Université de Valenciennes et du Hainaut-Cambrésis
Institut des Sciences et Techniques de Valenciennes **2015-2016**

Master Technologies Nouvelles des Systèmes d'Information — Semestre 9
Métaheuristique — Saïd Hanafi & Christophe Wilbaut

Problème du Sac-à-Dos Multidimensionnel avec contraintes de Demandes

Introduction

Ce document vise à présenter synthétiquement le problème que vous allez traiter dans le cadre du module *Heuristiques - Métaheuristiques*, ainsi que les modalités de travail et d'évaluation. Lisez attentivement ce document, et n'hésitez pas à poser des questions si certains points ne vous semblent pas clairs.

L'objectif est de vous mettre "en compétition" afin de vous motiver d'avantage que lors de séances de TP plus *classique*. Cet esprit de compétition doit vous pousser à chercher à obtenir les meilleurs résultats possibles lors de la résolution d'un problème **réel** sous certaines contraintes (essentiellement organisationnelles et temporelles).

Voici quelques règles à respecter :
- Vous devez former des équipes de 3 (**minimum**) à 5 (**maximum**) personnes.
- Vous pourrez développer vos approches pendant les séances de TD et TP du module. Ces séances sont également l'occasion de discuter avec l'intervenant présent sur les éventuels points de blocage ou sur vos pistes de réflexion.
- Vous avez la possibilité d'échanger entre vous les résultats obtenus sur les instances tests du problème.
- Vous pouvez programmer en `C`, `C++` ou `Java`.
- Les évaluateurs ne sont pas tenus d'utiliser un logiciel de développement particulier pour pouvoir générer votre exécutable. La compilation sera effectuée en ligne de commandes (voir la partie "Modalités d'évaluation").

Deux objectifs majeurs peuvent être identifiés : d'une part la mise en place d'un travail **collaboratif** au sein de chaque équipe de manière à obtenir les meilleurs résultats possibles, et d'autre part la mise en application de différents concepts abordés dans les enseignements liés à la recherche opérationnelle que vous avez suivis ces dernières années.

En termes d'attentes, votre équipe doit obtenir les meilleurs solutions **réalisables** possibles en exécutant votre méthode avec un temps limite fixé. Cet aspect de ***compétition*** sera pris en compte lors de l'évaluation, et chacun a donc intérêt à faire avancer son équipe au mieux.

Organisation

Composition des équipes : elle est **libre**, à condition qu'elle soit établie rapidement. Dans le cas contraire un tirage au sort sera organisé.

Évaluation du travail : elle portera sur plusieurs éléments...
- Un (petit) rapport de projet par équipe (5 à 10 pages) qui doit :
  - Présenter succinctement le problème.

- Présenter les différents points que vous aurez abordés (structures de données, langage de programmation, méthodes de résolution envisagées, choisies, difficultés, ...).
- Permettre d'identifier "qui à fait quoi" au sein d'une équipe.
- Reporter les résultats obtenus sur l'ensemble des instances utilisées, avec les caractéristiques du matériel employé (système d'exploitation, RAM, temps CPU).

— Une présentation par équipe (5 à 10 minutes) devant l'ensemble des étudiants participants au module. Chaque membre d'une équipe devra présenter une partie du travail réalisé de manière à pouvoir être évalué.
— Le code produit, ainsi que la documentation de celui-ci.
— La qualité des résultats obtenus.

Chaque équipe devra déposer son rapport et son code (uniquement les fichiers sources + spécifications) sur la plate-forme Moodle.

Important : la note obtenue sur ce mini projet sera une note **individuelle** prenant en compte le poids de chacun dans le travail de l'équipe.

Présentation du problème

Le problème traité dans ce projet est le problème du sac-à-dos multidimensionnel avec contraintes de demandes (**SADMD**). Il s'obtient en ajoutant au problème du sac-à-dos multidimensionnel un ensemble de contraintes dites de demande (de type $\geq$). La formulation du SADMD est la suivante :

$$
(\text{SADMD}) \left[\begin{array}{lll} \max & \sum_{j=1}^{n} p_j x_j & \\ \text{s.c. :} & \sum_{j=1}^{n} w_{ij} x_j \leq c_i & \forall i \in M^1 \\ & \sum_{j=1}^{n} w_{ij} x_j \geq c_i & \forall i \in M^2 \\ & x_j \in \{0,1\} & j = 1, \ldots, n \end{array} \right.
$$

avec $n = \mid N \mid$ le nombre de variables (d'objets), $M = M^1 \cup M^2$ l'ensemble des contraintes, décomposé en deux sous-ensembles : M^1 pour les contraintes de capacité, et M^2 les contraintes de demande.
Les coefficients p_j, c_i et w_{ij} sont des entiers, $\forall i \in M, \forall j \in N$, et font référence, respectivement, au profit de l'objet j, au membre droit de la contrainte i, et au poids de l'objet j dans la contrainte i.

Le SADMD fait partie de la classe des problèmes NP-difficiles, et l'utilisation d'heuristiques est recommandée même pour des instances de taille moyenne. C'est pourquoi **votre travail est de concevoir et implémenter un algorithme approché (heuristique ou métaheuristique)** permettant d'obtenir la meilleure solution réalisable possible dans un temps imparti.

