N° d'ordre : ....

**UNIVERSITÉ MOHAMMED SEDDIK BENYAHIA**
**JIJEL**
**FACULTÉ DE SCIENCES EXACTES ET D'INFORMATIQUE**

**MEMOIRE DE MASTER**

Pour l'obtention du diplôme de :

**MASTER**

En **INFORMATIQUE**

**Option :** Systeme d'Informattique et Aide a la Decision

Par :

**-BECHKIT Rabah**

**-BOUTANA Halim**

Thème

# Résolution heuristique du probléme de tournées de techniciens (STRSP)

Soutenue publiquement, le 12/09/2022, devant le jury composé de :

| | | | |
|---|---|---|---|
| Mr | Later Azzedine | à l'université de jijel | Président |
| Mr | Kerroum Ali | à l'université de jijel | Encadrant |
| Mme | Kerada Ouidad | à l'université de jijel | Examinatrice |

**Promotion 2022**

# *Dedicaces*

*Je dédie ce travail*

*A ma très chère mère Hassiba, Quoi que je fasse ou que je dise, je ne saurai point te remercier comme il se doit. Ton affection me couvre, ta bienveillance me guide et ta présence à mes côtés a toujours été ma source de force pour affronter les différentes obstacles.*

*A mon très cher père Mohamed, merci pour les valeurs nobles, l'éducation et le soutient. Aucune dédicace ne saurait exprimer l'amour, l'estime et le respect que j'ai toujours pour toi. Merci d'être toujours là pour moi.*

*A mon frère Bilel puisse Dieu vous donne santé, bonheur, courage et surtout réussite.*

*A mes sœurs Amina et Amel, je vous souhaite du fond de mon cœur une belle vie pleine de joie, de bonheur et d'amour.*

*A toute ma famille et mes chers amis pour leur soutien tout au long de mon parcours universitaire.*

*A tout qui m'aiment et que j'aime.*

***Rabah***

# *Dedicaces*

*Je dédie ce travail*

*A la mémoire de mon père, que je n'oublierai jamais.*

*A ma mère, dont l'amour et les sacrifices n'ont pas cessé de combler ma vie, que dieu la protège et la garde pour moi, quelle trouve ici l'expression de ma profonde gratitude. Je n'arriverai jamais à t'exprimer mon amour sincère et ma reconnaissance, pour tout ce que tu as fait pour moi.*

*A mes frères et sœurs, qui n'ont pas cessé de me conseiller, de m'encourager et de me soutenir. Que dieu vous offre la chance et le bonheur et surtout la santé.*

*A tous mes neveux et nièces. Je souhaite du fond de mon cœur que Dieu vous protège et vous donne joie et réussite.*

*A toute ma famille.*

*A tous mes amis et collègues de l'université de Jijel.*

*A tous ceux qui m'ont encouragé et ont participé à la réalisation de ce travail .*

*A tous ceux qui me sont chers.*

***Halim***

# *Remerciements*

*Tout d'abord nous remercions* ***ALLAH****, le tout puissant qui a illuminé notre chemin et qui nous a accordé le courage et la patience pour accomplir ce travail.*

*Nous tenons à remercier infiniment notre encadrant* ***Mr : Kerroum Ali****, pour sa disponibilité, son suivi et ses conseils judicieux qui nous ont été très utiles tout le long de la réalisation de ce projet. Permettez nous Monsieur de vous exprimer notre reconnaissance et notre gratitude.*

*Nous remercions également le président et les membres du jury qui nous ont honorés en acceptant de juger notre travail.*

*Nous adressons nos vifs remerciements à nos enseignants du département d'informatique qui nous ont encadré durant notre cursus universitaire.*

*Nous remercions toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.*

*Nous tenons à remercier profondément nos familles pour leurs conseils et encouragements durant toutes les années de nos études.*

# Résumé

Le problème de tournées des techniciens STRSP (Service Technician Routing Scheduling problem ) est un problème courant des entreprises qui doivent être en mesure fournir des services à un grand nombre de clients. Avec leurs ressources limitées, ces entreprises doivent essayer de fournir des services rapides, abordables et fiables pour maximiser la satisfaction des clients.Le Problème STRSP implique l'acheminement des techniciens en réponse aux demandes de service, en tenant compte des fenêtres de temps et des compétences. L'utilisation des méthodes d'optimisation pour résoudre le problème STRSP améliore les décisions d'affectation, de planification et d'acheminement, tout en obtenant une satisfaction client élevée.

Pour résoudre le problème STRSP, nous avons proposé et implémenté deux algorithmes basée sur l'heuristique de recherche adaptative à large voisinage (ALNS). Les résultats d'expérimentations menées sur les instances de la littérature montrent que les deux algorithmes proposés sont compétitifs avec les méthodes proposées dans la littérature pour résoudre le problème STRSP.Plusieurs nouvelles meilleures solutions sont trouvées.

***Mots-clés*** *: problème de tournées de techniciens STRSP, heuristique, Recherche adaptative à large voisinage*

# Abstract

The Service Technician Routing Scheduling problem (STRSP) is a common problem for companies that must be able to provide services to a large number of customers. With limited resources, these companies must try to provide fast, affordable and reliable services to maximize customer satisfaction.The STRSP Problem involves dispatching technicians in response to service requests, taking into account time windows and skill sets. Using optimization methods to solve the STRSP problem improves assignment, scheduling, and routing decisions, while achieving high customer satisfaction.

To solve the STRSP problem, we proposed and implemented two algorithms based on the Adaptive Large Neighborhood Search (ALNS) heuristic. The results of experiments carried out on the instances of the literature show that the two proposed algorithms are competitive with the methods proposed in the literature to solve the STRSP problem. Several new best solutions are found.

***Keywords :*** *Service Technician Routing Scheduling problem, heuristic, Adaptative Large neighborhood search*

# TABLE DES MATIÈRES

# TABLE DES FIGURES

# LIST OF ALGORITHMS

# LISTE DES TABLEAUX

# LISTE DES ACRONYMES

| | |
|---|---|
| **STRSP** | Service Technician Routing and Scheduling Problems |
| **VRP** | Vehicle Routing Problems |
| **LNS** | Large Neighborhood Search |
| **ILS** | Iterated Local Search |
| **ALNS** | Adaptive Large Neighborhood Search |
| **AG** | Algorithme Génétique |
| **SA** | Simulated Annealing |

# INTRODUCTION GÉNÉRALE

Actuellement, les prestations de service auprès des clients, jouent un rôle important dans la vie des sociétés modernes. Plusieurs entreprises ont investi massivement dans la recherche scientifique, afin d'actualiser et de développer le service client et d'améliorer la qualité de l'entreprise aux yeux du client. Offrir ces services peut toutefois, d'un côté séduire les clients et les fidéliser et d'un autre côté cela engendre des coûts importants. C'est ainsi, que les problèmes de tournées des techniciens sont apparus afin de fidéliser les clients en leur offrant un service efficace et rapide tout en minimisant les couts engendrés [1]. Ils sont rencontrés dans différents domaines (domaine des télécommunications, domaine médical, service après-vente...etc.)

Les problèmes de tournées des techniciens font partie de la classe des problèmes de tournées de véhicules (VRPs : Vehicle Routing Problems) et s'apparentent en particulier aux VRPs avec fenêtres de temps. Ils ont cependant, leurs caractéristiques propres comme les compétences des techniciens pour les différentes tâches. Chaque problème de tournées des techniciens, rencontré en pratique, possède des caractéristiques spécifiques et les algorithmes développés, pour la résolution de ces problèmes, doivent considérer, les paramètres en rapport avec l'objectif : Le coût, le temps et la distance, ainsi que les contraintes comme, les temps de service, les fenêtres de temps, et les compétences des techniciens,...etc.

Dans ce mémoire on s'intéresse à un problème particulier de tournées des techniciens appelé STRSP (Service Technician Routing and Scheduling Problems). Dans un problème STRSP, Chaque technicien de service se spécialise souvent dans différents domaines de compétence et possède des compétences à différents niveaux. Chaque tâche nécessite un technicien ayant les compétences appropriées au moins aux niveaux requis. Pour résoudre le problème STRSP, les entreprises doivent être en mesure de prendre simultanément trois types de décisions : l'affectation des tâches à l'ensemble des techniciens, l'acheminement de chaque technicien et la planification des différentes tâches. L'objectif principal de notre travail est de résoudre le problème STRSP, dans le but de réduire le coût total des tournées, en développant des algorithmes heuristiques, en raison de la complexité de ce problème.

Notre mémoire est composé de trois chapitres :

Le premier chapitre présente le problème de tournée des techniciens STRSP, son modèle mathématique ainsi que ses domaines d'application. Dans la dernière partie, nous décrivons quelques méthodes de résolution de ce type de problèmes.

Le deuxième chapitre présente l'approche utilisée pour la résolution de ce problème STRSP, en se basant sur deux versions de l'heuristique ALNS et en expliquant les différentes étapes de tous les algorithmes utilisés.

Le dernier chapitre de ce mémoire présente les outils logiciels et matériels, utilisés pour la résolution du problème de tournée des techniciens sans équipe, les interfaces et les résultats obtenus, ainsi que des comparaisons avec d'autres travaux antérieurs.

Enfin, nous terminons par une conclusion générale et les perspectives proposées.

# CHAPITRE 1

# PROBLÈME DE TOURNÉES DES TECHNICIENS

## 1.1 Introduction

Les problèmes de tournées de techniciens (STRSPs :Service Technician Routing and Scheduling Problems) sont apparus afin de fidéliser les clients en leur offrant un service de maintenance efficace et rapide tout en minimisant les couts engendrés. Appartenant à la classe des problèmes d'optimisation combinatoire NP-difficile [1], le principe général de ce problème consiste, en optimisant le coût, à construire un ensemble de tournées, pour un nombre fini de Techniciens , commençant et finissant à un dépôt.

Dans ce chapitre, nous présentons une définition générale du problème STRSP . Par la suite, nous présentons le modèle mathématique, ses domaines d'application ainsi que quelques méthodes de résolution des problèmes d'optimisation existantes.

## 1.2 Définition du problème STRSP

Un problème STRSP consiste à planifier les routes d'un groupe de techniciens afin de servir les requêtes des clients localisées dans différents endroits à moindre coût.

Tous les techniciens proviennent du centre de dépôt pour accomplir les tâches qui leur sont assignées et qui reviennent au dépôt une fois tous les travaux terminés. Un technicien a des compétences multiples avec différents niveaux à respecter pendant l'horizon de planification.

Une tâche de service a un temps de traitement connu, une fenêtre de temps spécifiée par le client dans lequel le service doit commencer, une durée limite du service, et ses exigences en matière de compétences. Et dans le cas où le technicien arrive à la tâche avant l'heure indiquée, il doit attendre l'heure du début de la tâche.

Une tâche est attribuée à au plus un technicien qui possède les compétences requises, mais en respectant la disponibilité du technicien c'est à dire la durée totale de sa tournée ne peux être dépassée.

Dans le cas où les techniciens ne sont pas en mesure de réaliser une tâche, il est possible de l'externaliser avec un prix coûteux, effectuer la tâche par un sous-traitant.

Pour résoudre un problème STRSP, les entreprises doivent être en mesure de prendre simultanément trois types de décisions :

- L'affectation des tâches à un technicien dans le dépôt.
- L'acheminement de chaque technicien, c'est-à-dire la séquence / l'ordre des tâches qu'un technicien doit visiter une fois.
- La planification des tâches (temps de début et de fin de chaque tâche).

L'objectif est de minimiser le coût opérationnel total qui comprend le coût du routage et le coût d'externalisation.

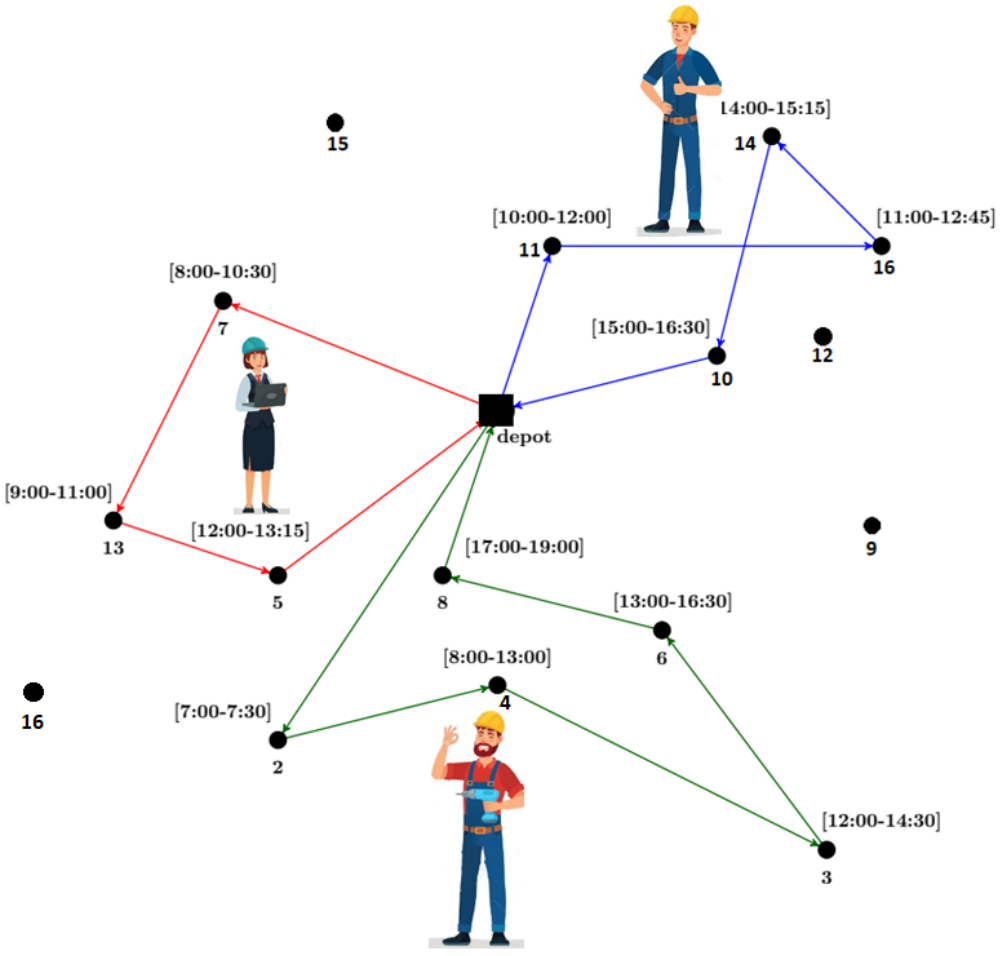



FIGURE 1.1 – Un exemple d'un STRSP

La Figure 1.1 montre un exemple du problème STRSP avec **un dépôt unique, 16 tâches et trois itinéraires de techniciens.**

**Dans les routes 1, 2 et 3 :** Les techniciens possèdent toutes les compétences requises pour effectuer toutes les tâches.

Pour les tâches 9, 12, 15 et 16 les techniciens sont incapables de les réaliser, il faut donc externaliser ces tâches à un coût élevé.

## 1.3 Modèle mathématique du problème STRSP

Dans cette section, nous fournissons d'abord une description formelle du STRSP que nous abordons. Nous formulons ensuite, un modèle de programmation mixte en nombres entiers (MIP) pour notre problème.

Le STRSP est défini sur un graphe complet $G = \{V, A\}$, où $V = \{0, 1, .., n+1\}$ est un ensemble de sommets et $A = \{(i, j) : i, j \in V, i \neq j\}$ est un ensemble d'arcs. Le sommet 0 désigne le dépôt et le sommet $n+1$ est une copie du dépôt, et $C = V \setminus \{0, n+1\}$ représente l'ensemble des sommets qui ont chacun une tâche unique. Selon le contexte, on se réfère à une tâche $i$ ou à un sommet $i$ pour tout $i \in C$. Un ensemble $K$ de techniciens est disponible pour effectuer les tâches. Chaque technicien est spécialisé dans un certain nombre de domaines de compétences à différents niveaux de compétences. A chaque tâche $i \in C$ est associée une durée de service $d_i$ , une fenêtre temporelle $[e_i, l_i]$ dans laquelle le service doit commencer, et un besoin en compétences. Le dépôt et sa copie ont également des fenêtres temporelles, qui définissent l'heure de départ au plus tôt $e_0$ et le dernier temps de retour $l_{n+1}$ de n'importe quel technicien. Aussi, la durée de trajet de chaque technicien ne doit pas dépasser un temps donné $D$. A chaque arc $(i, j) \in A$ est associé un coût $C_{ij}$ et un temps de parcours $t_{ij}$ .

Pour déterminer si un technicien serait en mesure de réaliser une tâche donnée, nous définissons un paramètre binaire $q_i^k$ , où $q_i^k = 1$ si le technicien $k \in K$est qualifié pour effectuer la tâche $i \in C$, et $q_i^k = 0$ sinon. Les valeurs de $q_i^k$ peuvent être facilement calculées en fonction des compétences des techniciens et des compétences requises pour les tâches. Enfin, toute tâche $i \in C$ peut être externalisée en engageant un coût $o_i$ , dans le cas où les ressources sont insuffisantes ou trop chères pour entreprendre toutes les tâches.

Le STRSP peut être formulé comme un modèle de programmation mixte en nombres entiers qui contient les variables binaires suivantes [21] :

$$x_{ij}^k = \begin{cases} 1 & \text{Si arc } (i,j) \text{ est parcouru par le technicien } k, \quad \forall (i,j) \in A, k \in K \\ 0 & \text{Sinon} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{Si la tâche } i \text{ est externalisée,} \\ 0 & \text{Sinon} \end{cases} \qquad \forall i \in C$$

Et la variable continue $b_i^k$ , $\forall i \in V - \{0, n+1\}, k \in K$, represente le temps de début du service de la tâche $i$.

Si $i = 0$ et $i = n+1$, $b_i^k$ represente le temps de départ depuis le dépôt ; ou le temps de retour du technicien $k$ au dépôt réspectivement.

Le modèle mathématique se présente comme suit [21] :

$$\min = \sum_{k \in K} \sum_{(i,j) \in A} C_{ij} x_{ij}^k + \sum_{i \in C} o_i y_i \tag{1.1}$$

avec les contraintes :

$$\sum_{k \in K} \sum_{j \in V} x_{ij}^k + y_i = 1 \qquad \forall i \in C \tag{1.2}$$

$$\sum_{j \in V} x_{ij}^k \leq q_i^k \qquad \forall k \in K \quad , \forall i \in C \tag{1.3}$$

$$\sum_{j \in V} x_{0.j}^k = 1 \qquad \forall k \in K \tag{1.4}$$

$$\sum_{i \in V} x_{i.n+1}^k = 1 \qquad \forall k \in K \tag{1.5}$$

$$b_i^k + (d_i + t_{ij})\, x_{ij}^k \leq b_j^k + l_i \left(1 - x_{ij}^k\right) \qquad \forall k \in K \quad \forall (i,j) \in A \tag{1.6}$$

$$e_i \leq b_i^k \leq l_i \qquad \forall k \in K \quad \forall i \in V \tag{1.7}$$

$$b_{n+1}^k - b_0^k \leq D \qquad \forall k \in K \tag{1.8}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall k \in K, \quad \forall (i,j) \in A \tag{1.9}$$

$$y_i \in \{0,1\} \qquad \forall i \in C \tag{1.10}$$

$$b_i^k \geq 0 \qquad \forall k \in K, \quad \forall i \in V \tag{1.11}$$

**La fonction objectif** (L'équation 1.1) pour minimiser le coût opérationnel total comprenant les coûts de routage et d'externalisation.

• L'équation (1.2) garantit que chaque tâche est soit visitée exactement une fois, soit externalisée.

• L'équation (1.3) garantit que les tâches ne peuvent être effectuées que par des techniciens satisfaisant aux exigences de compétences.

• Les équations (1.4) et (1.5) garantissent que chaque technicien quitte le dépôt et retourne à la copie du dépôt après avoir terminer leurs services.

• L'équation (1.6) définit les variables de temps $b_i^k$.

• L'équation (1.7) applique les restrictions de la fenêtre de temps.

