M2 Androide - UPMC Université Paris 6
2022-2023

Module MAOA:
Modèles et Applications en Ordonnancement et optimisation combinAtoire

Projet

"Autour du problème de Production et Distribution Intégré"

Le projet qui vous est proposé dans ce module MAOA présente un problème d'Optimisation Combinatoire en Recherche Opérationnelle. Son objectif est de traiter un sujet issu d'un problème industriel jusqu'à la réalisation d'un logiciel de résolution en utilisant différentes méthodes approchées et exactes. Il s'agit ainsi de développer un logiciel permettant de résoudre des instances du problème en tenant compte de sa vitesse d'exécution et surtout sur la qualité de la solution obtenue.

Nous nous intéressons ici au problème d'Optimisation Combinatoire connu dans la littérature scientifique sous le nom de "Production-Routing problem" que l'on peut traduire par "Problème de Production et Distribution Intégré".

1 Cadre du sujet

Le problème de Production et Distribution Intégré, que l'on notera par la suite PDI, est un problème de planification opérationnelle, c'est-à-dire qu'il s'intéresse à la fois aux décisions de planification mais aussi de manière conjointe aux aspects de distribution et de routage.

Une chaîne logistique classique consiste à une suite d'activités de production, de stockage et de distribution. Chacune de ces activités est planifiée et optimisée en utilisant les décisions de l'activité qui la précède: par exemple, une planification de production entraîne une répartition par lots de livraison et ces lots seront les entrées du problème de distribution. Il a été montré que cet enchaînement de décisions ne permet pas une optimisation globale de la chaîne logistique. On appelle "chaîne logistique intégrée" (integrated supply chain) le fait de considérer conjointement l'ensemble des décisions aux différents niveaux de la chaîne dans un même problème d'optimisation: à l'échelle de grands groupes industriels, une telle prise de décision intégrée peut engendrer des réductions de coût significatives.

La difficulté de résolution d'un problème d'optimisation intégré dans le cadre des chaînes logistiques est néanmoins importante car elle demande de produire des solutions de coût minimal mais qui soient bien entendu réalisables et bien adaptées aux contraintes de production et de distribution (délais de production, dates de livraison, gestion des trajets urbains,...).

Le problème PDI a pour cadre applicatif la gestion des décisions de planification d'un fournisseur (Vendor Managed Inventory) par exemple un producteur de produits frais. Le fournisseur surveille les stocks de ses revendeurs (ou détaillants) et décide d'une politique d'approvisionnement pour chacun de ses revendeurs. Le fournisseur est alors le décideur central qui doit résoudre le problème PDI en ayant une connaissance complète des données (état des stocks, besoins,...) fournies par les revendeurs.

Le PDI intègre ainsi deux problèmes très classiques d'Optimisation Combinatoire en Recherche Opérationnelle: le problème de dimensionnement de lots (Lot-Sizing Problem ou LSP) et le problème de tournées de véhicules (Vehicle Routing Problem ou VRP). Ce problème est apparu récemment dans la littérature scientifique et représente un challenge en RO de part sa difficulté à combiner les difficultés du LSP et du VRP.

1.1 Le Problème de Lot-Sizing (LSP)

On considère l'ensemble $\mathcal{N}$ constitué du fournisseur (indice 0) et des n revendeurs $\mathcal{N} = \{0, 1, \cdots, n\}$ d'un produit donné sur un horizon de planification discret $\mathcal{T} = \{1, \cdots, l\}$. La demande de chaque revendeur i à la période t est définie par d_{it}. On suppose que le revendeur i dispose d'une zone de stockage pour stocker le produit à un coût de stockage unitaire h_i. La capacité de stockage maximale chez le revendeur i (resp. fournisseur) est définie par L_i (resp. L_0). Chaque unité de produit fabriquée par le fournisseur induit un coût fixe f par période de production (appelé coût de setup) et un coût unitaire u.

