Logistique amont d'une usine chez Renault

1. Stratégies

On divise ce problème en 3 parties, c'est-à-dire, classifier un fournisseur est sous-traité ou non sous-traité : puis regrouper tous les fournisseurs qui ne sont pas sous-traités ; finalement, calculer un coût minimum pour chaque group

1.1 Classifier un fournisseur est sous-traité ou non sous-traité.

Il n'y a que 2 possibilités pour un fournisseur, sous-traité et non sous-traité. Donc, on prend un tableau, dont la longueur est égale au nombre de fournisseurs, pour représenter la classification. De plus, pour la classe sous-traiter, on peut directement obtenir son coût avec des coûts pour sous-traiter : par contre, pour non sous-traiter, on entre dans l'étape 2 et l'étape 3. On prend l'algorithme génétique pour résoudre cette partie.

Algorithme 1 : sous-traiter ou non sous-traiter

1 : Input : un nombre de fournisseur $N_{forunisseur}$, une matrice des coûts du transport $C_{transporter}$, un ensemble des coûts pour sous-traiter $C_{sous-traiter}$, un nombre de générations $N_{génération,1}$, un nombre de solutions $N_{solution}$, un nombre de gagnant $N_{gagnant}$

2 : Output : une solution

3 : Initialiser $N_{solution}$ solutions au hasard en format de tableau rempli par 0 et 1 et mettre-les dans une liste $L = \{N_{solution}\}$, mettre un compteur $i = 0$

4 : Calculer un fitness pour chaque solution

$$fitness = solution \times C_{sous-traiter} + \text{Coût}_{non\ sous-traiter}$$

$\text{Coût}_{non\ sous-traiter}$ a besoin de l'algorithme 2 et 3

5 : while $i < N_{génération,1}$ do

6 : Sélectionner $N_{gagnant}$ solutions en fonction de son coût $L = \{N_{gagnant}\}$

7 : Former $\frac{N_{gagnant} \times (N_{gagnant}-1)}{2}$ pairs non répétés avec $N_{gagnant}$ solutions

8 : for chaque pair de solutions do

9 : Faire un crossover entre 2 solutions pour produire deux postérités

10 : Faire une mutation pour chaque postérité, calculer son fitness

11 : Mettre chaque postérité dans L

12 : end for

13 : end while

14 : return la solution avec le minimum coût dans L

1.2 Regrouper les fournisseurs non sous-traités

En fonction de l'étape 1, on obtient un ensemble de fournisseurs $N_{non\ sous-traié}$ qui ne sont pas sous-traité. L'objective de cette partie est à trouver un meilleur groupement pour ces fournisseurs. Dans ce cas-là, on prend une matrice G ($N_{non\ sous-traié} \times N_{non\ sous-traié}$), dont chaque ligne représente un group et chaque colonne représente un fournisseur qui n'est pas sous-traité.

$$G_{gf} = \begin{cases} 1, & si\ la\ colone\ f\ est\ dans\ la\ ligne\ g \\ 0, & sinon \end{cases}$$

De plus, on prend aussi l'algorithme pour cette partie. Précisément, on continuer de faire un crossover entre deux lignes (c'est-à-dire, deux groupes) et une mutation pour engendrer une nouvelle matrice G'. En même temps, on vérifie si G' est légale ou pas. En effet, chaque groupe détient au moins d'un fournisseur et au plus de 4 et chaque fournisseur n'est que dans un groupe.

Algorithme 2 : regrouper les fournisseurs non sous-traités

1 : Input : un tableau S remplit par 0 et 1, une matrice des coûts du transport $C_{transporter}$, un nombre de générations $N_{génération,2}$
2 : Output : une matrice G représentant tous les groups
3 : Initialiser G comme une matrice d'identité, mettre un compteur $i = 0$, calculer un coût total du transport C pour chaque group avec l'algorithme 3
4 : while $i < N_{génération,2}$ do
5 : Sélectionner 2 lignes de G pour un crossover et une mutation afin d'avoir G'
6 : if G' est légale do
7 : Calculer un coût du transport C' pour chaque group
8 : if $C' < C$ do
9 : $G = G',\ C = C'$
10 : end if
11 : end if
12 : end while
13 : Return G et C

1.3 Calculer un coût du transport pour un group

D'après l'étape 2, on a un groupe de fournisseurs. Comme un nombre maximal de fournisseur est 4, on peut trouver tous les chemins pour tous les fournisseurs. Par exemple, si un groupe comprend 4 fournisseurs, il y a 64 chemins possibles.

$$A_4^4 + A_4^3 + A_4^2 + A_4^1 = 64$$

On prend l'algorithme glouton pour cette partie. En effet, chaque groupe frome un réseau qu'on parcourt pour trouver un chemin avec un charge maximum et un coût minimum jusqu'à tous les stocks nuls.

Algorithme 3 : calculer un coût minimum du transport pour un group des fournisseurs

1 : Input : un ensemble C représentant $\|C\|$ fournisseurs dans un même groupe, une matrice des coûts du transport $C_{transporter}$, une matrice $Stock$ pour chaque fournisseur pendant H semaines
2 : Output : un coût minimum pour ce groupe et une séquence de tournée
3 : Initialiser $\text{totalCost} = 0$ et tournée $P = \{\}$. Récupérer un ensemble $Paths$ qui comprend toutes les possibilités de chemins pour $\|C\|$ fournisseurs

4 : for chaque semaine H do
5 : while $Stock$ pour cette semaine et ce groupe n'est pas nulle do
6 : Sélectionner un chemin de $Paths$ en fonction des coûts
7 : Mettre à jour $Stock$
8 : end while
9 : Mettre une séquence de tournées pour cette semaine dans P
10 : end for
11 : Return C et P

---

2. Évaluation des algorithmes

2.1 Complexité en temps

$$O(N_{génération,1} {N_{gagnant}}^2 N_{génération,2} N_{forunisseur})$$

On note que l'algorithme 3 ne dépend pas de $N_{forunisseur}$, parce que chaque groupe détient au plus de 4 fournisseurs. De plus, $N_{génération,1}$, $N_{gagnant}$ et $N_{génération,2}$ sont 3 hyperparamètres qui ne dépendent pas aussi de $N_{forunisseur}$. Mais les trois influencent la complexité en temps, parce qu'il faut exiger beaucoup d'itérations afin d'obtenir une solution intéressante. Dans notre cas, on met

$$N_{génération,1} = 1000, N_{gagnant} = 5, N_{génération,2} = 100000$$

2.2 Exactitude

Algorithme génétique (Algorithme 1 et algorithme 2) appartient à une méthode heuristique. En ce qui concerne des atouts, il peut efficacement éviter un piège de la solution locale, c'est-à-dire, il arrive à beaucoup approcher une solution maximale à condition de générations suffisantes : par contre, son inconvénient inévitable est à exiger beaucoup de ressources à calculer à cause de nombreuses générations. Dans notre cas, on a mis 1 millions générations.