Projet IT45 : Problème d'affectation d'employés
Rapport final

Sommaire

I. Rappel du dossier d'analyse conception

Les modifications apportées au cours du développement sont notées en rouge.

A. 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.

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

1. 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 avions 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.

2. 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.

C. Le paramétrage de l'algorithme

1. 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

2. 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.

3. 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. Finalement, après quelques tests nous avons conclu qu'avoir une grande population ne nous était pas très utile, en effet cela nous faisait rester bloqués longtemps pour trouver assez de solutions valides pour passer à la génération suivante. Nous avons alors fait le choix de réduire la population en fixant sa taille au nombre de formations. Ainsi nous générons plus vite de nouvelles générations, ce qui permet un affichage des améliorations plus fréquent. En effet, il n'y a qu'un seul affichage par génération même si plusieurs solutions sont mieux que la meilleure solution de l'itération précédente.

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.

4. 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.

Pour insérer de nouveaux individus dans la population, nous avons également décidé d'utiliser la sélection par tournoi, en remplaçant l'individu ayant la plus grande fitness parmi un dixième des individus de la population tirés au hasard, par un nouvel individu.

5. Voisinage

a) 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.

b) 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 0.5 est plutôt bien adaptée elle permet d'explorer sans pour autant perdre une trop grande partie du patrimoine génétique hérité des générations précédentes.

6. Le temps limite d'exécution

Nous avons choisi de fixer la limite de temps d'exécution de notre algorithme à 15 min. En testant notre algorithme, il apparaît que trouver une meilleure solution prend de plus en plus de temps. Ainsi avoir un temps d'exécution très long n’améliore que peu le résultat tandis qu’il est contraignant au vu du nombre de tests à effectuer. De ce fait, nous avons finalement réduit le temps d'exécution maximal à 5 min. Il est important de noter que cette limite de temps s’applique uniquement sur la partie qui recherche les améliorations.

II. Résultats obtenus

Dans un premier temps, nous avons réalisé une série de tests permettant d'évaluer l'impact du nombre de formation ainsi que du nombre de centres de formation sur la fitness. Par la suite, nous nous sommes penchés sur l'amélioration de notre algorithme en testant d'autres paramètres que ceux par défaut.

Pour obtenir des résultats plus précis, chaque test à été répété 3 fois pour obtenir la moyenne ainsi que l'écart type. Afin de ne pas fausser les résultats, un seul membre du groupe de projet a été chargé de la réalisation des tests. En effet, cela nous permet de nous assurer que tous les tests ont été réalisés avec les mêmes performances de calculs, puisqu'ils sont exécutés sur une unique machine.

A. Paramétrage par défaut

Nous avons réalisé de nombreux tests en faisant varier le nombre d'apprenants ainsi que le nombre de centres de formations afin de tester l'efficacité de notre méthode de résolution.

Il a été décidé de faire varier le nombre d'apprenants de 20 jusqu'à 200. Nous avons choisi de ne pas commencer plus bas car le nombre d'intervenants étant trop faible, il est rare d'obtenir ne serait-ce qu'une seule solution valide.

En ce qui concerne le nombre de centres de formation, nous avons testé chaque paramétrage du nombre d'apprenant pour 3 et 5 centres.

Dans chacun de nos tests, le nombre de formation par apprenant est de 1. En effet, créer plusieurs créneaux de formation par apprenant aurait pris du temps car il faut tenir compte de la cohérence entre les créneaux de formations d'un même apprenant.