• L'équation (1.8) garantit que la durée du parcours de chaque technicien ne dépasse pas le temps maximum autorisé.

• Les équations (1.9) et (1.10) représentent les restrictions binaires sur les variables $x_{i.j}^k$ et $y_i$.

• Et L'équation (1.11) est la contrainte de non négativité sur les variables $b_i^k$ .

## 1.4 Domaines d'application

Les STRSPs sont des problèmes impliquant l'ordonnancement des tournées d'un groupe de techniciens, afin de satisfaire les demandes des clients au coût le plus bas possible. ils peuvent être trouvés dans une variété d'applications, tels que :

1. **Le domaine médical**
   • Les auteurs des travaux [6] [7] ont présenté l'exemple des infirmières qui rendent visite aux patients à leur domicile pour leur administrer des médicaments ou leur prodiguer des soins.

2. **Les services publics**
   • Hadji constantinou et Roberts [5] ont présenté un problème de tournées de techniciens rencontré dans le domaine des services publics.

3. **Le domaine de maintenance des télécommunications**
   • Les auteurs des travaux [2] [3] ont présenté ce problème dans le domaine de télécommunications que les techniciens effectuent des travaux de maintenance d'installation , de construction et de réparation .

4. **Le domaine d'énergie**
   • Weintraub et al. [8] ont fait une étude d'orientation et de planification des techniciens de service pour les fournisseurs d'énergie au Chili.

5. **Clubs automobiles**
   • Grötschel et al. [9] ont présenté un problème TRSP pour les techniciens de service travaillant dans des clubs automobiles.

## 1.5 Méthodes de résolution d'un problème d'optimisation

L'optimisation est un ensemble de techniques et un domaine des mathématiques qui vise à représenter, analyser et résoudre des problèmes impliquant la minimisation ou la maximisation d'une fonction sur un ensemble, de manière analytique ou numérique.

De nombreux systèmes optimisés peuvent être représentés par un modèle mathématique. L'optimisation est importante dans la recherche opérationnelle :

- Réduction des coûts opérationnels
- Amélioration du niveau de satisfaction client
- Réduction des délais

La pertinence du modèle, le bon choix des variables à optimiser, l'efficacité de l'algorithme et les moyens de traitement numérique conditionnent la qualité des résultats et des prévisions.

Les méthodes d'optimisation combinatoire peuvent être classées en deux grandes familles :

**les méthodes exactes** et **les méthodes approchées**. (Figure 1.2)

La figure 1.2 illustre la taxonomie des méthodes de résolution des problèmes d'optimisation.

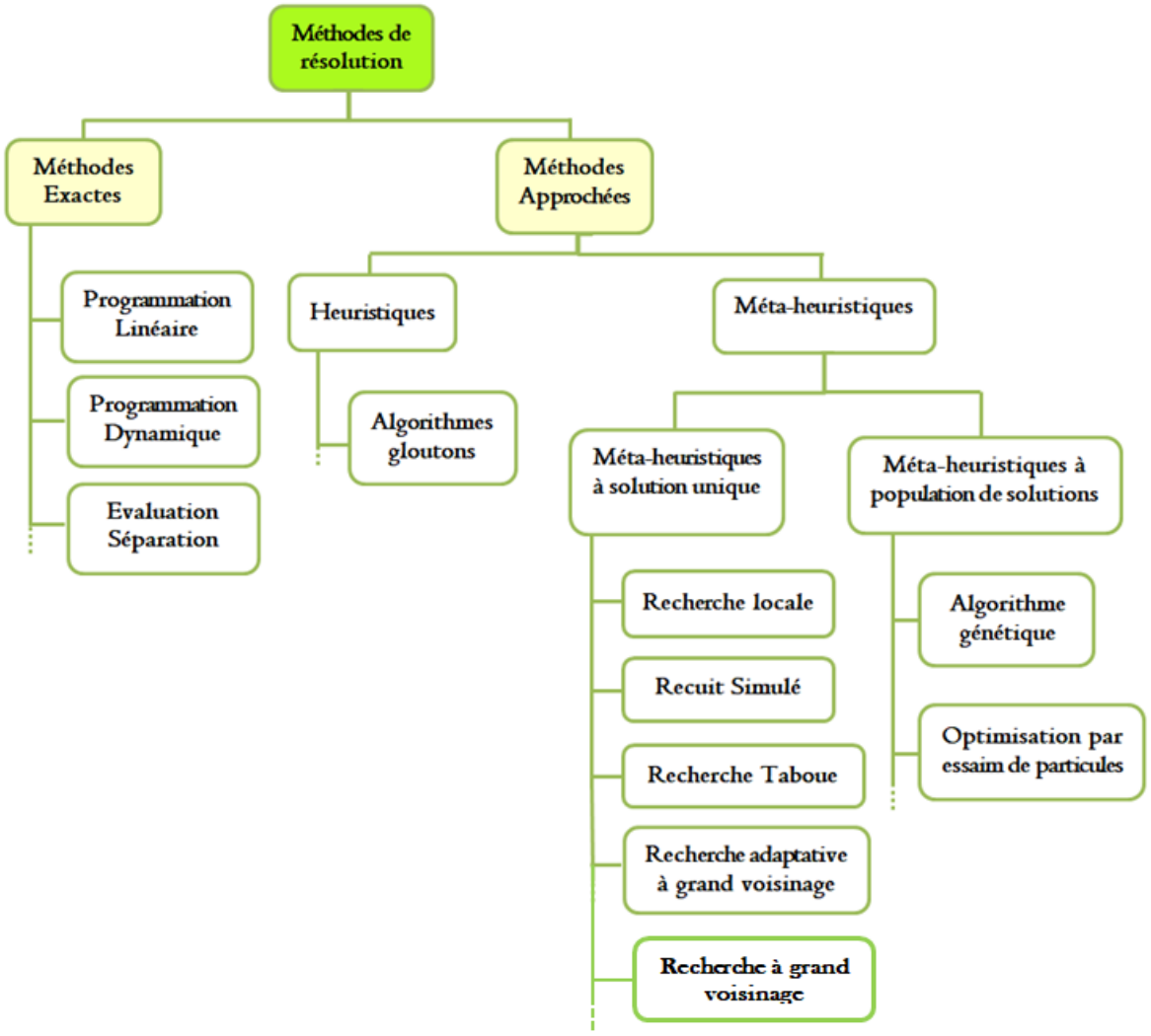



FIGURE 1.2 – Taxonomie des méthodes de résolution des problèmes d'optimisations

### 1.5.1 Méthodes exactes

Les méthodes exactes (appelées aussi complètes) donnent la meilleure réponse à un problème d'optimisation donné. Elles sont généralement basées sur des recherches arborescentes et une énumération partielle de l'espace des solutions.

Elles sont employées dans la recherche d'au moins une solution optimale à un problème donné. La technique de séparation et d'évaluation (branch and bound), la programmation dynamique et la programmation linéaire sont les algorithmes précis les plus connus. L'inconvénient majeur de ses méthodes est l'explosion combinatoire : Le nombre de combinaisons augmente avec l'augmentation de la dimension du problème.

L'efficacité de ces algorithmes n'est prometteuse que pour les instances de problèmes de petites tailles.

### 1.5.2 Méthodes approchées

Les méthodes approchées permettent d'identifier rapidement une solution réalisable à un problème. Cependant, la solution trouvée n'est pas toujours la solution optimale.

Le but et l'objectif est de découvrir une solution aussi proche que possible à l'aide d'une technique précise et plus rapide. Par conséquent, la qualité d'une méthode approchée sera déterminée par la différence entre sa solution et l'optimale.

Les méthodes approchées peuvent être classées en deux grandes familles : heuristiques et méta-heuristiques (Figure 1.2).

#### 1.5.2.1 Heuristiques

Le mot heuristique, dérivé de la langue grecque, vient du verbe heuriskein qui signifie trouver. Une heuristique est un algorithme qui permet de trouver dans un temps polynomial une solution réalisable, tenant compte d'une fonction objectif, pas nécessairement optimale pour un problème d'optimisation difficile. Une heuristique est spécifique au problème résolu et ne peut pas être généralisée [10].

#### 1.5.2.2 Méta-heuristiques

Le mot méta-heuristique est composé de deux mots grecs : méta et heuristique. Le mot méta est un suffixe signifiant au-delà c'est-à-dire de niveau supérieur [10].

Les méta-heuristiques sont des méthodes généralement inspirées de la nature. Contrairement aux heuristiques, elles s'appliquent à plusieurs problèmes de natures différentes. Pour cela, on peut dire qu'elles sont des heuristiques modernes, de plus haut niveau, dédiées particulièrement à la résolution des problèmes d'optimisation. Leur but est d'atteindre un optimum global tout en évitant d'être bloqués dans un optimum local. Les méta-heuristiques regroupent des méthodes qui peuvent se diviser en deux classes [10] :

- **Méta-heuristiques à solution unique :** Ces méthodes traitent une seule solution à la fois, afin de trouver la solution optimale.
- **Méta-heuristiques à population de solutions :** Ces méthodes utilisent une population de solutions à chaque itération jusqu'à l'obtention de la solution globale.

Ci-dessous, quelques méta-heuristiques parmi les plus utilisées [11] (Figure1.2) :

1. • Recuit simulé.
2. • Recherche taboue.
3. • Algorithme génétique.
4. • Recherche locale.
5. • Recherche à large voisinage.
6. • Recherche Adaptative à large voisinage.

1. **Recuit simulé**

   La méthode recuit simulé (Simulated Annealing) est une méta-heuristique, inspirée du processus utilisé en métallurgie (Recuit métallurgique), qui améliore l'heuristique de recherche locale pour s'échapper du piège de l'extrema (minima ou maxima) local, ce qui permet l'amélioration de la méthode de descente du gradient. L'idée principale est d'accepter les solutions dégradées avec une certaine probabilité dans l'espoir d'échapper aux optima locaux et d'explorer l'espace des solutions le mieux possible [10]. L'algorithme 1 présente les étapes de la méthode du recuit simulé.

**Algorithm 1** Algorithme du recuit simulé

**Déterminer** une configuration aléatoire $S$
**Choix** des mécanismes de perturbation d'une configuration
Initialiser la température T
**Tant que** Critère d'arrêt n'est pas satisfait **faire**
    **Tant que** l'equilibre n'est pas atteint **faire**
        **Tirer** une nouvelle configuration $S^{'}$
        **Appliquer** la régle de Metropolis
        **Si** $f(S^{'} < f(S))$ **Alors**
            $S_{min} \leftarrow S^{'}$
            $f_{min} \leftarrow f(S^{'})$
            **Fin si**
    **Fin Tant que**
    Décroitre la température
**Fin Tant que**

2. **Recherche taboue**

   La recherche taboue a été introduite par Glover (1989) [13]. L'exploitation du voisinage permet de se déplacer de la solution courante vers son voisin. Ce dernier n'est pas forcément meilleur que la solution courante. Cette méthode permet un déplacement d'une solution S vers la meilleure solution S' appartenant à son voisinage V(S).

   Le déplacement interdit, d'où vient le mot tabou, consiste au non-retour vers une solution récemment visitée. Pour cela, les solutions visitées sont temporairement interdites et stockées dans une liste taboue. Une fois la liste taboue est remplie, la solution la plus ancienne est retirée (selon le principe d'une file d'attente) FIFO. La taille de cette liste est un paramètre crucial qui affecte la résolution du problème. Elle peut être statique ou dynamique.

   ▷ Si la liste taboue est courte :

   - Il y a moins d'interdictions.
   - Intensification (On risque de n'exploiter que peu de distances).
   - Le risque de tourner en cycle, est plus grand.

▷ Si la liste taboue est longue :

   - Il y a d'avantage d'interdictions.
   - Diversification (On parcourt de plus grandes distances)
   - Le risque de cycles est réduit.

▷ Le comportement de l'algorithme dépend de :

- La longueur de la liste taboue.
- La taille du voisinage.

3. **Algorithme génétique**

Les algorithmes génétiques (AG) sont une famille de modèles informatiques inspirés du théorème d'évolution. Ces algorithmes, sur une structure de données simple semblable à un chromosome, encodent une solution potentielle à un problème particulier et appliquent des opérateurs de recombinaison à ces structures d'une manière qui préserve les détails essentiels. Bien que la variété des problèmes auxquels les AG sont utilisés soit assez complète, les AG sont le plus souvent considérés comme des optimiseurs de fonction [14].

La mise en œuvre d'un AG commence par une population (généralement aléatoire) de chromosomes. Ces structures sont ensuite évaluées et se voient attribuer des opportunités de reproduction afin que les chromosomes qui offrent une meilleure solution au problème cible aient une plus grande chance de se reproduire que les chromosomes qui sont de mauvaises solutions.

Les étapes de l'algorithme génétique representé dans la figure 3 :

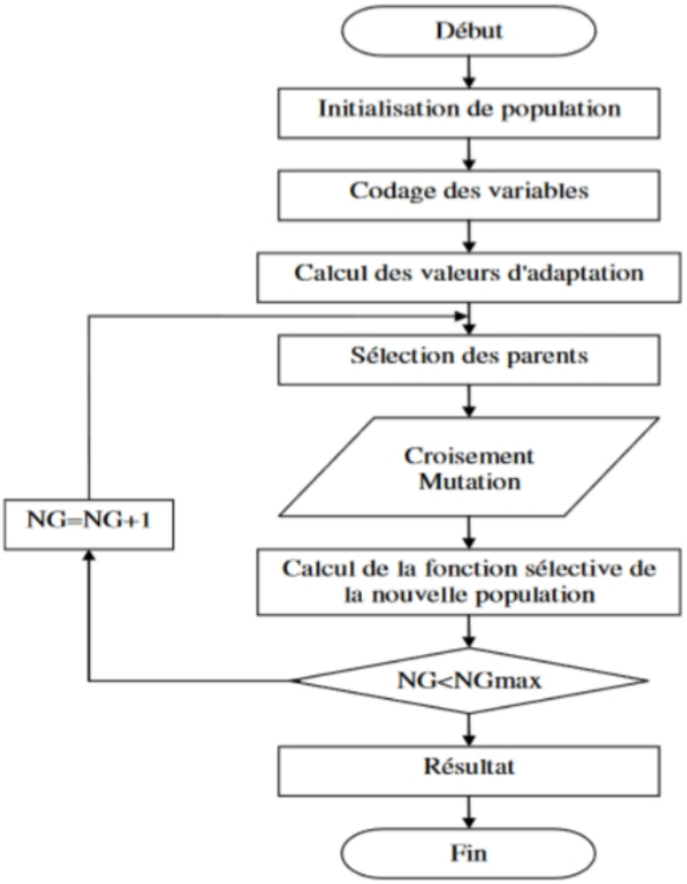



FIGURE 1.3 – Organigramme d'un algorithme génétique [12]

4. **Recherche locale**

La méthode de recherche locale est aussi appelée méthode de recherche de voisinage. Elle démarre par une solution initiale complète et construit une suite de meilleures solutions selon une fonction objectif [1]. Le choix se fait dans un ensemble localement proche. En d'autres termes, nous construisons une séquence limitée de solutions $S_n$ qui sont considérées comme plus proches d'une solution de départ $S_0$.

L'algorithme 2 résume cette idée.

**Algorithm 2** Algorithme général d'une méthode de recherche locale

```
1: Résultat : S : Solution
2: Données :
3:         S0 : Solution initiale ;
4:         S : Solution optimale ;
5:         S' : Solution voisine ;
6:         fit : réel ;
7: Début
8:    Choisir S0 une solution initiale ;
9:    S ← S0 ;
10:   fit ← f(S) ;
11:   Tant que (∃S' dans le voisinage de S) et (acceptable(S')) faire
12:      S ← S' ;
13:      fit ← f(S) ;
14:   fin
15: fin
```

5. **Recherche à large voisinage**

Les heuristiques de recherche locale ont déjà fait l'objet de nombreuses études. La recherche locale améliore une solution initiale en ne modifiant qu'une petite partie de celle-ci. Large Neighborhood Search (LNS) a été proposée par Shaw [15] et utilise une stratégie similaire, mais au lieu de petits ajustements, ce sont de grandes parties des solutions qui sont modifiées. Pour ce faire, on utilise un mélange de techniques dites de destruction et de réparation. La fonction de destruction contient généralement de l'aléatoire pour que différentes parties de la solution soient détruites à chaque invocation de la fonction. Une itération d'LNS correspond donc à l'application d'un opérateur de suppression (destruction) suivi d'un opérateur d'insertion (Réparation, Reconstruction), et à l'acceptation ou non de la solution obtenue.

Le schéma de la méthode LNS est présenté dans l'algorithme 3 :
L'algorithme maintient trois variables pour manipuler les solutions.

(a) La variable $S_{best}$ est utilisée pour maintenir la meilleure solution trouvée.

(b) la variable $S$ est utilisée pour maintenir la solution courante, et la remplace chaque fois qu'une meilleure solution est trouvée.

(c) tandis que $S'$ est une solution temporaire qui peut être rejetée ou promue pour être une solution courante.

La fonction $d()$ retourne une copie de $S$ partiellement détruite. Appliquer la fonction $r()$ sur la solution détruite retourne une solution faisable construite à partir de la solution détruite. Pour chaque itération, une fonction *accept*() est appelée sur la solution temporaire $S$.

Cette fonction peut être implémentée de différentes façons, la plus simple est d'accepter uniquement des solutions améliorées et c'est la plus importante de l'ensemble de l'algorithme.

**Algorithm 3** La recherche à large voisinage(LNS)

```
1: Générer une solution initiale S
2: S_best ← S
3: Tant que Condition d'arrêt n'est pas atteinte faire
4:     S' ← r(d(S))
5:     Si Accept(S, S') Alors
6:         S ← S';
7:     fin
8:     Si f(S') < f(S_best) Alors
9:         S_best ← S'
10:    fin
11: fin Tant que
```

## 6. Recherche Adaptative à large voisinage

En s'appuyant sur les travaux de [15], Ropke et Pisinger [20] ont développé la recherche adaptative à large voisinage (Adaptative Large Neighborhood Search en anglais ou ALNS). Tout en gardant le même principe de suppression et de réinsertion, ALNS emploie une stratégie de sélection d'opérateurs se basant sur les précédents poids (performances) de ces derniers. Concrètement, l'heuristique démarre avec des poids pi identiques pour chaque opérateur i et à chaque fois qu'une opération de suppression/insertion est effectuée, les poids des opérateurs concernés sont incrémentés.

Le pseudo-code de la méthode ALNS est illustré dans l'algorithme 4.

**Algorithm 4** La Recherche Adaptative à large voisinage(ALNS)

Générer une solution initiale S
$S_{best} \leftarrow S$
Initialiser les poids de tous les opérateurs.
**Tant que** Condition d'arrêt n'est pas atteinte **faire**
    Choisir un opérateur de destruction $d \in D$ par la méthode de roulette
    Choisir un opérateur de réparation $r \in R$ par la méthode de roulette
    $S' \leftarrow r\,(d\,(S))$
    **Si** $Accept(S, S')$ **Alors**
        $S \leftarrow S'$;
    **fin**
    **Si** $f(S') < f(S_{best})$ **Alors**
        $S_{best} \leftarrow S'$
    **fin**
Mettre à jour les poids des deux opérateurs $d$ et $r$.
**fin**

## 1.6 Quelques méthodes proposées pour la résolution du problème STRSP

Dans cette section, nous citions quelques travaux et quelques méthodes proposées pour la résolution du problème STRSP :

1. Kovacs et al [16] ont utilisé la méthode de Recherche adaptative à grand voisinage pour résoudre deux variantes d'un problème réel de tournée de techniciens.

   Une première version de l'algorithme est conçue pour les tournées effectuées par un seul technicien, tandis qu'une deuxième version est conçue pour les tournées effectuées par des équipes de travail. Dans les deux situations, les auteurs proposent des algorithmes de type ALNS employant diverses stratégies de destruction et de reconstruction de solutions.

   Contrairement à Ropke et Pisinger dans [19] , un score est relié à une paire d'opérateurs (destruction, reconstruction) qui est mise à jour à chaque itération en fonction de la qualité de la solution produite.

