MINES
Saint-Étienne

1816
2016

Challenge Optim 2024

24/06/2024

INSPIRING
INNOVATION
SINCE 1816

Le contexte

Un problème d'affectation de processus à des machines

source image GNT

Inspiré du challenge Google ROADEF/EURO 2012 :
https://www.roadef.org/challenge/2012/fr/

Le sujet

Données d'entrée :

- **{0,…,M-1}** : ensemble de M machines
- **{0,…,P-1}** : ensemble de P processus
- **{0,…,R-1}** : ensemble de R ressources
- **r(p,r)** : consommation de la ressource r par le processus p
- **c(m,r)** : capacité de la machine m concernant la ressource r

Solution : Affecter chaque processus à une machine en respectant les contraintes de capacité pour chaque machine et chaque ressource
[ainsi que d'autres contraintes et en minimisant une certaine fonction de coût] (slides suivants)

Contraintes de capacité :

→ Pour chaque machine m et chaque ressource r : $\sum_{p \text{ affecté à } m} r(p,r) \leq c(m,r)$

MINES Saint-Étienne | INSPIRING INNOVATION SINCE 1816

Le sujet

Exemple d'instance à 5 processus, 2 machines et 2 ressources

c(m,r)	Machine 0	Machine 1
Ressource 0	8	5
Ressource 1	6	6

r(p,r)	0	1	2	3	4
Ressource 0	3	3	1	1	2
Ressource 1	4	2	1	3	1

Exemple de solution, les quatre contraintes de capacité sont respectées

- Machine 0 : processus 0, 1
- Machine 1 : processus 2, 3, 4

Consommation	Machine 0	Machine 1
Ressource 0	6	4
Ressource 1	6	5

Le sujet

Données additionnelles :

- **{0,…,S-1}** : ensemble de S services
- **s_p** : service du processus p

Les services forment une partition des processus : chaque processus appartient a un et un seul service

Contraintes de conflit :

→ Deux processus du même service ne peuvent pas être affecté à la même machine

MINES Saint-Étienne | INSPIRING INNOVATION SINCE 1816

Le sujet

Exemple d'instance à 5 processus, 2 machines, 2 ressources et 3 services

c(m,r)	Machine 0	Machine 1
Ressource 0	8	5
Ressource 1	6	6

r(p,r)	0	1	2	3	4
Ressource 0	3	3	1	1	2
Ressource 1	4	2	1	3	1

Service	0	1	2
Processus	{0,1}	{2,3}	{4}

Exemple de solutions

- Machine 0 : processus 0, 1
- Machine 1 : processus 2, 3, 4

Non réalisable

Consommation	Machine 0	Machine 1
Ressource 0	6	4
Ressource 1	6	5

Exemple de solutions

- Machine 0 : processus 0, 2, 4
- Machine 1 : processus 1, 3

Consommation	Machine 0	Machine 1
Ressource 0	6	4
Ressource 1	6	5

Le sujet

Données additionnelles :

- **{0,…,L-1}** : ensemble de L localisations
- **l_m** : localisation de la machine m
- **loc_s** : nombre minimum de localisations auxquelles les processus du service s doivent être affectés

Les localisations forment une partition des machines : chaque machine appartient à une et une seule localisation

Contraintes de répartition géographique :

→ Les processus de chaque service doivent être affectés à des machines localisées sur au moins loc_s localisations

Le sujet

Exemple d'instance à 5 processus, 2 machines, 2 ressources, 3 services et 2 localisations

c(m,r)	Machine 0	Machine 1
Ressource 0	8	5
Ressource 1	6	6

r(p,r)	0	1	2	3	4
Ressource 0	3	3	1	1	2
Ressource 1	4	2	1	3	1

Service	0	1	2
Processus	{0,1}	{2,3}	{4}

Localisation	0	1
Machine	{0}	{1}

Service	0	1	2
loc_s	2	2	1

Réalisable

Exemple de solutions

- Machine 0 : processus 0, 2, 4
- Machine 1 : processus 1, 3

Consommation	Machine 0	Machine 1
Ressource 0	6	4
Ressource 1	6	5

Le sujet

Données additionnelles :

- **sc(m,r)** : valeur maximale souhaitée dans l'utilisation de la ressource r pour la machine m

Remarque : sc(m,r) ≤ c(m,r)

Valeur d'une solution (à minimiser)

