Université de Valenciennes et du Hainaut-Cambrésis
Institut des Sciences et Techniques de Valenciennes **2016-2017**

Master Cyber-Défense et Sécurité de l'Information Semestre 8
Complexité Saïd Hanafi & Christophe Wilbaut

Mini-Projet : Problème du Sac-à-Dos Multidimensionnel à Choix Multiples

---

Introduction

Dans le cadre du module *Complexité* de cette année, vous allez travailler **en équipe** sur un mini-projet. Votre travail sera évalué sur plusieurs aspects (voir plus loin) et conduira à la note du module. Ce document vise à présenter synthétiquement le problème que vous allez traiter ainsi que les modalités de travail et d'évaluation. Lisez le attentivement et n'hésitez pas à poser des questions si certains points ne vous semblent pas clairs.

Toutes les séances restantes du module vont être consacrées à ce mini-projet. Un des objectifs est de vous mettre "*en compétition*" pour 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. La composition des équipes doit être fixée et communiquée aux enseignants au lancement du mini-projet.
- Vous aller développer vos algorithmes 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 (entre les équipes) les résultats obtenus sur les instances tests du problème, si vous souhaitez avoir un point de comparaison.
- Le langage de programmation est **libre**. Attention toutefois à bien commenter votre code. S'agissant d'un travail en équipe sur une durée plus longue que les TPs "classiques", un code non commenté (ou pas suffisamment) sera pénalisé.
- Les évaluateurs ne sont pas tenus d'utiliser un logiciel de développement particulier pour pouvoir générer votre exécutable. La compilation doit donc **impérativement** pouvoir être effectuée en ligne de commandes (voir la partie "Modalités d'évaluation").

D'un point de vue pratique, l'objectif de votre équipe est d'obtenir les meilleurs solutions **réalisables** (ou faisables) possibles en exécutant votre méthode avec un temps limite fixé. Un classement des méthodes, en fonction des résultats obtenus, sera réalisé et pris en compte lors de l'évaluation. 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 par les enseignants.

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 choisi, méthodes de résolution envisagées, choisies, difficultés rencontrées, résultats obtenus, ...).
  - Permettre d'identifier "qui à fait quoi" au sein de l'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 de l'équipe devra présenter une partie du travail réalisé.
- Le code produit, ainsi que la documentation de celui-ci.
- La qualité des résultats obtenus (évaluée par le classement des équipes).

Chaque équipe devra déposer son rapport, son code (uniquement les fichiers sources + spécifications) et sa présentation sur la plate-forme Moodle, au plus tard le jour de la soutenance.

Attention : bien qu'étant un travail en équipe, la note obtenue sur ce mini projet est une note **individuelle**. Les enseignants peuvent pondérer la note de chacun en fonction du travail réalisé au sein de l'équipe.

Présentation du problème

Le cryptosystème de Merkle-Hellman proposé à la fin des années 70 fut l'un des tous premiers protocoles à clé publique publié. Cette approche repose sur un cas particulier du problème du Sac-à-Dos (SaD) (le problème de la somme de sous-ensembles). Bien que ce protocole ait été cassé dans les années 80, il est un bon exemple d'utilisation d'un problème d'optimisation pour construire un algorithme de cryptographie. D'autres méthodes et extension utilisant des variantes du problème du sac-à-dos ont depuis été proposées dans la littérature.

Dans ce projet nous vous proposons de travailler sur une autre variante du SaD, qui intègre des contraintes particulières pour l'affectation des variables. Il s'agit du problème du Sac-à-Dos Multidimensionnel à Choix Multiples (**SaDMCM**). La formulation mathématique de ce problème est la suivante :

$$\text{(SaDMCM)} \left[\begin{array}{lll} \max & \sum_{i=1}^{g} \sum_{j \in G_i} c_{ij} x_{ij} & \\ \text{s.c. :} & \sum_{i=1}^{g} \sum_{j \in G_i} a_{ij}^k x_{ij} \leq b_k & \forall k = 1, \ldots, m \\ & \sum_{j \in G_i} x_{ij} = 1 & \forall i = 1, \ldots, g \\ & x_{ij} \in \{0, 1\} & \forall i = 1, \ldots, g, j \in G_i \end{array} \right.$$

Dans ce problème un objet est sélectionné complètement ou pas du tout, les variables x_{ij} associées aux objets sont donc binaires. L'objectif du problème est de trouver un sous-ensemble des objets qui maximise le profit total (la sommes des c_{ij}), en satisfaisant un ensemble de contraintes de capacités (limitées pas les valeurs b_k) et de contraintes dites de choix. L'ensemble G des objets est divisé en g groupes (ou classes) : $G = G_1 \cup G_2 \cup \ldots G_g$, et il n'est possible de sélectionner qu'un seul objet dans chaque groupe (mais il en faut forcément 1 !). Soit g_i le nombre d'objets dans le group G_i. Le nombre total d'objets est défini par $n = \sum_{i=1}^{g} g_i$. Il faut noter que les groupes sont disjoints : $G_i \cap G_j = \oslash, 1 \leq i \leq j \leq g$. Les coefficients a_{ij}^k représentent la consommation des objets dans la contrainte de capacité k.

Le SaDMCM fait clairement partie de la classe des problèmes NP-difficiles (puisqu'il s'agit d'une extension du problème du SaD), 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 (ou plusieurs) algorithme approché (heuristique ou métaheuristique)** permettant d'obtenir la meilleure solution réalisable possible dans un temps imparti.

