui correspond au cas (ii) de la proposition 4. Tandis qu'aucune des deux arêtes (1,3) et (3,4) n'en fait partie, ce qui correspond au cas (i) de la proposition 4. Aucune nouvelle solution n'est donc engendrée.

5. $\lambda = \frac{2}{5}$: On transpose les arêtes (2,4) et (3,4). Suite à cela on obtient la séquence S_5 :
   $$S_5 = (1,4),(2,3),(1,2),(1,3),(2,4),(3,4)$$
   Les deux arêtes ne faisant pas partie de la dernière solution T_2. Aucune nouvelle solution n'est trouvée.

6. $\lambda = \frac{1}{3}$: Deux paires d'arêtes indépendantes partagent ce poids critique. On permuttera alors l'arête (1,4) avec (2,3), et l'arête (1,3) avec (2,4). La nouvelle séquence S_6 est donc la suivante :
   $$S_6 = (2,3),(1,4),(1,2),(2,4),(1,3),(3,4)$$
   Le prmeier couple d'arêtes étant inclu dans la dernière solution (cas (ii) de la proposition 4), et le second ne l'étant pas (cas (ii) de la proposition 4), aucune nouvelle solution n'est trouvée.

7. $\lambda = \frac{1}{4}$: Transposition des arêtes (1,2) et (1,4). On obtient alors la nouvelle séquence S_7 :
   $$S_7 = (2,3),(1,2),(1,4),(2,4),(1,3),(3,4)$$

Les deux arêtes faisant toutes deux partie de la dernière solution trouvée (cas (ii) de la proposition 4). Aucune nouvelle solution n'est trouvée.

Après avoir fini le parcours des poids critiques. On obtient la liste des solutions supportées extrêmes du problème. Dans le cas de cet exemple, cette liste se limite à deux éléments.

3.4 Cas où on a des poids critiques identiques

Nous relâchons maintenant l'hypothèse que les poids critiques sont tous distincts.

Remarque 4. *Il est possible qu'il existe plusieurs couples d'arêtes qui possédent le même poids critique, et quand on a des poids critiques indentiques, il y en a nécessairement plusieurs.*

Proposition 5. *Si on a* $\lambda(i,j) = \lambda(j,k)$ *alors* $\lambda(i,j) = \lambda(j,k) = \lambda(i,k)$.

Preuve. On sait que pour un poids $\lambda(i,j)$ on a :

$$\lambda c_i^1 + (1-\lambda)c_i^2 = \lambda c_j^1 + (1-\lambda)c_j^2 \tag{6}$$

et pour un poids $\lambda(j,k)$ on a :

$$\lambda c_j^1 + (1-\lambda)c_j^2 = \lambda c_k^1 + (1-\lambda)c_k^2 \tag{7}$$

Par transitivité, on obtient :

$$\lambda c_i^1 + (1-\lambda)c_i^2 = \lambda c_k^1 + (1-\lambda)c_k^2 \tag{8}$$

Cela équivaut à dire que : $\lambda(i,j) = \lambda(j,k) = \lambda(i,k)$.

Remarque 5.

(i) Ce résultat se généralise à des sous-séquences plus longues. Par exemple, pour une sous-séquence d'arêtes (i,j,k,l)*, on peut avoir* $\lambda(i,j) = \lambda(i,k) = \lambda(i,l) = \lambda(j,k) = \lambda(j,l) = \lambda(k,l)$.

(ii) On peut aussi avoir des sous-séquences indépendantes pour un même poids critique. Par exemple, on peut avoir $\lambda(i,j) = \lambda(k,l).t$

Tout comme on permute les sous-séquences d'arêtes indépendantes deux à deux, dans le cas où on a une sous-séquence qui possède 3 arêtes ou plus, le but sera d'inverser la séquence en appliquant une série de transpositions, et rechercher de nouvelles solutions en appliquant la proposition 4 à chaque transposition.

La question à se poser est donc s'il est possible, ou pas d'appliquer les permuations dans n'importe quel ordre.

Pour appliquer les résultats mentionnés dans la proposition 4, il faut que les arêtes soient consécutives dans la séquence. Par exemple, afin d'inverser une sous-séquence (i,j,k,l) avec le même poids critique, on effectuera les transpositions suivantes dans l'ordre :

$$(i,j) \to (i,k) \to (i,l) \to (j,k) \to (j,l) \to (k,l)$$

Ce qui donne :

$$(i,j,k,l) \to (j,i,k,l) \to (j,k,i,l) \to (j,k,l,i) \to (k,j,l,i) \to (k,l,j,i) \to (l,k,j,i)$$

On peut voir que la séquence S a été inversée après cette suite de transpositions d'arêtes consecutives.

Cependant, si on effectue les transpositions sur des arêtes non-consécutives, la sous-séquence résultante peut ne pas correspondre à l'inverse de la séquence initiale. Par exemple, pour la même sous-séquence (i,j,k,l), si on effectue les transpositions :

$$(i,j) \to (j,k) \to (k,l) \to (i,l) \to (i,k) \to (j,l)$$

Cela donne :

$$(i,j,k,l) \to (j,i,k,l) \to (k,i,j,l) \to (l,i,j,k) \to (i,l,j,k) \to (k,l,j,i) \to (k,j,l,i)$$

On peut voir qu'on obtient la sous-séquence (k,j,l,i), qui ne correspond pas à l'inverse de S.

4 Expérimentation numérique et résultats

4.1 Conception et mise en place de l'expérimentation

Afin de pouvoir évaluer l'efficacité de la nouvelle méthode proposée pour résoudre le problème bi-objectif d'arbre couvrant de poids minimum, il est nécessaire de la comparer à une méthode de référence. Dans cette étude, la méthode dichotomique a été choisie comme méthode de référence.

