Université de Technologie de Belfort Montbéliard

Département informatique

IT45

Problèmes d'Affectation, de Planification et de Routage des tournées des Employés

---

Analyse et Conception

Présenté par :
Maureen Grandidier
Simon Nemo

Enseignants :
Mira Bou-Saleh
Olivier Grunder

Table des matières

Présentation du problème

1 Objectif

Notre objectif est de créer un circuit hamiltonien pour chacun des employés en :

— Harmonisant la charge de travail et la distance parcourue pour chaque employé

— Minimisant le nombre d'affectation dont la spécialité est insatisfaite.

— Minimisant le nombre d'heures supplémentaires, heures perdues, et la distance totale parcourue.

2 Contraintes dures

Les contraintes dures sont les contraintes qui se doivent d'être respectées dans la partition en sortie de l'algorithme.

— Un intervenant ne peut réaliser qu'une mission à la fois

— Toutes les missions doivent être affectées

— Une mission ne peut être effectuée que par un intervenant ayant les bonnes compétences

— Respecter les heures maximales de travail par jour (Temps plein = 8h, temps partiel = 6h)

3 Contraintes souples

Les contraintes souples sont les contraintes qui peuvent être ou ne pas être respectées dans la partition en sortie de l'algorithme (elles sont considérées comme des "préférences").

— Une mission doit être effectuée si possible par un intervenant ayant les bonnes spécialités

— Chaque intervenant doit avoir une pause d'au moins 1h entre midi et 14h

— Respecter la limite des heures supplémentaires autorisées (10h/semaine, 2h/jour)

— L'amplitude d'une journée de travail ne doit pas dépasser 12h

— Un intervenant doit avoir le temps de se déplacer d'une mission à une autre.

— Une mission est effectuée par un et un seul intervenant (une mission n'est affectée qu'une fois)

Chapitre I : Méthodes de résolution

1 Introduction

Ce problème s'apparente à un VRP ou Vehicle Routing Problem. C'est une extension classique du problème du voyageur de commerce, et il fait partie de la classe des problèmes NP-complet. Ce qui signifie que tous les algorithmes connus pour résoudre ce problème ont un temps d'exécution exponentiel avec la taille des données d'entrée, et sont donc inexploitables en pratique même pour des instances de taille modérée. Pour ce type de problème, on distingue les méthodes exactes et les méthodes approchées. Les méthodes exactes fournissent la meilleure solution mais sont limitées à des petites instances de problèmes. En effet, le temps de résolution augmente rapidement lorsque le nombre de variable de décision augmente également. Or, pour un établissement SESSAD, le jeu de données est beaucoup trop grand. Il faut donc mettre en place un algorithme permettant d'obtenir une solution approchée.

2 Heuristiques

Des méthodes heuristiques ont été conçues pour explorer seulement des parties de l'espace de recherche, se concentrant dans ces parties qui semblent contenir une amélioration des solutions. De ce fait, le temps requis pour obtenir une solution est réduit mais la solution trouvée est rarement optimale. De plus, ces algorithmes ne permettent pas de générer plusieurs solutions, une optimisation en cascade est alors impossible.

3 Métaheuristiques

Une métaheuristique est une heuristique générique qui peut s'appliquer à n'importe quel problème d'optimisation mais qui nécessite une adaptation au problème concerné (contrairement aux heuristiques qui sont en général associées à un problème particulier).

Les métaheuristiques partagent en général trois particularités :

— Elles partent d'une ou plusieurs solutions initiales générées aléatoirement dans la plupart des cas, ou construites à partir d'une heuristique dans certains cas.

— Elles mémorisent des solutions ou des caractéristiques des solutions visitées durant le processus de recherche afin de ne pas tomber dans des solutions déjà visitées ou dans un optimum local.

— Elles utilisent une procédure qui permet de créer une nouvelle solution à partir de la solution courante et des informations mémorisées. Il s'agit d'un élément essentiel de la recherche car ceci permet une meilleure exploration de l'espace des solutions.