1. Résultats pour 3 Centres de formations

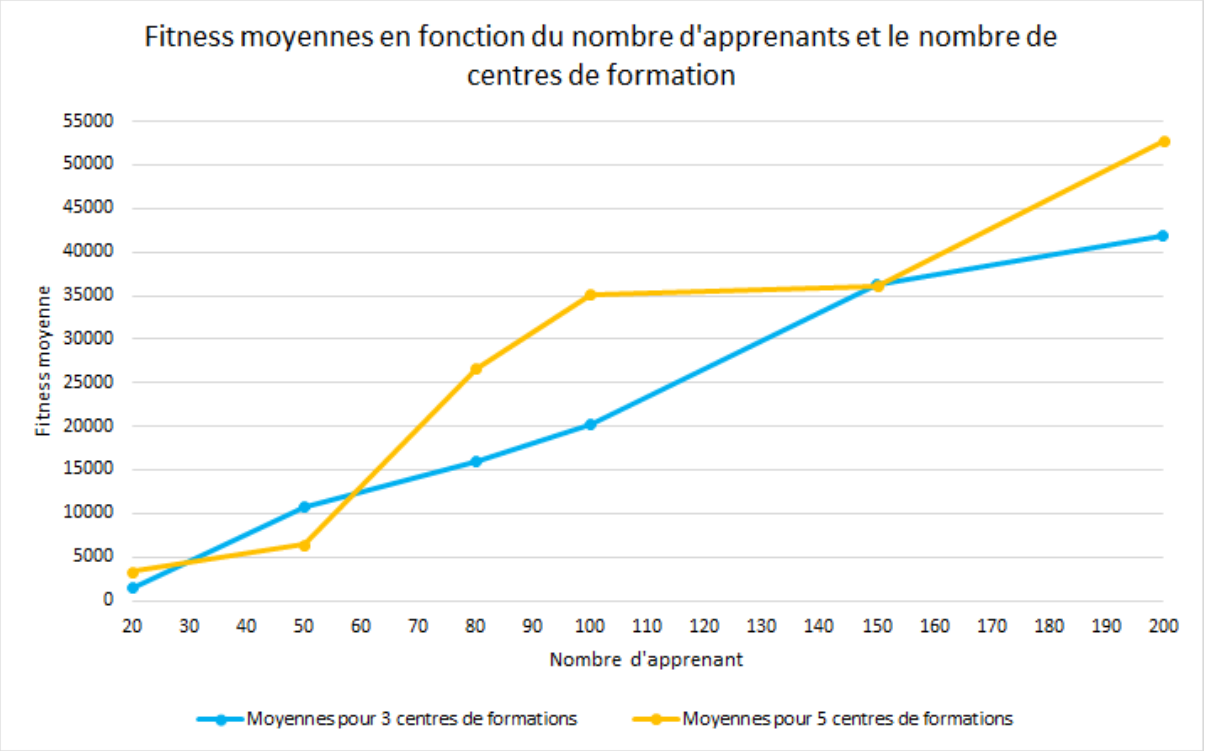
Paramétrage		Répétition			Statistiques	
Nombre d'apprenant	Nombre de centre	1	2	3	Moyenne	Ecart-type
20	**3**	1074.72	1365.95	1647.74	1362.80	233.94
50	**3**	10559.3	10883.6	10505	10649.3	167.15
80	**3**	15961.4	15322.2	16563.1	15948.9	506.67
100	**3**	19932	20496.7	20489.2	20305.9	264.45
150	**3**	35954.3	36294.4	36300.6	36183.1	161.80
200	**3**	42021.6	42288.6	41490.7	41933.6	331.62

2. Résultats pour 5 Centres de formations

Paramétrage		Répétition			Statistiques	
Nombre d'apprenant	Nombre de centre	1	2	3	Moyenne	Ecart-type
20	**5**	3371.19	3364.59	3351.91	3362.56	8.00
50	**5**	6564.26	5827.72	7053.51	6481.83	503.80
80	**5**	26166.6	28020.1	25725.3	26637.3	994.22
100	**5**	35328.3	34388	35750.6	35155.6	569.52
150	**5**	34739.4	36828.6	36823.3	36130.4	983.61
200	**5**	52620.5	53616.2	51943.6	52726.7	686.95

3. Graphique des résultats

Pour une meilleure visualisation des résultats, nous avons réalisé un graphique des moyennes obtenues pour chaque test.



On peut observer qu'en général on obtient de meilleurs résultats lorsque le nombre de centres de formation est diminué. Cela est cohérent avec ce qui était attendu car réduire le nombre de spécialité permet d'influer sur plusieurs paramètres de la fonction objectif :

- La distance moyenne parcourue est réduite car s'il y a moins de centres, il y a plus de probabilité que les interfaces fassent des interventions en restant dans le même centre au cours de la journée.
- Le nombre de spécialités non satisfaites baisse puisque l'on a plus de facilité à trouver un intervenant disponible avec la bonne spécialité.

Enfin, il est intéressant de noter que la fitness semble évoluer globalement de manière linéaire quel que soit le nombre de centres de formation, bien que les résultats pour 3 centres soient plus réguliers.

B. Amélioration des paramètres

Pour chacun des tests qui suivent, on utilise les paramètres de base du générateur d'instance. Ainsi le nombre d'apprenants est de 80, chacun suivant 1 formation par semaine et il y a 5 centres de formation.