   Les auteurs [16] emploient les techniques de destruction suivantes :
   - destruction aléatoire,
   - destruction des pires tâches,
   - destruction avec dépendance,
   - destruction par clusters.

   En ce qui concerne les méthodes de reconstruction, ils ont utilisé :
   - Greedy heuristic.
   - Sequential insertion heuristic.
   - Regret heuristics.

2. Fulin Xie et al [21] ont utilisé la méthode de recherche locale itérée (ILS : Iterated local search) pour résoudre un problème réel de tournée de techniciens STRSP. L'algorithme 5 de la méthode ILS se compose de trois éléments principaux :

   (a) La construction de la solution initiale.

   (b) La procédure de recherche locale .

   (c) Le mécanisme de perturbation.

   À chaque itération de la boucle principale entre les lignes (4 à 18), une solution initiale réalisable $s$ est construite pour la boucle **ILS** (lignes 6 à 14). A chaque itération **ILS**, la procédure de recherche locale prend en entrée la solution $s$, et renvoie une solution améliorée $s'$, qui est acceptée comme la nouvelle meilleure solution courante si elle est faisable et a une valeur $f(s')$ qui est strictement plus petite que celle de la solution titulaire $\bar{s}$, dénotée par $f(\bar{s})$. Ensuite, une nouvelle solution de départ $s$ pour la procédure de recherche locale est générée en perturbant la solution courant $\bar{s}$ (ligne 12) [21].

   La boucle **ILS** se répète jusqu'à ce que le nombre maximal d'itérations sans amélioration $MaxIt_{NI}$ soit atteint. Ensuite, la solution en place $\bar{s}$ remplace la meilleure solution globale $s^*$ si $f(\bar{s}) < f(s^*)$. Cette procédure se répète jusqu'à ce qu'un nombre prédéfini $MaxIt$ d'itérations ait été exécuté [21].

**Algorithm 5** Iterated Local Search(ILS)

```
1:  procedure ILS
2:    IT ← 1
3:   f(s*) ← +∞
4:   For IT ← 1 To MaxItdo
5:        generate initial solution s
6:        While It_NI < MaxIt_NI do
7:           s' ←Local Search (s)
8:           If ((s' is feasible) and (f(s') < f(s⁻))) Then
9:              s⁻ ← s'
10:             It_NT ← It_NT + 1
11:          end If
12:          s ← Perturb(s⁻)
13:          It_NT ← It_NT + 1
14:       end While
15:       If (f(s⁻) < f(S*)) Then
16:         S* ← s⁻
17:      end If
18:  end For
19:  Return s*
20: end procedure
```

## 1.7 Conclusion

Dans ce chapitre, nous avons présenté une vue générale des problèmes de tournées de techniciens STRSP : la définition de ce genre de problèmes. Par la suite, nous avons exposé les formulations mathématiques qui régissent ce problème. Enfin, nous avons présenté quelques domaines d'application réelles, et certaines méthodes de résolution.

# CHAPITRE 2

# RÉSOLUTION DU PROBLÈME STRSP PAR LA MÉTHODE ALNS

## 2.1 Introduction

Dans ce chapitre, nous introduisons notre approche de résolution du problème STRSP en détaillant les heuristiques utilisées.

Par conséquent, l'objectif de ce chapitre est une description détaillée des deux heuristiques utilisées à savoir deux variantes d'ALNS.

## 2.2 Heuristiques utilisées pour résoudre STRSP (les deux variantes d'ALNS)

A partir des formulations mathématiques proposées dans le chapitre 1, ainsi que les méthodes de résolution d'un problème d'optimisation, nous avons construit une méthode de résolution pour STRSP.

Pour cela, nous avons développé deux algorithmes basés sur la méthode ALNS. L'utilisateur aura le choix d'utiliser l'un ou l'autre **(ALNSv1-ALNSv2)**.

### 2.2.1 Algorithme ALNSv1

Le pseudo-code de ALNSv1 fonctionne dans l'algorithme : 6

**Algorithm 6** ALNSv1

$S_{init}, S', S, S_{best}$ :Solution  $T$ : réel // Température
$D$ : ensemble des opérateurs destruction ;
$R$ : ensemble des opérateurs réparation ;
$r : R$ , $d : D$ ;
$q$ : nombre des taches à supprimer
$u$ :liste des taches supprimées
$ext$ :liste des Taches à externaliser
Nbr-itération, max-itération :entier ;
Segment : nombre d'itérations dans lesquelles on modifie les poids
**Début**
Générer une solution initiale $S_{init}$
$S_{best} \leftarrow S_{init}$ ;
$S \leftarrow S_{init}$ ;
Initialiser $T$  // Température Initiale
Initialiser les scores ,poids et probabilité pour les opérateurs $D$
Initialiser les scores ,poids et probabilité pour les opérateurs $R$
Nbr-itération = 0 ;
**répéter**
    $S' \leftarrow S$ ;
    $q$ = tirer un nombre aléatoire
    choisir un opérateur de destruction $d \in D$ par la roulette
    $u \leftarrow d(s', q)$ ;
    $u \leftarrow u \cup ext$
    Incrémenter le nombre d'utilisation de l'opérateur $d$
    choisir un opérateur de réparation $r \in R$ par la roulette
    $r(s', u)$ ;
    Incrémenter le nombre d'utilisation de l'opérateur $r$ ;
    **Si** $f(S') < f(S_{best})$ **Alors**
        $S_{best} \leftarrow S'$ ;
        $S \leftarrow S'$ ;
        Mettre à jour le Score des opérateurs $d$ et $r$ ;
    **Sinon**
        **Si** $f(S') < f(S)$ **Alors**
            $S \leftarrow S'$ ;
            Mettre à jour le Score des opérateurs $d$ et $r$ ;
        **Sinon**
            **Si** $S'$ répond aux critères d'acceptation **Alors**
                //(acceptation par Recuit simulé)
                $S \leftarrow S'$ ;
                Mettre à jour le Score des opérateurs $d$ et $r$ ;
            **fin si**
        **fin si**
    **fin si**
    Incrémenter Nbr-itération
    **Si** ( Nbr-itération **MOD** segment=0 ) **Alors**
       Mettre à jour les poids et probabilité des deux opérateurs $d$ et $r$
       Réinitialiser les scores nmbr d'utilisation des deux opérateurs $d$ et $r$
    **fin si**

48 : **jusqu'à** (Nbr-itération > Max-itération )
49 : **retourner** $S_{best}$
50 : **FIN**

Dans la première étape, une solution initiale réalisable est construite.

Grâce à l'option d'externalisation prévue, nous pouvons toujours développer une solution réalisable.

La méthode **ALNSv1** est basée sur deux opérateurs destruction et réparation, qui améliorent la solution initiale.

Tout d'abord, nous commençons par initialiser les scores, les poids et les probabilités des opérateurs $D$ et $R$ ainsi que la température initiale $T_0$ qui dépend de la solution initiale.

À chaque itération, les opérateurs de destruction et de réparation $d$ et $r$ sont choisis, selon le concept de la roulette en fonction de leurs performances (mesurées par les scores et les poids) lors des tours précédents (ligne 15 et 16).

• Trois opérateurs de destruction sont proposés pour sélectionner les tâches à supprimer de la solution courante : **Random removal, Worst removal et worst removal unrandomised** ( voir ligne 22).

Les tâches sont d'abord retirées de $S'$, en utilisant l'opérateur de destruction sélectionné $d$ et ajoutées à la collection des tâches externalisées.

Ensuite, elles sont réinsérées dans les itinéraires, en utilisant l'opérateur de réparation approprié $r$. L'opérateur de réparation utilise deux stratégies : meilleure insertion (best insertion) et second meilleure insertion (second best insertion). (ligne 27).

Dans la ligne 28, incrémenter le nombre d'utilisation de l'opérateur $r$.

Si la solution résultante $S'$ est meilleure que la meilleure réponse identifiée jusqu'à présent, elle prend la place de $S_{best}$

Si la solution résultante $S'$ n'est pas meilleure que $S_{best}$ mais meilleure que $S$, elle prend la place de $S$.

Nous pouvons accepter une solution $S'$ , légèrement plus faible que $S$, à condition de satisfaire les conditions adoptées (Nous avons utilisé le recuit simulé), elle prend alors la place de $S$. (ligne 38).

Dans les trois cas, nous mettons à jour les scores des opérateurs. Dans le premier cas, nous ajoutons au score un nombre plus important que les autres (ligne 32). Par contre, dans les cas contraires, nous ajoutons au score un nombre inférieur (ligne 36et 40).

Ensuite, à chaque segment (certain nombre d'itérations), les scores et les poids des opérateurs $d$ et $r$ sont ajustés en fonction de la qualité de la solution résultante et la recherche passe à l'itération suivante. Cette opération est répétée jusqu'à ce qu'une condition d'arrêt soit remplie (nous avons fixé un nombre d'itérations donné).

Nous prenons le meilleur résultat, qui est la valeur de $S_{best}$.

### 2.2.2 Algorithme ALNSv2

Cette variante ALNSv2 est une heuristique que nous avons développée et adoptée dans notre résolution. Elle utilise les poids des opérateurs fixes et choisis par nous-même.

Le schéma de cette heuristique est présenté dans l'algorithme 7 suivant :

**Algorithm 7** ALNSv2

$S_{init}, S', S, S_{best}$ :Solution
$T$ : réel // Température
$D$ : ensemble des opérateurs *destruction* ;
$R$ : ensemble des opérateurs *réparation* ;
$r : R$ ;
$d : D$ ;
$q$ : nombre des taches à supprimer
$u$ :liste des taches supprimées
$ext$ :liste des Taches à externaliser
Nbr-itération, max-itération :entier ;
Segment : nombre d'itérations dans lesquelles on modifie les poids
**Début**
Générer une solution initiale $S_{init}$
$S_{best} \leftarrow S_{init}$ ;
$S \leftarrow S_{init}$ ;
Initialiser $T$ // Température Initiale
Nbr-itération $= 0$ ;
**répéter**
    $S' \leftarrow S$ ;
    $q =$ tirer un nombre aléatoire
    choisir un opérateur de destruction $d \in D$ par la roulette
    $u \leftarrow d(s', q)$ ;
    $u \leftarrow u \cup ext$
    choisir un opérateur de réparation $r \in R$ par la roulette
    $r(s', u)$ ;
    Incrémenter le nombre d'utilisation de l'opérateur $r$ ;
    **Si** $f(S') < f(S_{best})$ **Alors**
        $S_{best} \leftarrow S'$ ;
        $S \leftarrow S'$ ;
    **Sinon**
        **Si** $f(S') < f(S)$ **Alors**
            $S \leftarrow S'$ ;
        **Sinon**
            **Si** $S'$ répond aux critères d'acceptation **Alors**
                //(acceptation par Recuit simulé)
                $S \leftarrow S'$ ;
            **fin si**
        **fin si**
    **fin si**
    Incrémenter Nbr-itération
**jusqu'à** (Nbr-itération > Max-itération )
**retourner** $S_{best}$
**FIN**

L'algorithme prend la solution initiale $S$ et la remplace chaque fois qu'une meilleure solution est trouvée. Mais l'algorithme ne cherche pas toujours dans le voisinage de la meilleure solution actuelle $S_{best}$. Au lieu de cela, il recherche le voisinage de $S$, ce qui donne la solution $S'$.

Alors que les solutions $S$ et $S_{best}$ sont initialement égales, des différences peuvent apparaître lorsqu'une solution non améliorée $S'$ est acceptée. Dans ce cas, $S'$ est affectée à $S$ et $S_{best}$ reste la même. Nous pouvons remarquer que la fonction de voisinage $N$ a été remplacée par une combinaison des opérateurs de destruction $d$ et de réparation $r$. Ces fonctions sont la partie la plus importante de l'ensemble de l'algorithme.

### 2.2.3 Destruction et réparation

C'est la partie la plus importante de l'algorithme ALNS

Le but est de détruire une partie de la solution en la supprimant de celle-ci et en la réparant par la suite. Par conséquent, l'opérateur de destruction a pour but de supprimer les portions qui permettront à l'opérateur de réparation d'améliorer la solution dans la mesure du possible. Dans la littérature, ce concept a été désigné sous plusieurs noms, notamment détruire et réparer [18], réparer et optimiser [17], déchirer et recoudre [24], enlever et réinsérer [25] à ruiner et recréer [26]. Bien entendu, pour ces deux opérateurs, différentes heuristiques sont concevables. Elles seront expliquées dans les sections suivantes.

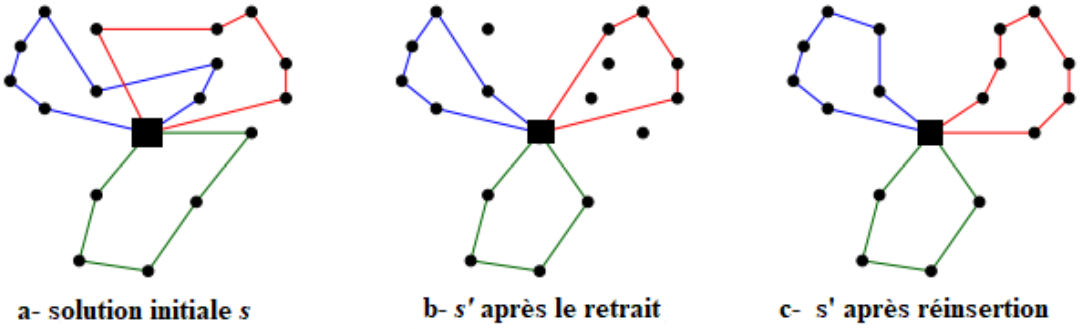



FIGURE 2.1 – Un exemple du concept de voisinage dans STRSP.

- La figure (a) montre une solution sous-optimale initiale $s$ de l'instance STRSP donnée avec le dépôt au centre et trois routes.
- Dans la figure (b) , les composants de la solution ont été supprimés des itinéraires. Le voisinage $N(s)$ contient toutes les solutions qui peuvent être créées en réinsérant la tâche dans les routes.
- La figure (c) représente une telle solution $s' \in N(s)$.

#### 2.2.3.1 Opérateurs de destruction

Dans la littérature, différents opérateurs de destruction ont été proposés.

Avant de lancer une méthode de destruction, il faut spécifier le nombre de tâches à éliminer sur les différentes routes. La lettre $q$ représente ce nombre.

à chaque itération, il est choisi aléatoirement entre $\min\{0.1|V^a|, 0.3\}$ et $\min\{0.4|V^a|, 0.6\}$. L'ensemble $V^a$ contient toutes les tâches qui sont actuellement assignées à une route.

Dans notre travail, nous avons utilisé trois types d'opérateurs de destruction, l'opérateur suppression aléatoire, l'opérateur suppression pire et l'opérateur suppression pire avec probabilité.

1. **Opérateur random removal (randRem)**

   Random removal est l'opérateur de destruction le plus simple. Désigné par **randRem**, il retire de manière aléatoire une tâche de la route d'un technicien et la déplace vers la liste des tâches externalisées. Cette opération est répétée jusqu'à ce que toutes les tâches $q$ sont déplacées. Il s'agit d'un opérateur très important pour l'algorithme ALNS car il peut supprimer toute solution indépendamment des coûts. De ce fait, la suppression aléatoire crée une diversification qui est très importante lors de l'exploration d'un grand espace de recherche.

**Algorithm 8** Random Removal

```
Function Random Removal (S :solution, q :entier) :Liste de taches
L = Liste de Toutes les taches planifiées dans la solution S ;
r :tache ;u :Liste de taches ;y :entier ;
Tant que q > 0 faire
    choisir un nombre aléatoire y dans l'intervalle [0, |L| − 1] ;
    r = L[y] ;
    supprimer r de la solution S et de la liste L ;
    q = q − 1 ;
    u = u ∪ {r}
fin tant que
return u ;
```

2. **Opérateur worst removal (worstRem)**

   L'opérateur **worstRem** supprime des tâches des différentes routes qui ont le pire impact sur la valeur de la fonction objectif.

   Soit $f(s)$ la valeur de la fonction objectif de la solution actuelle $s$, et $f_{-i}(s)$ la valeur de la fonction objectif de cette solution en l'absence de la tâche $i$. La différence en termes de valeur de la fonction objectif pour le transfert de la tâche $i$ actuellement programmée dans la liste des tâches externalisées est alors donnée par :

$$d(i,s) = f_{-i}(s) - f(s) \tag{2.1}$$

   Les tâches actuellement insérées sont placées dans une liste $L$, qui est ordonnée par valeurs décroissantes de $d(i,s)$.

A chaque itération, un nombre aléatoire $y$ de l'intervalle [0, 1[ est choisi. Ensuite, la tâche $i := L\left[y^{\rho_{worst}}|L|\right]$ est retirée de sa route, et les valeurs $d\left(i,s\right)$ de toutes les tâches restantes sont modifiées. La quantité de randomisation est déterminée par le paramètre $\rho_{worst}$ .

Les étapes ci-dessus sont poursuivies jusqu'à ce que toutes les tâches $u$ aient été retirées de leurs chemins.

L'opérateur de suppression pire avec probabilité (**Worst Removal**) est représentée en pseudo-code dans l'algorithme 9

**Algorithm 9** Worst Removal

```
Function Worst Removal (S :solution, q : entire, p : réel) :Liste de taches
r :tache ; u :Liste de taches ; y :entier ;
Tant que q > 0 faire
    L = Liste de Toutes les taches planifiées triées par coût décroissant d(i, s) ;
    choisir un nombre aléatoire y dans l'intervalle [0, 1) ;
    r = L[y^p|L|] ;
    supprimer r de la solution S et de la liste L ;
    q = q − 1 ;
    u = u ∪ {r}
fin tant que
return u ;
```

3. **Opérateur worst removal unrandomised (UnrandWorstRem)**

   Cet opérateur est une variante de la précédente qui a été adoptée par nous-même. Les tâches actuellement insérées sont placées dans une liste $L$, ou les plus pires des tâches sont supprimées.

   En d'autres termes, nous avons trié les tâches réalisées par ordre décroissant de leur coût puis nous avons supprimé les tâches les plus coûteuses en respectant toujours le taux de suppression précédent et enfin nous avons calculé le coût de suppression. Toujours les taches d'externalisation ont été ajoutées à la liste de suppression.

**Algorithm 10** Worst Removal unrandomised

```
Function Worst Removal unrandomised (S :solution, q : entire) :Liste de taches
r :tache ; u :Liste de taches ; y :entier ;
Tant que q > 0 faire
    L = Liste de Toutes les taches ,triées par coût décroissant d(i, s) ;
    r = L[0] ;
    supprimer r de la solution S et de la liste L ;
    q = q − 1 ;
   u = u ∪ {r}
fin tant que
return u ;
```

#### 2.2.3.2 Opérateurs de réparation

Plusieurs opérateurs de réparation sont utilisés pour réparer une solution partiellement détruite. Toutes les heuristiques tentent d'insérer autant de tâches que possible qui ne sont actuellement allouées à aucun itinéraire.

Dans notre travail, nous avons utilisé l'opérateur de meilleure insertion ( Greedy insertion) et l'opérateur du second insertion.

1. **Opérateur Greedy insertion (greedyIns)**

   L'opérateur **greedyIns** insère de manière répétée une tâche à la position la moins chère possible.

   Nous introduisons :

   $f_\Delta(i,k)$ = changement de la valeur objectif pour l'insertion de la tâche $i$ (qui est actuellement externalisée) à la position la moins chère possible dans l'itinéraire $k$.

   Soit $f_\Delta(i,k) = \infty$, si la tâche $i$ ne peut pas être insérée dans la route $k$.

   A Chaque itération, parmi l'ensemble des tâches actuellement externalisées $V^0$, on choisit la tâche $i^*$ et la route $k^*$ pour les quelles :

   $$(i^*, k^*) := arg_{i \in V^0} \min_{k \in K} f_\Delta(i,k) \tag{2.2}$$

   La tâche $i^*$ est alors placée au point de coût le plus bas de la route $k^*$. Ce processus est répété jusqu'à ce que toutes les tâches aient été insérées ; Dans le cas où une tâche n'est pas insérée, elle est externalisée.

2. **Opérateur Random insertion (RanIns)**

   Dans ce cas, nous choisissons, de manière aléatoire, une tâche supprimée dans les opérateurs de destruction et nous l'insérons dans la meilleure position possible.