→ La valeur d'une solution est la somme des dépassements par rapport aux valeurs maximales souhaitées sc(m,r)

$$\text{valeur} = \sum_{m=0}^{M-1} \sum_{r=0}^{R-1} max\left\{0, \left(\sum_{p \text{ affecté à } m} r(p,r) \right) - sc(m,r)\right\}$$

niveau de consommation de r sur m

Le sujet

Exemple d'instance à 5 processus, 2 machines, 2 ressources, 3 services et 2 localisations

c(m,r)	Machine 0	Machine 1
Ressource 0	8	5
Ressource 1	6	6

r(p,r)	0	1	2	3	4
Ressource 0	3	3	1	1	2
Ressource 1	4	2	1	3	1

Service	0	1	2
Processus	{0,1}	{2,3}	{4}

Localisation	0	1
Machine	{0}	{1}

Service	0	1	2
loc_s	2	2	1

sc(m,r)	Machine 0	Machine 1
Ressource 0	5	4
Ressource 1	4	6

Exemple de solutions de valeur 3

- Machine 0 : processus 0, 2, 4
- Machine 1 : processus 1, 3

Consommation	Machine 0	Machine 1
Ressource 0	6	4
Ressource 1	6	5

Excédent	Machine 0	Machine 1
Ressource 0	1	0
Ressource 1	2	0

Les entrées / sorties

15 instances sont proposées

1 fichier data**n**.txt par instance, où **n** est le numéro de l'instance

Format (dans l'ordre du fichier)

nbLocalisations	4
nbRessources	2
nbMachines	4
nbProcessus	100
nbServices	79

Valeurs L, R, M, P et S

*******Machines*******				
2	4420	4322	3729	3677
3	2888	3106	2717	2323
1	3213	4634	2277	3836
0	3973	5354	3591	4320

Données machines
(de la machine 0 à la machine M-1)

(localisation, capacité par ressource, valeur maximale souhaitée par ressource)

*******Services*******
3
0
…

Donnée loc_s
(du service 0 au service S-1)

Les entrées / sorties

Format (suite)

*******Processus*******		
10	1	1
11	425	435
12	1	1
13	3	3
…	…	…

(service, consommation par ressource)

Données processus
(du processus 0 au processus P-1)

Les entrées / sorties

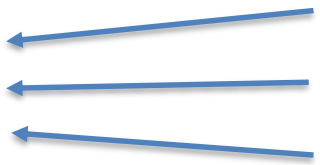
Sorties : un fichier par instance résolue

Nom proposé pour les fichiers solutions : **res_n.txt** où n est le numéro de l'instance

Attention : extension .txt obligatoire

Format du fichier :

EQUIPE	ZeProf			
INSTANCE	1			
0	3	0	1	…



Nom de l'équipe

Numéro de l'instance

Machine choisie
(du processus 0 au processus P-1)

Attention : Nom de l'équipe au choix mais **sans caractères spéciaux ni accents**

Algorithme ALGO1

```
pour chaque processus p de P
    s ← service auquel appartient p
    si le service s ne couvre pas encore loc_s localisations
        essayer d'affecter p à une machine m d'une localisation non couverte par s
        si échec
            essayer d'affecter p à n'importe quelle machine m
        fin si
    sinon
        essayer d'affecter p à n'importe quelle machine m
    fin si
    si aucune affectation possible pour p
        return false
fin pour
return true
```

Une affectation est possible lorsqu'elle respecte les contraintes de capacité et de conflit

Attention : cet algorithme se contente de chercher une solution, il n'optimise pas sa valeur

Pseudo code détaillé

```
pour chaque service s
    nb_loc[s] ← 0   // nombre de localisations couvertes par le service s
fin pour
pour chaque processus p de P
    s ← service auquel appartient p
    new_loc ← false // booléen : affectation nouvelle loc. possible ou non
    si nb_loc[s] < loc_s
        m ← testNewLoc(p) // on essaie d'affecter p dans une nouvelle loc.
        si m != machineneutre // affectation possible (cf slide suivant)
            machine[p] ← m  // affectation de p à m
            new_loc ← true
            nb_loc[s] ← nb_loc[s]+1
            updateRessources(p,m)
        fin si
    fin si
    si new_loc = false
        m ← testAll(p)  // on essaie d'affecter p n'importe où
        si m != machineneutre // affectation possible
            machine[p] ← m  // affectation de p à m
            updateRessources(p,m)
        else
            return false // aucune affectation possible pour p, échec de l'algorithme
        fin si
    fin si
fin pour
return true // tous les processus ont été affectés
```