Si l'on ne s'intéresse qu'aux décisions de production du fournisseur (problème LSP), les variables de décision à prendre en compte sont les suivantes:

p_t (Variable de production): quantité produite à la période t.

y_t (Variable de lancement): variable binaire qui vaut 1 si une production est lancée à la période t, et 0 sinon.

I_{it} (Variable de stockage): quantité en stock à la fin de la période t pour i.

q_{it} (Variable d'approvisionnement): quantité produite pour le revendeur i à la période t.

Le problème LSP consiste alors à minimiser la somme totale de coûts de production, de lancement et de stockage de façon à satisfaire la demande des revendeurs.

Il existe une formulation classique pour ce problème qui possède de bonnes propriétés (voir cours).

$$
\begin{aligned}
\min \quad & \sum_{t=1}^{l} (up_t + fy_t + \sum_{i=1}^{n} h_i I_{it}) \\
s.t. \quad & I_{0,t-1} + p_t = \sum_{i=1}^{n} q_{it} + I_{0,t} \qquad \forall t \in \{1, \ldots, l\} && (1) \\
& I_{i,t-1} + q_{it} = d_{it} + I_{i,t} \qquad \forall i \in \{1, \ldots, n\}, \forall t \in \{1, \ldots, l\} && (2) \\
& p_t \leq M_t y_t \qquad \forall t \in \{1, \ldots, l\} && (3) \\
& I_{0,t-1} \leq L_0 \qquad \forall t \in \{1, \ldots, l\} && (4) \\
& I_{i,t-1} + q_{it} \leq L_i \qquad \forall i \in \{1, \ldots, n\}, \forall t \in \{1, \ldots, l\} && (5) \\
& I_{it}, q_{it} \in \mathbb{R}^+ \qquad \forall i \in \{1, \ldots, n\}, \forall t \in \{1, \ldots, l\} \\
& p_t \in \mathbb{R}^+, y_t \in \{0, 1\} \qquad \forall t \{1, \ldots, l\}
\end{aligned}
$$

La fonction objectif consiste à minimiser les coûts de production, de lancement et de stockage. Les contraintes (1) et (2) sont les contraintes de conservation de flot au niveau du fournisseur et des revendeurs respectivement. Les contraintes (3) sont les contraintes qui

relient les variables de production et de lancement. Les contraintes (4) et (5) garantissent le respect des capacités de stockage chez le fournisseur et les revendeurs.

1.2 Le problème de tournée de véhicules (VRP)

Soit un graphe orienté complet $G = (\mathcal{N}, A)$ possédant $n+1$ sommets notés $\mathcal{N} = \{0, 1, ..., n\}$: on appelle le sommet 0 le dépôt et les sommets $\mathcal{N}_C = \mathcal{N} \setminus \{0\}$ les revendeurs (ou clients). Les arcs sont donc l'ensemble $A = \{(i, j) \mid i, j \in \mathcal{N}, i \neq j\}$. A chaque client $i \in \mathcal{N}_C$, on associe une valeur réelle positive d_i appelée *demande* du sommet i. A chaque arc $(i, j) \in A$, on associe un *coût* c_{ij} strictement positif correspondant au coût de transport du sommet i au sommet j. Le *problème de tournées de véhicules* (Vehicule Routing problem ou VRP) consiste à déterminer des tournées de véhicules partant du dépôt et passant livrer la quantité demandée par chacun des revendeurs.
On considère une flotte $K = \{1, \ldots, m\}$ de m véhicules identiques qui ont chacun une capacité maximale Q: un véhicule peut transporter au plus Q quantité de produits à partir du dépôt. On appelle *tournée* un circuit non vide de G passant par le sommet 0 et dont la somme des demandes des sommets du cycle est inférieure à Q. Comme le graphe G est complet, on peut écrire une tournée T comme une séquence $(i_0, i_1, ..., i_p)$ de sommets où $i_0 = 0$ et $\sum_{l=1}^{p} d_{i_l} \leq Q$.

