UPMC SORBONNE UNIVERSITÉS

Module RP
Résolution de Problèmes

Projet

Google Hash Code : "Optimize a data center"

Le "Google Hash Code" est une compétition de programmation pour des équipes d'étudiants ou de non-étudiants. La compétition est basée sur un problème, le plus souvent en optimisation combinatoire, et sur une instance de taille importante à résoudre. Le gagnant de cette compétition est celui qui a le meilleur "score", c'est-à-dire une solution optimisant la fonction objetuf sur cette instance.

Pour ce projet du module "Résolution de Problèmes", nous vous proposons un problème directement issu du "Qualification Round" du Google Hash Code 2015.

On considère dans ce projet la résolution du problème d'affectation de serveurs dans un centre de données en utilisant des techniques de résolution heuristique et exacte. Vous pouvez trouver la version du projet à la page

`http://hashcode.withgoogle.com/past_editions.html`

sous la forme d'un sujet et d'une instance appelée dc.in (data-center.input). Une version française est aussi en ligne ici :

`https://sites.google.com/site/hashcode2015/tasks`

Un barême indicatif est indiqué pour chacun des exercices.
La page web de ce projet est

`http://www-desir.lip6.fr/~fouilhoux/documentens.php`

où seront mis en ligne toutes les informations et indications utiles, ainsi que l'affichage des meilleurs scores obtenus lors du projet !

1 Définition du problème

L'énoncé suivant reprend celui du sujet initial du Google HashCode en introduisant certaines notations utiles nouvelles.

1.1 Enoncé du problème

Un entrepôt de données est composé d'un ensemble de ressources informatiques, appelées *slots*. Chaque slot peut être utilisé pour installer un *serveur*, c'est-à-dire une machine virtuelle. L'en-

semble de tous les serveurs doit être partitionné en "pool" : chaque pool correspondra à une activité indépendante.

Un entrepôt est organisé en *rangées* où les slots sont placés les uns après les autres. Les slots d'une même rangée sont reliés entre eux par les ressources électriques et le câblage informatique : ainsi, en cas de panne d'une rangée, tous les serveurs de cette rangée sont en panne.

Le problème est à la fois de localiser les serveurs sur les slots et d'affecter les serveurs aux pools. Cette double affectation est réalisée de manière à résister le plus possible à une panne d'une rangée : pour cela, pour chaque pool, on va répartir les serveurs de ce pool sur plusieurs rangées pour que la panne d'une rangée permette au mieux de poursuivre les activités de ce pool.

Une instance du problème est donnée par :

- le nombre R de rangées, numérotées de 0 à $R-1$;
- le nombre S de slots par rangée (qui est le même pour chaque rangée). Pour une rangée donnée, les slots sont numérotés de 0 à $|S|-1$;
- l'ensemble $\mathcal{U}$ de slots indisponibles : chaque slot indisponible est donné par un emplacement (r_i, s_i), $i = 0, \ldots, |\mathcal{U}|-1$, indiquant la rangée r_i, et le slot s_i du slot indisponible ;
- le nombre P de pools à créer ;
- l'ensemble $\mathcal{M}$ des serveurs à positionner : chaque serveur est caractérisé par un couple $(z_m, c_m), m = 0, \ldots, |\mathcal{M}|-1$, où
  z_m : le nombre de slots nécessaires pour le serveur, $1 \leq z_m \leq |S|$
  c_m : capacité du serveur (la quantité de ressources fournie par le serveur)
  avec $1 \leq c_m \leq 1000$.

On appelle *affectation* $\mathcal{A}$ un tableau contenant pour chacun des serveurs $m = 0, \ldots, |\mathcal{M}|-1$

- soit $\mathcal{A}[m] = (ar_m, as_m, ap_m)$ où
  ar_m : numéro de la rangée où le serveur est localisé
  as_m : numéro du premier slot de la rangée où le serveur est affecté sur z_m slots consécutifs
  ap_m : numéro du pool où le serveur est affecté
- soit $\mathcal{A}[m] = (\times)$ désignant le fait que le serveur n'a pas été affecté.

