# Prosit 4 : pas de déjeuner gratuit

## Mots Clés :

* Métaheuristique :
* **Hill Climbing :** Un algorithme d'optimisation qui commence avec une solution arbitraire et essaie d'améliorer cette solution par des modifications incrémentielles. L'objectif est de toujours minimiser la fonction de coût .
* **Heuristique Glouton :** Une méthode qui construit une solution en prenant une série de choix locaux optimaux dans l'espoir que cette stratégie mènera à une solution globale optimale.
* **Etape d’intensification :** l'algorithme se concentre sur l'exploration approfondie d'une région prometteuse de l'espace de recherche pour trouver une meilleure solution
* **Test statistique :** Une méthode pour déterminer si les résultats obtenus par un algorithme ou une expérience sont significatifs ou dus au hasard.
* **Etape de diversification :** Une phase dans une métaheuristique visant à explorer de nouvelles régions de l'espace de recherche pour éviter les optima locaux et favoriser la découverte de solutions globalement optimales.
* **Optimum local :** Une solution qui est meilleure que toutes les solutions voisines immédiates, mais qui n'est pas nécessairement la meilleure solution globale possible.
* Modélisation du voisinage :
* Métaheuristique :
* **Trajectoire :** Algorithmes qui suivent une seule solution à la fois, comme le recuit simulé ou l'algorithme du tabou.
* **Population :** Algorithmes qui maintiennent et évoluent un ensemble de solutions simultanément, comme les algorithmes génétiques ou les colonies de fourmis.
* **Algorithme génétique :** Une métaheuristique inspirée des processus de sélection naturelle et de génétique. Il utilise des opérations comme la sélection, le croisement et la mutation pour évoluer un ensemble de solutions.
* **Méthode par construction :** Une approche qui construit une solution à partir de rien en ajoutant progressivement des éléments, en général en suivant une heuristique gloutonne.
* **Méthode de recherche perturbative :** Une méthode qui améliore une solution existante en apportant des modifications (perturbations) pour échapper aux optima locaux.
* **Méthode de recherche locale :** Une technique qui améliore une solution en explorant son voisinage immédiat pour trouver des solutions meilleures.
* **Algo du tabou :** Une méthode de recherche locale qui utilise une mémoire pour éviter de revisiter les solutions récemment explorées, ce qui permet d'éviter les cycles et d'échapper aux optima locaux.
* **Recuit simulé :** Une métaheuristique inspirée du processus de refroidissement des métaux. Elle accepte temporairement des solutions moins bonnes pour éviter de rester coincé dans des optima locaux, avec une probabilité qui diminue au fil du temps.
* **GRASP (Greedy Randomized Adaptive Search Procedure) :** Une méthode itérative qui combine des heuristiques gloutonnes et des recherches locales. Chaque itération commence par la construction d'une solution gloutonne aléatoire, suivie d'une phase de recherche locale.
* **VRP géométrique (Vehicle Routing Problem) :** Un problème de recherche opérationnelle où l'objectif est de trouver les chemins optimaux pour une flotte de véhicules qui doivent desservir un ensemble de clients avec certaines contraintes.
* **Colonie de fourmi :** Une métaheuristique inspirée du comportement des fourmis cherchant de la nourriture. Les fourmis virtuelles déposent des phéromones sur les chemins qu'elles parcourent, influençant ainsi les choix des autres fourmis et convergeant vers des solutions optimales.
* **rugosité du voisinage :** Une mesure de la difficulté d'explorer le voisinage d'une solution en termes de variation des valeurs de la fonction objectif. Un voisinage rugueux a de grandes variations, rendant plus difficile la recherche d'une solution optimale.
* **Paramètre de génération :** Les paramètres qui régissent la génération des solutions initiales ou des nouvelles solutions dans un algorithme, comme la taille de la population initiale dans un algorithme génétique.
* **Plan d’expérience :** Une méthodologie statistique pour planifier, mener et analyser des expériences de manière à optimiser les ressources et obtenir des conclusions valides sur les facteurs influençant les résultats

## Contexte :

On a abandonné la programmation linéaire nous essayons de résoudre le problème à l’aide des métaheuristiques. Nous recherchons une méthode métaheuristique en vue de résoudre le problème.