Le problème VRP consiste à déterminer un ensemble d'au plus m tournées $(T_1, ..., T_k)$, $k \leq m$, telles que, chaque revendeur soit servi par exactement une tournée et que la somme totale des coûts des arcs utilisés est minimum.
On suppose qu'un revendeur peut être servi par une unique tournée, *i.e.* $d_i \leq Q$ pour tout $i \in \mathcal{N}_C$. On peut noter que dans une solution, les tournées n'ont aucun client en commun en dehors du sommet 0.

Il existe plusieurs formulations PLNE classiques pour ce problème, en voici deux qui nous intéressent pour ce projet
- une formulation compacte basée sur les inégalités de Miller-Tucker-Zemlin (MTZ)
- une formulation à nombre exponentiel de contraintes, basée sur les coupes

D'autres formulations existent, comme celles indexées sur les véhicules (qui peuvent être vues comme des formules où l'on désagrège les variables des deux citées ci-dessus), ou des formulations à nombre exponentiel de variables qui se résolvent par génération de colonnes.

1.2.1 Métaheuristique pour le VRP

- Heuristique gloutonne:

Remarquons qu'une méthode gloutonne pour le VRP n'est pas évidente: en effet le problème de partition des clients en tournées (réalisables) est déjà un problème difficile. Il s'agit du problème du Bin-Packing qui revient à déterminer s'il existe une répartition des clients en m boîtes satisfaisant chacune la contrainte de sac-à-dos de la capacité d'un véhicule.
Les heuristiques gloutonnes suivantes vont avoir le défaut d'utiliser peut-etre trop de véhicules: une étape ultérieure de l'algorithme (par exemple itérative comme indiquée juste après).
Ces trois heuristiques gloutonnes ne produises qu'une partition des clients, à la fin, il est donc nécessaire, pour chaque élément de la partion de déterminer un circuit unique passant par tous les sommets de l'élément de la partition: c'est-à-dire résoudre le problème du voyageur

de commercer (TSP) limité à ces sommets: on peut le faire par une heuristique gloutonne au plus proche sommet en démarrant du dépôt par exemple (ou prendre un outil exact vu la rapidité de ces outils).

Heuristique gloutonne "Bin packing"
résoudre un "bin packing" c'est-à-dire en ne regardant que les demandes d_i remplir de manière gloutonne une première boîte de clients tant que la somme des d_i ne dépasse par Q et réitérer (on peut donc obtenir plus de m boîtes).

Heuristique gloutonne Heuristique de Clark-Wright
Cette heuristique repose, pour chaque couple de clients, sur la valeur (i, j) $s_{ij} = c_{0i} + c_{0j} - c_{ij}$ qui représente "l'économie" (savings) réalisée par le fait de regrouper i et j dans une même tournée plutôt que prendre i et j chacun dans un aller-retour de 0 à i et un aller-retour de 0 à j.
L'algorithme consiste à:
- partir d'une solution à n circuits définis comme les n allers-retours de 0 à chaque client i.
- Puis, dans l'ordre décroissant des s_{ij}, si i et j sont dans deux circuits différents et si la réunion de leurs deux circuits ne dépassent pas la demande transportable par un seul véhicule, réunir en un seul circuit les deux circuits.
Ainsi à chaque itération où une fusion a lieu, on regroupe deux circuits, l'algorithme peut également avoir plus de m circuits. Il est aussi possible d'utiliser $s_{ij} = c_{0i} + c_{0j} - \alpha c_{ij} + \beta|c_{0i} + c_{0j}| + \gamma\frac{d_i+d_j}{d}$ avec α, β, γ pris entre 0 et 2 afin de prendre un peu en compte le fait que la fusion dépend plus ou moins des distances et plus ou moins des valeurs des demandes.

Heuristique sectorielle
Cette heuristique n'a de sens que lorsque le dépôt est plus ou moins au centre du nuage des clients (ce qui est souvent le cas).
On veut tracer (virtuellement) des secteurs angulaires centrés en O qui contienne chacun un nombre de clients réalisables par une tournée. Pour cela, on prend 0 comme centre d'un repère, on classe les clients par angle à partir de l'axe des abscisses (si deux clients sont sur sur le même angle, on peut les mettre au hasard l'un après l'autre). Puis, dans cet ordre, on construit un ensemble de clients tant que la demande ne dépasse pas Q, puis un deuxième etc... On s'arrêt quand tous les clients sont affectées.

