Université Toulouse III - Paul Sabatier

Faculté Sciences et Ingénierie

Algorithmique Avancée

Compte rendu TP Algorithme Avancée

Problème de Chasse aux Trésors

par

LEGINYORA Iban

Année universitaire 2024 - 2025

Table des matières

Introduction

Ce problème de chasse aux trésors modélise un parcours optimal entre plusieurs points (villes), où l'objectif est de maximiser la valeur totale collectée tout en respectant une contrainte de budget en distance parcourue.

Proposition : L'algorithme implémenté dans ce code est une **recherche tabou** visant à résoudre un problème de collecte de trésors à travers différentes villes, avec un budget donné. L'objectif est de maximiser la valeur des trésors collectés tout en minimisant le coût du parcours, c'est-à-dire la distance parcourue. Ce problème relève de l'optimisation combinatoire.

Modèle en Programmation Linéeaire en Nombres Entiers (PLNE)

Le problème de la chasse aux trésors est modélisé sous forme d'un modèle en Programmation Linéaire en Nombres Entiers (PLNE). Pour supprimer les sous-cycles, deeux formulations différentes sont utilisées :

— `MTZ (Miller-Tucker-Zemlin)` : basée sur des variables continues pour les sous-cycles ;

— `DFJ (Dantzig-Fulkerson-Johnson)` : formulation avec des contraintes sur les sous-ensembles.

Le choix du modèle appliqué pour la gestion des sous-cycles se fait via une précision dans les paramètres suivants :

```
python3 chasse_tresors_plne.py <file_path> [-MTZ | -DFJ] [-display] (by default : -MTZ)
```

1.1 Présentation du modèle en PLNE

1.1.1 Notation :

— N : Nombre total de villes, y compris le point de départ (ville 0).

— B : Budget maximal en distance totale parcourue.

— $d_{i,j}$: Distance euclidienne entre les villes i et j, définie par :

$$d_{i,j} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

— v_k : Valeur associée à la ville k (butin collecté).

— $x_{i,j} \in \{0, 1\}$: Variable binaire indiquant si l'arc entre les villes i et j est emprunté.

— $y_k \in \{0, 1\}$: Variable binaire indiquant si la ville k est visitée.

— $u_k \in R$: Variable continue utilisée pour éliminer les sous-circuits avec la méthode `MTZ`.

1.1.2 Variables de décision :

— $x_{i,j} \in \{0, 1\}, \quad \forall i, j \in \{0, \ldots, N\}, i \neq j, d_{i,j} \leq B$: vaut 1 si l'arc entre les villes i et j existe.

— $y_k \in \{0, 1\}, \quad \forall k \in \{0, \ldots, N\}$: vaut 1 si la ville k est visitée.

— $u_k \in [0, N], \quad \forall k \in \{1, \ldots, N\}$: variables continues (`MTZ`) pour éliminer les sous-cycles.

— $villes\_indices \subseteq \{1, \ldots, N\}$: L'ensemble des indices des villes à visiter, et excluant le dépôt (ville 0). Est utile pour la méthode DFJ.

1.1.3 Contraintes :

1. Conservation des flux (visite unique des villes). Chaque ville est visitée au plus une fois, ce qui implique qu'il y a un arc d'entrée et un arc de sortie entre deux villes visitée communicantes :

$$\sum_{\substack{j=1 \\ j\neq i,\ (i,j)\in x}}^{N} x_{i,j} = y_i, \quad \forall i \in \{1,\dots,N\}$$

$$\sum_{\substack{j=1 \\ j\neq i,\ (j,i)\in x}}^{N} x_{j,i} = y_i, \quad \forall i \in \{1,\dots,N\}$$

