Solveurs multi-objectif

Eliot Braud et Elizabeth Gandibleux

Octobre 2024

Contents

Contexte: Quatre méthodes sont présentées dans ce rapport technique, accompagnées de leur visualisation sur quatre instances : sppw02.txt, sppw03.txt, sppnw19.txt et sppnw40.txt. Une conclusion finale est également fournie.

1 La méthode dichotomique

On introduit une méthode dichotomique basée sur l'algorithme Aneja and Nair, dans le but d'obtenir un meilleur générateur.

1.1 Étude comparative entre ϵ-contrainte et la dichotomie pour le calcul de générateurs

Points d'observations:

- Distribution globalement semblable entre ϵ-contrainte bidirectionnelle et dichotomie.
- Pourcentage très généralement moindre avec la dichotomie.
- Nombre de solutions admissibles trouvées avec la dichotomie $\geq$ ϵ-contrainte bidirectionnelle.
- Moins de temps avec la dichotomie, mais ϵ-contrainte peut être améliorable.

1.2 Visualisation

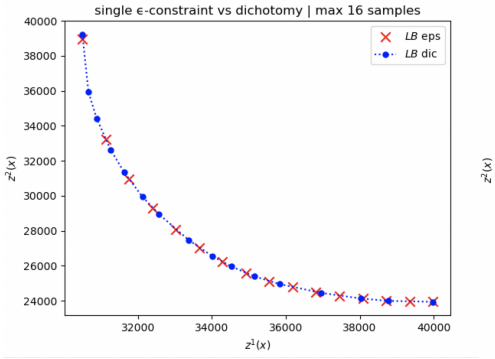

Figure 1: Plot de l'instance **sppw02.txt**

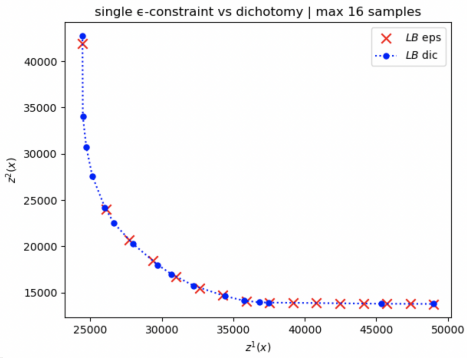

Figure 2: Plot de l'instance **sppw03.txt**

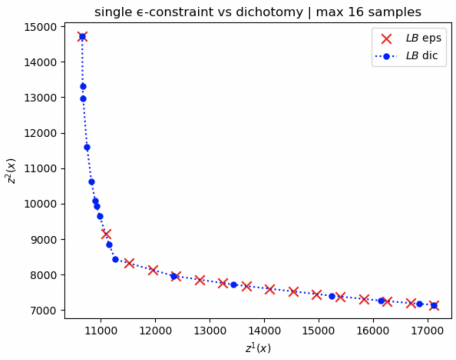

Figure 3: Plot de l'instance **sppnw19.txt**

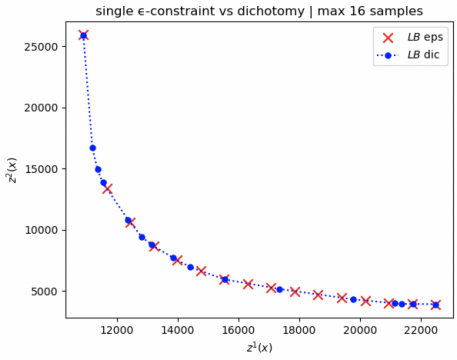

Figure 4: Plot de l'instance **sppnw40.txt**

1.3 Conclusion

En conclusion, l'étude comparative entre la méthode ϵ-contrainte et la méthode dichotomique a révélé des résultats globalement similaires en termes de distribution des solutions, avec toutefois un avantage pour la dichotomie. Celle-ci génère un nombre légèrement supérieur de solutions admissibles, mais offre des gains significatifs en temps de calcul. Bien que la méthode ϵ-contrainte puisse être optimisée, la méthode dichotomique semble plus efficace pour obtenir un générateur rapide et performant.