- Méthode itérative:

A l'issue d'une étape gloutonne, on obtient une solution réalisable ou alors une solution utilisant plus de m tournées (chacune réalisable).
Une métaheuristique itérative peut alors être utilisée sur la base de plusieurs voisinages:
- les voisinages classiques du TSP (2-opt) pour améliorer chaque tournée indépendemment.
- la possiblité pour un client de changer de tournées
- supprimer une tournée vide
- éventuellement ajouter une tournée vide (si le nombre est inférieur à m ou si on veut permettre une transition négative)

Si le nombre de tournées de la solution initiale est supérieur à m, il est possible d'utiliser dans un premier temps une fonction objective artificielle pour forcer à réduire le nombre de

tournées.

D'autres méta-heuristiques sont bien entendu possible (algorithme génétique, etc).

1.2.2 Formulations PLNE pour le VRP

Plusieurs formulations existent pour le RP.
• *Formulation MTZ*

On considère des variables binaires x_{ij} si un véhicule utilise l'arc $(i,j) \in A$ et des variables réelles w_i associées à chaque sommet $i \in \mathcal{N}_C$.

$$
\begin{aligned}
\min \sum_{(i,j)\in A} c_{ij}x_{ij} & & \\
\sum_{j=1}^{n} x_{0j} \leq m & & (6)\\
\sum_{i=1}^{n} x_{i0} \leq m & & (7)\\
\sum_{j=0}^{n} x_{ij} = 1 & \quad \forall i \in \mathcal{N}_C, & (8)\\
\sum_{i=0}^{n} x_{ij} = 1 & \quad \forall j \in \mathcal{N}_C, & (9)\\
w_i - w_j \geq d_i - (Q+d_i)(1-x_{ij}) & \quad \forall i \in \mathcal{N}_C, \forall j \in \mathcal{N}_\mathcal{C}, & (10)\\
0 \leq w_i \leq Q & \quad \forall i \in \mathcal{N}_C & \\
w_i \in I\!R & \quad \forall i \in \mathcal{N}_C & \\
x_{ij} \in \{0,1\} & \quad \forall (i,j) \in A. &
\end{aligned}
$$

Les contraintes de degré (6)-(9) imposent qu'il y ait exactement un arc entrant et un arc sortant pour chaque revendeurs, alors que le dépôt peut avoir jusqu'à m arcs entrants et m arcs sortants. Ces inégalités suffisent à décrire des tournées partant du dépôt, malheureusement il peut y avoir des sous-tournées (comme dans le problème du voyageur de commerce) qui ne passe pas par le dépôt. Pour briser les sous-tournées, on ajoute ici les contraintes MTZ (10) qui indiquent qu'à chaque arc $(i,j) \in A$ avec $i,j \neq 0$, pris dans la solution, le sommet j a une valeur w_j inférieure à $w_i - d_i$ (si $x_{ij} = 1$, alors $w_j \leq w_i - d_i$). Ainsi, tout circuit qui ne passe pas le sommet 0 est impossible. En revanche, si l'arc n'est pas pris, l'inégalité reste bien valide (si $x_{ij} = 0$, alors $w_j - w_i \leq Q$). Les contraintes MTZ ont une seconde utilité pour la formulation: elles permettent de limiter les tournées à charger au plus Q unités de produit par véhicule. En effet, on peut voir que w_i est une borne supérieure sur la quantité chargée dans le véhicule après le passage d'un véhicule au sommet i: sur une tournée $t = (i_0, i_1, ..., i_p)$ si on somme les contraintes MTZ des arcs, on obtient $w_{i_1} - w_{i_p} \geq \sum_{l=1}^{p} d_{i_l}$, donc si w_{i_p} est la charge du véhicule au retour de sa tournée, w_{i_1} est la charge de la tournée, or $w_i \leq Q$ pour chaque revendeur i.