2. Point de départ et retour au point 0. Le parcours commence et finit au point de départ (ville 0) :

$$\sum_{j=1}^{N} x_{0,j} = 1$$

$$\sum_{j=1}^{N} x_{j,0} = 1$$

3. Contrainte de budget sur la distance totale. La distance totale parcourue ne doit pas dépasser le budget B :

$$\sum_{(i,j)\in x} x_{i,j} \cdot d_{i,j} \leq B$$

4. Élimination des sous-circuits (`MTZ`). Pour éviter les sous-circuits, on utilise les variables u_k avec les contraintes suivantes :

$$u_i - u_j + (N-1) \cdot x_{i,j} \leq N-2, \quad \forall i,j \in \{1,\dots,N\},\ i \neq j$$

5. Élimination des sous-circuits (DFJ). Pour empêcher la formation de sous-cycles dans les villes visitées :

$$\sum_{\substack{i,j\in Q \\ i\neq j,\ (i,j)\in x}} x_{i,j} \leq |Q| - 1, \quad \forall Q \subseteq \{1,\dots,N\},\ |Q| \geq 2$$

où :

— Q : un sous-ensemble de villes.

— $|Q|$: taille de l'ensemble Q.

— $villes\_indices$: ensemble des indices des villes.

1.1.4 Objectif :

L'objectif maximise la somme des valeurs des villes visitées tout en respectant la distance maximale B.

$$\max \sum_{k=1}^{N} \mathsf{v}_k \cdot y_k$$

1.2 Implémentation et résolution

Le modèle PLNE est implémenté en Python avec le solveur **SCIP**. Deux formulations (`MTZ` et `DFJ`) sont paramétrables, et le coût appliqué entre les villes est représenté par la distance euclidienne.

Métaheuristique

L'algorithme utilisée dans le cadre de l'implémentation du problème en métaheuristique est une recherche tabou. Celui-ci repose sur plusieurs fonctions qui sont implémentées dans le fichier `chasse_tresors_mh.py` avec 3 paramètres à préciser : l'instance à tester, le nombre d'itérations maximum et la taille de la liste tabou :

```
python3 chasse_tresors_mh.py <file_path> <max_iteration> <size_tabu> [-display]
```

2.1 Structures de données utilisées

- `solution` : liste ordonnée des villes visitées ;
- `liste_tabou` : mémoire des dernières opérations interdites pour éviter de revenir en arrière.
- `villes` : liste contenant les valeurs de chaque ville `(id, x, y, butin)`.

2.2 Méthodes implémentées

2.2.1 Calcul de la solution initiale

La solution initiale est générée à l'aide d'un algorithme `greedy` qui est glouton. La méthode consiste à choisir une ville aléatoirement parmi les villes restantes et teste toutes les positions possibles pour insérer cette ville dans la solution, en vérifiant que le coût total ne dépasse pas le budget B. Ce processus continue jusqu'à ce que toutes les villes aient été insérées dans la solution.

Cette méthode est implémentée dans la fonction `greedy(villes, B, N)` qui retourne un vecteur de solution.

2.2.2 Exploration du voisinage

Une fois la solution initiale obtenue, l'algorithme explore le voisinage en appliquant plusieurs opérations pour générer de nouvelles solutions voisines.

Les opérations possibles sont les suivantes :

— **Ajouter un point (**`ajouter_point(solution, villes, B, N, liste_tabou)`**)** : Cette opération consiste à ajouter une ville à la solution en cherchant la position qui minimise le coût total, tout en respectant le budget B. Si la ville a déjà été visitée, elle est ignorée.

— **Supprimer un point (**`supprimer_point(solution, villes, B, N, liste_tabou)`**)** : Il s'agit de supprimer une ville de la solution, à condition que celle-ci ne fasse pas déjà partie de la liste tabou. Les paramètres `villes`, `B` et `N` sont présents uniquement pour correspondre à la signature de l'appel (ligne 160).

— **Échanger un point (**`echanger_point(solution, villes, B, N, liste_tabou)`**)** : Cette opération consiste à échanger une ville visitée contre une autre ville non visitée, en vérifiant que le coût total de la nouvelle solution reste inférieur au budget B.