2 Les méthodes basées sur le destroy and repair

2.1 Un type de résolution

Dans cette section, nous allons présenter une adaptation de l'heuristique "Destroy and repair". Après avoir résolu le problème avec la méthode dichotomique, nous fixerons les variables dont la valeur est 0 à 0 et celles dont la valeur est 1 à 1. Ensuite, nous procéderons à une résolution exacte en ne traitant que les variables fractionnaires restantes.

2.1.1 Étude

Fixer la valeur de la majorité des variables rend le problème très petit et rapide à résoudre. De plus, en ayant une large composante en commun avec les solutions des générateurs, nous supposons que ces nouvelles solutions entières auront un fort potentiel pour être de bonne qualité.

2.1.2 Visualisation

Aucune solutions réalisables sont trouvées, il n'y a donc pas de nouvelles solutions, on obtient alors les mêmes plots que celles dans la section précédente.

2.1.3 Conclusion

Fixer à 0 toutes les variables qui ont une valeur de 0 dans le générateur lors de la recherche de solutions entières surcontraint le modèle et rend la résolution impossible.

2.2 Deux types de résolutions

2.2.1 Étude

Toujours dans la même veine, on part de la solution obtenue par la méthode de dichotomie, mais ici, l'objectif est de fixer les variables fractionnaires en variables binaires et de résoudre le problème avec un modèle d'optimisation en nombre entier mixte (MILP). Parallèlement, les variables déjà déterminées à 0 ou 1 dans le générateur sont résolues de manière continue.

Dans notre implémentation, nous vérifions d'abord la réalisabilité de la solution. Si elle est réalisable, nous calculons les valeurs objectives. Sinon, nous développons un nouveau modèle d'optimisation qui recherche des solutions conformes aux contraintes imposées, en utilisant une variable aléatoire λ pour enrichir l'exploration des solutions. Cela permet d'identifier de nouveaux points réalisables non détectés par la méthode de dichotomie ou celle de l'ϵ-contrainte.