• *Renforcement*

Une formulation plus forte peut venir renforcer la formulation MTZ précédente en utilisation des inégalités brisant les sous-tournées, c'est-à-dire empêchant l'existence de circuit ne passant par le dépôt 0.

Par exemple les "blossom inequalities" qui sont utilisées dans le polyèdre des couplages et dans celui du voyageur de commerce.

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1 \qquad \forall S \subset \mathcal{N}_C, |S| \geq 2, \tag{11}$$

Ces inégalités indiquent que, dans tout sous ensemble de s clients, on ne peut pas avoir plus de s arcs: en effet, si on a s arcs pour s sommets, cet ensemble de clients induit un circuit.

En utilisant les égalités de degré (8), on peut transformer les inégalités "blossom" en inégalités dites de coupes. En effet, considérons $S \subset \mathcal{N}_C$, avec $|S| \geq 2$, alors sommons les inégalités (8) sur S, on obtient:

$$\sum_{i\in S}\sum_{j=1}^{n} x_{ij} = |S|$$

or on peut remarquer que les arcs issus d'un sommet de S sont partitionnés entre ceux allant vers un autre sommet de S et ceux sortant de S, c'est-à-dire

$$\sum_{i\in S}\sum_{j=1}^{n} x_{ij} = \sum_{i\in S}\sum_{j\in S} x_{ij} + \sum_{i\in S}\sum_{j\notin S} x_{ij}$$

En remplaçant ces valeurs dans l'inégalité (11), on obtient alors les inégalités de coupes

$$\sum_{i\in S}\sum_{j\notin S} x_{ij} \geq 1 \qquad \forall S \subset \mathcal{N}_C, |S| \geq 2, \tag{12}$$

Ces inégalités de coupes (12) sont donc exactement les inégalités de blossom (**??**: il n'y a donc aucune différence concernant leurs impacts dans la formulation. Ces deux inégalités sont en même quantité exponentielle dans la formulation. En revanche, elles différent sur leur algorithme de séparation: les inégalités de blossom sont séparables en temps polynomial, mais avec un degré de polynôme élevé par l'algorithme de Edmonds et Trotter; alors que les inégalités de coupes sont séparables plus efficacement au prix de la recherche d'une coupe minimum dans un graphe (voir cours).
De plus ces inégalités sont dominées par les inégalités de la sous-section suivante: il est donc plus efficace d'ajouter les inégalités (13) que les inégalités (11) ou (12).

• *Formulation par les coupes*

Pour obtenir une formulation non-compacte, on remplace totalement les contraintes MTZ et les variables MTZ par l'inégalité suivante:

$$\sum_{i\in W}\sum_{j\in\{0,1,\dots,n\}\setminus W} x_{ij} \geq \left\lceil \frac{\sum_{i\in W} d_i}{Q} \right\rceil \qquad \forall W \subset \{1,\dots,n\}, W \neq \emptyset, \tag{13}$$

Ces contraintes (13) sont dites *contraintes de coupes du CVRP*. L'idée principale est la suivante: la demande totale d'un ensemble W de clients est $\sum_{i\in W} d_i$, si elle est non nulle, le nombre de tournées nécessaires pour servir les clients de W est supérieur à $\frac{\sum_{i\in W} d_i}{Q}$. On peut montrer que cette inégalité est suffisante pour formuler le problème VRP (voir exo de cours à venir).

Dans le cas où l'on souhaite séparer un vecteur x^* entier, séparer ces inégalité est polynomiale: voici la description de la méthode. En posant $F = \{ij \in A \mid x^*_{ij} = 1\}$, on obtient un ensemble de m circuits par les inégalités de degré de la formulation. En effet, on peut alors prouver que chaque sommet client appartient à un unique circuit. On cherche pour chaque client i les sommets C constituant le circuit qui passe par lui. Il y a alors 3 cas: soit pour un i donné le circuit C ne passe pas le dépôt: dans ce cas, les sommets du circuit forme un ensemble $W = C$ pour lequel l'inégalité (13) est violée par x^*; soit pour un i donné le circuit passe par le dépôt mais le circuit a une trop grosse demande pour un seul véhicule (i.e. $W = C \setminus \{0\}$ viole l'inégalité (13) qui lui correspond); soit **pour tout** client i, son circuit passe par le dépôt et a une demande inférieure à celle de la capacité d'un véhicule. Dans ce dernier cas, la solution est donc valide et x^* ne viole aucune inégalité.