Une *solution* du problème est une affectation telle que

- Chaque slot contient au plus un serveur
- Aucun serveur n'est affecté à un slot indisponible
- Un serveur ne s'étend pas au-delà des slots de sa rangée.

1.2 Score d'une solution

Etant donnés une affectation $\mathcal{A}$ et un pool $i \in \{0, \ldots, P-1\}$, on appelle *capacité garantie du pool i* pour l'affectation $\mathcal{A}$ la capacité totale des serveurs du pool en cas de panne d'une

rangée, *i.e.*

$$gc_i(\mathcal{A}) = \min_{r \in \{0,\dots,R-1\}} \sum_{\substack{m=0 \\ \mathcal{A}[m] = (ar_m, as_m, i) \\ ar_m \neq r}}^{|\mathcal{M}|-1} c_m$$

On définit le *score* d'une affectation comme la capacité garantie pour l'ensemble des pools, c'est-à-dire

$$score(\mathcal{A}) = \min_{i \in \{0,\dots,P-1\}} gc_i(\mathcal{A}).$$

L'objectif de ce problème est de déterminer une affectation qui résiste le mieux possible à une panne de l'une des rangées : on choisit de maximiser la capacité garantie de ses pools, *i.e.*

$$\max_{\text{solution } \mathcal{A}} score(\mathcal{A}).$$

1.3 Notations utiles supplémentaires

Il est utile pour ce projet de déduire les informations suivantes de l'instance :

- Pour un serveur $m \in \mathcal{M}$, on considère l'ensemble L_m des slots (r, s) à partir duquel le serveur peut-être localisé, *i.e.* tel qu'il y ait au moins z_m slots disponibles à partir de (r, s).
- Pour un slot (r, s), on considère l'ensemble K_{rs} des serveurs pouvant être localisés au slot (r, s).
- L'ensemble $\mathcal{F} = \{(r, s) \mid r \in \{0, \dots, R-1\}, s \in \{0, \dots, S-1\}, (r, s) \notin \mathcal{U}\}$ des slots disponibles

Pour les informations citées ci-dessus, il est utile de posséder des structures de données efficaces pour stocker et accéder aux informations.

2 Instances

Q 0.1. Récupérer l'instance dc.in de la page du Google Hash Code. Le format de l'instance est précisé dans les documents du Google Hash Code.

Q 0.2. Créer des fonctions lisant p% de cette instance : c'est-à-dire lisant $\left\lfloor \frac{p \times Param}{100} \right\rfloor$ des valeurs

R : lire les $\left\lfloor \frac{pR}{100} \right\rfloor$ premières lignes du fichier concernant les rangées

P : nombre de pools $= \left\lfloor \frac{pP}{100} \right\rfloor$

$|\mathcal{M}|$: lire les $\left\lfloor \frac{p|\mathcal{M}|}{100} \right\rfloor$ premières lignes du fichier concernant les serveurs

Conserver la longueur totale des rangées en S slots. Bien entendu, les slots indisponibles à considérer sont ceux localisés sur les lignes lues de l'instance.

Q 0.3. Réaliser une visualisation de ces instances et surtout des solutions.

3 Techniques de résolutions

Le projet consiste à étudier différentes techniques de solutions pour le problème. Vous pouvez utiliser le langage de programmation et un solveur PLNE de votre choix [1].

Exercice 1 : Méthodes gloutonnes (3 pts)

Q 1.1. Implémenter la méthode gloutonne suivante :
- Trier les serveurs par ordre décroissant de capacités
- Localiser, dans l'ordre décroissant, un à un les serveurs au premier slot disponible de haut en bas, gauche à droite ; affecter alternativement les serveurs aux pools

Q 1.2. Proposer au moins 1 autre méthode gloutonne plus performante (complexité maximale en $O(|\mathcal{M}|^2 R^2 S^2)$).