2.2.2 Visualisation

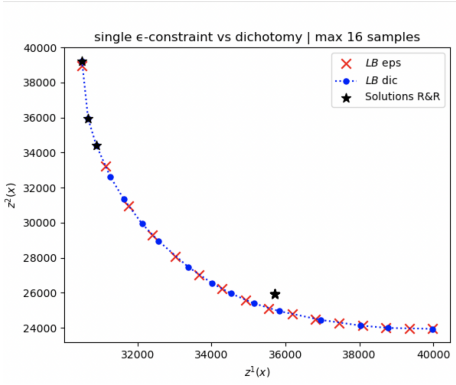

Figure 5: Plot de l'instance **sppw02.txt**

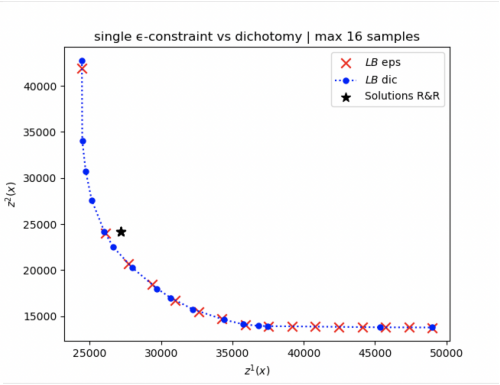

Figure 6: Plot de l'instance **sppw03.txt**

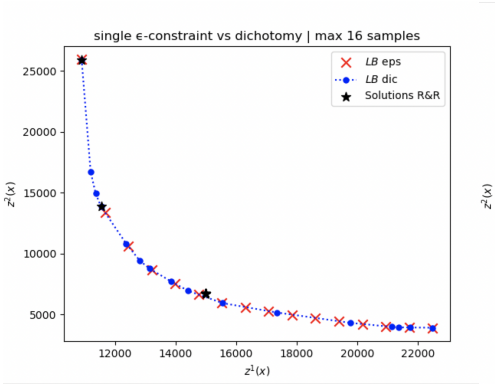

Figure 7: Plot de l'instance **sppnw19.txt**

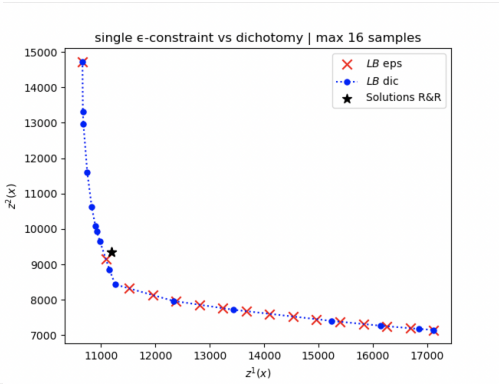

Figure 8: Plot de l'instance **sppnw40.txt**

2.2.3 Conclusion

La nouvelle méthode de résolution développée présente plusieurs avantages par rapport à l'approche précédente. En revisitant les solutions identifiées par la dichotomie, elle parvient également à découvrir de nouveaux points réalisables et optimaux. Cette capacité à explorer davantage l'espace de solutions est renforcée par l'introduction de la variable aléatoire λ, qui joue un rôle crucial dans la recherche de solutions. En conclusion, cette approche permet d'optimiser les performances tout en garantissant une meilleure qualité des solutions trouvées.

3 La méthode de la pénalité

3.1 Étude

En réponse à la conclusion du premier essai (de la section 2), nous ne fixerons plus les variables, mais nous pénaliserons leur changement de valeur. Les variables des générateurs, étant issues de la relaxation linéaire d'un problème 0-1 LP, peuvent prendre une valeur comprise entre 0 et 1. Plus précisément, nous découperons cet intervalle en trois ensembles qui nous intéressent : les variables avec une valeur de 0, celles avec une valeur de 1, et celles ayant des valeurs flottantes dans l'intervalle ouvert]0, 1[.

L'objectif est le suivant: en partant d'un générateur, nous allons résoudre le modèle en variable entière avec une fonction objectif visant à minimiser les différences avec le générateur initial. Pour ce faire, nous pénaliserons toute solution entière si l'une des variables y prend une valeur différente de celle qu'elle avait dans la solution du générateur.

- **Avantages:** Comme nous ne fixons plus les valeurs des variables, le modèle est beaucoup moins contraint et a donc une meilleure chance de trouver des solutions admissibles.

- **Inconvénients:** Bien que mono-objectif, le nouveau modèle contient presque le double de variables et un nombre non négligeable de contraintes supplémentaires, ce qui rend la résolution du problème plus lente.

3.2 Modèle avec pénalitées

Notations:

- $j \in J$, l'ensemble des variables;
- $i \in I$, l'ensemble des contraintes;
- $A_{ij}, \quad \forall i \in I, \forall j \in J$, la matrice des contraintes;
- c_{1j} et $c_{2j}, \quad \forall j \in J$, les valeurs des fonctions objectives;
- $j \in J^{(0)}$, l'ensemble des variables fixées à 0 dans le générateur;
- $j \in J^{(1)}$, l'ensemble des variables fixées à 1 dans le générateur.

Variables de décisions:

- $x_j, \quad \forall j \in J$, est égal à 1 si la ressource i est utiliser, sinon 0.
- $y_j, \quad \forall j \in J^{(0)} \cup J^{(1)}$, est égal à 1 si la variable d'indice j a une valeur différente de celle du générateur, sinon 0.

Modèle:

$$\min \textsc{penality}(y) \tag{1}$$

s.t.

$$\sum_{j \in J} x_j \cdot A_{ij} = 1, \quad \forall i \in I \tag{2}$$

$$x_j \leq y_i, \quad \forall j \in J^{(0)} \tag{3}$$

$$x_j \geq 1 - y_i, \quad \forall j \in J^{(1)} \tag{4}$$

Pour rester exaustif nous essayerons trois différentes fonctions objectives:

- **Pénalité unitaire:** Le changement de valeur d'une variable entraîne une pénalité de 1 dans la fonction objectif.

$$\textsc{penality}^{(\texttt{ones})}(y) = \sum_{j \in J} y_j$$

- **Pénalité pondérée:** La pénalité engendrée par le changement de valeur d'une variable dépend des valeurs objectives (pour les deux fonctions)

associées à cette variable.

$$\text{PENALITY}^{(\texttt{wsum})}(y) = \sum_{j \in J^{(0)} \cup J^{(1)}} y_j \cdot \frac{c_{1j} + c_{2j}}{\max_{k \in J}(c_{1k}) + \max_{k \in J}(c_{2k})}$$

Il est important de noter que, dans les expérimentations numériques actuelles, les deux objectifs sont du même ordre de grandeur. Par conséquent, nous n'avons pas pondéré un objectif plus qu'un autre, une décision qui pourrait être remise en question si l'un des objectifs avait un ordre de grandeur différent par exemple.