   Dans le cas où il existe plusieurs positions similaires, le choix a été fait de manière aléatoire.

   Toutes les tâches non insérées, ont été déplacées vers l'externalisation.

### 2.2.4 Critères d'acceptation

La méthode d'acceptation des deux algorithmes **ALNSv1** et **ALNSv2** est basée sur le **recuit simulé** (Simulated Annealing (SA)) où toute solution d'amélioration est acceptée Si $f(S') < f(S)$.

Si $f(S') > f(S)$, $S'$ est accepté avec probabilité $e^{(-(f(S')-f(S))/T)}$ où $T$ est la température. La température réduite à chaque itération d'un facteur $\phi$; ($\phi = 0.99975$).

Pour la méthode acceptation cela signifie qu'au début, la probabilité d'accepter des solutions moins bonnes que la solution actuelle est élevée et diminue progressivement. Plus précisément, pour une différence de coût arbitraire mais fixe $f(S) - f(S')$ la probabilité d'acceptation décroît exponentiellement $e^{(-(f(S')-f(S))/T)}$ . La température initiale dépend de la solution initiale et elle est utilisée comme suit :

$$T = -\frac{\omega_T}{\ln 0.5} f_c(S) \tag{2.3}$$

Selon Ropke and Pisinger [20] , $\omega_T = 0.05$. ce qui signifie qu'une solution qui est 5% plus pire que $f_c(S)$ est autorisée avec une probabilité de 0,5.

L'intention est de laisser l'algorithme chercher dans l'espace de recherche pendant un certain temps en acceptant également une solution légèrement pire que la solution actuelle et de le faire se concentrer sur des solutions plus prometteuses plus tard. Et tend à éviter de s'enfermer trop vite dans la recherche d'un optimum local.

### 2.2.5 Choix des opérateurs de destruction d et de réparation r

Comme Ropke et Pisinger [ [19], [20]], nous avons utilisé des scores et des poids séparés pour chaque opérateur de destruction et de réparation :

- De l'ensemble des opérateurs de destruction :
  - RandRem,
  - WorstRem,
  - UnrandWorstRem,
- Et de l'ensemble des opérateurs de réparation :
  - GreedyIns,
  - RanIns.

Les probabilités de choisir des opérateurs de destruction $d$, sont proportionnelles à $\rho_d$ où $\rho_d$ désigne le poids de chaque opérateur $d$ ;

Soit $n_d$ désignent le nombre respectif d'opérateurs de destruction employés, alors la probabilité $\phi_d$ de choisir un opérateur donné est calculée comme suit :

$$\phi_d = \frac{\rho_d}{\sum_{d'}^{n_d} \rho_{d'}} \tag{2.4}$$

En utilisant la sélection par roulette, les deux opérateurs de destruction sont choisis à chaque itération de l'algorithme ALNS.

Comme dans la référence [20], sur la base des scores obtenus, les poids sont ajustés dynamiquement pendant la recherche. Au début, les poids $\rho_d$ de tous les opérateurs $d$ sont fixés à 1 et les scores $\psi_d$ sont fixés à zéro. À la fin de chaque itération ; (les scores $\psi_d$ des opérateurs employés $d$ sont mis à jour comme suit :

$$\psi_d = \begin{cases} \psi_d + \sigma_1 & \text{Si les opérateurs de destruction ont donné une solution qui a amélioré la meilleure solution globale } S_{best}. \\ \psi_d + \sigma_2 & \text{Si les opérateurs de destruction ont donné une solution qui a amélioré la solution courante } S. \\ \psi_d + \sigma_3 & \text{Si les opérateurs de destruction ont donné une solution qui a été acceptée comme la nouvelle solution courante S, bien qu'elle soit pire (Critère d'acceptation : Recuit simulé),} \end{cases} \tag{2.5}$$

$\sigma_1$ , $\sigma_2$ et $\sigma_3$ sont des paramètres définis par l'utilisateur.

Toutes les 100 itérations, les poids $\rho_d$ sont mis à jour sur la base des scores obtenus au cours du dernier segment temporel, c'est-à-dire les 100 dernières itérations. Au début de chaque nouveau segment temporel, les scores $\psi_d$ sont remis à zéro. A la fin de chaque segment temporel, les poids sont mis à jour comme suit :

$$\rho_d = (1 - \rho_{react})\, \rho_d + \rho_{react} \frac{\psi_d}{\max(1, \Theta_d)} \tag{2.6}$$

Où $\Theta_d$ désigne le nombre de fois où les opérateurs de destruction $d$ ont été utilisée dans le dernier segment temporel ; $\rho_{react}$ contrôle l'influence respective des informations historiques et nouvelles.

De la même façon on calcule les scores, les poids et les probabilités pour les opérateurs de réparation $r$

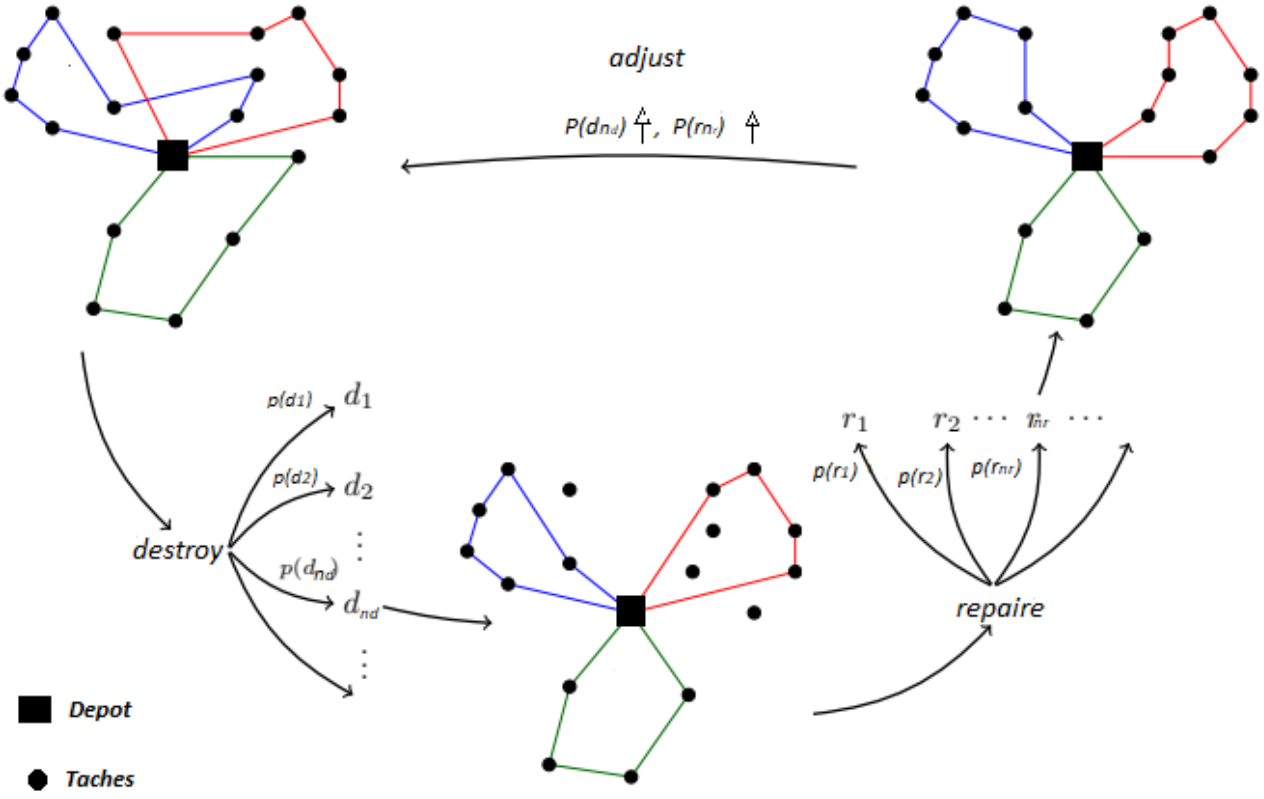



FIGURE 2.2 – Le schéma d'ALNS destroy-repair

La figure 2.2 montre le schéma d'ALNS. Une première solution passe par le processus de destruction et de réparation.

Les opérateurs de destruction et de réparation sont choisis parmi un ensemble d'opérateurs. Les probabilités de sélection sont basées sur leur succès.

Après un certain nombre d'itérations, les probabilités sont ajustées en fonction du succès de l'opérateur.

## 2.3 Solution initiale adoptée

Pour construire la solution initiale, nous suivons les étapes suivantes : Tout d'abord, une liste de tâches L1 est créée et triée, selon le temps au plutôt. La deuxième étape consiste à construire la liste des techniciens L2. Dans la troisième étape, un tableau des compétences des techniciens L3 est élaboré. Ensuite, pour chaque tâche, nous calculons le cout engendré par l'affectation fictive de cette tache à la fin du trajet de chaque technicien, tout en respectant les contraintes :

- Fenêtres de temps.
- Le technicien doit être capable d'effectuer cette tâche.
- Ne pas dépasser le temps maximum autorisé.

Enfin, l'affectation réelle est celle qui engendre le coût minimal. Lorsqu'une tâche donnée ne peut être effectuée par aucun des techniciens, alors elle est affectée à un sous-traitant (externalisation).

## 2.4 Conclusion

Dans ce chapitre, nous avons présenté notre approche pour la résolution du problème STRSP. Pour cela, nous avons appliqué deux méthodes de **ALNS** : **ALNSv1** et **ALNSv2**.

Les résultats de notre résolution seront présentés et discutés dans le chapitre suivant.

# CHAPITRE 3

# EXPÉRIMENTATION ET RÉSULTATS

## 3.1 Introduction

Dans le chapitre précédent, nous avons présenté notre algorithme développé pour la résolution du problème STRSP, qui est basé sur deux variantes **ALNSv1** et **ALNSv2** ; cettes dernières fonctionnent avec les opérateurs de destruction et de réparation d et r.

Dans ce chapitre nous présenterons les outils matériels et logiciels utilisés pour développer les méthodes proposées, ainsi que la description détaillée des instances utilisées et les interfaces graphiques. A la fin, tous les résultats obtenus seront discutés et comparés avec ceux trouvés par les travaux antérieurs issus de la littérature.

## 3.2 Représentation des outils

### 3.2.1 Outils logiciels

Un langage de programmation représente l'interface entre le programmeur et l'ordinateur. il est écrit pour exploiter les ordinateurs et leurs capacités de résolution des problèmes.

Heureusement, il y a beaucoup d'outils open source disponibles dans ce domaine qui peuvent faciliter le processus de développement de notre application, et voici l'ensemble des outils utilisés pour le faire.

#### 3.2.1.1 Langage java

Java 3.1 est un langage de programmation purement orienté objet, développé par SUN Microsystems, vendu à oracle en 2009 ; c'est un langage simple et puissant qui possède de nombreux points forts.

Java est un langage compilé et interprété de haut niveau, multi paradigme et multi plateforme, la raison de sa grande renommée dans la société des développeurs est due à sa simplicité de développement d'applications dans différents domaines qui fonctionnent sur différentes plateformes et à la disponibilité d'une bibliothèque open source très bien

conçue, utilisée pour de multiples sous-domaines de l'informatique. C'est l'un des principaux langages de programmation utilisés aujourd'hui en entreprise.

Les utilisations du langage java sont nombreuses, parmi lesquelles : le développement web, le développement Android /iOS, finance e-commerce, big data et Jeux vidéo.


FIGURE 3.1 – logo java.

#### 3.2.1.2 JDK

Le JDK 3.2(Java Développement Kit) est l'outil de base pour tout développement Java. . Il est proposé gratuitement par Oracle, il peut être chargé sur le site JavaSoft.

Ce kit contient tout le nécessaire pour développer des logiciels, des applications ou des applets Java : Le compilateur (en ligne de commandes), une machine virtuelle, un ensemble complet de classes de bases regroupées en packages.


FIGURE 3.2 – logo jdk.

#### 3.2.1.3 NetBeans

NetBeans 3.3 est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en 2000 sous licence CDDL (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web).

NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X et Open VMS.

NetBeans est lui-même développé en Java, ce qui peut le rendre assez lent et gourmand en ressources mémoires[50].

FIGURE 3.3 – logo NetBeans.

#### 3.2.1.4 Visual studio code

Visual studio code 3.4 est un éditeur de code extensible développé par Microsoft pour Windows, Linux et macOS2.

Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les snippets, la refactorisation du code et Git intégrer. Il est proposé gratuitement.

Les utilisateurs peuvent modifier le thème, les raccourcis clavier, les préférences et installer des extensions qui ajoutent des fonctionnalités supplémentaires.

Dans le sondage auprès des développeurs réalisé par Stack Overflow en 2021, Visual Studio Code a été classé comme l'outil d'environnement de développement le plus populaire [51].

FIGURE 3.4 – Logo Visual studio code.

#### 3.2.1.5 MXGraph et JFREECHART

**1-MXGraph :**3.5 est une bibliothèque de diagrammes JavaScript qui permet de créer rapidement des applications de graphiques et de cartographie interactives qui s'exécutent nativement dans n'importe quel navigateur majeur pris en charge par son fournisseur.


FIGURE 3.5 – logo MXGraph.

**2-JFREECHART :** 3.6 est une bibliothèque de graphiques Java gratuite, ce qui permet aux développeurs d'afficher facilement des graphiques de qualité professionnelle dans leurs applications.

JFREECHART prend en charge les graphiques circulaires (2D et 3D), les graphiques à barres (horizontaux et verticaux, réguliers et empilés), les graphiques de ligne, les graphiques de dispersion, les graphiques de séries chronologiques, les graphiques à clôture à haut-low thermomètres, cadrans et plus encore. JFREECHART peut être utilisé dans les applications côté client et côté serveur.


FIGURE 3.6 – Logo JFREECHART.

### 3.2.2 Matériel informatique

Dans le but de minimiser le coût de tournées, nous avons implémenté les méta-heuristiques ALNSv1 et ALNSv2. Toutes leurs fonctions sont implémentées en java par l'éditeur NetBeans IDE 8.2 et exécutées sur deux ordinateurs portables, de configurations suivantes :

- **CPU :** Intel(R) Core (TM) i5-4310U CPU @ 2.00GHz 2.60 GHz
- **RAM :** 8 Go.
- **OS :** Système d'exploitation 64 bits (Windows 10).

- **CPU :** Intel(R) Core (TM) i5-3210M CPU @ 2.50GHz 2.50 GHz
- **RAM :** 4 Go.
- **OS :** Système d'exploitation 64 bits (Windows 8).

## 3.3 Interfaces Graphiques

Dans notre projet, nous avons utilisé la bibliothèque Swing à cause de sa facilité de créer des applications avec interface graphique. Notre application se compose des fenêtres suivantes :

**Fenêtre principale**

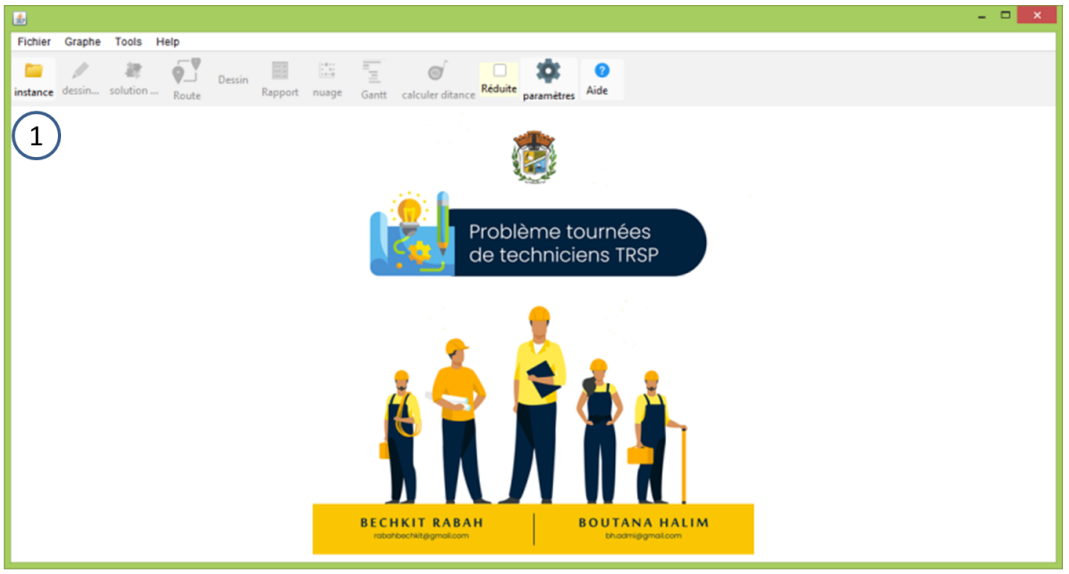



FIGURE 3.7 – Fenêtre d'acceuil

Si, l'utilisateur choisit le champ de instance (1) alors la fenêtre suivante apparaît :

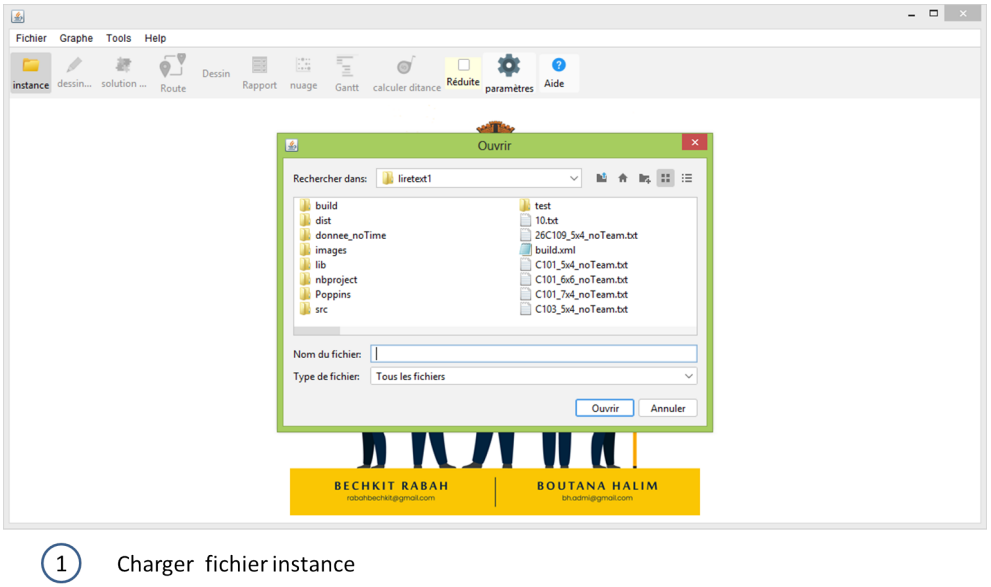



FIGURE 3.8 – Ouvrir instance fichier

La fenêtre suivante c'est **la fenêtre principale**. la fenêtre principale se compose de quatre parties :

