Metaheuristique
Compte rendu DM3
GA (Genetic algorithm) pour le Set Packing Problem (SPP)

MBACKE Baye Lahad

December 2, 2023

Université de Nantes — UFR Sciences et Techniques

Master informatique parcours Apprentissage et Traitement Automatique de la Langue (ATAL)

Année académique 2023-2024

1 Introduction

Le présent rapport vise à présenter notre implémentation de l'algorithme génétique (GA) comme métaheuristique pour résoudre le Set Packing Problem (Set Packing Problem, SPP). Nous décrirons en détail notre approche, mettant en lumière la façon dont nous avons adapté l'algorithme génétique spécifiquement pour le SPP

2 Description de l'algorithme

L'Algorithme Génétique (GA) est une métaheuristique inspirée par le processus de sélection naturelle. L'algorithme est initialisé avec une population de solutions aléatoires. Dans le contexte du Set Packing Problem (SPP), une solution est un vecteur binaire où chaque élément représente si un ensemble est inclus dans la solution ou non.

Pour un nombre donné de générations, l'algorithme effectue les étapes suivantes :

1. **Calcul de la fitness** : La fitness de chaque individu dans la population est calculée en utilisant la fonction `calculate_fitness`. Cette fonction évalue à quel point la solution satisfait les contraintes du SPP.

2. **Sélection** : Deux parents sont sélectionnés de la population en fonction de leur fitness. La fonction `rank_selection` est utilisée à cette fin, qui sélectionne les individus avec une probabilité plus élevée s'ils ont une fitness plus élevée.

3. **Crossover** : Une nouvelle solution (descendance) est créée en combinant les parents. La fonction `crossover` est utilisée à cette fin.

4. **Mutation et réparation** : La fonction `mutate_and_repair` est utilisée pour introduire une variation dans la descendance et pour s'assurer que la descendance satisfait toujours les contraintes du SPP.

5. **Remplacement** : L'ancienne population est remplacée par la nouvelle population.

Après que toutes les générations ont été traitées, l'algorithme retourne la meilleure solution trouvée.

L'algorithme est utilisé avec une taille de population de 500, un maximum de 100 générations et une probabilité de mutation de 0.01. Les données pour le SPP sont chargées à partir du fichier "Data/pb_100rnd0100.dat".

3 Expérimentation et Analyse des Résultats

Nous avons utilisé un algorithme génétique pour résoudre le le Set Packing Problem (SPP) représentés par ces instances. Les paramètres de l'algorithme varient selon les instances, avec des valeurs différentes pour la taille de la population, le nombre maximal de générations et la probabilité de mutation.

Les résultats obtenus pour chaque instance sont répertoriés dans le tableau, présentant la meilleure solution trouvée pour chacune d'entre elles.

Nous avons observé que pour les huit premières instances, une probabilité de mutation de 0,05 a été utilisée, tandis que pour les deux dernières, cette probabilité était de 0,01. Dans certains cas, une probabilité de mutation de 0,05 a généré des solutions de meilleure qualité que celle de 0,01, même lorsque la taille de la population et le nombre maximal de générations étaient identiques.

Notre analyse des résultats implique une comparaison des performances pour chaque instance en fonction des différents paramètres utilisés. Nous avons examiné comment la probabilité de mutation a affecté la qualité des solutions, si la taille de la population ou le nombre maximal de générations ont eu un impact significatif sur la convergence vers une solution optimale, et si certains paramètres se sont avérés plus efficaces pour certaines instances que pour d'autres.

Nous avons également exploré les raisons pour lesquelles une probabilité de mutation plus élevée a conduit à de meilleures solutions pour certaines instances, envisageant que cela puisse être attribuable à la nature spécifique de ces problèmes.

En résumé, notre analyse approfondie nous a permis de :

Comparer les performances pour chaque instance en fonction des différents paramètres utilisés. Évaluer l'impact de la probabilité de mutation, de la taille de la population et du nombre maximal de générations sur la qualité des solutions. Examiner les raisons pour lesquelles certaines instances ont été mieux résolues avec une probabilité de mutation plus élevée malgré des paramètres similaires. Ces investigations nous ont permis de mieux comprendre comment les paramètres de notre algorithme génétique influent sur la résolution des problèmes spécifiques représentés par ces instances.