- **Pénalité sur l'intérêt de la variable de SPA:** La pénalité dépend de l'intérêt accordé à la variable par le SPA.

$$\text{PENALITY}^{(\texttt{wspa})}(y) = \sum_{j \in J} y_j \cdot \frac{c_{1j} + c_{2j}}{\sum_{i \in I} A_{ij}}$$

3.3 Visualisation

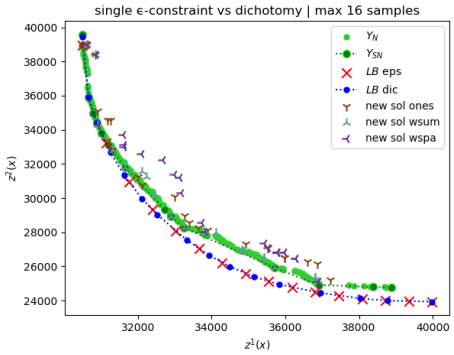

Figure 9: Instance **sppaa02.txt**

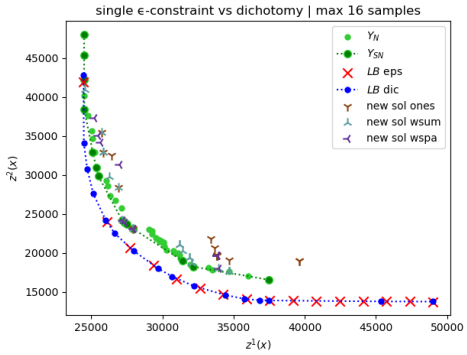

Figure 10: Instance **sppaa03.txt**

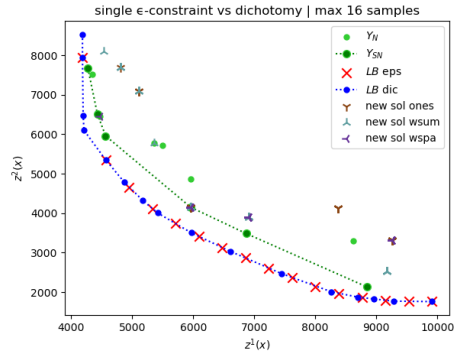

Figure 11: Instance **sppnw19.txt**

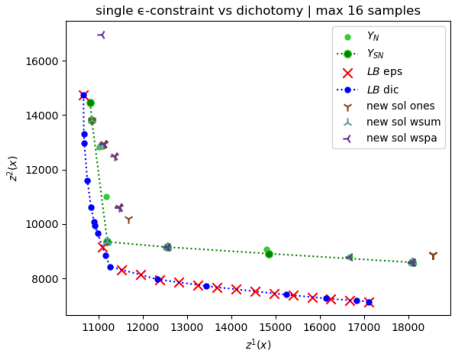

Figure 12: Instance **sppnw40.txt**

3.4 Conclusion

Cette méthode présente deux inconvénients majeurs:

- **Taille du modèle:** Un grand nombre de variables et de contraintes sont ajoutés au modèle, ce qui le rend significativement plus volumineux et allonge le temps de résolution.
- **Perte de la fonction objectif:** Dans le nouveau modèle, la fonction objective disparaît presque complètement. Cette perte d'information se manifeste par le fait que la solution optimale obtenue par le modèle est rarement la meilleure solution admissible, comparée à celles trouvées par le modèle avec les meilleures valeurs en termes de fonction objective.
- **Filtrage:** En réalité, toutes les solutions admissibles trouvées par le modèle sont filtrées, ce qui peut être fastidieux et exigeant en termes de temps de calcul.

Un point positif de cette méthode est que la résolution à l'aide d'un modèle permet de trouver un grand nombre de solutions admissibles (comme illustré dans la Figure 13), avec la possibilité d'obtenir également des solutions de bonne qualité.
Un second point important est que la solution optimale du modèle n'est pas nécessairement la meilleure solution au regard des fonctions objectives de départ. Pour palier à ce problème toutes les solutions admissibles trouvées par le modèle sont prisent en compte lors du filtrage des points dominés. Dans la Figure 14 les solutions optimales des modèles sont marqués par des étoiles et les solutions non efficaces par des trièdres.

