École Nationale Supérieure d'Informatique

People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

École Nationale Supérieure d'Informatique (ESI)

Département d'Informatique

2ème année, Spécialité SIQ Groupe 02

Optimisation Combinatoire

Métaheuristique : Recherche Locale

Réalisé par :

- Bayadh Hadjer
- Rahal Nour Elhouda
- Medfouni Khitem
- Nakib Ibtihel
- Yazi Lynda Mellissa
- Benmachiche Khaled

Encadré par :
Amine KECHID

Année Académique
2024 – 2025

Table des matières

1. Introduction

Dans le domaine de l'optimisation combinatoire, la recherche locale représente une catégorie d'approches heuristiques permettant de trouver des solutions de qualité acceptable pour des problèmes complexes, souvent NP-difficiles, où les méthodes exactes sont trop coûteuses en termes de temps de calcul. Ces techniques, basées sur l'exploration de solutions voisines d'une solution donnée, sont particulièrement adaptées aux problèmes de grande taille, offrant un compromis entre qualité de la solution et efficacité de calcul.

Les principales méthodes de recherche locale incluent :

— Le Recuit Simulé (Simulated Annealing)
— La Recherche Tabou (Tabu Search)
— La Recherche Voisinage Variable (VNS - Variable Neighborhood Search)

Ces méthodes d'optimisation, adaptées à différents types de problèmes, sont largement utilisées dans les domaines de la planification, de la logistique, du routage ou encore de l'affectation de ressources, où la rapidité d'exécution est cruciale tout en garantissant des solutions suffisamment bonnes.

2. Modélisation générale du problème de coloration

Le problème de coloration de graphe consiste à attribuer une couleur à chaque sommet dun graphe de manière à ce que deux sommets adjacents (reliés par une arête) naient pas la même couleur. Lobjectif est de minimiser le nombre total de couleurs utilisées, autrement dit, trouver le nombre chromatique du graphe.

2.1. Définition formelle

Soit un graphe non orienté $G = (V, E)$, où :

— V est lensemble des sommets,
— $E \subseteq V \times V$ est lensemble des arêtes.

Une fonction de coloration est une application :

$$c : V \to \{1, 2, \ldots, k\}$$

telle que :

$$\forall (u, v) \in E,\ c(u) \neq c(v)$$

3. Approche par recherche locale

Deux grandes approches peuvent être adoptées pour modéliser ce problème dans le cadre d'une résolution par heuristiques :

3.1. Approche 1 : Minimisation des conflits avec k fixé

Cette première approche suppose un nombre de couleurs k fixé à l'avance. L'objectif est de minimiser le nombre de conflits, cest-à-dire le nombre de paires de sommets adjacents partageant la même couleur.

Fonction objectif

La fonction objectif dans ce cas est la suivante :

$$f(c) = \sum_{(u,v)\in E} \delta(c(u), c(v))$$

où

$$\delta(a,b) = \begin{cases} 1 & \text{si } a = b \\ 0 & \text{sinon} \end{cases}$$

Cette fonction pénalise chaque conflit, cest-à-dire chaque arête reliant deux sommets ayant la même couleur.

Stratégie d'amélioration

On applique une heuristique locale pour minimiser $f(c)$, et lorsque le nombre de conflits atteint zéro, on peut essayer de diminuer la valeur de k pour trouver une solution valide avec moins de couleurs. Ce processus est répété jusquà ce quaucune solution valide ne soit trouvée pour une valeur inférieure de k.

3.2. Approche 2 : Intégration directe du nombre de couleurs dans la fonction objectif

Contrairement à lapproche précédente, cette méthode nimpose pas un nombre de couleurs fixe. Elle combine directement deux objectifs dans la fonction à minimiser :

— Le nombre de conflits (validité de la solution),
— Le nombre total de couleurs utilisées (objectif global de minimisation).

Fonction objectif composite

On définit une fonction objectif pondérée :

$$f(c) = \alpha \cdot k + \beta \cdot \sum_{(u,v)\in E} \delta(c(u), c(v))$$

où :

— α et β sont des coefficients pondérant respectivement limportance du nombre de couleurs et des conflits.
— δ est définie comme précédemment.

3.3. Comparaison des deux approches

— **Approche 1** : Simple, facile à contrôler, efficace lorsquon cherche uniquement à valider une solution pour un k donné. Toutefois, nécessite plusieurs exécutions avec différentes valeurs de k pour approcher le nombre chromatique.
— **Approche 2** : Plus flexible et directe, car elle cherche à minimiser k dès le départ tout en garantissant la validité via une fonction composée. Adaptée aux contextes où léquilibre entre qualité et optimalité est essentiel.