## Problématique :

Comment concevoir une méthode de résolution à l’aide des métaheuristiques pour le problème du TSP ?E

## Contraintes :

* Résolution rapide

## Généralisation :

* Algorithme métaheuristique
* Plan d’expérimentation

## Livrable :

* Choix de la méthode métaheuristique
* Programme de résolution du problème
* Plan d’expérimentation

## Pistes de solution :

* Trouver le ratio précision – temps d’exécution en faisant varier la solution
* Déterminer un critère de précision
* Comparer les solutions entre elles
* Calculer borne inférieur et supérieur (utilisation solveur), pire et meilleur solution optimal

## Plan d’actions:

1. Lister les métaheuristiques (description, différence, …)
2. Choisir la plus adapter (génétique, fourmi, glouton)
3. Justifier le choix (source, biblio, …)
4. Implémenter la méthode
5. Plan d’expérimentation(comportement de l’algorithme, test , évaluation de la méthode, qualité et temps d’exécution).
6. Génération d’instance

## Réalisation :

Métaheuristique cités précédemment dans les mots clés :

**Algorithme génétique :**

Principe : Inspiré par la sélection naturelle et la génétique.

Mécanismes : Utilise des opérations de sélection, croisement et mutation pour évoluer une population de solutions.

Recherche : Explore plusieurs solutions simultanément.

Avantages : Bon pour explorer de grands espaces de recherche, s’adapte bien aux problèmes complexes.

Inconvénients : Peut nécessiter un réglage fin des paramètres (taux de mutation, taille de la population).

**Algorithme du tabou :**

Principe : Utilise une mémoire pour éviter de revisiter des solutions déjà explorées.

Mécanismes : Maintient une liste taboue pour stocker les solutions récemment visitées, évite les cycles.

Recherche : Trajectoire unique, suit une seule solution

Avantages : Efficace pour échapper aux optima locaux, évite les boucles et redondances.

Inconvénients : Peut être sensible à la longueur de la liste taboue, risque de nécessiter un ajustement des paramètres.

**Recuit simulé :**

Principe : Inspiré par le processus de refroidissement des métaux (recuit).

Mécanismes : Accepte temporairement des solutions de moindre qualité avec une probabilité qui diminue au fil du temps.

Recherche : Trajectoire unique, suit une seule solution tout en explorant de nouvelles.

Avantages : Bon pour échapper aux optima locaux, simple à implémenter.

Inconvénients : Nécessite des paramètres de refroidissement appropriés, peut être lent si mal configuré.

**GRASP (Greedy Randomized Adaptive Search Procedure):**

Principe : Combine des heuristiques gloutonnes et des recherches locales.

Mécanismes : Construit des solutions gloutonnes aléatoires puis applique une recherche locale pour l'amélioration.

Recherche : Itérative, génère et améliore plusieurs solutions successives.

Avantages : Simple et flexible.

Inconvénients : La qualité des solutions peut varier, dépend de la qualité des heuristiques gloutonnes utilisées.

**Colonie de fourmi :**

Principe : Inspiré par le comportement des fourmis cherchant de la nourriture.

Mécanismes : Les fourmis artificielles déposent des phéromones sur les chemins, influençant les décisions futures.

Recherche : Population, plusieurs fourmis explorent l’espace simultanément.

Avantages : Efficace pour les problèmes de graphes et de routage, adaptable et parallèle.

Inconvénients : Peut nécessiter un réglage fin des paramètres, peut converger prématurément.

### Rappel des contraintes :

* Problème du voyageur possédant de multiple camion
* Multiple point de collecte et de livraisons.
* Chaque camion possède ses propres colis.

### Choix de la solution :

L’algorithme qui répondrait le mieux à nos contraintes, serait le Ant Colony Optimization, il permet de gérer notre contrainte de ramassage et de livraison mais également sur la taille des données d’entrée avec une variation du nombre de camion ou encore du nombre de villes.

Les résultats indiquent que l'ACO aboutit à des solutions de qualité en termes de rapidité et de précision, en comparaison à un algorithme simplex écrit en C (CPLEX) étant considéré comme exact, affirmant ainsi son utilité dans des scénarios réalistes et à grande échelle et surtout dans des cas pratiques. (ref: Table 1. Result summaries of different instances.)

Dans l’article, en plus de prendre en compte la fenêtre de temps, les multiples points de collecte ou de livraison sont pris en compte.

Source : <https://www.mdpi.com/2305-6290/5/2/28>

Ant Colony Optimization for Multiple Pickup and Multiple Delivery Vehicle Routing Problem with Time Window and Heterogeneous Fleets

L’algorithme propose par l’article est le suivant :