La ligne en gras dans chaque tableau correspond au paramètre offrant le meilleur résultat.

1. Taille de la population

On commence par tenter de trouver une taille de population fournissant de meilleurs résultats. Les taux de mutation et croisements sont donc fixés aux valeurs par défaut, respectivement 0.01 et 0.5. La ligne avec un fond coloré correspond aux résultats que nous avions obtenus lors du test avec le paramétrage par défaut, il s'agit donc de notre ligne de référence.

Paramétrage	Répétition			Statistiques	
Taille de la population	1	2	3	Moyenne	Ecart-type
40	20448.3	23784.8	23332.8	22521.9	1477.86
80	26166.6	28020.1	25725.3	26637.3	994.22
160	**21433.2**	**20484**	**19522.7**	**20479.9**	**779.96**

Nous pouvons constater que, que l'on baisse ou augmente la taille de la population, la fitness tend à diminuer. Cependant, cette diminution est plus conséquente lors de l'augmentation du paramètre.

On peut donc conclure que rester sur notre première idée d'avoir une taille de population égale au double du nombre de formations aurait pu nous permettre d'obtenir de meilleurs résultats. Pour améliorer l'algorithme nous pourrions donc fixer ce paramètre par défaut au double du nombre de formation.

2. Taux de croisement

Pour améliorer le paramétrage du taux de croisement on tient compte du résultat obtenu dans la partie précédente, ainsi on fixe la taille de la population à 160. Le taux de mutation reste fixé encore une fois à 0.01, la valeur par défaut. Notre ligne de référence correspond donc à la meilleure du tableau de l'étape précédente.

Paramétrage	Répétition			Statistiques	
Taux de croisement	1	2	3	Moyenne	Ecart-type
0.25	21921	22253.9	22878.4	22351.1	396.85
0.5	**21433.2**	**20484**	**19522.7**	**20479.9**	**779.96**
0.75	22389.8	20940.3	18571.8	20633.9	1573.67

On observe que lorsque le taux de croisement diminue, on obtient des valeurs de fitness plus élevées. Cela provient du fait qu'en diminuant cette probabilité, peu d'exploration est faite, donc l'amélioration de la meilleure solution stagne. Lorsque l'on augmente ce taux, il peut arriver d'obtenir de meilleurs résultats comme par exemple le 18571.8. Cependant, la fitness moyenne est moins bonne que notre valeur de référence, de plus les résultats obtenus sont plus disparates car l'écart-type augmente significativement.

Nous pouvons en conclure que nous avons bien fait de changer notre valeur par défaut en la faisant passer de 0.8 lors de la phase d'analyse à 0.5 lors de la phase d'implémentation.

3. Taux de mutation

On finit par se pencher sur le paramétrage du taux de mutation. La taille de la population et le taux de croisement sont respectivement fixés à 160 et 0.5 de par les résultats précédents. A nouveau, notre ligne de référence correspond à la meilleure du tableau de l'étape précédente.

Paramétrage	Répétition			Statistiques	
Taux de mutation	1	2	3	Moyenne	Ecart-type
0.001	20007.7	22856.2	20944	21269.3	1185.42
0.01	**21433.2**	**20484**	**19522.7**	**20479.9**	**779.96**
0.1	22837.5	21424.1	22385.5	22215.7	589.37

Cette fois, on constate que lors de la diminution, tout comme l'augmentation du taux de mutation, les résultats que l'on obtient sont moins bons. Cela peut s'expliquer par le fait qu'un taux trop bas engrange trop peu d'exploration, tandis qu'un taux trop élevé nous fait perdre une trop grande partie des améliorations apportées par les générations précédentes.

On en conclut que nous avons bien fait de garder la valeur par défaut que nous nous étions fixés à 0.01 lors de la phase d'analyse conception.

4. Conclusion

Pour une instance constituée de 80 créneaux de formations, 1 par apprenant, réparties dans 5 centres de formation, le paramétrage initial de notre algorithme semble être assez bon puisque seule l'augmentation de la taille de la population semble offrir une amélioration notable des performances. Toutefois, l'ensemble des paramètres pourraient être raffinés en multipliant les tests pour différentes valeurs, ce que nous n'avons pas fait faute de temps.

Utiliser un autre fichier d'instances, avec un nombre de formations différentes par exemple, pourrait cependant donner lieu à un paramétrage optimal différent de celui que nous avons exposé ici. Il conviendrait donc d'affiner le paramétrage de l'algorithme selon les paramètres de l'instance étudiée.