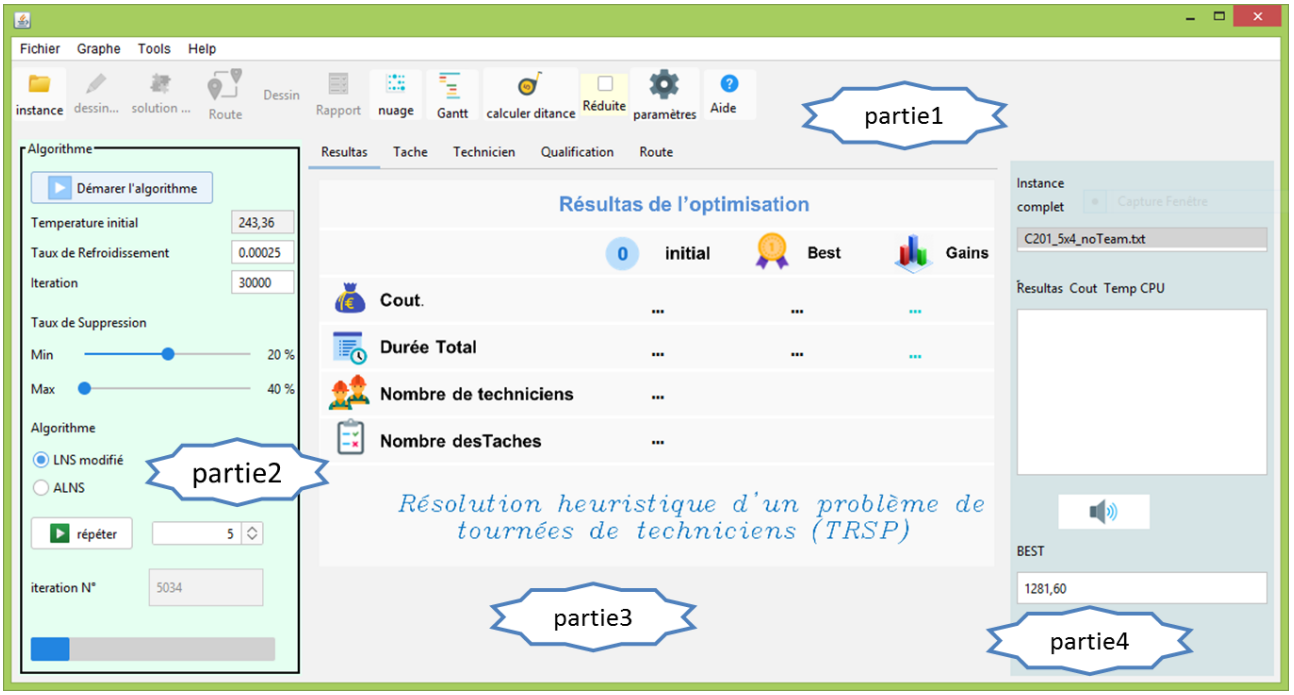
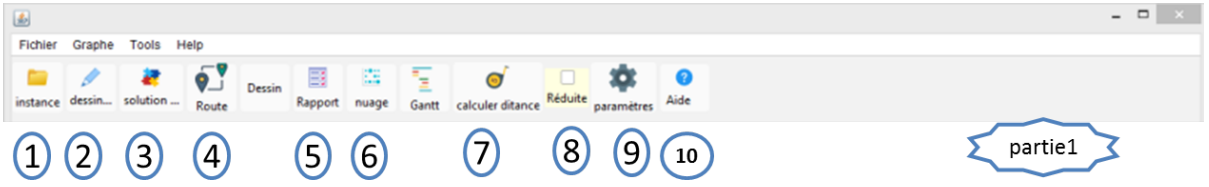


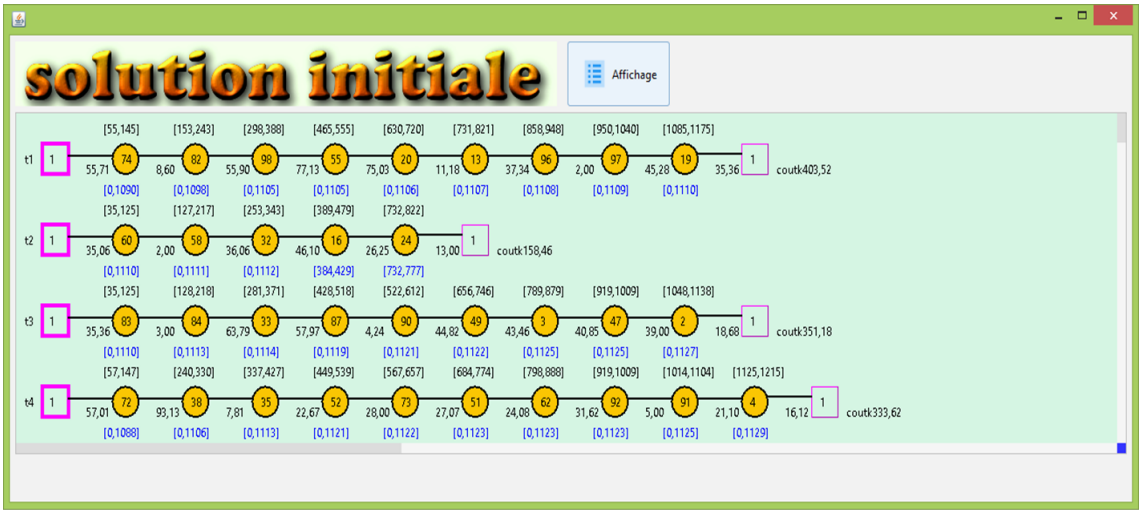

FIGURE 3.9 – Fenêtre principale

La partie 1 de la figure 3.10 est la barre d'outils, comprenant les numéros de 1 à 10 :

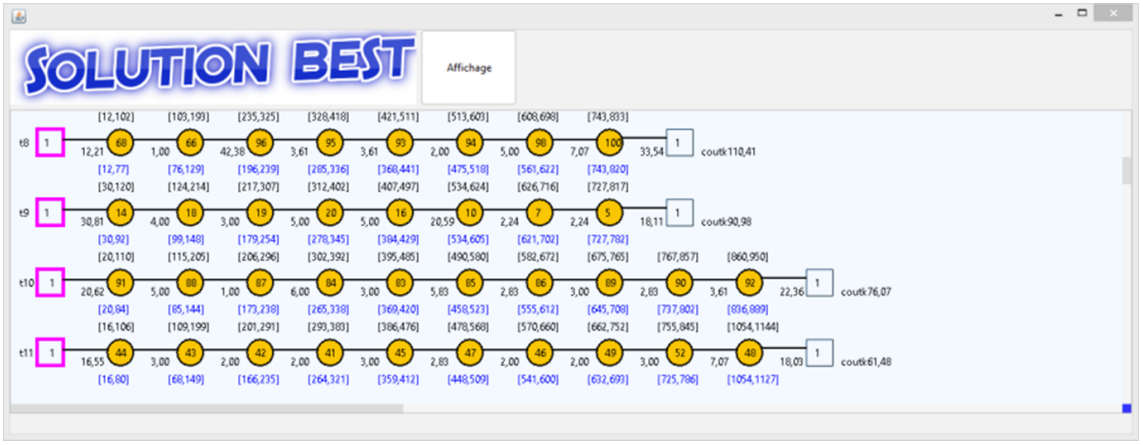



1. Charger instance
2. Présentation graphique de la solution initiale
3. Présentation graphique de la solution Best
4. itinéraire de chaque technicien
5. Rapport
6. Nuage de points
7. Outil de calcul des distances
8. Utiliser l'instance réduite
9. Réglage des paramètres
10. Aide à comprendre le fonctionnement du logiciel

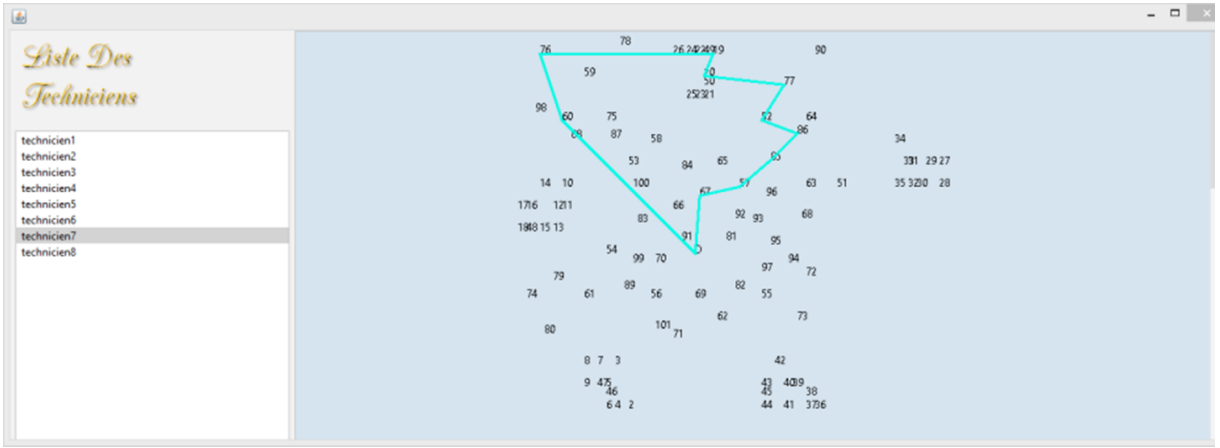
FIGURE 3.10 – Barre d'outils




2 présentation graphique de la solution initiale

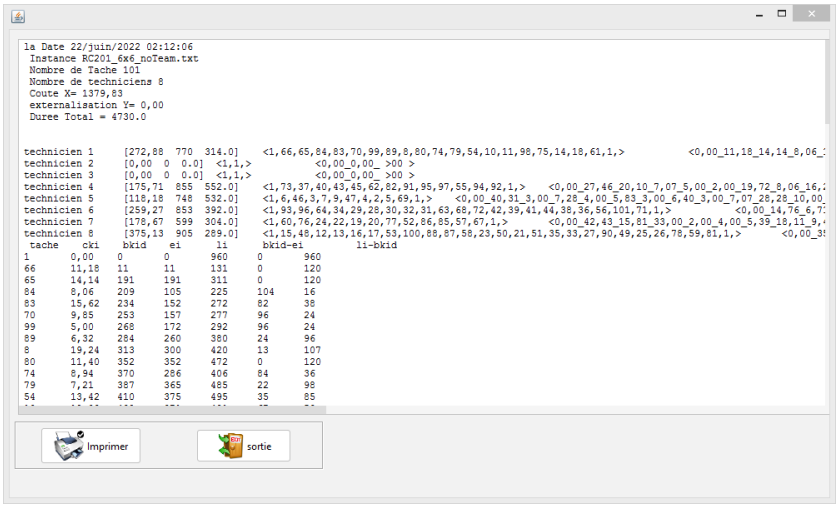



3 Présentation graphique de la solution Best

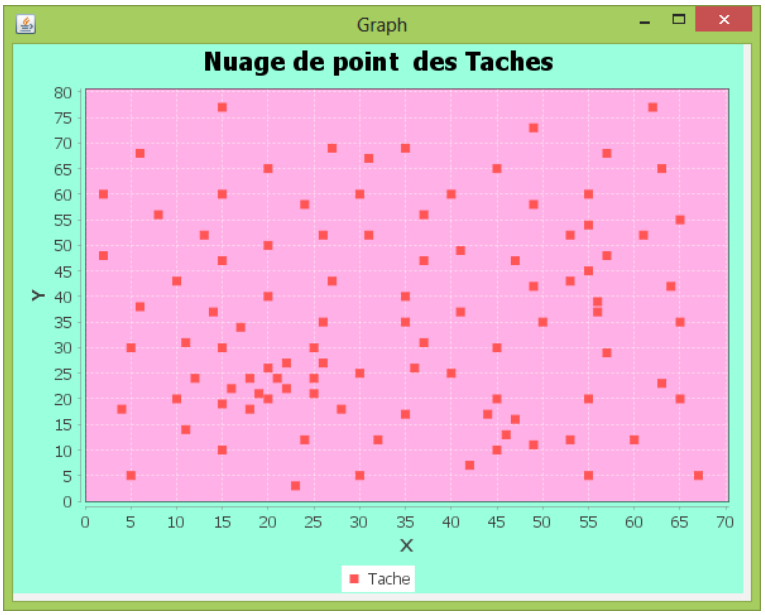



4 Itinéraire de chaque technicien

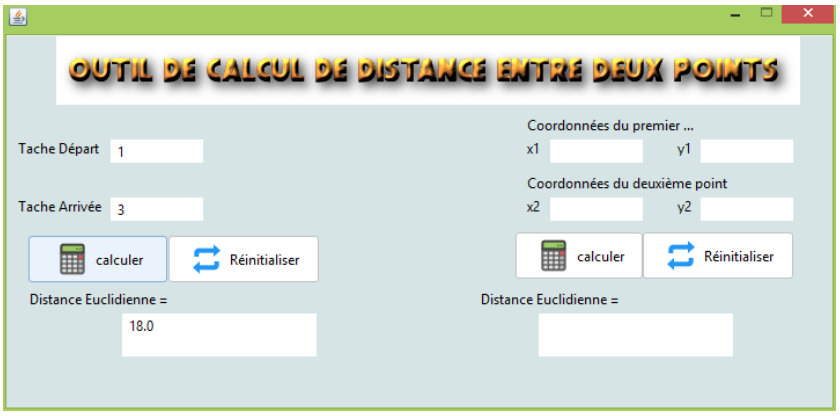
5 : Rapport de la solution best



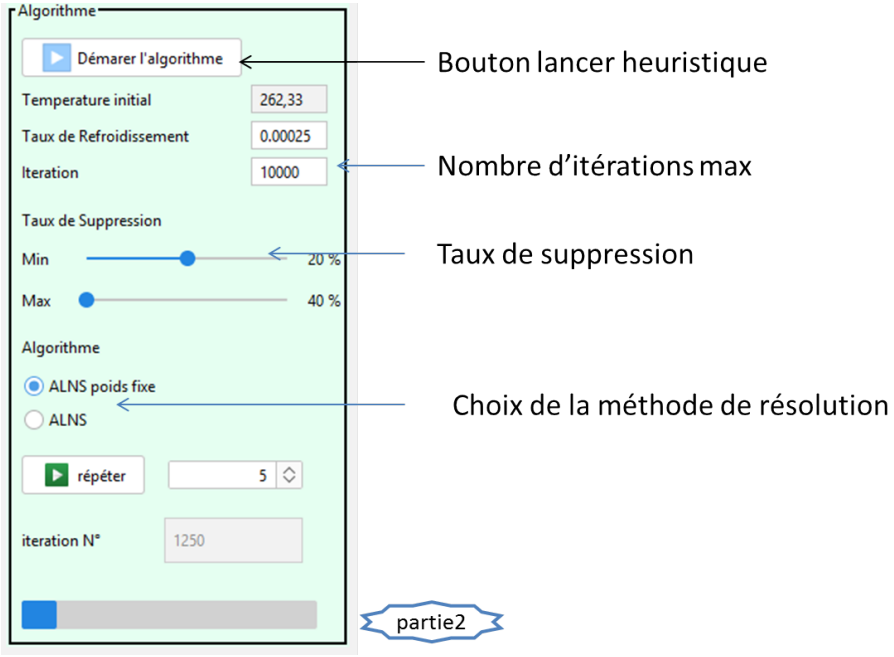

6 :Nuage de points des taches



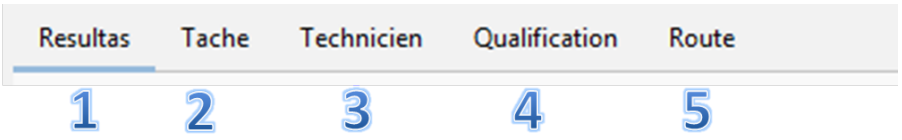

7 : Outil pour calculer la distance entre deux points



La fenêtre suivante présente la partie 2 :




La partie 3 :

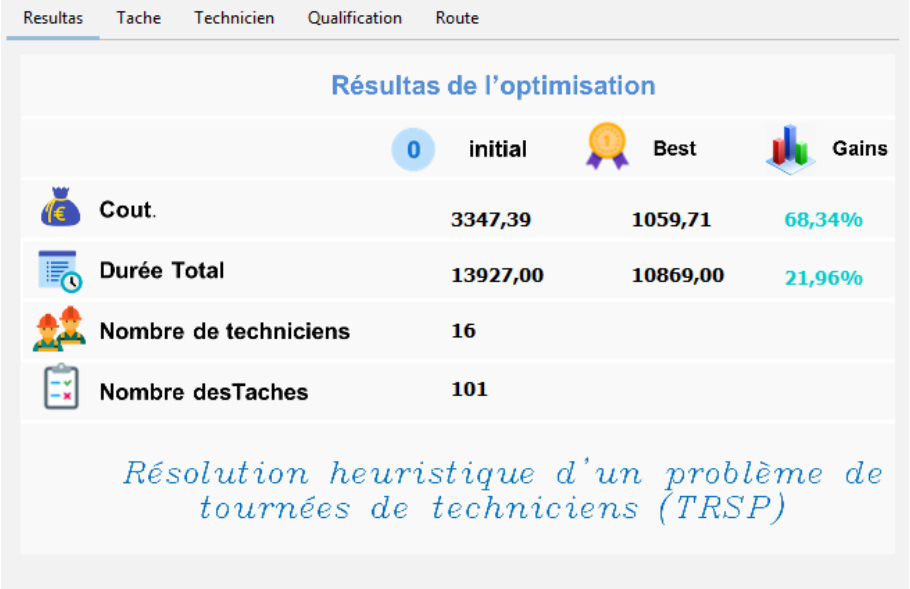



1:Résultats de l'optimisation

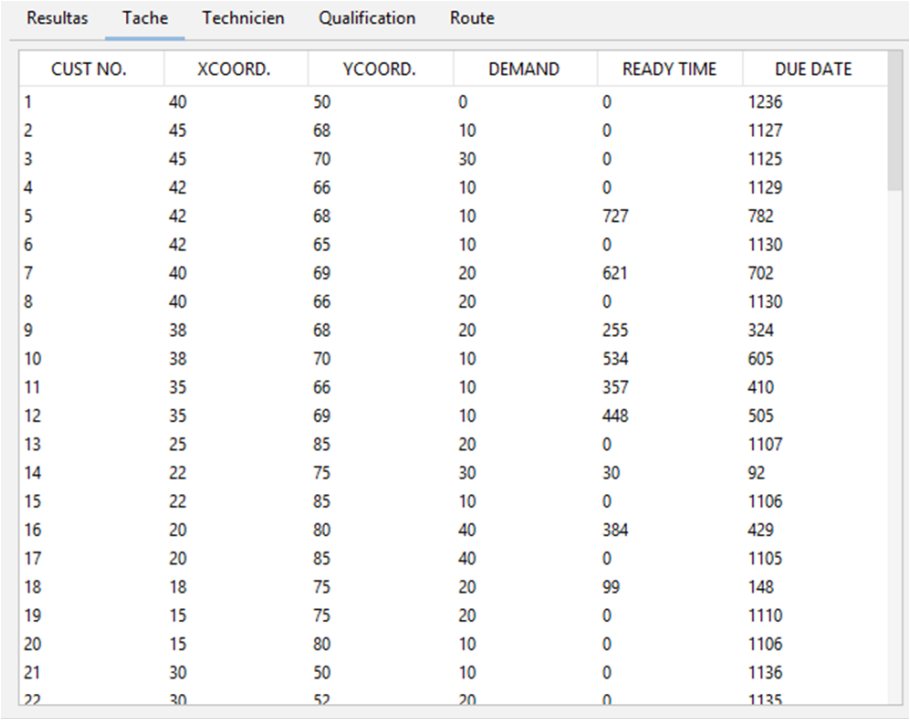
2:Tableau des tâches

3:Tableau des techniciens

1 : Résultats de l'optimisation du STRSP :