Vous avez à votre disposition sur Moodle trois articles scientifiques décrivant des approches existantes pour résoudre le MDMKP [1, 3, 4]. Une des premières choses à faire est de les lire et d'en comprendre le fonctionnement global. Il est autorisé de s'inspirer d'une de ces approches.

Modalités

Un ensemble conséquent d'instances est disponible sur le site de la *OR-Library*[2]. Dans notre cas, nous nous limiterons à tester les algorithmes sur un sous-ensemble de 18 instances que vous pouvez

récupérer sur Moodle. Chaque instance est associée à **un fichier texte** qui respecte le format suivant :

nb_variables nb_contraintes_capacité nb_contraintes_demande
p_j pour chaque $j \in N$
c_i pour chaque $i \in M^1$
c_i pour chaque $i \in M^2$
pour chaque $i \in M^1$, w_{ij} pour chaque $j \in N$
pour chaque $i \in M^2$, w_{ij} pour chaque $j \in N$

Exemple de fichier d'instance :
5 2 1
20 18 15 14 12
35 40
15
15 16 12 12 10
22 21 16 14 15
8 2 6 2 5

Cette instance correspond au problème :

$$\left[\begin{array}{lll} \max & 20x_1 + 18x_2 + 15x_3 + 14x_4 + 12x_5 & \\ \text{s.c. :} & 15x_1 + 16x_2 + 12x_3 + 12x_4 + 10x_5 & \leq 35 \\ & 22x_1 + 21x_2 + 16x_3 + 14x_4 + 15x_5 & \leq 40 \\ & 8x_1 + 2x_2 + 6x_3 + 2x_4 + 5x_5 & \geq 15 \\ & x_j \in \{0,1\} & j = 1, \ldots, 5 \end{array}\right.$$

Votre algorithme doit **produire un fichier texte** contenant la meilleure solution obtenue et sa valeur, en respectant le format suivant :

nom_du_fichier_en_entrée
nb_variables
pour chaque $j \in N$ la valeur de x_j
valeur_de_la_solution

Modalités d'évaluation des algorithmes

- La machine utilisée pour les tests est équipée d'un processeur `Intel Core i7` 2.8 GHz avec 16 Go de RAM, et **Windows 7**. La compilation se fera en ligne de commande avec `gcc` ou `g++` (`Makefile` utilisable par l'intermédiaire de `MinGW`) ou `javac`.
- Vous devrez fournir, par l'intermédiaire de `Moodle`, l'ensemble des fichiers **sources** permettant de générer l'exécutable correspondant à votre programme, avec un `Makefile` ou au minimum un `ReadMe` expliquant comment générer cet exécutable (en précisant notamment les options de compilation à utiliser). Vous déposerez également une version de celui-ci sur `Moodle`.
- Votre exécutable doit obligatoirement respecter la syntaxe d'appel suivante, en ligne de commande :
  - en C/C++ :
    *miniprojet_* < n°_du_groupe > *.exe nom_instance fichier_sortie temps*
  - en Java :
    *java miniprojet_* < n°_du_groupe > *nom_instance fichier_sortie temps*
  - le paramètre *nom_instance* permettra d'indiquer quelle instance doit être résolue

— le paramètre $fichier\_sortie$ donne le nom du fichier qui contiendra votre solution
— le paramètre *temps* est le temps de recherche accordé à l'algorithme, en secondes.
— **Attention :** vous devez vous assurer que votre algorithme s'arrête une fois ce temps écoulé. Nous pouvons par exemple fixer ce temps à 60 secondes sur des instances de petite taille.

Côté évaluation, votre note finale sera calculée sur la base de plusieurs notes portant sur :

1. le rapport (/5),
2. la présentation (/5),
3. la qualité du code et la maîtrise technique (code et méthodes mises en œuvres (/5)),
4. la performance de vos algorithmes (/5).

La note du point 3 prendra également en compte le respect des consignes, et la note du point 4 sera directement lié à l'aspect "compétition" du mini-projet.

Pour toute question relative à ce mini-projet n'hésitez pas à nous contacter : Saïd Hanafi et/ou Christophe Wilbaut.

Références

[1] Arntzen, H., Hvattum, L. M. et Lokketangen, A. Adaptive memory search for multidemand multidimensional knapsack problems. Computers & Operations Research, 33 : 2508-2525, 2006.

[2] Beasley, J. E. OR-Library. http ://people.brunel.ac.uk/˜mastjjb/jeb/info.html.

[3] Cappanera, P. et Trubian, M. A Local-Search-Based Heuristic for the Demand-Constrained Multidimensional Knapsack Problem. INFORMS Journal on Computing, 17(1) : 82-98, 2005.

[4] Hvattum, L. M. et Lokketangen, A. Experiments using scatter search for the multidemand multidimensional knapsack problem. Dans : Doerner, K. F. et al. (éditeurs), *Metaheuristics : Progress in Complex Systems Optimization*, Operations Research/Computer Science Interfaces Series, Vol. 39, pp. 3-24. Springer, Berlin, NY, 2007.