4. Heuristiques initiales

La qualité de la solution initiale joue un rôle déterminant dans la performance des algorithmes de recherche locale. Une bonne solution de départ permet non seulement daccélérer la convergence, mais aussi dorienter la recherche vers des zones prometteuses de lespace des solutions, réduisant ainsi le risque de stagnation dans des minima locaux peu intéressants.

Dans le contexte de la coloration de graphe, plusieurs heuristiques classiques sont utilisées pour construire cette solution initiale :

Greedy

Lapproche greedy colore les sommets un à un, en attribuant à chaque sommet la plus petite couleur disponible qui respecte les contraintes de coloration vis-à-vis de ses voisins déjà traités. Bien que cette méthode soit très rapide et simple à implémenter, elle tend à produire des solutions sous-optimales, notamment lorsque lordre de parcours des sommets est défavorable.

Welsh-Powell

Cette variante améliore lapproche greedy en réordonnant les sommets par degré décroissant (nombre de voisins). L'idée est de colorier d'abord les sommets les plus contraints, dans lespoir de mieux répartir les couleurs sur le reste du graphe. Cette stratégie permet souvent dutiliser un nombre de couleurs inférieur à celui obtenu par lapproche greedy naïve.

DSatur (Degree of Saturation)

DSatur est lune des heuristiques les plus efficaces pour générer des solutions initiales de haute qualité. Elle repose sur le degré de saturation : à chaque étape, on sélectionne le sommet ayant le plus grand nombre de couleurs différentes déjà utilisées par ses voisins. En cas dégalité, on choisit le sommet de plus haut degré. Cette méthode privilégie les zones du graphe les plus contraintes, ce qui permet en général de produire une solution initiale très compétitive, idéale pour servir de point de départ à des méthodes doptimisation plus avancées.

En résumé, bien que les heuristiques simples comme Greedy puissent suffire pour des graphes peu complexes, des méthodes plus sophistiquées comme DSatur offrent un meilleur compromis

entre coût de calcul et qualité de la solution initiale, ce qui se traduit souvent par des gains significatifs en performance lors des étapes doptimisation ultérieures.

5. Implémentation des algorithmes

Les heuristiques de recherche locale constituent des méthodes puissantes pour améliorer des solutions initiales dans des problèmes NP-difficiles tels que la coloration de graphe. Leur efficacité repose sur l'exploration intelligente de l'espace des solutions à l'aide de mécanismes de diversification et d'intensification. Nous présentons ici trois approches couramment utilisées : le recuit simulé, la recherche tabou et la recherche à voisinage variable (VNS).

5.1. Recuit simulé (Simulated Annealing)

Inspiré du processus physique de recuit des métaux, le recuit simulé est un algorithme probabiliste permettant déviter les minima locaux en acceptant temporairement des solutions de moindre qualité.

Ce metaheuristique a été implémenté en suivant les deux approches décrites précédemment : une version en deux phases (minimisation des conflits à k fixé) et une version intégrée (optimisation pondérée conjointe du nombre de couleurs et des conflits).

Paramètres principaux :

- Température initiale T_0
- Taux de refroidissement $\alpha \in (0, 1)$
- Critère darrêt : température minimale $T_{\min}$ ou nombre maximal ditérations

Mécanisme : À chaque itération, une solution voisine c' est générée à partir de la solution courante c. La nouvelle solution est acceptée si elle améliore la fonction objectif, ou bien avec une probabilité décroissante :

$$P = \exp\left(-\frac{f(c') - f(c)}{T}\right)$$

La température est ensuite réduite progressivement :

$$T \leftarrow \alpha T$$

5.2. Recherche tabou (Tabu Search)

La recherche tabou repose sur une stratégie déterministe guidée par une mémoire à court terme (liste tabou), permettant déviter les cycles et dexplorer efficacement lespace des solutions.

Cette méthode a également été implémentée en suivant les deux variantes précédemment introduites : lapproche en deux phases et lapproche intégrée basée sur une fonction objectif pondérée.

Éléments fondamentaux :

- **Liste tabou :** enregistre les derniers mouvements pour éviter les retours en arrière immédiats.
- **Critère daspiration :** permet de passer outre les interdictions si un mouvement mène à une meilleure solution globale.
- **Diversification :** introduit de la variété dans les trajectoires de recherche.

5.3. Recherche à Voisinage Variable (VNS)

La recherche à voisinage variable (VNS) repose sur lidée que lefficacité de la recherche dépend de la diversité des structures de voisinage explorées. L'implémentation de la VNS a été réalisée dans le cadre d'une version en deux phases (minimisation des conflits à k fixé), dans laquelle l'objectif est de minimiser les conflits tout en respectant une contrainte sur k.

Principe :

- Définir une séquence de voisinages $N_1, N_2, ..., N_k$
- Appliquer une recherche locale dans N_i, puis passer au suivant si aucune amélioration nest trouvée.
- En cas damélioration, revenir au premier voisinage.