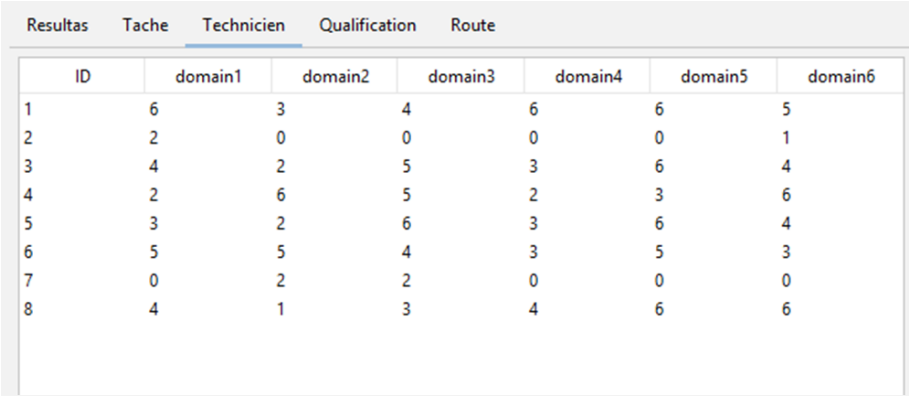

2 : Tableau des tâches :




| CUST NO. | XCOORD. | YCOORD. | DEMAND | READY TIME | DUE DATE |
|---|---|---|---|---|---|
| 1 | 40 | 50 | 0 | 0 | 1236 |
| 2 | 45 | 68 | 10 | 0 | 1127 |
| 3 | 45 | 70 | 30 | 0 | 1125 |
| 4 | 42 | 66 | 10 | 0 | 1129 |
| 5 | 42 | 68 | 10 | 727 | 782 |
| 6 | 42 | 65 | 10 | 0 | 1130 |
| 7 | 40 | 69 | 20 | 621 | 702 |
| 8 | 40 | 66 | 20 | 0 | 1130 |
| 9 | 38 | 68 | 20 | 255 | 324 |
| 10 | 38 | 70 | 10 | 534 | 605 |
| 11 | 35 | 66 | 10 | 357 | 410 |
| 12 | 35 | 69 | 10 | 448 | 505 |
| 13 | 25 | 85 | 20 | 0 | 1107 |
| 14 | 22 | 75 | 30 | 30 | 92 |
| 15 | 22 | 85 | 10 | 0 | 1106 |
| 16 | 20 | 80 | 40 | 384 | 429 |
| 17 | 20 | 85 | 40 | 0 | 1105 |
| 18 | 18 | 75 | 20 | 99 | 148 |
| 19 | 15 | 75 | 20 | 0 | 1110 |
| 20 | 15 | 80 | 10 | 0 | 1106 |
| 21 | 30 | 50 | 10 | 0 | 1136 |
| 22 | 30 | 52 | 20 | 0 | 1135 |

Tableau des tâches

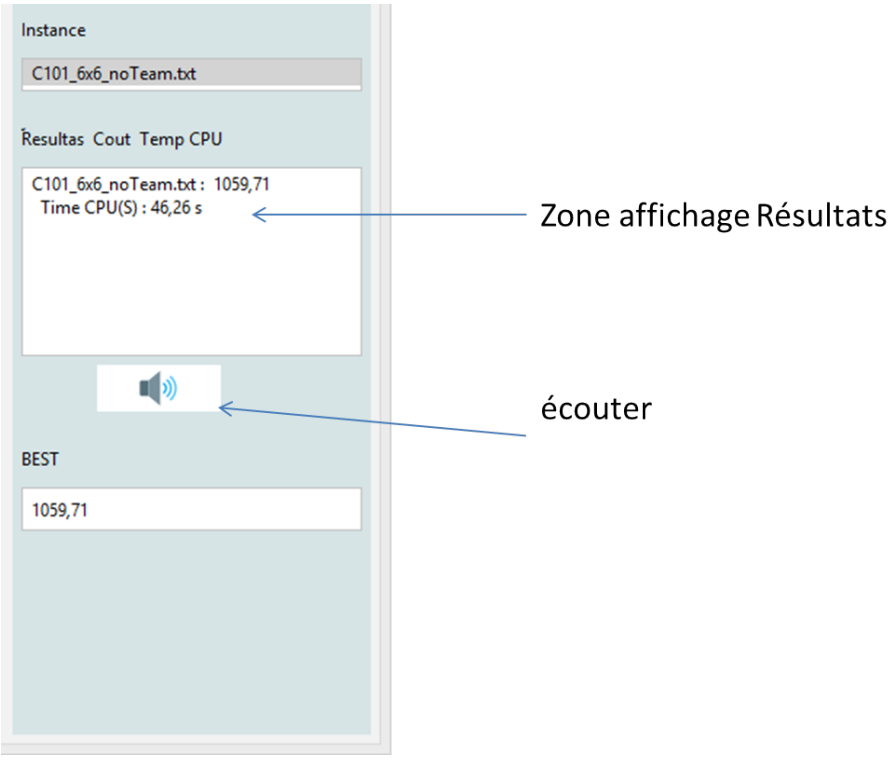
3 : Tableau des techniciens :




| ID | domain1 | domain2 | domain3 | domain4 | domain5 | domain6 |
|---|---|---|---|---|---|---|
| 1 | 6 | 3 | 4 | 6 | 6 | 5 |
| 2 | 2 | 0 | 0 | 0 | 0 | 1 |
| 3 | 4 | 2 | 5 | 3 | 6 | 4 |
| 4 | 2 | 6 | 5 | 2 | 3 | 6 |
| 5 | 3 | 2 | 6 | 3 | 6 | 4 |
| 6 | 5 | 5 | 4 | 3 | 5 | 3 |
| 7 | 0 | 2 | 2 | 0 | 0 | 0 |
| 8 | 4 | 1 | 3 | 4 | 6 | 6 |

Tableau des techniciens

La partie 3 :



## 3.4 Instances de test

Les expériences sont menées à l'aide des instances de problème de routage et d'ordonnancement des techniciens (TSRP) introduites par Kovacs et al [16]. Ces instances sont adaptées à partir des instances de référence de Solomon [27] pour le VRPTW et des instances de test fournies pour le challenge ROADEF [28]. Elles sont disponibles en ligne sur : `http://prolog.univie.ac.at/research/STRSP/`

L'ensemble des instances de Kovacs et al [16] sont générés à l'aide de 12 instances de Solomon [27], à savoir, R101, R103, R201, R203, C101, C103, C201, C203, RC101, RC103, RC201, RC203, où R, C et RC représentent respectivement, l'aléatoire (Random), regroupés (Clusters) et un mélange de paramètres géographiques aléatoires et regroupés. Les ensembles d'instances avec les préfixes R1, C1 et RC1 ont un horizon de planification court, tandis que ceux avec les préfixes R2, C2 et RC2 ont un horizon de planification long. Les deux derniers chiffres du nom de l'instance indiquent la densité de la fenêtre temporelle. Dans les instances 01, tous les clients sont associés à des créneaux horaires, alors que dans les ensembles 03, seuls 50% des clients ont des créneaux horaires. En termes de compétences requises, Kovacs et al. [16] génèrent trois types de matrices de compétences requises représentées par $5 \times 4$, $6 \times 6$ et $7 \times 4$ sur la base des données ROADEF, où les lignes des matrices correspondent aux domaines de compétences et les colonnes correspondent aux niveaux de compétences sous chaque domaine de compétences. Les données client des instances de Solomon sont couplées de manière aléatoire avec les données de compétence, ce qui donne un total de 36 instances de test. Toutes les instances ont 100 clients et un seul dépôt. Pour chaque cas, Kovacs et al. [16] définissent une version « équipe » et une version « sans équipe ». Puisque notre étude ne considère pas la possibilité d'équipes (team building), nous n'utilisons que la version "sans équipe" des instances dans nos expérimentations. Pour chaque instance, il existe deux ensembles de données sur les techniciens : l'un a un nombre suffisant de techniciens pour que la faisabilité puisse être réalisée sans externalisation, tandis que l'autre a un nombre limité de techniciens de sorte qu'il est impossible d'effectuer toutes les tâches sans l'utilisation de l'option d'externalisation. Le coût d'externalisation d'une tâche $i$ est défini comme $o_i = 200 + \mu_i^{1.5}$ , où $\mu_i$ mesure la difficulté de la tâche $i$, et est calculée comme la somme des compétences requises pour $i$ dans la matrice des compétences. Le coût d'externalisation est toujours supérieur au coût d'attribution d'une tâche à un technicien.

Le tableau 3.1 donne un aperçu du nombre de techniciens disponibles dans chacun des ensembles de données. Nous pouvons noter que chaque technicien a des compétences et des niveaux de compétences différents.

TABLE 3.1 – Nombre de techniciens par Dataset

| | Sans équipe | |
|---|---|---|
| | Complete | Reduced |
| C1*_5x4 | 17 | 8 |
| C2*_5x4 | 8 | 4 |
| R1*_5x4 | 25 | 12 |
| R2*_5x4 | 7 | 3 |
| RC1*_5x4 | 22 | 11 |
| RC2*_5x4 | 9 | 5 |
| | | |
| C1*_6x6 | 16 | 8 |
| C2*_6x6 | 7 | 4 |
| R1*_6x6 | 26 | 13 |
| R2*_6x6 | 7 | 4 |
| RC1*_6x6 | 24 | 12 |
| RC2*_6x6 | 8 | 4 |
| | | |
| C1*_7x4 | 17 | 9 |
| C2*_7x4 | 8 | 4 |
| R1*_7x4 | 28 | 14 |
| R2*_7x4 | 10 | 5 |
| RC1*_7x4 | 23 | 12 |
| RC2*_7x4 | 9 | 5 |

## 3.5 Réglage des paramètres

Les paramètres de l'ALNS sont réglés sur les valeurs données dans le tableau 3.2

TABLE 3.2 – Les paramètres de la méthode ALNS

| Paramètres | $\rho_{worst}$ | $\sigma 1$ | $\sigma 2$ | $\sigma 3$ | $\rho_{react}$ | Segment |
|---|---|---|---|---|---|---|
| Valeur | 3 | 33 | 9 | 13 | 0.1 | 100 |

L'ALNS proposée est basée sur un certain nombre de paramètres :

Dans worstRem le paramètre $\rho_{worst}$, contrôle la quantité de randomisation dans le mécanisme d'ajustement de poids adaptatif, la quantité d'ajustement de poids est contrôlée par les paramètres $\sigma\mathbf{1}$, $\sigma\mathbf{2}$, et $\sigma\mathbf{3}$, et le facteur de réaction $\rho_{react}$ qui doivent être définis.

## 3.6 Comparaison des Résultats

Cette section présente les résultats de l'évaluation de nos méta-heuristiques ALNSv1 (la colonne [ALNSv1]) et ALNSv2 (la colonne [ALNSv2]) par rapport à l'algorithme ALNS de Kovacs (2012) (la colonne [ALNS-2012]) et l'algorithme ILS de Xie (2017) (la colonne [ILS]) à l'aide d'instances de référence contenant 25, 50 et 100 tâches et un nombre suffisent de techniciens.

Dans les tableaux présentés ci-après, le premier groupe de colonnes indique l'identifiant de l'instance, le nombre de tâches |C|, et le nombre maximum de techniciens |K|. Colonnes Opt. et AVG. montrent, pour chaque instance, la valeur de solution optimale trouvée par CPLEX et les valeurs de solutions moyennes trouvées par ( l'ALNSv1, l'ALNSv2) , l'ALNS-2012 et l'ILS sur cinq exécutions aléatoires. Les colonnes ImpC/A ,Imp$A_{12}$/A et ImpI/A donnent les différences relatives en pourcentage entre les valeurs des solutions (ALNSv1, ALNSv2) et les solutions CPLEX ,les solutions ALNS-2012, et les solutions ILS respectivement. le temps CPU moyen en secondes sont rapportés dans la colonne intitulée CPU. Le gras dans les colonnes de l'ALNSv1 ou l'ALNSv2 est utilisé pour mettre en évidence les valeurs qui correspondent à une amélioration par rapport aux valeurs correspondantes de l'ALNS-2012 ou l'ILS.

. Le tableau 3.3 fournit une Comparaison des solutions exactes, ALNS-2012, ILS et **ALNSv1** sur de petites instances avec 25 tâches.
. Le tableau 3.4 fournit une Comparaison des solutions exactes, ALNS-2012, ILS et **ALNSv2** sur de petites instances avec 25 tâches.

. Le tableau 3.5 fournit une Comparaison des solutions exactes, ALNS-2012, ILS et **ALNSv1** sur de petites instances avec 50 tâches.
. Le tableau 3.6 fournit une Comparaison des solutions exactes, ALNS-2012, ILS et **ALNSv2** sur de petites instances avec 50 tâches.

TABLE 3.3 – Comparaison des solutions exactes, ALNS-2012, ILS et **ALNSv1** sur de petites instances avec 25 tâches

| **Instance** | **\|C\|** | **\|K\|** | **CPLEX** | | **ALNS-2012** | | **ILS** | | **ALNSv1** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **Opt.** | **CPU** | **Avg.** | **CPU** | **Avg.** | **CPU** | **Avg.** | **ImpC/A** | **ImpA$_{12}$/A** | **ImpI/A** | **CPU** |
| **C101_5x4** | 25 | 4 | 271.70 | 0.16 | 271.70 | 1.63 | 272.96 | 0.11 | **271,70** | 0,00 | 0,00 | 0.46 | 0,20 |
| **C201_5x4** | 25 | 2 | 863.08 | 0.09 | 863.08 | 2.45 | 863.08 | 0.09 | 863,08 | 0,00 | 0,00 | 0,00 | 0,11 |
| **C203_5x4** | 25 | 2 | 835.83 | 214.45 | 835.83 | 0.00 | 835.83 | 0.15 | 835,83 | 0,00 | 0,00 | 0,00 | 0,06 |
| **R101_5x4** | 25 | 4 | 2195.04 | 0.34 | 2195.04 | 1.26 | 2195.04 | 0.22 | 2195,04 | 0,00 | 0,00 | 0,00 | 0,34 |
| **R201_5x4** | 25 | 2 | 1091.07 | 0.86 | 1092.41 | 1.45 | 1091.07 | 0.03 | **1091,07** | 0,00 | 0,12 | 0,00 | 0,0,9 |
| **RC101_5x4** | 25 | 4 | 862.21 | 13.51 | 950.81 | 1.41 | 862.21 | 0.37 | 869,56 | -0,85 | 8,55 | -0,85 | 0,81 |
| **RC201_5x4** | 25 | 3 | 465.25 | 1.12 | 465.25 | 1.71 | 465.31 | 0.06 | **465,25** | 0,00 | 0,00 | 0,01 | 0,66 |
| **C101_6x6** | 25 | 4 | 927.35 | 0.14 | 927.35 | 1.77 | 927.35 | 0.09 | 927,35 | 0,00 | 0,00 | 0,00 | 0,72 |
| **C201_6x6** | 25 | 2 | 1217.10 | 0.13 | 1217.10 | 4.56 | 1217.10 | 0.01 | 1217,10 | 0,00 | 0,00 | 0,00 | 0,08 |
| **C203_6x6** | 25 | 2 | 930.60 | 5.73 | 930.60 | 1.86 | 930.60 | 0.03 | 930,60 | 0,00 | 0,00 | 0,00 | 0,61 |
| **R101_6x6** | 25 | 4 | 2857.05 | 0.17 | 2977.63 | 1.28 | 2868.19 | 0.30 | **2857,05** | 0,00 | 4,05 | 0,39 | 0,30 |
| **R201_6x6** | 25 | 2 | 1377.42 | 0.53 | 1377.42 | 1.76 | 1422.57 | 0.05 | **1377,96** | -0,04 | -0,04 | 3,14 | 0,32 |
| **RC201_6x6** | 25 | 3 | 1228.89 | 22.79 | 1228.89 | 1.40 | 1228.89 | 0.14 | 1228,89 | 0,00 | 0,00 | 0,00 | 0,58 |
| **C101_7x4** | 25 | 4 | 789.08 | 0.16 | 789.08 | 1.86 | 789.08 | 0.06 | 789,08 | 0,00 | 0,00 | 0,00 | 0,34 |
| **C103_7x4** | 25 | 4 | 671.06 | 3993.12 | 671.06 | 2.03 | 671.06 | 0.11 | 671,06 | 0,00 | 0,00 | 0,00 | 0,08 |
| **C201_7x4** | 25 | 2 | 738.35 | 0.05 | 738.35 | 1.60 | 738.35 | 0.02 | 738,35 | 0,00 | 0,00 | 0,00 | 0,36 |
| **C203_7x4** | 25 | 2 | 684.98 | 190.13 | 684.98 | 1.86 | 684.98 | 0.03 | 684,98 | 0,00 | 0,00 | 0,00 | 0,09 |
| **R101_7x4** | 25 | 4 | 2447.74 | 0.09 | 2447.74 | 1.27 | 2447.74 | 0.12 | 2447,74 | 0,00 | 0,00 | 0,00 | 0,21 |
| **R201_7x4** | 25 | 2 | 959.51 | 0.19 | 959.51 | 1.49 | 959.51 | 0.07 | 959,51 | 0,00 | 0,00 | 0,00 | 0,08 |
| **R203_7x4** | 25 | 2 | 849.47 | 496.32 | 849.47 | 1.88 | 849.47 | 0.03 | 849,47 | 0,00 | 0,00 | 0,00 | 0,06 |
| **RC201_7x4** | 25 | 3 | 967.60 | 5.46 | 967.60 | 2.00 | 967.60 | 0.08 | 967,60 | 0,00 | 0,00 | 0,00 | 0,50 |
| **Average** | | | 1106.21 | 235.50 | 1116.23 | 1.74 | 1108.95 | 0.10 | 1106,58 | -0,04 | 0,60 | 0,15 | 0,31 |

TABLE 3.4 – Comparaison des solutions exactes, ALNS-2012, ILS et **ALNSv2** sur de petites instances avec 25 tâches

| **Instance** | **\|C\|** | **\|K\|** | **CPLEX** | | **ALNS-2012** | | **ILS** | | **ALNSv2** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **Opt.** | **CPU** | **Avg.** | **CPU** | **Avg.** | **CPU** | **Avg.** | **ImpC/A** | **ImpA$_{12}$/A** | **ImpI/A** | **CPU** |
| **C101_5x4** | 25 | 4 | 271.70 | 0.16 | 271.70 | 1.63 | 272.96 | 0.11 | 274.84 | -1.16 | -1.16 | -0.69 | 0,20 |
| **C201_5x4** | 25 | 2 | 863.08 | 0.09 | 863.08 | 2.45 | 863.08 | 0.09 | 863,08 | 0,00 | 0,00 | 0,00 | 0,11 |
| **C203_5x4** | 25 | 2 | 835.83 | 214.45 | 835.83 | 0.00 | 835.83 | 0.15 | 835,83 | 0,00 | 0,00 | 0,00 | 0,06 |
| **R101_5x4** | 25 | 4 | 2195.04 | 0.34 | 2195.04 | 1.26 | 2195.04 | 0.22 | 2195,04 | 0,00 | 0,00 | 0,00 | 0,34 |
| **R201_5x4** | 25 | 2 | 1091.07 | 0.86 | 1092.41 | 1.45 | 1091.07 | 0.03 | **1091,07** | 0,00 | 0,12 | 0,00 | 0,0,9 |
| **RC101_5x4** | 25 | 4 | 862.21 | 13.51 | 950.81 | 1.41 | 862.21 | 0.37 | **862,21** | 0,00 | 9,32 | 0,00 | 0,81 |
| **RC201_5x4** | 25 | 3 | 465.25 | 1.12 | 465.25 | 1.71 | 465.31 | 0.06 | **465,25** | 0,00 | 0,00 | 0,01 | 0,66 |
| **C101_6x6** | 25 | 4 | 927.35 | 0.14 | 927.35 | 1.77 | 927.35 | 0.09 | 927,35 | 0,00 | 0,00 | 0,00 | 0,72 |
| **C201_6x6** | 25 | 2 | 1217.10 | 0.13 | 1217.10 | 4.56 | 1217.10 | 0.01 | 1217,10 | 0,00 | 0,00 | 0,00 | 0,08 |
| **C203_6x6** | 25 | 2 | 930.60 | 5.73 | 930.60 | 1.86 | 930.60 | 0.03 | 930,60 | 0,00 | 0,00 | 0,00 | 0,61 |
| **R101_6x6** | 25 | 4 | 2857.05 | 0.17 | 2977.63 | 1.28 | 2868.19 | 0.30 | **2857,05** | 0,00 | 4,05 | 0,39 | 0,30 |
| **R201_6x6** | 25 | 2 | 1377.42 | 0.53 | 1377.42 | 1.76 | 1422.57 | 0.05 | **1377,96** | -0,04 | -0,04 | 3,14 | 0,32 |
| **RC201_6x6** | 25 | 3 | 1228.89 | 22.79 | 1228.89 | 1.40 | 1228.89 | 0.14 | 1228,89 | 0,00 | 0,00 | 0,00 | 0,58 |
| **C101_7x4** | 25 | 4 | 789.08 | 0.16 | 789.08 | 1.86 | 789.08 | 0.06 | 789,08 | 0,00 | 0,00 | 0,00 | 0,34 |
| **C103_7x4** | 25 | 4 | 671.06 | 3993.12 | 671.06 | 2.03 | 671.06 | 0.11 | 671,06 | 0,00 | 0,00 | 0,00 | 0,08 |
| **C201_7x4** | 25 | 2 | 738.35 | 0.05 | 738.35 | 1.60 | 738.35 | 0.02 | 738,35 | 0,00 | 0,00 | 0,00 | 0,36 |
| **C203_7x4** | 25 | 2 | 684.98 | 190.13 | 684.98 | 1.86 | 684.98 | 0.03 | 684,98 | 0,00 | 0,00 | 0,00 | 0,09 |
| **R101_7x4** | 25 | 4 | 2447.74 | 0.09 | 2447.74 | 1.27 | 2447.74 | 0.12 | 2447,74 | 0,00 | 0,00 | 0,00 | 0,21 |
| **R201_7x4** | 25 | 2 | 959.51 | 0.19 | 959.51 | 1.49 | 959.51 | 0.07 | 959,51 | 0,00 | 0,00 | 0,00 | 0,08 |
| **R203_7x4** | 25 | 2 | 849.47 | 496.32 | 849.47 | 1.88 | 849.47 | 0.03 | 849,47 | 0,00 | 0,00 | 0,00 | 0,06 |
| **RC201_7x4** | 25 | 3 | 967.60 | 5.46 | 967.60 | 2.00 | 967.60 | 0.08 | 967,60 | 0,00 | 0,00 | 0,00 | 0,50 |
| **Average** | | | 1106.21 | 235.50 | 1116.23 | 1.74 | 1108.95 | 0.10 | 1106,38 | -0,06 | 0,59 | 0,14 | 0,31 |