En cours, nous avons vu 2 algorithmes de ce type :

— La recherche tabou

— Les algorithmes génétiques

3.1 Tabou

Les avantages de cet algorithme sont qu'il est très efficace sur de nombreux problèmes d'optimisation et qu'il est simple à comprendre et donc à mettre en place. Cependant, les paramètres peuvent être difficiles à régler, donc difficile à le rendre efficace et il est gourmand en ressource (taille de la liste tabou). Cet algorithme n'assure aucune convergence.

3.2 Génétique

Les algorithmes génétiques sont des algorithmes d'optimisation s'appuyant sur des techniques dérivées de la génétique et de l'évolution naturelle : croisements, mutations, sélection, etc. Les avantages de cet algorithme sont qu'il permet de créer de nombreuses solutions (intéressant pour une optimisation en cascade) et qu'il permet de jouer sur de nombreux paramètres. Cependant, ce type d'algorithme est difficile à mettre en place et n'assure en aucun cas d'avoir la solution optimale malgré un nombre d'itération élevé. Le temps de calcul peut être élevé car la fonction d'évaluation est évaluée de nombreuses fois. Il existe également le risque d'être coincé dans un optima local mais cet inconvénient peut être contourné en lançant plusieurs fois l'algorithme et en jouant avec les paramètres (ex : taux de mutation) permettant par ailleurs d'augmenter le nombre de solutions trouvées que l'on pourra utilisée pour l'optimisation en cascade.

L'algorithme génétique semble plus adapté à notre problème car il permet de produire de nombreuses solutions permettant une optimisation en cascade et est plus flexible. C'est donc cet algorithme que nous allons choisir. Il faudra prêter une attention particulière à la fonction d'évaluation afin de limiter le temps de calcul.

4 Codage des solutions

Chaque individu de la population représente une solution au problème à optimiser. Un individu est représenté par un chromosome (un chromosome représente une solution). Ce chromosome est constitué de gènes (un gène représente un élément d'une solution). A chaque individu est associé une évaluation (appelée aussi fitness) qui mesure la qualité de la solution. L'évaluation représente la performance de l'individu vis-à-vis du problème. La pertinence du codage va dépendre du choix des opérateurs de reproduction et l'efficacité globale de l'algorithme.

Il existe plusieurs types de codage pour les valeurs attribuées aux gènes : binaire, réelle, par structure arborescente ou encore par permutation.

Dans la permutation, les chromosomes contiennent une séquence de gênes, dans laquelle l'ordre est significatif, et où les gênes sont des nombres entiers.

Un chromosome ici est une suite de nœuds. Il y a autant de noeud que de missions, les chiffres représentent l'id des missions et ils sont classés dans l'ordre dans lequel les intervenants doivent réaliser les missions. Ce type de codage est appelé codage par liste de permutation.

Nous utilisons également un codage direct : toute l'information de la solution est présente dans le chromosome. Chaque permutation contient à la fois les missions et des séparateurs de tournées. Nous pensons utiliser -1 comme séparateur de tournées, ce qui représente le fait de partir le matin du SESSAD et d'y retourner le soir pour finir sa journée de travail. Le reste des chiffres représentent les missions. Ainsi chaque intervenant a 5 tournées (lundi, mardi, mercredi, jeudi, vendredi) séparées par des -1 dans le chromosome.

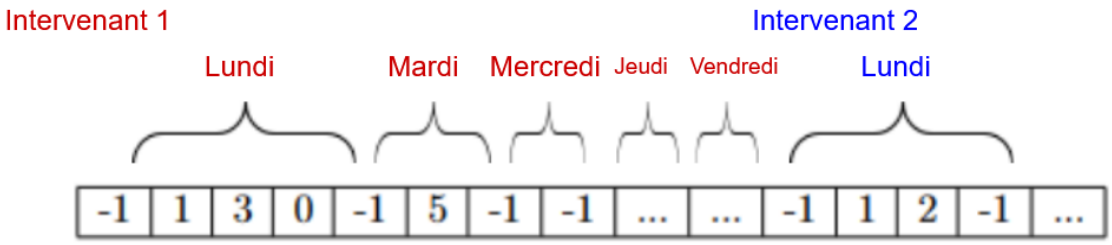

FIGURE 1 – Codage direct

On peut également utiliser -10, -11, -12, ..., -15 pour représenter le séparateur de l'intervenant 1 et -20, -21, ..., -25 pour l'intervenant 2 etc, nous verrons au moment de la programmation ce qui nous aide le plus.