Nous avons testé les performances des deux méthodes sur une série de 10 instances, allant de 50 à 500 sommets. Ces instances ont été obtenues à l'aide d'un générateur de graphes programmé en language C, développé par Richard Johnsonbaugh et Martin Kalin du Department of Computer Science and Information Systems de l'université DePaul à Chicago, IL. Ce générateur nous garantit que les graphes soient connexes, et permet de choisir le coût maximum des arêtes. Nous avons alors fait le choix d'avoir des coûts allant de 1 à 10000. Par ailleurs, les deux objectifs ont des coûts générés indépendamment. En effet, les graphes générés ne possédant qu'un seul coût par arête, nous avons rajouté les coûts du second objectif pour les rendre adaptés au problème bi-objectif d'arbre couvrant de poids minimum.

Les deux méthodes ont été implémentées en utilisant le langage de programmation Julia dans sa version 1.8.1. Les tests ont été réalisés sur une machine équipée d'un CPU Intel® Core™ i7-9750H de 9ème génération (6 coeurs, 12 threads, avec une fréquence de 2.6 GHz).

Les résultats de l'évaluation comparative des deux méthodes sont présentés dans le tableau 2. Le nombre de solutions supportées non-extrêmes en plus qu'offre la nouvelle méthode par rapport à la méthode dichotomique est précisé entre parenthèses.

4.2 Résultats obtenus et comparaison avec la méthode dichotomique

Instances	Nb de sommets	Nb d'arêtes	Dichotomie		Nouvelle méthode	
			Nb Sol	CPUt(s)	Nb Sol	CPUt(s)
BiMST50.dat	50	122	33	0.016	33	0.004
BiMST100.dat	100	495	131	0.042	131	0.158
BiMST150.dat	150	1117	231	0.127	232(+1)	1.078
BiMST200.dat	200	1990	337	0.236	339(+2)	4.782
BiMST250.dat	250	3112	475	0.476	478(+3)	13.577
BiMST300.dat	300	4485	586	0.829	597(+11)	36.870
BiMST350.dat	350	6107	719	1.475	747(+28)	86.555
BiMST400.dat	400	7980	863	2.257	981(+118)	168.213
BiMST450.dat	450	10102	1014	3.131	1277(+263)	315.052
BiMST500.dat	500	12475	1163	4.266	1756(+593)	550.607

Tableau 2 – Résultats des tests sur les instances

Sur des instances relativement petites (de 50 et 100 sommets), on peut voir que la nouvelle méthode offre le même nombre de solutions supportées que la méthode dichotomique. Les temps de calculs quant à eux, sont également du même ordre de grandeur.

Par ailleurs, pour les instances allant de 150 à 500 sommets, on peut voir une évolution du nombre de solutions supportées non-extrêmes obtenues avec la nouvelle méthode. En effet, elle retourne un nombre supérieur de solutions par rapport à la méthode dichotomique, et plus l'instance est grande, plus cette différence est conséquente, jusqu'à obtenir +593 solutions supportées non-extrêmes pour l'instance à 500 sommets. Cependant, à cette augmentation s'accompagne une augmentation dans les temps de calculs, seul compromis de la méthode.

5 Conclusion

Ce travail de recherche a permis d'avoir une vue d'ensemble sur le problème d'arbre couvrant de poids minimum et ses propriétés fondamentales, en particulier dans un contexte bi-objectif. Il nous a également permis de revenir sur des notions élémentaires de la théorie des graphes et de l'optimisation multi-objectif, qui sont des domaines de recherche très importants en informatique.

En exploitant les propriétés structurelles du problème, nous avons proposé une nouvelle méthode qui repose sur l'utilisation de sommes pondérées. En explorant efficacement l'espace des solutions, cette méthode permet d'obtenir l'ensemble des solutions supportées extrêmes ainsi que certaines solutions supportées non-extrêmes.

Bien que la méthode proposée présente certaines limites, notamment au niveau des temps de calculs, les résultats obtenus restent très intéressants. En effet, nous avons pu observer que la méthode proposée est capae de fournir un nombre supérieur de solutions supportées non-extrêmes comparé à la méthode dichotomique.

En perspective de ce travail, un profilage et une révision du code réalisé sont prévus par la suite. Le but étant d'apporter des améliorations à ce dernier.

Pour conclure, je tiens tout particulièrement à remercier le professeur A. PRYZBYLSKI pour son soutien et sa présence durant la réalisation de ce travail.

Références

[1] Matthias Ehrgott. *Multicriteria optimization*. T. 491. Springer Science & Business Media, 2005.

[2] Joseph B. Kruskal, Laurent Beauguitte et Marion Maisonobe. *Joseph B. Kruskal, Jr., 1956, On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Version bilingue et commentée*. 1956. URL : https://hal.science/hal-03163666.

[3] Y. P. Aneja et K. P. K. Nair. « Bicriteria Transportation Problem ». In : *Management Science* 25.1 (1979), p. 73-78. ISSN : 00251909, 15265501. URL : http://www.jstor.org/stable/2630529.

[4] Stefan Ruzika et Horst W. Hamacher. « A Survey on Multiple Objective Minimum Spanning Tree Problems ». In : *Algorithmics of Large and Complex Networks : Design, Analysis, and Simulation*. Sous la dir. de Jürgen Lerner, Dorothea Wagner et Katharina A. Zweig. Berlin, Heidelberg : Springer Berlin Heidelberg, 2009, p. 104-116. ISBN : 978-3-642-02094-0. DOI : 10.1007/978-3-642-02094-0_6. URL : https://doi.org/10.1007/978-3-642-02094-0_6.