TABLE 3.5 – Comparaison des solutions exactes, ALNS-2012, ILS et **ALNSv1** sur de petites instances avec 50 tâches

| **Instance** | **\|C\|** | **\|K\|** | **CPLEX** | | **ALNS-2012** | | **ILS** | | **ALNSv1** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **Opt.** | **CPU** | **Avg.** | **CPU** | **Avg.** | **CPU** | **Avg.** | **ImpC/A** | **ImpA$_{12}$/A** | **ImpI/A** | **CPU** |
| **C101_5x4** | 50 | 6 | 830,00 | 2,61 | 838,11 | 6,98 | 830,00 | 1,09 | **830,86** | -0,10 | 0,87 | -0,10 | 1,21 |
| **C201_5x4** | 50 | 4 | 859,54 | 0,59 | 859,54 | 7,60 | 859,54 | 0,78 | 859,54 | 0,00 | 0,00 | 0,00 | 0,17 |
| **R201_5x4** | 50 | 4 | 1107,51 | 107,52 | 1107,51 | 7,49 | 1112,25 | 2,13 | **1107,51** | 0,00 | 0,00 | 0,43 | 0,63 |
| **C101_6x6** | 50 | 6 | 1154,84 | 251,94 | 1181,31 | 7,68 | 1154,84 | 1,95 | **1154,84** | 0,00 | 2,24 | 0,00 | 4,12 |
| **C201_6x6** | 50 | 4 | 1203,93 | 0,42 | 1203,93 | 8,15 | 1203,93 | 0,66 | 1203,93 | 0,00 | 0,00 | 0,00 | 0,49 |
| **R201_6x6** | 50 | 4 | 1647,07 | 794,06 | 1647,00 | 8,27 | 1649,95 | 1,99 | **1647,27** | -0,01 | -0,02 | 0,16 | 0,65 |
| **C101_7x4** | 50 | 6 | 1356,54 | 7,61 | 1367,75 | 7,71 | 1367,75 | 1,26 | 1369,18 | -0,93 | -0,10 | -0,10 | 2,21 |
| **C201_7x4** | 50 | 4 | 1312,21 | 0,42 | 1312,21 | 7,87 | 1312,21 | 0,37 | 1312,21 | 0,00 | 0,00 | 0,00 | 0,60 |
| **R201_7x4** | 50 | 4 | 1553,23 | 129,01 | 1553,23 | 7,19 | 1553,23 | 0,93 | 1553,23 | 0,00 | 0,00 | 0,00 | 1,23 |
| **Average** | | | 1224.99 | 143.80 | 1230.07 | 7,66 | 1226.51 | 1,24 | 1226.51 | -0,12 | 0,33 | 0,04 | 1,07 |

TABLE 3.6 – Comparaison des solutions exactes, ALNS-2012, ILS et **ALNSv2** sur de petites instances avec 50 tâches.

| **Instance** | **\|C\|** | **\|K\|** | **CPLEX** | | **ALNS-2012** | | **ILS** | | **ALNSv2** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | **Opt.** | **CPU** | **Avg.** | **CPU** | **Avg.** | **CPU** | **Avg.** | **ImpC/A** | **ImpA$_{12}$/A** | **ImpI/A** | **CPU** |
| **C101_5x4** | 50 | 6 | 830,00 | 2,61 | 838,11 | 6,98 | 830,00 | 1,09 | **830,00** | 0,00 | 0,97 | 0,00 | 1,10 |
| **C201_5x4** | 50 | 4 | 859,54 | 0,59 | 859,54 | 7,60 | 859,54 | 0,78 | 859,54 | 0,00 | 0,00 | 0,00 | 0,21 |
| **R201_5x4** | 50 | 4 | 1107,51 | 107,52 | 1107,51 | 7,49 | 1112,25 | 2,13 | **1107,51** | 0,00 | 0,00 | 0,43 | 0,67 |
| **C101_6x6** | 50 | 6 | 1154,84 | 251,94 | 1181,31 | 7,68 | 1154,84 | 1,95 | **1154,84** | 0,00 | 2,24 | 0,00 | 5,51 |
| **C201_6x6** | 50 | 4 | 1203,93 | 0,42 | 1203,93 | 8,15 | 1203,93 | 0,66 | 1203,93 | 0,00 | 0,00 | 0,00 | 0,70 |
| **R201_6x6** | 50 | 4 | 1647,07 | 794,06 | 1647,00 | 8,27 | 1649,95 | 1,99 | **1647,27** | -0,01 | -0,02 | 0,16 | 0,45 |
| **C101_7x4** | 50 | 6 | 1356,54 | 7,61 | 1367,75 | 7,71 | 1367,75 | 1,26 | **1356,54** | 0,00 | 0,82 | 0,82 | 1,91 |
| **C201_7x4** | 50 | 4 | 1312,21 | 0,42 | 1312,21 | 7,87 | 1312,21 | 0,37 | 1312,21 | 0,00 | 0,00 | 0,00 | 0,51 |
| **R201_7x4** | 50 | 4 | 1553,23 | 129,01 | 1553,23 | 7,19 | 1553,23 | 0,93 | 1553,23 | 0,00 | 0,00 | 0,00 | 1,49 |
| **Average** | | | 1224.99 | 143.80 | 1230.07 | 7.66 | 1227.08 | 1.24 | 1225.01 | -0,001 | 0.45 | 0,16 | 1,25 |

Pour les tableaux 3.3 et 3.4 : (avec 25 tâches et petites instances)

Le tableau 3.3 donne des résultats expérimentaux sur de petites instances contenant 25 tâches. Par rapport à CPLEX, notre algorithme **ALNSv1** trouve systématiquement des solutions optimales dans les cinq exécutions aléatoires pour **19** des **21** instances et produit un écart moyen global de -0.04% sur toutes les instances. Par rapport à l'ALNS-2012, notre algorithme **ALNSv1** donne de meilleures solutions pour trois instances avec écart moyen global de 0.60% sur toutes les instances, en particulier $RC101 - 5 \times 4$ (améliorent la solution de 8,55% ). En comparant ILS avec ALNSv1, notre algorithme donne de meilleures solutions pour quatres instances avec écart moyen global de 0.15% sur toutes les instances.

Le tableau 3.4 donne des résultats expérimentaux sur de petites instances contenant 25 tâches. Par rapport à CPLEX, notre algorithme **ALNSv2** trouve systématiquement des solutions optimales dans les cinq exécutions aléatoires pour **19** des 21 instances et produit un écart moyen global de -0.06% sur toutes les instances. Par rapport à l'ALNS-2012, notre algorithme **ALNSv2** donne de meilleures solutions pour trois instances avec écart moyen global de 0.59% sur toutes les instances, en particulier $RC101-5\times4$ (améliorent la solution de 9,32%). Entre ILS et **ALNSv2**, notre algorithme donne de meilleures solutions pour trois instances avec un écart moyen global de 0.14% sur toutes les instances.

Pour les tableaux 3.5 et 3.6 : (avec 50 tâches et petites instances)

Le tableau 3.5 présente les résultats des expérimentations sur des instances avec **50 tâches**. Sur les **9** instances, notre algorithme **ALNSv1** découvre des solutions optimales pour **6** et produit un écart moyen global de **-0,12 %** par rapport à CPLEX. L'écart moyen de l'ALNSv1 par rapport à l'ALNS-2012 en termes de valeurs de solution est de **0,33%**, et il trouve de meilleures solutions pour **2** instances. Par rapport à l'ILS, notre algorithme **ALNSv1** donne de meilleures solutions pour **2** instances avec un écart moyen global de **0.04%** sur toutes les instances.

Le tableau 3.6 présente les résultats des expérimentations sur des instances avec **50 tâches**. Sur les **9** instances, notre algorithme **ALNSv2** découvre des solutions optimales pour **8** et produit un écart moyen global de **-0,001 %** par rapport à CPLEX. L'écart moyen de l'ALNSv2 par rapport à l'ALNS-2012 en termes de valeurs de solution est de **0,45%**, et il trouve de meilleures solutions pour **3** instances. Par rapport à l'ILS, notre algorithme **ALNSv2** donne de meilleures solutions pour **3** instances avec moyen global de **0.16%** sur toutes les instances.

Pour les instances avec 100 tâches, les tableaux 3.7, 3.8, 3.9 et 3.10 rapportent les résultats de calcul sur 36 instances avec un nombre de techniciens suffisant et insuffisant (complete / reduced).

Avec les instances « **sans équipe / complete** » :

. Le tableau 3.7 fournit une comparaison de l'ALNSv2 avec l'ALNS-2012 et l'ILS
. Le tableau 3.8 fournit une comparaison de l'ALNSv1 avec l'ALNS-2012 et l'ILS

Avec les instances « **sans équipe / reduced** » :

. Le tableau 3.9 fournit une comparaison de l'ALNSv2 avec l'ALNS-2012 et l'ILS
. Le tableau 3.10 fournit une comparaison de l'ALNSv1 avec l'ALNS-2012 et l'ILS

De plus, nous rapportons les valeurs des solutions minimales trouvées par l'ALNS-2012 ,ILS ,ALNSv2 et ALNSv1 ; sur cinq exécutions aléatoires dans des colonnes intitulées Min, et les différences correspondantes en pourcentage entre les valeurs des solutions ALNS-2012 et ILS, ALNSv2 et ALNSv1 sont présentées dans la colonne Imp$A_{12}$/A et Imp A/I.

Dans les colonnes de l'ALNSv1 ou de l'ALNSv2 : **le gras coloré** est utilisé pour mettre en évidence les valeurs qui correspondent à une amélioration par rapport aux valeurs correspondantes de l'ALNS-2012 ou de l'ILS. **Le gras** est utilisé pour les valeurs identiques par rapport aux valeurs correspondantes de l'ALNS-2012 ou de l'ILS et **le gras souligné** pour les résultats optimaux globaux .

TABLE 3.7 – Comparaison des solutions ALNS-2012, ILS et **ALNSv2** aux instances « **sans équipe / complete** » avec 100 tâches

| | CPLEX | ALNS-2012 | ILS | ALNSv2 | | |
|---|---|---|---|---|---|---|
| Instance | Best | Min | Min | Min | $ImpA_{12}/A$ | ImpI/A |
| C101_5x4 | 1096,85 | 1098,71 | 1097,67 | 1099,19 | -0,04 | -0,14 |
| C103_5x4 | 2604,28 | 1018,61 | 1012,86 | **1017,31** | 0,13 | -0,44 |
| C201_5x4 | 1157,56 | 1158,97 | 1157,56 | **1157,56** | 0,12 | 0,00 |
| C203_5x4 | 2449,89 | 1046,93 | 1054,21 | **1046,93** | 0,00 | 0,69 |
| R101_5x4 | 1652,13 | 1687,68 | 1658,93 | **1652,26** | 2,10 | 0,40 |
| R103_5x4 | 2604,28 | 1238,67 | 1235,05 | 1258,08 | -1,57 | -1,86 |
| R201_5x4 | 1569,42 | 1440,30 | 1431,16 | **1430,84** | 0,66 | 0,02 |
| R203_5x4 | 2477,25 | 1098,00 | 1097,55 | **1096,48** | 0,14 | 0,10 |
| RC101_5x4 | 2503,98 | 1702,51 | 1673,94 | 1727,73 | -1,48 | -3,21 |
| RC103_5x4 | 6810,66 | 1337,99 | 1321,66 | 1344,65 | -0,50 | -1,74 |
| RC201_5x4 | 1849,86 | 1601,89 | 1589,24 | **1587,96** | 0,87 | 0,08 |
| RC203_5x4 | 2724,35 | 1161,53 | 1162,95 | **1162,12** | -0,05 | 0,07 |
| C101_6x6 | 972,89 | 989,21 | 972,89 | **978,36** | 1,10 | -0,56 |
| C103_6x6 | 3897,18 | 893,94 | 900,82 | 909,65 | -1,76 | -0,98 |
| C201_6x6 | 821,55 | 821,55 | 821,55 | **821,55** | 0,00 | 0,00 |
| C203_6x6 | 2164,58 | 689,60 | 689,6 | 690,01 | -0,06 | -0,06 |
| R101_6x6 | 1648,27 | 1658,27 | 1657,55 | **1652,17** | 0,37 | 0,32 |
| R103_6x6 | 15070,3 | 1223,63 | 1216,08 | **1222,31** | 0,11 | -0,51 |
| R201_6x6 | 1462,9 | 1261,94 | 1265,56 | **1260,75** | 0,09 | 0,38 |
| R203_6x6 | 3329,47 | 932,35 | 930,74 | **917,26** | 1,62 | 1,45 |
| RC101_6x6 | 2317,68 | 1679,13 | 1663,3 | **1674,95** | 0,25 | -0,70 |
| RC103_6x6 | 5614,42 | 1281,55 | 1297,09 | 1294,94 | -1,04 | 0,17 |
| RC201_6x6 | 1849,86 | 1395,40 | 1367,89 | **1393,62** | 0,13 | -1,88 |
| RC203_6x6 | 2337,76 | 1001,04 | 1003,81 | 1006,64 | -0,56 | -0,28 |
| C101_7x4 | 1357,05 | 1357,05 | 1357,05 | 1372,45 | -1,13 | -1,13 |
| C103_7x4 | 3506,12 | 1215,70 | 1220,19 | **1216,28** | -0,05 | 0,32 |
| C201_7x4 | 1256,30 | 1256,30 | 1256,3 | **1256,30** | 0,00 | 0,00 |
| C203_7x4 | 2401,88 | 1150,85 | 1137,07 | **1133,47** | 1,51 | 0,32 |
| R101_7x4 | 1764,78 | 1776,46 | 1781,13 | **1773,57** | 0,16 | 0,42 |
| R103_7x4 | 7412,24 | 1346,80 | 1337,92 | **1335,23** | 0,86 | 0,20 |
| R201_7x4 | 1411,36 | 1398,14 | 1401,68 | **1396,08** | 0,15 | 0,40 |
| R203_7x4 | 2476,19 | 1164,90 | 1160,51 | **1163,28** | 0,14 | -0,24 |
| RC101_7x4 | 1902,94 | 1821,90 | 1805,39 | 1857,37 | -1,95 | -2,88 |
| RC103_7x4 | 3367,93 | 1435,63 | 1427,4 | 1451,36 | -1,10 | -1,68 |
| RC201_7x4 | 1823,30 | 1697,82 | 1697,82 | **1697,82** | 0,00 | 0,00 |
| RC203_7x4 | 2070,17 | 1239,45 | 1230,63 | **1219,44** | 1,61 | 0,91 |
| Average | 2826,05 | 1285,57 | 1280,35 | 1285,44 | 0,02 | -0,33 |

TABLE 3.8 – Comparaison des solutions ALNS-2012, ILS et **ALNSv1** aux instances « **sans équipe / complete** » avec 100 tâches

| | CPLEX | ALNS-2012 | ILS | ALNSv1 | | |
|---|---|---|---|---|---|---|
| Instance | Best | Min | Min | Min | $ImpA_{12}/A$ | ImpI/A |
| C101_5x4 | 1096,85 | 1098,71 | 1097,67 | **1096,86** | 0,17 | 0,07 |
| C103_5x4 | 2604,28 | 1018,61 | 1012,86 | 1047,3 | -2,82 | -3,40 |
| C201_5x4 | 1157,56 | 1158,97 | 1157,56 | **1157,56** | 0,12 | 0,00 |
| C203_5x4 | 2449,89 | 1046,93 | 1054,21 | **1046,93** | 0,00 | 0,69 |
| R101_5x4 | 1652,13 | 1687,68 | 1658,93 | **1661,72** | 1,54 | -0,17 |
| R103_5x4 | 2604,28 | 1238,67 | 1235,05 | 1244,16 | -0,44 | -0,74 |
| R201_5x4 | 1569,42 | 1440,30 | 1431,16 | **1432,52** | 0,54 | -0,10 |
| R203_5x4 | 2477,25 | 1098,00 | 1097,55 | 1101,13 | -0,29 | -0,33 |
| RC101_5x4 | 2503,98 | 1702,51 | 1673,94 | 1713,68 | -0,66 | -2,37 |
| RC103_5x4 | 6810,66 | 1337,99 | 1321,66 | 1350,07 | -0,90 | -2,15 |
| RC201_5x4 | 1849,86 | 1601,89 | 1589,24 | **1591,59** | 0,64 | -0,15 |
| RC203_5x4 | 2724,35 | 1161,53 | 1162,95 | 1168,91 | -0,64 | -0,51 |
| C101_6x6 | 972,89 | 989,21 | 972,89 | **989,21** | 0,00 | -1,68 |
| C103_6x6 | 3897,18 | 893,94 | 900,82 | 932,55 | -4,32 | -3,52 |
| C201_6x6 | 821,55 | 821,55 | 821,55 | **821,55** | 0,00 | 0,00 |
| C203_6x6 | 2164,58 | 689,60 | 689,6 | **689,60** | 0,00 | 0,00 |
| R101_6x6 | 1648,27 | 1658,27 | 1657,55 | **1656,4** | 0,11 | 0,07 |
| R103_6x6 | 15070,3 | 1223,63 | 1216,08 | **1223,09** | 0,04 | -0,58 |
| R201_6x6 | 1462,90 | 1261,94 | 1265,56 | **1260,19** | 0,14 | 0,42 |
| R203_6x6 | 3329,47 | 932,35 | 930,74 | **919,99** | 1,33 | 1,15 |
| RC101_6x6 | 2317,68 | 1679,13 | 1663,30 | **1674,56** | 0,27 | -0,68 |
| RC103_6x6 | 5614,42 | 1281,55 | 1297,09 | **1277,33** | 0,33 | 1,52 |
| RC201_6x6 | 1849,86 | 1395,40 | 1367,89 | **1395,39** | 0,00 | -2,01 |
| RC203_6x6 | 2337,76 | 1001,04 | 1003,81 | **1002,7** | -0,17 | 0,11 |
| C101_7x4 | 1357,05 | 1357,05 | 1357,05 | 1364,98 | -0,58 | -0,58 |
| C103_7x4 | 3506,12 | 1215,70 | 1220,19 | 1220,64 | -0,41 | -0,04 |
| C201_7x4 | 1256,30 | 1256,30 | 1256,30 | **1256,3** | 0,00 | 0,00 |
| C203_7x4 | 2401,88 | 1150,85 | 1137,07 | **1150,85** | 0,00 | -1,21 |
| R101_7x4 | 1764,78 | 1776,46 | 1781,13 | **1778,83** | -0,13 | 0,13 |
| R103_7x4 | 7412,24 | 1346,80 | 1337,92 | **1342,27** | 0,34 | -0,33 |
| R201_7x4 | 1411,36 | 1398,14 | 1401,68 | **1399,43** | -0,09 | 0,16 |
| R203_7x4 | 2476,19 | 1164,90 | 1160,51 | **1162,5** | 0,21 | -0,17 |
| RC101_7x4 | 1902,94 | 1821,90 | 1805,39 | **1815,66** | 0,34 | -0,57 |
| RC103_7x4 | 3367,93 | 1435,63 | 1427,40 | 1448,63 | -0,91 | -1,49 |
| RC201_7x4 | 1823,30 | 1697,82 | 1697,82 | **1697,82** | 0,00 | 0,00 |
| RC203_7x4 | 2070,17 | 1239,45 | 1230,63 | **1232,20** | 0,58 | -0,13 |
| Average | 2826,05 | 1285,57 | 1280,35 | 1286,81 | -0,16 | -0,52 |