Nous pensons traiter les séparateurs comme les autres gênes lors des croisements ou autre, et nous appliquerons sûrement une procédure de réparation si les enfants ne respectent pas les contraintes. Sinon, si cela nous est trop difficile, nous pourrons toujours essayer un codage indirect et un découpage.

5 Génération de population

Au niveau de la génération de la population, on a le choix entre plusieurs possibilités :

— Initialisation aléatoire

— utilisation d'heuristiques

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Étant donné que notre optimum dans l'espace d'état du problème actuel nous est totalement inconnu, il est naturel de faire une initialisation aléatoire la plus uniforme possible afin de favoriser une exploration de l'espace de recherche maximum.

6 Fonction d'adaptation (fitness)

Notre problème est de type multi-objectifs, puisque nous cherchons par exemple à minimiser le nombre d'affectations dont la spécialité est insatisfaite, à atteindre un équilibre entre les employés, entre les heures supplémentaires de chacun, les heures non-travaillées, la distance parcourue etc.

L'algorithme optimise à partir de la 1ère fitness donnée et on applique des pénalités pour chaque contrainte souple non respectée. L'algorithme générera plusieurs solutions considérées comme "bonne" puis on optimisera en cascade avec le second critère, on retiendra de nouveau plusieurs solutions considérées comme les meilleures, puis on finira par optimiser avec le troisième critère pour obtenir la solution optimale qui convient aux trois fitness.

Voici les trois fitness que nous allons utiliser :

$$f_{employees} = \frac{\zeta \cdot \sigma_{WH}(s) + \gamma \cdot \sigma_{OH}(s) + \kappa \cdot \sigma_D(s)}{3}$$

$$f_{students} = \alpha \cdot penalties(s)$$

$$f_{SESSAD} = \frac{\beta \cdot sumWOH(s) + \kappa \cdot moyD(s) + \kappa \cdot maxD(s)}{3}$$

avec :

- $\sigma_{WH}(s)$ = écart type des heures non-travaillées (wasted hours) des employés pour s
- $\sigma_{OH}(s)$ = écart type des heures supplémentaires des employés pour s
- $\sigma_D(s)$ = écart type des distances des employés pour s
- $penalties(s)$ = nombre d'affectations dont la spécialité est insatisfaite pour s
- $sumWOH(s)$ = somme des heures non-travaillées et des heures supplémentaires de tous les employés pour s
- $moyD(s)$ = distance moyenne parcourue par les employés pour s
- $maxD(s)$ = distance maximale parcourue par les employés pour s
- $\alpha, \beta, \gamma, \zeta$ et κ sont des facteurs de corrélation.
  $\alpha = 100$ / nombre total de missions dans le problème
  $\beta = 100$ / nombre total d'heures qu'un employé peut travailler en semaine (45h)
  $\gamma = 100$ / nombre total des heures supplémentaires tolérées (10h)
  $\zeta = 100$ / moyenne des heures du quota du travail des employés
  $\kappa = 100$ / moyenne de toutes les distances ($\frac{\sum_{m \in M}(d_{(center,m)} + d_{(m,center)}}{nombred'intervenants}$)

7 Sélection des individus

Le principe des algorithmes génétiques repose sur le principe de la sélection naturelle.

Les individus sont sélectionnés pour la phase de reproduction avec une probabilité proportionnelle à leur adaptation.

Il existe de nombreuses techniques de sélection, nous allons présenter celles que nous avons vu en cours et bien entendu notre choix final :

— La sélection aléatoire : comme son nom l'indique, ce type de sélection choisit le chromosome selon une distribution uniforme.

— La sélection par roulette : elle consiste à associer à chaque chromosome une probabilité d'être sélectionné proportionnelle à son fitness. Avec cette technique, les individus avec les meilleures fitness ont plus de chance d'être sélectionné. Cependant, avec cette méthode on risque d'obtenir un "super-héros", un individu avec une probabilité de sélection très élevée qui sera presque systématiquement toujours choisi (et fera perdre en diversité), ce qui empêche l'exploration de possible meilleures solutions.

