Projet IT45 : Problème d'affectation d'employés
Dossier d'analyse / conception

Sommaire

I. Les paramètres du problème

Plusieurs paramètres sont à prendre en compte dans notre problème :

- La distance parcourue par chaque intervenant.e.s au cours de la semaine doit être minimisée.
- L'intervenant.e envoyé doit posséder la compétence adaptée à la personne aidée.
- La satisfaction des spécialités doit être maximisée.
- Le planning des intervenant.e.s doit respecter les lois sur les horaires de travail. Les intervenant.e.s sont à temps plein, cela signifie qu'ils peuvent travailler au maximum 35h par semaine. Une journée ne peut pas dépasser 8h de travail sur une amplitude de 12h et il doit y avoir une pause de minimum 1h à midi. Après examen du générateur d'instance, on en conclut que les horaires de formations maximales sont 8h - 12h puis 13h - 19h. Les problèmes de l'amplitude de 12h et de la pause de midi ne seront donc pas à vérifier lors de l'admissibilité d'une solution.

II. Le choix de l'algorithme utilisé pour la résolution

A. Les différentes pistes qui ont été envisagées

Notre première idée fut de coupler un algorithme de plus court chemin avec un algorithme d'affectation. En effet pour minimiser la distance à parcourir nous avions dans l'idée d'implémenter l'algorithme de Little puisque chaque intervenant.e réalise une boucle dont le point de départ et d'arrivé est le SESSAD. Pour ce qui est de la partie affectation des intervenant.e.s aux différents apprenants nous avons pensé à l'algorithme de Gale et Shapley qui était notamment utilisé par APB. Dans notre cas, au lieu de réaliser un classement des intervenant.e.s par personne en situation de handicap, nous avions réfléchi à un système avec 2 listes. La première contenant les intervenants qui possèdent à la fois les bonnes compétences et spécialités. Tandis que la deuxième aurait été composée des intervenants possédant uniquement la bonne compétence. Le classement au sein des listes aurait alors été fait en fonction des distances ce qui nous aurait permis de lier les deux algorithmes. Cependant cette idée a été abandonnée car elle ne permettait pas de tenir compte des contraintes horaires.

B. Le choix qui a été fait et pourquoi

Finalement, nous avons fait le choix d'utiliser un algorithme génétique pour la résolution du problème. En effet, parmi les différents paramètres que l'on doit prendre en compte dans la recherche de la solution optimale, il y a le problème de planification horaire. Après plusieurs recherches sur internet et d'après le diapo du cours, l'algorithme génétique semble être la solution la plus fiable pour trouver une solution optimale à un problème de planification. Donc même s'il existe plusieurs algorithmes permettant de résoudre les autres problèmes tels que le problème des distances (le plus court circuit pour un intervenant), il nous a semblé plus intéressant de partir sur un algorithme génétique.

III. Le paramétrage de l'algorithme

A. La fonction objectif

La fonction objectif a été donnée dans le sujet. Pour rappel elle se présente comme suit :
$min\, z = 0.5 * (moy_d(s) + ecart_d(s)) + 0.5 * f_{corr} * penalite(s)$

avec :
s: la solution
$moy_d(s)$: distance moyenne parcourue par les employés pour s
$ecart_d(s)$: ecart type des distances des employés pour s
f_{corr}: facteur de corrélation correspondant à la moyenne de toutes les distances
$penalite(s)$: nombre de spécialités non satisfaites pour s

B. La représentation d'une solution

On considère que tous nos apprenants suivent le même nombre de cours N dans la semaine. Une solution est un tableau d'entier dont l'indice est calculé à partir du numéro de l'apprenant et le numéro d'un de ses cours.

```
indice = numéro de l'apprenant * N + numéro du cours
```

Il nous ait alors simple de retrouver facilement l'identifiant d'un apprenant ainsi que le numéro de son cours à partir de l'indice du tableau de la manière suivante :

```
numéro de l'apprenant = indice / N
numéro du cours = indice [N]
```

Enfin, en valeur du tableau se trouve le numéro de l'intervenant qui l'aide.

```
[indice] = numéro de l'intervenant.e
```

Ainsi on stocke pour chaque créneau de formation qu'elle interface est affectée. Les horaires, spécialités, compétences, etc. sont ensuite facilement accessibles depuis les différents tableaux du fichier d'instances.

C. Génération de la population initiale

Nous avons pu observer lors du TP sur les algorithmes génétiques que l'on obtient de meilleures solutions lorsque la taille de la population est supérieure au nombre de gènes. Nous avons ainsi choisi de fixer la taille de la population proportionnellement au nombre de créneaux de formation, en prenant tout simplement le double.

On génère ensuite autant de solutions que la taille de la population. Chacune de ses solutions doit être admissible, on ne génère alors pas les différentes solutions directement de manière aléatoire. Pour générer une solution admissible on parcourt la liste des apprenants pour leur attribuer à chacun une interface. On cherche pour chacun d'eux, la liste des interfaces libres sur ce créneau horaire, dont l'affectation à ce créneau de formation

ne déroge pas aux lois du travail et dont la compétence est la bonne. On sélectionne ensuite aléatoirement l'interface que l'on affecte à l'apprenant parmi celles qui respectent les contraintes d'horaires et de compétences.

D. Le mécanisme de sélection des individus

Nous avons choisi d'implémenter la méthode du tournoi pour la sélection car celle-ci nous évite de rencontrer le problème du super-héros. Nous avons alors un champ d'exploration qui sera certes réduit mais diversifié.

E. Voisinage

1. Type et probabilité de mutation

Pour l'originalité nous nous sommes inspirés de l'opérateur de mélange proposé par Syswerda en 1991. Il s'agit de mélanger aléatoirement les gènes entre deux positions choisies au hasard. Nous avons choisi de ne pas implémenter de fonction réparatrice par soucis de temps, nous générons donc des solutions jusqu'à en trouver des valides. Or avec l'opérateur de mélange, la probabilité de tomber sur une solution valide est très faible, cette opération prendrait donc trop de temps. Nous avons alors changé son fonctionnement afin de limiter son impact. La première position est toujours choisie au hasard. Pour la deuxième position, on choisit aléatoirement si elle se trouve avant ou après la première, puis on tire aléatoirement la distance entre les deux positions (entre 3 et 5 inclus). Si cette deuxième position dépasse les limites du tableau, on prendra alors la plus proche extrémité.

Nous avons pris la décision de fixer la probabilité de mutation par défaut à 0,01. En effet, cette valeur est un exemple donné dans le rapport de recherche "Différentes opérateurs évolutionnaires de permutation : sections, croisements et mutations" du LIFC page 43. Il s'agit d'une valeur définie par Grefenstette en 1986.

2. Type et probabilité de croisement

Pour innover en matière de croisement, nous avons décidé de réaliser une double opération de croisement en n-points, n étant un nombre tiré aléatoirement entre 1 et 4 inclus. On tire alors un premier n aléatoire, on cherche les positions après lesquelles se trouvent les points de coupes, de telle manière que les n blocs soient de tailles égales, puis on réalise l'opération de croisement. Ensuite on répète cette opération une seconde fois, sur les deux enfants obtenus, en tirant une autre valeur de n différente de la précédente.

Concernant la probabilité de croisement nous avons estimé que la valeur de 0.8 qui était utilisée dans le TP5 était plutôt bien adaptée car dans la réalité le croisement génétique des deux parents est très fort. Nous avons donc gardé cette valeur comme paramètre par défaut.

F. Le temps limite d'exécution

Nous avons choisi de fixer la limite de temps d'exécution de notre algorithme à 15 min.