TABLE 3.9 – Comparaison des solutions ALNS-2012, ILS et **ALNSv2** aux instances « **sans équipe / reduced** » avec 100 tâches

| Instance | CPLEX | ALNS-2012 | ILS | ALNSv2 | | |
|---|---|---|---|---|---|---|
| | Best | Min | Min | Min | $ImpA_{12}/A$ | ImpI/A |
| C101_5x4 | 5857.35 | 5656,63 | 5589,76 | **5616,51** | 0,71 | -0,48 |
| C103_5x4 | 7874.13 | 2644,65 | 2650,69 | 2818,91 | -6,59 | -6,35 |
| C201_5x4 | 2755.52 | 2755,52 | 2755,52 | 2756,42 | -0,03 | -0,03 |
| C203_5x4 | 3630.92 | 2389,37 | 2383,01 | **2387,25** | 0,09 | -0,18 |
| R101_5x4 | 5446.89 | 5582,58 | 5561,83 | **5507,54** | 1,34 | 0,98 |
| R103_5x4 | 4856.22 | 1710,25 | 1658,23 | 1933,57 | -13,06 | -16,60 |
| R201_5x4 | 3295.67 | 2838,50 | 2839,54 | 2839,88 | -0,05 | -0,01 |
| R203_5x4 | 4042.66 | 2332,23 | 2335,76 | **2332,11** | 0,01 | 0,16 |
| RC101_5x4 | 5441.33 | 5127,79 | 4909,89 | **5106,96** | 0,41 | -4,01 |
| RC103_5x4 | 6142.71 | 2170,57 | 2301,86 | 2332,72 | -7,47 | -1,34 |
| RC201_5x4 | 3579.44 | 3088,23 | 3083,49 | **3069,8** | 0,60 | 0,44 |
| RC203_5x4 | 3579.20 | 2516,16 | 2512,64 | 2526,95 | -0,43 | -0,57 |
| C101_6x6 | 7660.86 | 7731,07 | 7660,86 | **7728,77** | 0,03 | -0,89 |
| C103_6x6 | 9050.71 | 4980,70 | 4971,66 | 5042,42 | -1,24 | -1,42 |
| C201_6x6 | 3315.30 | 3278,07 | 3298,68 | 3333,44 | -1,69 | -1,05 |
| C203_6x6 | 5443.79 | 2460,17 | 2468,53 | **2461,06** | -0,04 | 0,30 |
| R101_6x6 | 5944.91 | 5955,17 | 5948,71 | **5946,98** | 0,14 | 0,03 |
| R103_6x6 | 8799.51 | 2251,64 | 2225,67 | **2205,63** | 2,04 | 0,90 |
| R201_6x6 | 5169.21 | 3503,40 | 3510,36 | **3462,53** | 1,17 | 1,36 |
| R203_6x6 | 5595.34 | 2437,28 | 2443,97 | **2434,9** | 0,10 | 0,37 |
| RC101_6x6 | 7621.03 | 5276,34 | 4975,34 | **4982,13** | 5,58 | -0,14 |
| RC103_6x6 | 6878.60 | 2263,83 | 2113,03 | 2349,84 | -3,80 | -11,21 |
| RC201_6x6 | 7017.96 | 4422,86 | 4490,33 | 4526,31 | -2,34 | -0,80 |
| RC203_6x6 | 5683.37 | 2649,51 | 2671,23 | 2692,1 | -1,61 | -0,78 |
| C101_7x4 | 5208.99 | 5208,30 | 5246,13 | 5256,22 | -0,92 | -0,19 |
| C103_7x4 | 6657.08 | 2020,40 | 2009,86 | 2223,52 | -10,05 | -10,63 |
| C201_7x4 | 2773.41 | 2773,41 | 2781,07 | **2773,41** | 0,00 | 0,28 |
| C203_7x4 | 3048.56 | 2261,37 | 2262,00 | 2267,55 | -0,27 | -0,25 |
| R101_7x4 | 5079.67 | 5239,81 | 5127,29 | **5128,84** | 2,12 | -0,03 |
| R103_7x4 | 5603.74 | 2104,92 | 2139,77 | **2091,61** | 0,63 | 2,25 |
| R201_7x4 | 2790.76 | 2672,96 | 2664,93 | **2639,09** | 1,27 | 0,97 |
| R203_7x4 | 3209.88 | 2199,10 | 2209,32 | **2202,1** | -0,14 | 0,33 |
| RC101_7x4 | 5692.20 | 5531,06 | 5373,05 | **5363,72** | 3,03 | 0,17 |
| RC103_7x4 | 7660.38 | 2586,03 | 2591,39 | **2575,59** | 0,40 | 0,61 |
| RC201_7x4 | 3278.57 | 2919,83 | 2910,73 | **2888,67** | 1,07 | 0,76 |
| RC203_7x4 | 3739.80 | 2277,62 | 2305,62 | **2303,05** | -1,12 | 0,11 |
| Average | 5261,82 | 3439,37 | 3416,16 | 3447,45 | -0,84 | -1,30 |

TABLE 3.10 – Comparaison des solutions ALNS-2012, ILS et **ALNSv1** aux instances « **sans équipe / reduced** » avec 100 tâches

| Instance | CPLEX | ALNS-2012 | ILS | ALNSv1 | | |
|---|---|---|---|---|---|---|
| | Best | Min | Min | Min | $ImpA_{12}$/A | ImpI/A |
| C101_5x4 | 5857.35 | 5656,63 | 5589,76 | **5652,55** | 0,07 | -1,12 |
| C103_5x4 | 7874.13 | 2644,65 | 2650,69 | 2819,24 | -6,60 | -6,36 |
| C201_5x4 | 2755.52 | 2755,52 | 2755,52 | **2755,52** | 0,00 | 0,00 |
| C203_5x4 | 3630.92 | 2389,37 | 2383,01 | **2386,02** | 0,14 | -0,13 |
| R101_5x4 | 5446.89 | 5582,58 | 5561,83 | **5579,92** | 0,05 | -0,33 |
| R103_5x4 | 4856.22 | 1710,25 | 1658,23 | 1912,7 | -11,84 | -15,35 |
| R201_5x4 | 3295.67 | 2838,50 | 2839,54 | 2849,96 | -0,40 | -0,37 |
| R203_5x4 | 4042.66 | 2332,23 | 2335,76 | **2333,62** | -0,06 | 0,09 |
| RC101_5x4 | 5441.33 | 5127,79 | 4909,89 | **4892,23** | 4,59 | 0,36 |
| RC103_5x4 | 6142.71 | 2170,57 | 2301,86 | 2452,95 | -13,01 | -6,56 |
| RC201_5x4 | 3579.44 | 3088,23 | 3083,49 | **3083,74** | 0,15 | -0,01 |
| RC203_5x4 | 3579.20 | 2516,16 | 2512,64 | 2522,9 | -0,27 | -0,41 |
| C101_6x6 | 7660.86 | 7731,07 | 7660,86 | **7728,77** | 0,03 | -0,89 |
| C103_6x6 | 9050.71 | 4980,70 | 4971,66 | 5092,72 | -2,25 | -2,44 |
| C201_6x6 | 3315.30 | 3278,07 | 3298,68 | 3343,12 | -1,98 | -1,35 |
| C203_6x6 | 5443.79 | 2460,17 | 2468,53 | **2457,15** | 0,12 | 0,46 |
| R101_6x6 | 5944.91 | 5955,17 | 5948,71 | 6016,71 | -1,03 | -1,14 |
| R103_6x6 | 8799.51 | 2251,64 | 2225,67 | 2308,9 | -2,54 | -3,74 |
| R201_6x6 | 5169.21 | 3503,40 | 3510,36 | **3463,21** | 1,15 | 1,34 |
| R203_6x6 | 5595.34 | 2437,28 | 2443,97 | **2434,90** | 0,10 | 0,37 |
| RC101_6x6 | 7621.03 | 5276,34 | 4975,34 | **5083,87** | 3,65 | -2,18 |
| RC103_6x6 | 6878.60 | 2263,83 | 2113,03 | **2243,28** | 0,91 | -6,16 |
| RC201_6x6 | 7017.96 | 4422,86 | 4490,33 | 4501,36 | -1,77 | -0,25 |
| RC203_6x6 | 5683.37 | 2649,51 | 2671,23 | **2664,34** | -0,56 | 0,26 |
| C101_7x4 | 5208.99 | 5208,30 | 5246,13 | 5247,63 | -0,76 | -0,03 |
| C103_7x4 | 6657.08 | 2020,40 | 2009,86 | 2189,51 | -8,37 | -8,94 |
| C201_7x4 | 2773.41 | 2773,41 | 2781,07 | **2773,41** | 0,00 | 0,28 |
| C203_7x4 | 3048.56 | 2261,37 | 2262,00 | **2261,64** | -0,01 | 0,02 |
| R101_7x4 | 5079.67 | 5239,81 | 5127,29 | **5083,70** | 2,98 | 0,85 |
| R103_7x4 | 5603.74 | 2104,92 | 2139,77 | **2085,64** | 0,92 | 2,53 |
| R201_7x4 | 2790.76 | 2672,96 | 2664,93 | **2654,47** | 0,69 | 0,39 |
| R203_7x4 | 3209.88 | 2199,10 | 2209,32 | **2203,52** | -0,20 | 0,26 |
| RC101_7x4 | 5692.20 | 5531,06 | 5373,05 | **5444,89** | 1,56 | -1,34 |
| RC103_7x4 | 7660.38 | 2586,03 | 2591,39 | **2575,59** | 0,40 | 0,61 |
| RC201_7x4 | 3278.57 | 2919,83 | 2910,73 | **2888,67** | 1,07 | 0,76 |
| RC203_7x4 | 3739.80 | 2277,62 | 2305,62 | 2307,73 | -1,32 | -0,09 |
| Average | 5261,82 | 3439,37 | 3416,16 | 3452,67 | -0,96 | -1,41 |

## 3.7 Interprétation des résultats

Après avoir exécuté nos algorithmes ALNSv2 et ALNSv1, les résultats obtenus représentés dans les tableaux 3.7 3.8 3.9 et 3.10 sont comparés avec ceux trouvés par Kovacs(ALNS) [2012] [16] et Xie(ILS)[2017] [21], dans les deux cas (complete ) et (reduced).

Nous pouvons bien remarqué que nos résultats sont très satisfaisants et bien améliorés par rapport à ceux trouvés par ces travaux antérieurs.

En effet, les comparaisons pour les deux cas étudiés fait sortir les conclusions suivantes :

- Avec les instances « **sans équipe / complete** »

1. Par rapport à Kovacs (ALNS-2012) [16], nos résultats de **l'ALNSv2** montrent, **19** instances sur 36 sont améliorées, **4** instances sont identiques et **13** instances sont dans le même champ et proches de la meilleure valeur ; avec un écart moyen global de **0.02%** sur toutes les instances. (Tableau 3.7).

   Cependant, par comparaison avec les résultats de Xie 2017 **(ILS)** [21], nous avons amélioré **16** instances sur 36 , **4** instances sont identiques et **16** instances sont dans le même champ et proches de la meilleure valeur ; avec un écart moyen global de **-0.33%** sur toutes les instances.

   ET nous avons obtenu **3** résultats optimums globaux .(Tableau 3.7).

2. Par rapport à Kovacs (ALNS-2012) [16], nos résultats de **l'ALNSv1** montrent, **15** instances sur 36 sont améliorées, **8** instances sont identiques et **13** instances sont dans le même champ et proches de la meilleure valeur ; avec un écart moyen global de **-0.16%** sur toutes les instances.

   Cependant, par comparaison avec les résultats de Xie 2017 **(ILS)** [21], nous avons amélioré **9** instances sur 36 , **5** instances sont identiques et **22** instances sont dans le même champ et proches de la meilleure valeur ; avec un écart moyen global de **-0.52%** sur toutes les instances.

   ET nous avons obtenu **3** résultats optimums globaux .(Tableau 3.8).

- Avec les instances « **sans équipe / reduced** »

1. Par rapport à Kovacs **(ALNS-2012)** [16], nos résultats de **l'ALNSv2** montrent, **18** instances sur 36 sont améliorées, **1** instance sont identiques et **17** instances sont dans le même champ et proches de la meilleure valeur ; avec un écart moyen global de **-0,84%** sur toutes les instances.(Tableau 3.9).

   Cependant, par comparaison avec les résultats de Xie 2017 **(ILS)** [21], nous avons amélioré **16** instances sur 36, et **20** instances sont dans le même champ et proches de la meilleure valeur ; avec un écart moyen global de **-1.30%** sur toutes les instances. (Tableau 3.9).

   Et nous avons obtenu **un** résultat optimal global .

2. Par rapport à Kovacs **(ALNS-2012)** [16], nos résultats de **l'ALNSv1** montrent, **17** instances sur 36 sont améliorées, **2** instances sont identiques et **17** instances sont dans le même champ et proches de la meilleure valeur ; avec un écart moyen global de **-0.96%** sur toutes les instances.(Tableau 3.10).

Cependant, par comparaison avec les résultats de Xie 2017 **(ILS)** [16], nous avons amélioré **14** instances sur 36, **1** instance est identique et **21** instances sont dans le même champ et proches de la meilleure valeur ; avec un écart moyen global de **-1.41%** sur toutes les instances. (Tableau (Tableau 3.10).

Et nous avons obtenu **deux** résultats optimums globaux .

Enfin, nous pouvons conclure que parmi les 36 instances qui ont été résolues, dans chaque cas, par notre algorithme, il y a des instances meilleures, des instances idéales (optimales) et des cas proches de la meilleure valeur.

## 3.8 Conclusion

Dans ce chapitre nous avons présenté le matériel et les outils logiciels utilisés dans le développement de la méthode proposée, ainsi que les interfaces graphiques. En plus, nous avons présenté et discuté les résultats obtenus par l'application de nos algorithmes (ALNSv2 et ALNSv1) sur STRSP. A la fin, nous avons comparé les résultats de notre approche avec ceux trouvés par Kovacs [2012] [16] ainsi que ceux trouvés par Xie [2017] (ILS) [21]. Notre algorithme a donné des résultats beaucoup plus meilleurs que ceux obtenus par ces travaux antérieurs.

# CONCLUSION GÉNÉRALE ET PERSPECTIVES

Les problèmes de tournées des techniciens STRSP (Service Technician Routing and Scheduling Problems) font partie intégrante du quotidien des décideurs et planificateurs. Malgré leur apparente simplicité, les problèmes de tournées des techniciens sont des problèmes NP difficiles, et leur étude est utile pour comprendre et modéliser plusieurs problèmes de ce genre.

Dans ce mémoire, nous avons commencé d'abord par la définition et la description du problème de tournées de techniciens, ce qui nous a permis de mettre en place la formulation mathématique qui régie ce type de problèmes et d'étudier les méthodes de résolution de ces problèmes trouvées dans les travaux antérieurs. par la suite nous avons présenté la approche utiliser pour résudre le probleme STRSP.Cette approche est basée sur une méthode d'optimisation : ALNS (Recherche adaptative à large voisinage). Il s'agit d'utiliser des opérateurs de destruction et de réparation afin de modifier une solution courante et trouver la solution optimale. Dans ce travail, nous avons proposé deux algorithmes ALNSv1 et ALNSv2 , qui ont prouvé leur efficacité dans la résolution du problème STRSP.

Pour tester nos algorithmes, nous avons utilisé des instances de Kovacs, inspirées de celles de Solomon, impliquant 25, 50 et 100 tâches. Nous avons pu apporter des améliorations dans les résultats obtenus.

Les résultats obtenus sont très encourageants et prometteurs. Les comparaisons avec des travaux antérieurs, montrent que nos résultats sont bien compétitifs. Nous tenons à noter que plusieurs nouvelles meilleures solutions et plusieurs résultats optimaux globaux ont été trouvés.

Comme perspectives futures de ce travail, nous proposons la résolution d'autres versions de ce problème ( Version avec équipe).

# BIBLIOGRAPHIE


[1] Ines Mathlouthi. Méthodes de résolution exactes et heuristiques pour un problème de tournées de techniciens. 2018.

[2] Jean-François Cordeau, Gilbert Laporte, Federico Pasin, and Stefan Ropke. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4) :393–409, 2010.

[3] Jiyang Xu Steve Y. Chiu. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7 :495– 509, 2001.

[4] Arturo Castillo-Salazar, Dario Landa-Silva, and Rong Qu. A survey of workforce scheduling and routing problems. pp :283–302, 2012.

[5] Eleni Hadjiconstantinou and Daron Roberts. Routing under uncertainty : an application in the scheduling of field service engineers. In *The vehicle routing problem*, pages 331–352. SIAM, 2002.

[6] Sachidanand V Begur, David M Miller, and Jerry R Weaver. An integrated spatial dss for scheduling and routing home-health-care nurses. *Interfaces*, 27(4) :35–48, 1997.

[7] Eddie Cheng and Jennifer Lynn Rich. A home health care routing and scheduling problem. Technical report, 1998.

[8] Aboud J. Fernandez C. Laporte G. Ramirez Weintraub, A. An emergency vehicle dispatching system for an electric utility in chile. *Journal of the Operational Research Society*, 50 :690–696, 1999.

[9] Krumke S. O. Rambau J. Torres L. M.In U. LeopoldWildburger F. Rendl G. Wäscher (Eds.) Grötschel, M. Online-dispatching of automobile service unitsoperations research proceedings 2002 (sor 2002). *Berlin : Springer.*, pp :166–173, 2002.

[10] M Douiri, Souad Elbernoussi, and Halima Lakhbab. Cours des méthodes de résolution exactes heuristiques et métaheuristiques. *Université Mohamed V, Faculté des sciences de Rabat*, pages 5–87, 2009.

[11] Victor Pillac, Christelle Gueret, and Andrés L Medaglia. A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7) :1525–1535, 2013.

[12] Salhi Ahmed and TarekBouktir. *Contribution a l'optimisation de l'ecoulement de puissance en utilisant la logique floue associee aux reseaux de neurones ( neuro-flou.* PhD thesis, Université De Biskra, 2015.

[13] Fred Glover. Tabu search. *ORSA Journal on computing*, 1(3) :190–206, 1989.

[14] Whitley D. A genetic algorithm tutorial. *Stat Comput*, 4(2) :65–85, 1994.

[15] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer, 1998.

[16] Attila A Kovacs, Sophie N Parragh, Karl F Doerner, and Richard F Hartl. Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of scheduling*, 15(5) :579–600, 2012.

[17] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8) :2403–2435, 2007.

[18] David Pisinger and Stefan Ropke. Large neighborhood search. In *Handbook of metaheuristics*, pages 399–419. Springer, 2010.

[19] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4) :455–472, 2006.

[20] Stefan Ropke and David Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3) :750–775, 2006.

[21] Tolga Bektas, Fulin Xie, and Chris Potts. Iterated local search for workforce scheduling and routing problems. *Journal of Heuristics*, 23, 12 2017.

[22] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598) :671–680, 1983.

[23] Jean-Yves Potvin and Jean-Marc Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3) :331–340, 1993.

[24] William A Dees and Patrick G Karger. Automated rip-up and reroute techniques. In *19th Design Automation Conference*, pages 432–439. IEEE, 1982.

[25] George FD Duff. Differences, derivatives, and decreasing rearrangements. *Canadian Journal of Mathematics*, 19 :1153–1178, 1967.

[26] Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2) :139–171, 2000.

[27] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2) :254–265, 1987.

[28] ROADEF Challenge. Technicians and interventions scheduling for telecommunications, 2007.