La fonction `voisinage(solution, villes, B, liste_tabou)` applique ces trois opérations dans un ordre aléatoire et renvoie une nouvelle solution voisine différente de la solution actuelle.

2.2.3 Recherche locale avec la liste tabou

La recherche tabou explore des solutions voisines en utilisant la fonction `voisinage()`. Chaque nouvelle solution générée est comparée à la meilleure solution trouvée jusqu'à présent, et si elle est meilleure (en termes de butin des trésors collectés), elle devient la nouvelle meilleure solution.

— **Liste tabou :** Les villes récemment ajoutées ou supprimées de la solution sont ajoutées à une liste tabou. Cette liste empêche l'algorithme de revenir à des solutions récemment explorées. La taille de cette liste est définie par l'argument `taille_tabou`. Si la taille de la liste dépasse cette limite, l'élément le plus ancien est retiré.

— **Critères d'arrêt :** L'algorithme s'arrête après un nombre d'itérations spécifié par l'argument `max_iterations`.

Comparaison des approches

Nous comparons les deux approches sur plusieurs instances. Les critères d'évaluation incluent :

— La valeur de la meilleure solution trouvée ;

— Le temps de calcul ;

— Le nombre de nœuds explorés pour le PLNE ou le nombre d'itérations pour la MH.

Voici la liste exhaustive des résultats obtenus sur l'ensemble des instances.

3.1 Résultats sur des instances

Comparaison des Résultats : PLNE vs MH

				Modèle PLNE									Modèle MH					
Fichier	Nb Villes	Budget	Type	Modèle	Solution	Valeur	Distance	Temps (s)	Nodes	P. Bound	D. Bound	Gap	Solution	Valeur	Distance	Temps (s)	Max. itération	Taille liste Tabou
cat5_150	5	150	Optimal	MTZ	0-1-0	2	129	0.01					0-1-0	2	128	6.58	1	0
cat5_160	5	160	Optimal	MTZ	0-3-0	3	157	0.01					0-3-0	3	156	3.934	1	1
cat5_175	5	175	Optimal	MTZ	0-3-5-0	10	167	0.03					0-3-5-0	10	166	0.0004	100	5
cat5_180	5	180	Optimal	MTZ	0-3-2-4-5-0	28	180	0.13					0-5-4-2-3-0	28	179	0.0018	100	16
cat5_200	5	200	Optimal	MTZ	0-5-4-2-3-1-0	30	197	0.01					0-1-3-2-4-5-0	30	196	0.0004	100	1
cat5_500	5	500	Optimal	MTZ	0-2-1-3-4-5-0	30	256	0.01					0-1-3-2-4-5-0	30	196	0.0005	100	1
rc201-chasse-tresor-14-25	12	170	Optimal	DFJ	0-17-15-18-16-14-25-23-21-20-22-19-24-0	240	165	0.66					0-24-22-19-20-21-23-25-15-18-17-16-14-0	240	156	0.005	500	10
rc201-chasse-tresor-2-13	12	170	Optimal	DFJ	0-12-13-8-7-5-3-11-10-0	220	169	12.05					0-12-13-8-7-5-3-11-10-0	220	169	0.003	500	10
rc201-chasse-tresor-2-50	12	170	Non-optimal	MTZ				200.00	8418	200	636.298	218.15%	0-26-24-22-[...]-18-17-16-14-0	400	161	0.015	500	10
rc201-chasse-tresor_1000	100	1000	Non-optimal	MTZ				200.00	2278	142	172.000	21.41%	0-76-98-60-[...]-26-78-59-0	1724	687	0.338	500	10
rc201-chasse-tresor_150	100	150	Non-optimal	MTZ				200.00	1485	160	818.049	5012.81%	0-60-10-14-[...]-22-24-78-0	347	149	0.106	1000	30
rc201-chasse-tresor_170	100	170	Non-optimal	MTZ				200.00	1506	160	883.827	5423.92%	0-76-98-88-[...]-24-26-78-0	417	167	0.115	1000	30
rc201-chasse-tresor_500	100	500	Non-optimal	MTZ				216.00	1815	200	162.876	8043.78%	0-98-10-14-[...]-26-78-59-0	1084	500	0.122	500	10

Table 3.1 – Synthèse des résultats

Dans la suite de cette partie de comparaison des approches, une étude plus approfondie de deux instances est détaillée.