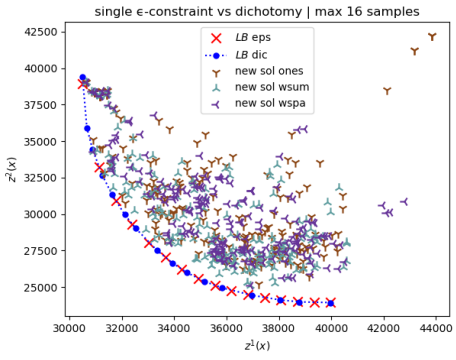

Figure 13: Instance **sppaa02.txt** sans filtrage des points dominées

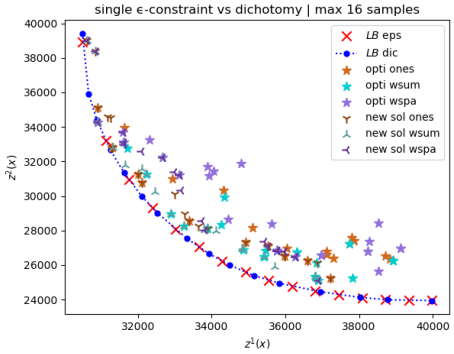

Figure 14: Instance **sppaa02.txt** optimum des modèles *vs* solution non dominées

4 Commandes

On se place dans un REPL Julia: pour lancer l'exécution, entrez la commande:

```
include("MOSmain.jl")
```

Dans les paramètres globaux on peut choisir laquelles des 3 méthodes on utilise pour la résolution:

```
global dichotomique      = true
global deux_resolutions  = false
global penalite_ponderee = false
```

Ici, on souhaite seulement avoir une résolution dichotomique.

5 Conclusion

Dans le cadre de notre étude sur la **Gravity Machine**, deux aspects ont été étudiés:

1. Tout d'abord, une méthode a été conçue pour obtenir un meilleur générateur de solutions, permettant ainsi d'explorer des solutions plus précises et plus diversifiées, que celles obtenues avec l' ϵ-contrainte.

2. Deux autres méthodes ont prouvé leur capacité à trouver des solutions entières optimales, en des temps raisonnables sur chacune des instances.

Ainsi, comme le montre notre graphique de comparaison en Figure 15, les résultats de notre approche dépassent ceux de **Gravity machine** en termes de performance et de qualité de solution, après de 188.61 secondes.

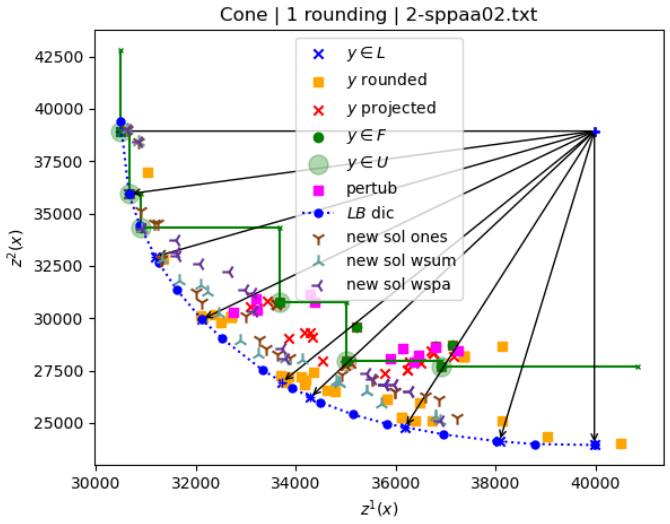

Figure 15: Comparaison de Gravity machine et de la méthode de pénalité pondérée sur **sppaa02.txt**

Perspectives d'Amélioration

1. Intégrer des approches de résolution hybrides pour combiner les forces des différentes méthodes et obtenir une meilleure précision dans les solutions entières optimales.

2. Utiliser la méthode de repair and repair en tant qu'heuristique de départ pour la méthode de la somme pondérée.

Ces améliorations pourraient renforcer l'efficacité de nos méthodes et les rendre encore plus compétitives face à des approches de gravity machine.