Dans le cas où l'on souhaite séparer un vecteur x^* quelconque (potentiellement fractionnaire). La séparation des inégalités de coupes CVRP est NP-difficile: néanmoins on peut les séparer heuristiquement par des méthodes approchées (glouton, taboo search). [1].

Ainsi, en ajoutant à la fois une séparation entière exacte (et efficace), ainsi qu'une méthode approchée de séparation fractionnaire, on obtient une méthode PLNE non compacte largement plus efficace que la formulation MTZ.

1.3 Le problème de Production et Distribution Intégré (PDI)

Pour le problème PDI, on appelle réseau la donnée d'un graphe complet orienté $G = (\mathcal{N}, A)$ où $\mathcal{N}$ représente l'ensemble composé du lieu de production (appelé souvent usine) et des clients. On note $\mathcal{N} = \{0, 1, \dots, n\}$ où 0 est l'usine et les clients sont l'ensemble $\mathcal{N}_C = \mathcal{N} \setminus \{0\}$. Les arcs sont donc l'ensemble $A = \{(i, j) \mid i, j \in \mathcal{N}, i \neq j\}$. On considère un horizon de planification discret $\mathcal{T} = \{1, \cdots, l\}$.
Un produit unique doit être délivré dans l'usine et livré par un ensemble de véhicules identiques $K = \{1, \dots, m\}$ aux clients pour satisfaire la demande à chaque période.
Les paramètres sont ainsi:

u coût unitaire de production

[1] On peut aussi s'intéresser aux inégalités où la borne est relaxée en $\frac{\sum_{i\in W} d_i}{Q}$ pour lesquelles le problème de séparation est alors plus simple.. mais l'inégalité est alors moins forte. A l'inverse, on peut augmenter encore le terme de droite des inégalités, mais le problème de séparation associé est alors encore plus difficile à résoudre

f coût fixe (setup) de production

h_i coût de stockage unitaire

c_{ij} coût de transport du nœud i au nœud j

d_{it} demande du client i à la période t

C capacité de production

Q capacité d'un véhicule

L_i capacité de stockage maximale au nœud i

I_{i0} quantité en stock initiale au nœud i (dans certains articles et dans les fichiers d'instances, elle est aussi noté $L0[i]$.)

Ce problème est décrit clairement avec un recensement très complet de la littérature dans l'article de Adulyasak et al. [1].

2 Contenu du projet

Le projet est divisé en trois parties: résolution heuristique, résolution exacte et évaluation expérimentale.

2.1 Résolution heuristique

Même si le but de ce projet (ainsi que de l'UE MAOA) est une résolution exacte du problème, il est utile de commencer votre étude par la mise en place d'une heuristique ou d'une méta-heuristique. Cela vous permettra d'apréhender facilement le problème et ses instances, ainsi que d'avoir un élément de comparaison pour le reste du projet.

D'autre part, la nature NP-difficile des problèmes traités issus d'applications réelles rendent nécessaires la mise au point d'une telle approche heuristique.
Les objectifs sont:
- permettre de produire une première solution utile aux méthodes exactes,
- produire une solution quelque soit la taille de l'instance à traiter,
- la comparaison entre cette méthode et les bornes obtenues par les méthodes exactes vous permettra dans certains cas d'avoir une garantie expérimentale de vos solutions.