3.2 Étude de l'instance : cat5_175

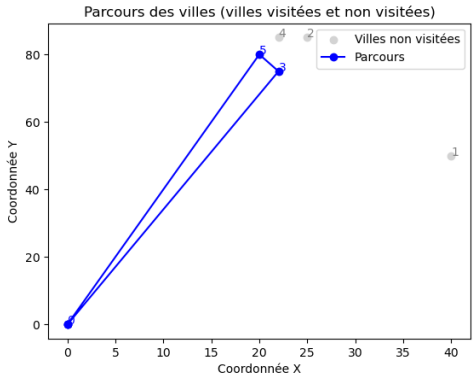
Résultats obtenus

Modèle	Solution	Valeur	Distance	Temps
MTZ	0-3-5-0	10	167	0.03

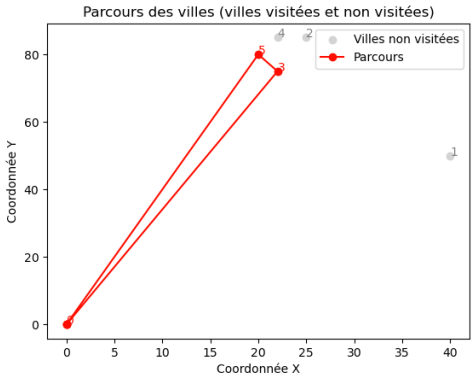
Table 3.2 – Résultats obtenus en PLNE

Solution	Valeur	Distance	Temps	Max Iteration	Taille liste Tabou
0-3-5-0	10	166	0.0004	100	5

Table 3.3 – Résultats obtenus en MH



(a) Parcours obtenu avec la méthode PLNE



(b) Parcours obtenu avec la méthode MH

Voici une analyse des résultats obtenus :

Points communs

1. **Structure de la solution** : Les deux approches aboutissent à une solution comportant un parcours similaire (*0-3-5-0*) avec une valeur optimale de 10. Cela montre que les deux méthodes fournissent des solutions comparables en termes de qualité pour cette instance.
2. **Convergence vers une solution satisfaisante** : Les deux approches trouvent une solution respectant les contraintes imposées par le problème, malgré leurs différences méthodologiques.

Différences

1. **Distance obtenue** : Une légère différence est constatée dans les distances calculées : 167 pour PLNE contre 166 pour MH. Mais cela est lié au fait que les résultats ont été arrondis.
2. **Temps de résolution** : MH est nettement plus rapide (0.0004 unités) que PLNE (0.03 unités). Cela illustre l'efficacité de la méthode MH, adaptée aux problèmes de grande taille.
3. **Stratégies d'optimisation** : MH utilise un mécanisme itératif avec une liste tabou de taille 5 et un maximum de 100 itérations, alors que PLNE exploite une résolution exacte (modèle MTZ), ce qui peut expliquer les écarts en termes de performance.
4. **Convergence et écart (Gap)** : PLNE affiche un écart de 0.03, qui signifie qu'il y a une légère sous-optimalité par rapport à la borne duale. En revanche, MH converge rapidement vers une solution réalisable.