Pseudo code détaillé

```
testNewLoc(processus p)

s ← service auquel appartient p
pour chaque machine m
    l ← localisation de la machine m
    si useLoc[l][s] = false // l est une nouvelles localisation pour le service
        si testRessources(p,m) = true // test des contraintes de capacité
            return m
        fin si
    fin si
fin pour
return machineneutre // convention à définir : aucune machine n'a été trouvée
```

Attention : initialisation de la matrice useLoc[l][s] à false en amont, mise à jour de useLoc[l][s] à faire en aval

Remarque : utiliser une nouvelle loc. garantit de ne pas avoir de conflit

Remarque : avec des structures de données adaptées il est possible de plutôt itérer sur les localisations puis sur les machines de la localisation courante

Pseudo code détaillé

```
testAll(processus p)

pour chaque machine m
    s ← service auquel appartient p
    si useMach[m][s] = false // test de conflit entre m et s
        si testRessources(p,m) = true // test des contraintes de capacité
            return m
        fin si
    fin si
fin pour
return machineneutre // convention à définir : aucune machine n'a été trouvée
```

Attention : initialisation de la matrice useMach[l][s] à false en amont, mise à jour de useMach[l][s] à faire en aval

Pseudo code détaillé

```
testRessources(processus p, machine m)

pour chaque ressource r
    si capa_res[m][r] < r(p,r)
        return false
    fin si
fin pour
return true
```

```
updateRessources(processus p, machine m)

pour chaque ressource r
     capa_res[m][r] ← capa_res[m][r] - r(p,r)
fin pour
```

Attention : initialisation de la matrice capa_res[m][r] à c(m,r) en amont

Algorithme (suite)

Structures de données

Il est conseillé d'introduire des types structurés pour les processus et pour les machines

- processus : identifiant du processus, données r(p,r), service s_p
- machine : identifiant de la machine, données c(m,r), données sc(m,r), localisation l_m

Ces structures seront en particulier très utiles si vous souhaitez faire des tris, sur les ressources par exemple, et continuer à accéder facilement aux données associées

Evaluation

Note : moyenne des notes sur l'ensemble des instances

Note pour une instance :

- Aucune solution aussi bonne que ALGO1 n'a été transmise : 8
- Une solution de score supérieur à ALGO1 a été transmise :

  Note proportionnelle à la qualité de la solution, sur l'intervalle [10,20]
  (note 10 pour un score ALGO1, note 20 pour la meilleure solution reçue)

Dans le cas où aucune solution n'aurait été soumise pour au moins une instance, la note sera définie sur la base des solutions déposées et du code

Organisation

1. A effectuer en binôme. Choisir son nom d'équipe.
2. Se rendre sur campus et récupérer les fichiers : le sujet (ces transparents), les 15 fichiers d'instance, le checker
3. Choisir son langage (et son environnement)
4. Mettre à jour les fichiers solutions obtenus sur campus jusqu'à **18h00**
5. 18h00 : remise des prix
6. Avant 20h : déposer votre code sur campus

Les meilleurs résultats sont projetés tout au long de la journée

Organisation

Fonctionnement du checker

check.exe res_1.txt res_2.txt …

ou

check.exe, puis indiquer le nom du fichier résultat à vérifier

Disponible sous Linux ou sous Windows

Pour une exécution sous Linux, bien vérifier les droits
(commande chmod)

Quelques conseils

Si vous êtes peu à l'aise en programmation, nous vous conseillons fortement de programmer en C et de suivre la méthodologie suivante :

1. Commencer par définir sur papier les structures de données qui seront utilisées
2. Faire un planning (horaire) de développement, étape par étape
3. Ne pas hésiter à faire appel aux enseignants pour valider ces structures et ce planning
4. Nous consulter si vous constatez un retard inquiétant par rapport à ce planning

Le challenge est noté mais les enseignants sont à votre disposition pour vous aider tout au long de la journée et répondre à vos questions, comme pour un TP

MINES Saint-Étienne | INSPIRING INNOVATION SINCE 1816

Je vous souhaite un EXCELLENT CHALLENGE