— Sélection par tournoi : Cette technique tire au hasard deux ou plusieurs individus de la population et le plus fort est sélectionné, c'est-à-dire celui ayant le fitness le plus intéressant. Cette technique a comme avantage de ne pas avoir à trier la population et d'éviter le risque du "super-héros" mais le meilleur individu peut ne pas être sélectionné (champ d'exploration réduit)

Pour ce projet, nous avons choisi la sélection par tournoi qui est l'une des sélections les plus utilisées dans les algorithmes évolutionnaires.

8 Reproduction

Les 2 méthodes les plus courantes sont :

— Croisement : produire 1 ou plusieurs enfants à partir d'un ou plusieurs parents

— Mutation : un individu mute et donne un nouvel individu

D'autres méthodes existent mais sont beaucoup plus complexes : recherche locales, algorithmes de colonies de fourmis, règles basées sur l'indicateurs de similarité entre individus...

Il faudra que l'on fasse attention lors des croisements et mutations que les enfants produits respectent bien les contraintes. Pour cela, nous pourrons toujours déplacer les séparateurs jusqu'à trouver une solution. Et s'il est impossible de les "réparer" nous les exclurons.

Au niveau des croisements, il en existe plus de 22. Dans un premier temps, nous utiliserons le croisement en un point, et selon les résultats que nous obtiendrons, nous changerons de stratégie ou non. (Nous pensons utiliser sinon le Partially Mapped Crossover).

8.1 Croisement en un point

La première étape consiste à choisir aléatoirement un point de coupure pour partager chaque parent en deux parties. Puis le premier enfant est construit en utilisant la première partie du premier parent et la deuxième partie du deuxième parent. A l'inverse, le deuxième enfant est une concaténation de la seconde partie du premier parent et de la première partie du second parent.

Mais cet opérateur s'il est appliqué aux problèmes représentés sous forme de permutations, comme le VRP, a de fortes chances de produire des enfants invalides, par duplication et/ou omission de certains éléments de la permutation. On doit donc adapté cet opérateur.

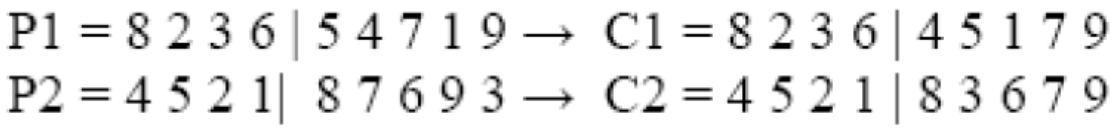

FIGURE 2 – Croisement en un point pour codage par permutation

8.2 Partially Mapped Crossover (PMX)

Le but de cet opérateur de croisement est de construire un enfant par le choix de sous séquences ordonnancées de l'un des parents et de préserver l'ordre et la position d'autant de sous séquences que possible des autres parents. La sous-séquence de l'ordonnancement est sélectionnée par le choix de deux points de coupure aléatoire, lesquels servent de frontière pour l'opération de substitution.

Considérons par exemple les deux parents :

P1 = 123 | 4567 | 89
P2 = 452 | 1876 | 93

Étape 1 : Ces deux parents vont produire deux enfants. Dans cette première étape, les segments compris entre les points de coupures sont échangés :