Exemples de voisinages :

- N_1 : changement de couleur dun sommet
- N_2 : permutation de couleurs entre deux sommets
- N_3 : recoloriage dun sous-ensemble de sommets

Cette approche favorise une exploration adaptative, ajustant dynamiquement la granularité des mouvements pour sortir des minima locaux.

6. Optimisation des paramètres

L'optimisation des paramètres est essentielle pour maximiser l'efficacité des heuristiques de recherche locale, notamment pour des problèmes complexes comme la coloration de graphes. Un bon réglage des hyperparamètres améliore la convergence et la qualité des solutions. Plusieurs approches peuvent être utilisées pour optimiser ces paramètres :

- **Recherche exhaustive (Grid Search)**
  La recherche par grille consiste à évaluer systématiquement toutes les combinaisons dun ensemble discret de valeurs pour chaque hyperparamètre.

  Cette méthode garantit la couverture de lespace de recherche défini, mais devient rapidement inefficace lorsque le nombre de paramètres ou la granularité des valeurs augmente. Elle est donc plus adaptée à de petits espaces de recherche.
- **Optimisation bayésienne avec *Optuna***
  `Optuna` est une bibliothèque moderne doptimisation dhyperparamètres, fondée sur des

techniques bayésiennes et de recherche adaptative. Contrairement aux méthodes de recherche exhaustive, elle construit un modèle probabiliste de la fonction objectif afin de concentrer les évaluations dans les régions les plus prometteuses de lespace des paramètres.

Exemple doptimisation sur le graphe `dsjc250.5`

- **Graphe** : 250 sommets, 15668 arêtes
- **Objectif** : minimiser le nombre de conflits dans une solution obtenue par recuit simulé
- **Meilleur essai trouvé** :
  - Nombre de conflits : **61.0** pour k = 28
  - Paramètres optimaux :
    - `initial_temp` : 4.6739
    - `cooling_rate` : 0.9989
    - `min_temp` : 0.00051
    - `max_iterations` : 20000

Visualisation des résultats

`Optuna` fournit également des outils de visualisation utiles pour lanalyse du processus doptimisation :

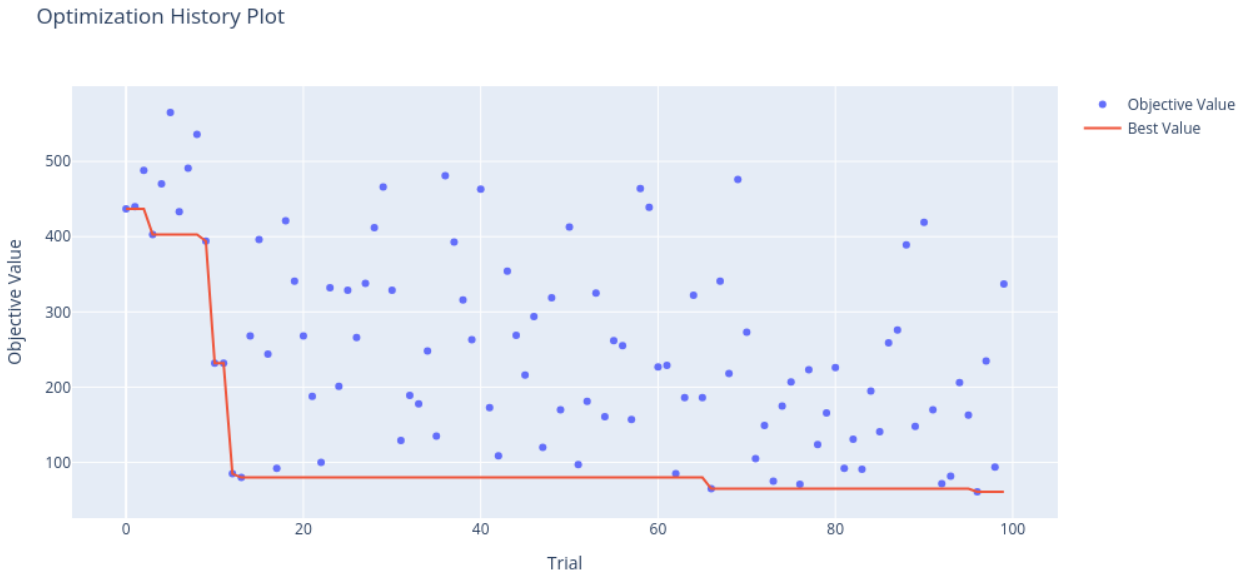

FIGURE 1 – Historique de l'optimisation

7. Résultats expérimentaux

Dans cette section, nous présentons les résultats expérimentaux obtenus en appliquant les différentes méthodes de recherche locale méta-heuristique sur plusieurs graphes types. Nous évaluons les performances des algorithmes en termes de temps d'exécution, nombre de conflits, et valeur de k atteinte, où k est le nombre de couleurs nécessaires pour colorier le graphe.