Une des toutes premières tâches à faire est de consulter les références bibliographiques accessibles sur Moodle et illustrant des approches applicables pour résoudre ce problème [1, 2, 3, 4, 5, 6]. Vous êtes autorisé à vous inspirer et à implémenter / adapter tout ou partie de ces approches.

Modalités

Un ensemble de 32 instances du SaDMCM sont disponibles sur Moodle pour évaluer vos algorithmes. Dans les instances utilisées dans ce projet tous les groupes ont le même nombre d'objets : $g_i = g_j, 1 \leq i \leq j \leq g$. On obtient donc le nombre total d'objets par $n = g \times g_i$. Chaque instance est associée à **un fichier texte** qui respecte le format suivant :

g-g_i-m /* nbre. de groupes - nbre. d'objets par groupe - nbre. de contraintes de capacité */
b_1-b_2-...-b_m /* membres droits des contraintes de capacité */
c_{11}-a^1_{11}-a^2_{11}-...-a^m_{11} /* profit de l'objet 1 du groupe 1, puis coef. de cet objet dans les contraintes */
c_{12}-a^1_{12}-a^2_{12}-...-a^m_{12} /* profit de l'objet 2 du groupe 1, puis coef. de cet objet dans les contraintes */
...
c_{21}-a^1_{21}-a^2_{21}-...-a^m_{21} /* profit de l'objet 1 du groupe 2, puis coef. de cet objet dans les contraintes */
...
c_{g1}-a^1_{g1}-a^2_{g1}-...-a^m_{g1} /* profit de l'objet 1 du groupe g, puis coef. de cet objet dans les contraintes */
c_{gg_i}-$a^1_{gg_i}$-$a^2_{gg_i}$-...-$a^m_{gg_i}$ /* profit de l'objet g_i du groupe g, puis coef. de cet objet dans les contraintes */

Un - (tiret du 6) sépare bien chaque valeur.

Exemple de fichier d'instance :
3-2-2
25-50
20-15-22
18-16-21
15-12-16
14-12-14
12-10-15
10-8-13

Cette instance contient 3 groupes de 2 objets (donc 6 objets), avec 2 contraintes de capacité. La formulation mathématique correspondante est :

$$\left[\begin{array}{lll} \max & 20x_{11} + 18x_{12} + 15x_{21} + 14x_{22} + 12x_{31} + 10x_{32} & \\ \text{s.c. :} & 15x_{11} + 16x_{12} + 12x_{21} + 12x_{22} + 10x_{31} + 8x_{32} & \leq 25 \\ & 22x_{11} + 21x_{12} + 16x_{21} + 14x_{22} + 15x_{31} + 13x_{32} & \leq 50 \\ & x_{11} + x_{12} = 1 & \\ & x_{21} + x_{22} = 1 & \\ & x_{31} + x_{32} = 1 & \\ & x_{ij} \in \{0, 1\}, \quad i = 1, 2, 3, j = 1, 2 & \end{array}\right.$$

Votre algorithme doit **produire un fichier texte** contenant la meilleure solution obtenue et sa valeur.

Soit x cette solution. Votre fichier devra respecter le format suivant :

nom_du_fichier_en_entrée
valeur_de_la_solution
pour i de 1 à g faire
  pour chaque j de 1 à g_i faire
    écrire(x_{ij})

La valeur de la solution correspond à la fonction objectif : $\sum_{i=1}^{g} \sum_{j \in G_i} c_{ij} x_{ij}$.

Modalités d'évaluation des algorithmes

- La machine utilisée pour les tests est équipée d'un processeur `Intel Core i7` 2.6 GHz avec 8 Go de RAM, et **Windows 7**.
- Vous devrez fournir, par l'intermédiaire de `Moodle`, l'ensemble des fichiers **sources** permettant de générer l'exécutable correspondant à votre programme, un `ReadMe` expliquant comment générer cet exécutable (en précisant notamment les options de compilation à utiliser si besoin). Vous déposerez également une version de l'exécutable sur `Moodle` (si compatible avec Windows, sinon inutile).
- Votre exécutable doit obligatoirement respecter la syntaxe d'appel suivante (exemples), en ligne de commande :
  - en C/C++ :
    *miniprojet_* < n° *_du_groupe* > *.exe chemin_instance fichier_sortie temps*
  - en Java :
    *java miniprojet_* < n° *_du_groupe* > *chemin_instance fichier_sortie temps*
  - le paramètre *chemin_instance* permettra d'indiquer quelle instance doit être résolue (les instances **ne seront pas nécessairement** dans le même répertoire que l'exécutable). Exemple de valeur possible : C :\M1CYBER\Tests\Instances\I1.txt
  - le paramètre *fichier_sortie* donne le nom du fichier qui contiendra votre solution (par défaut dans le même répertoire que l'exécutable).
  - le paramètre *temps* est le temps de recherche accordé à l'algorithme, en secondes (par défaut 60 secondes).

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] Hifi, M., Michrafy, M. et Sbihi, A. Heuristic algorithms for the multiple-choice multidimensional knapsack problem. Journal of the Operational Research Society, 55 : 1323–1332, 2004.

[2] Hifi, M., Michrafy, M. et Sbihi, A. A Reactive Local Search-Based Algorithm for the Multiple-Choice Multi-Dimensional Knapsack Problem. Computational Optimization and Applications, 33(2) : 271–285, 2006.

[3] Chu, P. C. et Beasley J. E. A Genetic Algorithm for the Multidimensional Knapsack Problem. Journal of Heuristics, 4 : 63–86, 1998.

[4] 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.

[5] 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.

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