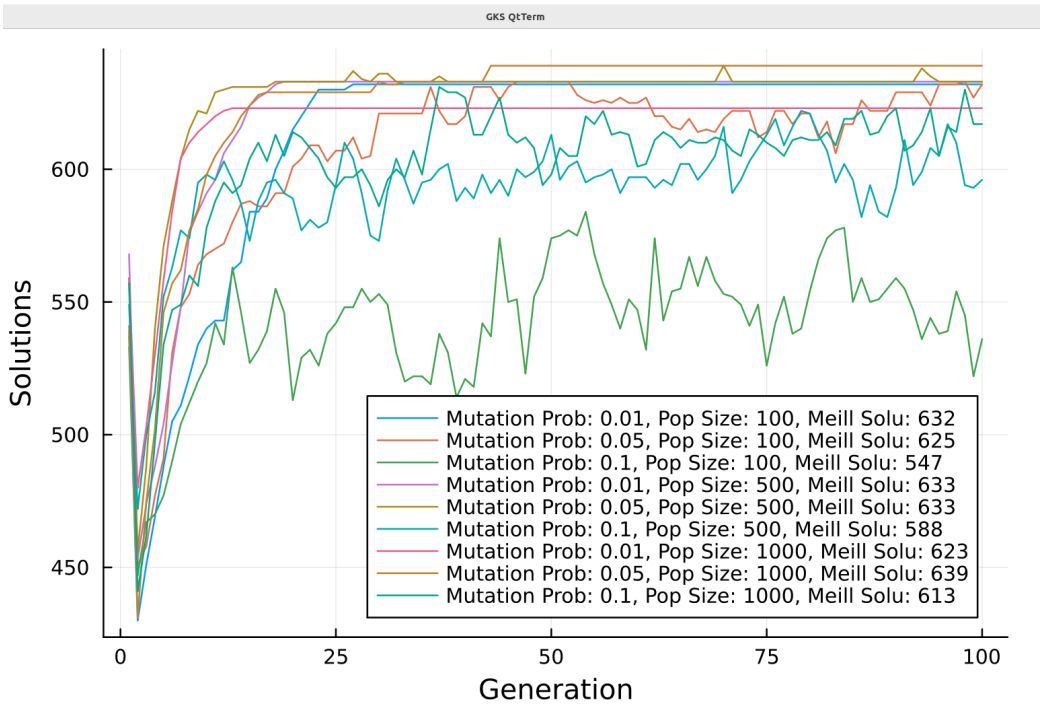

Figure 1: $\texttt{pb}_1 00 rnd 0500.dat$

4 Réglage des paramètres

L'algorithme génétique a été configuré en prenant en compte plusieurs paramètres pour explorer efficacement l'espace de recherche et obtenir des solutions de haute qualité pour le problème donné. Des expériences ont été menées en ajustant la probabilité de mutation et la taille de la population, avec des résultats notables sur des instances spécifiques.

4.1 Probabilité de Mutation

La probabilité de mutation, déterminant la fréquence à laquelle les gènes d'une solution sont modifiés pour maintenir la diversité génétique, a été évaluée avec des valeurs de 0.01, 0.05 et 0.1. Globalement, des valeurs inférieures (0.01 et 0.05) ont conduit à de meilleures solutions finales avec des fitness entre 623 et 639, par rapport à une valeur plus élevée de 0.1, produisant des solutions autour de 613 de fitness. Notamment, pour les instances

spécifiques `pb_500rnd0900.dat` et `pb_1000rnd0100.dat`, une probabilité de mutation plus faible (0.01 ou 0.05) a généré les solutions de meilleure qualité.

4.2 Taille de la Population

La taille de la population, représentant le nombre d'individus dans chaque génération, a été évaluée avec des valeurs de 100, 500 et 1000. Des populations plus grandes (500 et 1000) ont convergé vers des solutions légèrement meilleures, avec des fitness entre 633 et 639, par rapport à une population de taille 100, générant des solutions entre 547 et 632 de fitness.

4.3 Meilleurs Paramètres pour les Instances Spécifiques

Pour les instances `pb_500rnd0900.dat` et `pb_1000rnd0100.dat`, il a été observé que des valeurs plus basses de probabilité de mutation, telles que 0.01 ou 0.05, ont généré les solutions de meilleure qualité. Pour ces instances spécifiques, une combinaison de probabilité de mutation plus faible avec une taille de population de 500 ou 1000 semble offrir les meilleurs résultats, produisant des solutions plus performantes.

Il est important de souligner que ces résultats peuvent être spécifiques à ces instances de problème et que des expériences supplémentaires pourraient être nécessaires pour définir les paramètres optimaux dans d'autres contextes ou pour d'autres ensembles de données.

5 Comparaison de GRASP et de l'algorithme génétique

Instance	GRASP	Algorithme Génétique
	Zmax	Zmax
didactic.dat	30	30
pb_100rnd0100.dat	356	372
pb_100rnd0200.dat	30	34
pb_100rnd0500.dat	627	639
pb_100rnd1000.dat	37	39
pb_200rnd0100.dat	351	404
pb_200rnd1800.dat	16	18
pb_500rnd0100.dat	285	258
pb_500rnd0900.dat	2153	2195
pb_1000rnd0100.dat	49	51

Dans cette section, nous comparons les performances de GRASP et de l'algorithme génétique en utilisant plusieurs indicateurs pertinents.

5.1 Qualité de la solution

GRASP : Sur toutes les instances testées, GRASP n'a pas produit des solutions meilleures qu
GA : GA a généré des solutions légèrement meilleures que GRASP, par exemple avec `pb_`

5.2 Temps de calcul

GRASP : Nécessite une évaluation approfondie pour déterminer son temps de convergence et de résolution.

GA : Sa vitesse de convergence pourrait différer de GRASP. Une analyse comparative du temps de calcul est nécessaire.

5.3 Robustesse

GRASP : Sa stabilité sur différentes instances reste à évaluer, mais elle peut être moins sensible à certains paramètres.

GA : Peut présenter une robustesse supérieure sur un ensemble varié d'instances.

5.4 Complexité de l'implémentation

GRASP : Peut être plus simple à implémenter et ajuster en raison de sa structure heuristique plus directe.

GA : Son implémentation peut être plus complexe en raison de la gestion d'opérateurs génétiques et de paramètres spécifiques.

Cette comparaison s'est base les solutions obtenues par GRASP et l'algorithme génétique pour chaque instance du problème et aussi de la difficulté ou facilite de leur implementation. Ces résultats peuvent fournir des indications sur les performances relatives de chaque méthode dans des contextes spécifiques.

6 Conclusion

Notre étude montre que l'algorithme génétique est prometteur pour résoudre le Set Packing Problem (SPP), mais son efficacité dépend fortement des paramètres choisis. Le réglage précis de ces paramètres est crucial pour obtenir des solutions optimales.