M2 Androide - Sorbonne Université
2020-2021

Module MAOA:
Modèles et Applications en Ordonnancement et optimisation combinAtoire

Projet

"Autour du problème de Production et Distribution Intégré"

INSTANCES

Dans le survey [1], deux benchmarks d'instances sont évoqués :
- les instances de l'article Archetti et al. [8] : nous les appelerons instances A.
- les instances des articles de Boudia et al. [16] et [18] : nous les appelerons instances B.
La source de ces instances peut être retrouvée sur un site[1] tenu par Y. Adulyasak.

Pour vous faciliter le dévelopement, nous avons "unifié" ces deux benchmarks en un seul qui est en ligne **sur le site du module**. Le code C++ ayant servi à manipuler ces instances vous est également fourni et est décrit dans ce document.

• Différence entre les instances

Il y a quelques différences nécessaires à connaître entre les instances :

Pour les instances A :
- le coût de transport c_{ij} du client i au client j est donné par la formule

$$\left\lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + \frac{1}{2} \right\rfloor$$

Pour les instances B :
- le coût de transport c_{ij} du client i au client j est donné par une multiplication de la distance euclidienne par une constante mc

$$mc\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- la production en t devient disponible en $t+1$, sans coût de stockage sur ce pas de temps $[t, t+1]$ (c'est-à-dire que la production, stockée au lieu de production, n'entraîne pas de stockage sur ce pas de temps): **il faut donc ne comptabiliser en stockage que ce qui est conservé plus d'une période**.
- Ainsi la demande en 1 doit être satisfaite par le stock initial du dépôt : dans ces instances, ce stock initial est ainsi égal à la demande totale des clients sans aucun coût de stockage pour la période de 0 à 1.

• Caractéristiques utiles spécifiques

Il y a également des caractéristiques utiles à noter concernant les deux benchmarks :

[1] `https://sites.google.com/site/ayossiri/publications`

Pour les instances A :
- les instances ont soit 14, soit 50 ou 100 clients.
- toutes les instances ont un horizon de 6 périodes.
- la demande d'un client est fixe sur le temps (le problème PDI reste NP-difficile sur ces instances).
- le stock au dépôt n'est pas limité, la capacité de production est infinie.
- pour un nombre de clients n, il y a 96 instances. Ces instances sont divisées en 4 classes: classe I de 1 à 24; classe II de 25 à 48; classe III de 49 à 72 et classe IV de 73 à 96. La classe I sert de référence et les 3 autres lui sont identiques sauf : la classe II a un coût de production variable 10 fois plus élevé ; la classe III a un coût de trajet 5 fois plus grand ; la classe IV a un coût de stockage nul.

Pour les instances B :
- les instances ont soit 50, soit 100 ou 200 clients.
- toutes les instances ont un horizon de 20 périodes.
- le coût unitaire de production n'est pas indiqué : nous l'avons fixé à 0 pour correspondre à l'idée citée ci-dessus de l'absence de coût de stockage de la production sur la période de production.
- le coût de stockage au dépôt comme chez les clients est de 1.
- les capacités de stockage sont identiques sur chaque tiers de clients.

• **Format du benchmark "unifié"**

Sur le site du module, un fichier .tgz contient les deux benchmark unifiés en un seul benchmark avec un format unique de fichier (les programmes ayant servi à unifier ces deux benchmarks sont fournis à titre indicatif dans l'archive).
Le format .prp qui vous est proposé est assez simple à lire d'après les notations de l'article [1]: il s'agit d'un fichier .txt utilisant les noms des paramètres. Le nom des fichiers commencent par A ou B suivant la benchmark, suivi du nombre de clients et finissant par le nom originel du fichier dans la benchmark.

• **Code C++ de manipulation d'instance**

Le site du module propose également un code C++ exemple de manipulation d'instances :
- une classe PRP (PRP.h et PRP.cpp) permettant de stocker, lire le format .prp et afficher à l'écran une instance en mode texte.
- un programme d'affichage (PRP_viewer.cpp) qui utilise la classe PRP pour lire une instance au format .prp. et l'afficher (en mode texte) à l'écran : ce programme peut servir de base à vos programmes.

• **Affichage graphique**

Comme signalé dans le sujet du projet, il est souhaitable de prévoir un affichage des tournées de véhicules. Un moyen assez simple et très portable est de créer automatiquement un fichier SVG (voir classe `C_Graph` du TME), ou bien poscript [2]; ou alors d'utiliser GraphViz[3].

[2] `http://www-desir.lip6.fr/~fouilhoux/documents/doc_postscript.pdf`
[3] `http://www-desir.lip6.fr/~fouilhoux/documents/doc_graphviz.pdf`

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.