E1 = xxx | 1876 | xx
E2 = xxx | 4567 | xx

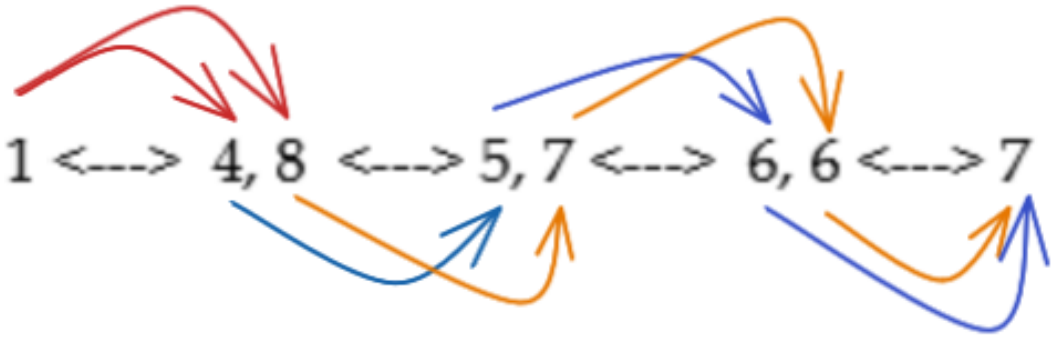

FIGURE 3 – Table de correspondance

Notons qu'en vue de procéder à l'étape 3 de résolution des conflits, il est possible d'établir entre les deux points de coupures une table de correspondances entre les allèles. Cette table est définie comme suit dans notre exemple :

(ainsi après l'allèle 1 il y a aura soit l'allèle 4 soit 8, après le 4 il y aura le 5, après le 8 il y aura le 7, après le 5 ou 7 il y aura le 6...)

Étape 2 : les chromosomes enfants sont complétés en transmettant les allèles non conflictuelles issus du second parent.

E1 = x23 | 1876 | x9
E2 = xx2 | 4567 | 93

Étape 3 : cette dernière étape consiste à terminer l'élaboration des descendants en résolvant les conflits à l'aide de la table de correspondance :

E1 = 423 | 1876 | 59
E2 = 182 | 4567 | 93

Au niveau des mutations, il en existe en tout 6 types. Nous commencerons notre programmation du problème avec l'opérateur de mutation par permutation qui est un des opérateurs de mutation les plus répandus. Bien entendu, nous changerons si besoin.

8.3 Mutation par permutation en codage réel

Lors de cette mutation, 2 positions sont sélectionnées au hasard et les gènes situés dans ces positions sont permutés.

Chromosome de départ : 1 **2** 3 4 5 6 7 **8** 9

Après permutation : 1 **8** 3 4 5 6 7 2 **9**

9 Remplacement

Après l'application des opérateurs de reproduction, les enfants sont évalués. Il faut ensuite déterminer la génération suivante (appelé remplacement) en déterminant quel individus devront disparaître de la population à chaque génération et quels individus fort survivent dans la population de la prochaine génération. Les quatre paragraphes suivants présentent quatre types de stratégies de remplacement.

9.1 Remplacement des stratégies d'évolution

Dans les stratégies d'évolution, la taille de la population de la prochaine génération est plus petite que la taille de la population des enfants. Soit on choisi les meilleurs individus de la population des enfants, soit on les choisit de la population des enfants et la population courante conjointes.

9.2 Remplacement de type Steady-state

A chaque génération, un (pu deux) enfant(s) seulement est (sont) généré(s). Ils remplacent le plus mauvais individu de la population courante. En d'autre termes, la population de la prochaine génération est la population courante, excepté le plus mauvais individu qui est remplacé par une ou deux solutions de la population des enfants. Ainsi la population de la prochaine génération est est peut-être plus grande que la population courante. Cependant, la plupart des algorithmes évolutionnaires sont appliqués sur des populations avec des tailles fixes en gardant au moins le meilleur individu dans la population courante.

9.3 Remplacement élitiste

L'élitisme est une façon de protéger la rémanence de bonnes solutions et d'assurer leur survie tout au long de la recherche. Ici, la population de la prochaine génération est choisie à partir de la population des enfants et de la population courante. Cette sélection à l'avantage de permettre une convergence plus rapide des solutions, mais au détriment de la diversité des individus.

9.4 Remplacement générationnel

La population des enfants remplace systématiquement la population courante. Ainsi, la population de prochaine génération est égale à la population des enfants. C'est cette stratégie que nous allons utiliser.

10 Conditions d'arrêt

Nous avons décidé mettre en place des conditions d'arrêts qui seront définies en argument dans la console.

Il y aura :

— Un critère d'arrêt sur le nombre d'itération (soit le nombre de génération)

— Un critère d'arrêt lorsque le temps maximal de calcul choisi est dépassé

Au critère d'arrêt, l'algorithme retourne la meilleure solution qu'il a obtenu.

Références

Table des figures