7.1. Tableau des performances

Les performances des différents algorithmes sont mesurées pour plusieurs graphes types. Nous avons utilisé des graphes représentatifs pour évaluer les capacités de chaque algorithme. Les performances sont comparées sur la base des trois critères principaux : temps d'exécution, nombre de conflits, et valeur de k, obtenue après l'optimisation de la coloration.

Le tableau ci-dessous présente les résultats de l'application de différentes heuristiques sur 2 graphes distincts :

Graphe	**Méthode**	**K trouvé**	**K optimal**	**Écart (%)**	**Conflits**	**Temps (s)**
dsjc250.5	SA1	37	28	32.14	0	16.41
	Tabu1	34	28	21.42	0	318.65
	VNS1	37	28	32.14	0	321.96
	SA2	37	28	32.14	0	12.86
	Tabu2	32	28	14.29	0	234.4
r250.5	SA1	68	65	1.54	0	20.58
	Tabu1	68	65	1.54	0	363.29
	VNS1	68	65	1.54	0	309.57
	SA2	68	65	1.54	0	10.82
	Tabu2	66	65	1.48	0	319.57

TABLE 1 – Comparaison des performances des variantes SA, Tabu et VNS selon les approches 1 et 2 sur différents graphes

7.2. Analyse des résultats

Les résultats présentés dans le tableau 1 montrent des différences notables entre les performances des différentes heuristiques appliquées aux graphes `dsjc250.5` et `r250.5`.

Comparaison de la valeur de k trouvée

Pour le graphe `dsjc250.5`, les variantes Tabu ont produit les valeurs de k les plus proches de loptimal, avec un minimum de 32 couleurs pour Tabu2, soit un écart de seulement 14.29%. En revanche, les autres méthodes (SA et VNS) ont convergé vers des valeurs plus éloignées du k optimal, toutes à 37 couleurs, soit un écart de 32.14%.

Concernant le graphe `r250.5`, toutes les méthodes SA, Tabu et VNS ont trouvé des solutions proches de loptimal (65 couleurs), avec un maximum de 68 couleurs et un écart maximal de 1.54%. Cela indique que ce graphe est potentiellement moins difficile à colorier de manière optimale ou que les heuristiques convergent plus facilement.

Temps d'exécution

On remarque que les approches Tabu sont les plus coûteuses en temps dexécution. Par exemple, Tabu1 prend plus de 5 minutes pour le graphe `dsjc250.5`, contre environ 16 secondes pour SA1. Cette différence est également observée pour le graphe `r250.5`, avec Tabu1 à 363.29 secondes contre seulement 10.82 secondes pour SA2.

Discussion

Les performances des algorithmes varient en fonction de la nature du graphe et de la configuration de lalgorithme.

- **Méthodes Tabu :** bien qu'elles soient performantes en termes de qualité de solution, elles souffrent de temps d'exécution élevés. Cela peut être attribué à la gestion complexe des mouvements interdits et à une exploration plus profonde de lespace de recherche.
- **Méthodes SA :** elles offrent un bon compromis entre qualité et temps. Notamment SA2 qui maintient un temps très faible tout en obtenant des résultats comparables aux autres variantes.
- **Méthode VNS :** les résultats sont similaires à ceux de SA pour certains graphes, mais le coût en temps est souvent plus élevé. Elle reste intéressante pour des scénarios où des améliorations marginales sont critiques.

Le choix de lheuristique dépend donc du contexte : pour un besoin rapide, SA2 serait privilégié. Pour une solution optimale, Tabu2 peut être préféré malgré le coût temporel.

8. Conclusion

Cette étude a mis en évidence lefficacité des méta-heuristiques de recherche locale pour aborder des problèmes combinatoires complexes de manière robuste, flexible et efficace. Les différentes approches mises en uvre ont toutes permis dobtenir des solutions valides, ce qui confirme la pertinence de ces techniques dans le traitement de problèmes NP-difficiles.

Au-delà des résultats obtenus, plusieurs pistes damélioration se dégagent pour renforcer encore les performances des méthodes explorées. Lhybridation des méta-heuristiques permettrait de combiner les forces de différentes approches pour mieux sadapter à la diversité des cas rencontrés. Lutilisation dhyper-heuristiques, capables de sélectionner dynamiquement les meilleures stratégies, ouvrirait la voie à des algorithmes plus autonomes et adaptatifs. Lintégration de lapprentissage automatique offrirait des moyens de guider plus intelligemment la recherche ou dajuster automatiquement les paramètres. Enfin, la parallélisation des algorithmes pourrait considérablement réduire les temps de calcul et améliorer leur scalabilité.

Ces perspectives dessinent un cadre prometteur pour le développement de solutions toujours plus efficaces, intelligentes et généralisables à une large variété de situations concrètes.