3.3 Étude de l'instance : rc201-chasse-tresor-14-25

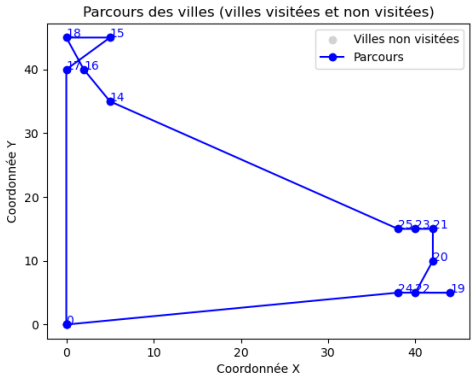
Résultats obtenus

Modèle	Solution	Valeur	Distance	Temps
DFJ	0-17-15-18-16-14-25-23-21-20-22-19-24-0	240	165	0.66

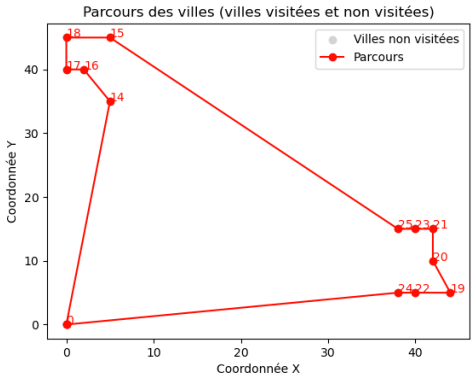
Table 3.4 – Résultats obtenus en PLNE

Solution	Valeur	Distance	Temps	Max Iteration	Taille liste Tabou
0-24-22-19-20-21-23-25-15-18-17-16-14-0	240	156	0.005	500	10

Table 3.5 – Résultats obtenus en MH



(a) Parcours obtenu avec la méthode PLNE



(b) Parcours obtenu avec la méthode MH

Voici une analyse des résultats obtenus :

Points communs

1. **Qualité de la solution** : Les deux approches obtiennent une valeur de solution optimale égale à 240, montrant que les deux méthodes convergent vers des solutions de haute qualité.
2. **Respect des contraintes** : Les solutions respectent le budget imposé (170 unités) et fournissent des parcours réalisables.

Différences

1. **Distance totale parcourue** : Une différence notable est observée dans les distances totales : 165 pour PLNE contre 156 pour MH. Cela montre que MH a trouvé un parcours légèrement plus efficace en termes de distance.
2. **Temps de résolution** : La méthode MH est beaucoup plus rapide (0.005 unités) que PLNE (0.66 unités), grâce à son approche itérative et approximative.
3. **Stratégies d'optimisation** : MH utilise un algorithme tabou avec une liste de taille 10 et un maximum de 500 itérations. En revanche, PLNE repose sur une méthode exacte avec le modèle DFJ, ce qui explique un temps de calcul plus important.

Conclusion

Ce rapport a permis d'étudier deux approches pour résoudre un problème d'optimisation combinatoire lié à la chasse aux trésors : le **modèle en Programmation Linéaire en Nombres Entiers (PLNE)** et une **métaheuristique basée sur la recherche tabou (MH)**.

Pour rappel, PLNE offre une garantie d'optimisation théorique grâce à des méthodes exactes comme MTZ et DFJ et elle convient aux problèmes nécessitant une solution précise, mais au prix d'un temps de calcul élevé, particulièrement pour des instances complexes.

Quant à la recherche Tabou (MH), elle fournit des solutions proches de l'optimalité en un temps réduit. et est adaptée aux problèmes de grande taille ou à des contraintes complexes, avec une flexibilité grâce à son approche itérative.

Pour comparaison, la méthode PLNE est très bonne en précision mais demande des ressources importantes. En revanche la métaheuristique est plus rapide tout en obtenant de très bonnes solutions, bien qu'elle n'atteint pas toujours l'optimalité théorique.

Donc pour les cas où une solution exacte est cruciale, PLNE est préférable. Les métaheuristiques peuvent être utilisée lorsqu'il y a une demande de besoins d'efficacité ou de temps de calcul.