Heuristique proposée:

Vous pourrez tester une heuristique itérative en deux phases fondée sur le principe suivant: une première phase consiste à la résolution du problème de lot-sizing dans lequel on intègre une approximation des coûts de visite aux clients. La seconde phase sera dédiée à la construction des tournées de véhicules. A l'issue de cette seconde phase, les coûts de visite seront mis à jour pour la première phase de l'itération suivante. Cette procédure est réitérée jusqu'à ce qu'un critère d'arrêt soit atteint. Ce critère est à définir, il peut consister en un nombre maximum d'itérations. Il faudra mémoriser la meilleure solution trouvée durant ce processus. On détaille dans ce qui suit ces deux phases:

1. Résoudre le problème LSP avec prise en compte des coûts de visite à partir de la formulation dite agrégée présentée précédemment en utilisant un solveur de PL. Cette phase permet de calculer les quantités produites, stockées et approvisionnées à partir de l'usine pour chaque client et pour chaque période de l'horizon de planification. On ajoutera à la fonction objectif le terme $\sum_{i,t} SC_{it} z_{it}$ représentant le coût total de visite des clients sur l'horizon de planification. Le paramètre SC_{it} représente le coût de visite du client i à la période t et la variable binaire z_{it} vaut 1 si le client i est visité à la période t. Il faudra ajouter des contraintes couplantes entre ces variables et les variables de production dans le modèle. A l'initialisation, $SC_{it} = c_{0i} + c_{i0}$ pour tout i pour éviter que les clients trop éloignés de l'usine soient servis trop fréquemment sur l'horizon de planification.

2. Une deuxième phase consiste alors à résoudre un VRP à chaque période. Les quantités à livrer aux clients sont calculées dans la phase de planification. Vu la difficulté de résolution du VRP, il est préférable de recourir à une méthode approchée pour résoudre le problème de tournées: attention il faut s'assurer que la capacité des véhicules est respectée.

3. Mise à jour des coûts SC_{it}. A l'issue de la deuxième phase, on obtient à chaque période le sous-ensemble de clients visités, que l'on note $\mathcal{S}_t$. Pour les sommets de $\mathcal{S}_t$, on note i^- (respectivement i^+), le prédécesseur (respectivement le successeur) de i dans la tournée. Une mise à jour possible pour SC_{it} serait de fixer la valeur de SC_{it} à $c_{i^-i} + c_{ii^+} - c_{i^-i^+}$ si $i \in \mathcal{S}_t$ et de fixer sa valeur à la valeur du coût de l'insertion la plus économique dans une des tournées à la période t, sinon.

Une phase de diversification peut être envisagée sur le paramètre SC_{it} à l'issue de cette méthode itérative qui pourra être relancée avec de nouvelles valeurs pour ce paramètre.

Le principe décrit ci-dessus fournit un cadre général. Vous êtes libres de tester d'autres idées soit pour la résolution des sous-problèmes, soit pour la mise à jour des coûts.

2.2 Résolution exacte et bornes par PLNE

Il vous est demandé de mettre en place au moins une méthode exacte **performante** ou un calcul de **bornes pertinentes**, en utilisant la programmation linéaire en nombres entiers (PLNE).
Les objectifs possibles sont:
- résoudre exactement des instances de tailles réduites,
- fournir des bornes pour le problème pour des tailles moyennes,
- utiliser les techniques d'arrondi à partir des solutions fractionnaires,
Les tailles réduites vous permettront également de vérifier si vos solutions heuristiques sont de bonne qualité.

Nous vous proposons, comme lors des séances de TME, d'utiliser l'outil CPLEX avec concert technology.

Formulations proposées:

Le papier de recherche [1] propose plusieurs formulations PLNE pour le PDI. Nous nous intéressons à deux d'entre elles:

- La formulation de Bard et Nananukul [10,12] donnée par les inégalités (1)-(16) qui est compacte par l'utilisation d'inégalités MTZ.
- La formulation de Boudia *et al.* [17,18] donnée par les inégalités (20)-(32) dont l'inégalité (29) est en nombre exponentiel.