Exercice 2 : Méta-heuristiques

Dans cette partie, on désire mettre en oeuvre des méthodes méta-heuristiques pour le problème. En plus du cours RP, vous trouverez une documentation dans le support de cours du module MAOA à partir de la page 57 du cours MAOA_ROOC_C_BB.pdf sur le site `http://www-desir.lip6.fr/~fouilhoux/documentens.php`.

Q 2.1. Implémenter la méthode de descente stochastique à partir de votre solution gloutonne à partir des voisinages suivants :

a - Enlever un serveur

b - Placer un serveur non encore affecté sur un emplacement libre tiré au sort et un pool tiré au sort

c - Affecter un serveur déjà affecté à un autre pool tiré au sort

A chaque itération de la descente, vous tirerez au sort quel voisinage appliquer ; les probabilités de choix de ces voisinages sont 30% pour le a, 30% pour le b et 40% pour le c.

Q 2.2. Proposer des améliorations de la méthode de descente proposée ci-dessous : ce peut-être par exemple d'autres voisinages, d'autres probabilités, un cadre différent de méta-heuristiques : recuit simulé, méthode tabou ou algorithme génétique,...

Exercice 3 : Formulations PLNE (*5 points)*

Plusieurs formulations en programmes linéaires en nombres entiers (PLNE) peuvent être obtenues pour ce problème.

Q 3.1. Déterminer une formulation PLNE basée sur les variables suivantes :

z_{mrsi} qui vaut 1 si le serveur m est localisé à partir du slot (r, s) (donc sur z_m slots consécutifs) et affecté au pool i ; et 0 sinon

Q 3.2. En utilisant un solveur de votre choix (gurobi, cplex, xpress, glpk), utioliser cette formulation pour résoudre les instances du problème.

1. Dans le concours Google Hash Code, seuls les solveurs publiques (libres ou open-source) peuvent être utilisés

Exercice 4 : Heuristiques d'arrondi*(3 points)*

Q 4.1. Implémenter la méthode d'arrondi pour le problème à partir des relaxations linéaires de la formulation entière de l'exercice précédent :

- Trier les variables d'affectations z_{mrsi} par ordre décroissant de valeur
- Fixer les variables z_{mrsi} dans cet ordre : à 1 si cette affectation mène à une solution ; et 0 sinon.

Q 4.2. Proposer une autre méthode d'arrondi. Cela peut-être une amélioration de la méthode précédente ou un couplage entre différentes méthodes déjà rencontrées.

Exercice 5 : A vous de jouer*(2 points + points bonus !)*

Q 5.1. Comparer les différentes méthodes des exercices précédents (capacités, rapidité). Déduisez quelles méthodes sont les plus adaptées pour résoudre les instances pour p allant de 10% à 100%.

Q 5.2. Pour les instances difficiles à résoudre par les méthodes précédentes, proposer VOTRE méthode... La meilleure méthode aura des points bonus !

4 Documents à fournir

Pour valider une solution trouvée pour une instance du problème, vous pouvez envoyer un mail à Pierre.Fouilhoux@lip6.fr en respectant le format de solution précisé dans le Google Hash Code (préciser également le nom de la méthode utilisée). Les valeurs des solutions seront publiées sur la page web du projet.

Pour le rendu final, vous rédigerez un rapport très synthétique contenant la description de votre travail, c'est-à-dire :
- une description synthétique de vos méthodes
- une descrition rapide de votre code et de la façon de l'utiliser
- vos analyses pratiques et théoriques
- et vos meilleurs résultats !
- joignez également votre code (commenté bien sûr)

Envoyer un mail à Evripidis.Bampis@lip6.fr et à Pierre.Fouilhoux@lip6.fr avant la séance prévue pour le rendu.
Votre mail doit comporter :
- un objet commençant par [RP]
- contenir 2 fichiers dont le nom commencent par nom1_nom2 en mettant les noms de votre binome par ordre alphabétique
- un des fichiers est l'archive tar.gz de votre code
- un des fichiers est votre rapport en pdf.

Dates du projet
Sujet distribué en TD **le 2 mars 2017**.
Une mini-soutenance aura lieu la semaine du **28 avril 2017** (date à préciser).