A partir de ces formulations, deux travaux vous sont proposés:

- Vous coderez la formulation (1)-(16) qui est facile à mettre en œuvre mais ne permet de résoudre que des instances de petites tailles.
- Afin de résoudre des instances de plus grande taille et d'obtenir de meilleures bornes encadrant la solution optimale, il est nécessaire d'approfondir en direction des algorithmes de Branch-and-Cut. Nous vous proposons deux pistes possibles:
  - soit enrichir la formulation (1)-(16) par l'ajout d'inégalités (17), (18) ou (19) dans le cadre d'un algorithme de Branch-and-Cut. Comme le problème de séparation des inégalités (17), (18) ou (19) est NP-difficile, il est préférable de conserver les inégalités MTZ et d'utiliser une méthode approchée de séparation pour (17), (18) ou (19).
  - soit utiliser la formulation (20)-(32) dans le cadre d'un algorithme de Branch-and-Cut.

2.3 Evaluation expérimentale

Comme évoqué ci-dessus, une évaluation expérimentale est nécessaire pour valider vos différentes méthodes approchées et exactes, en utilisant des bornes pour obtenir des garanties expérimentales. La meilleure façon de convaincre de la qualité de votre travail est de pouvoir donner des statistiques (courbes, chiffres éloquents,...) pour illustrer la performance de vos outils. N'hésitez pas à avoir recours à des tableurs ou des logiciels de tracé de courbes (gnuplot par exemple).

3 Instances

Voir sources des instances et document les décrivant sur le site du module.

Il vous est demandé d'utiliser un ensemble d'instances de tailles et de dimensions suffisantes pour effectuer des tests expérimentaux. La nature de ces instances (particularités, corrélation,...) est aussi à examiner de près.

Il est aussi très utile d'avoir une visualisation de ces instances (voir document concernant les instances).

4 Déroulement du projet

Le projet nécessite un travail personnel qui s'étale sur tout le semestre: il vous sera demandé:
- un rapport synthétique
- une production logicielle
- une validation expérimentale
- une présentation finale de votre projet devant vos camarades.

Le projet est à réaliser en binômes.

Il vous est demandé un travail régulier afin de pouvoir rendre le projet complet à temps: n'hésitez pas à discuter par mail avec les encadrants tout au long du projet sur des questions constructives ou des problèmes particuliers d'ordre théorique ou technique. Le sujet est assez libre pour laisser place à l'intuition et à l'initiative.

4.1 Evaluation du projet

Plusieurs facteurs seront pris en compte dans l'évaluation:
- Le but de ce projet est clairement d'aboutir à un produit fini: c'est-à-dire à un logiciel complet permettant de résoudre un problème de Recherche Opérationnelle allant de l'instance à une solution compréhensible. Pour cela, vous pouvez par exemple utiliser en entrée un fichier texte contenant les données du problème pratique et en sortie une visualisation compréhensible de la solution (graphique,...).
- Un des objectifs est d'obtenir un logiciel performant. Vous devez fournir une évaluation de ses performances par rapport aux instances fournies et à des instances générées aléatoirement, en utilisant des mesures expérimentales.
- L'apport d'idées nouvelles (originalité), de recherche dans la littérature scientifique, de mise au point d'algorithmes fins,... sera bien évidemment pris en compte.

Le projet proposé est vaste: à chacun de trouver la partie et le développement qui convient à ses capacités informatiques et mathématiques.

4.2 Rendu et Soutenance du projet

Il est attendu

- un rapport décrivant votre travail, votre code et les résultats expérimentaux obtenus
- votre code
- une soutenance (entre 5 et 10 min) où vous présentez votre travail

4.3 Calendrier du projet

Mercredi 5 octobre 2022: TME et début du projet.

Mercredi 12 octobre 2022: Séance questions-réponses en TD sur le projet

Mercredi 9 novembre 2022: Suivi du projet

Date limite de rendu de projet: Vendredi 7 décembre 2022

Soutenances: Mercredi 14 décembre 2022

References

[1] Y. Adulyasak and J-F Cordeau and R. Jans (2015). The production routing problem: A review of formulations and solution algorithms *Computers & Operations Research*, 55:141-152.