École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique
5e année
2013 - 2014

Rapport des Tests

Rapport des tests du problème SCP (hors-ligne)

Encadrants
Vincent T'Kindt
vincent.tkindt@univ-tours.fr

Étudiants
Lei SHANG
lei.shang@etu.univ-tours.fr

Université François-Rabelais, Tours

DI5 2013 - 2014

Version du 23 février 2014

Table des matières

Table des figures

Liste des tableaux

Valide inequalities

1.1 Cuts classiques

Les Cuts classiques sont des coupes qui ne dépendent pas la particularité du problème. Nous présentons dans cette section qu'un seul type de coupes : 1-Cuts.

1.1.1 1-Cuts

1-cuts par Osorio et al.(2002) sont des coupes générées à partir des contraintes de types $d^T x \leq b$ avec $d_1 \geq d_2 \geq \cdots \geq d_n > 0$. Ce sont donc des contraintes redondantes qui peut pourtant plus efficaces. Par exemple pour la contrainte $x_1 + 2x_2 + 2x_3 \leq 3$ dont les variables sont binaires, on peut en déduire un 1-cuts $x_2 + x_3 < 1$, car si $x_2 = x_3 = 1$ la contrainte originale sera violée.

Il existe déjà l'algorithme[?] pour générer automatiquement les 1-cuts, alors nous l'avons réalisé et ensuite appliqué sur les contraintes de ressources dans le Preprocessing. Le résultat de test montre que ces coupes ont bien un effet possitif pour fixer plus de variables surtout pour les permiers 3 scénarios. Cependant, nous avons aussi aperçu que le nombre de coupes générées est considérable pendant cette démarche, ce qui peut potentiellement avoir un effet négatif sur le temps de résolution, parce que quand nous avons de nombreux contraintes ajoutés, Cplex va alors mettre plus de temps pour traiter ces contraintes.

Pour résoudre ce problème, nous nous posons la question : combien de 1-cuts devons nous générer et quelles sont les contraintes prioritaires. Empiriquement, nous décitons de trier les contraintes par ordre croissante de la partie droite de l'équation (noté RHS) car quand le RHS est plus petit, ça génère des coupes plus contraignant. Après, en considérant la partie gauche de l'équation, nous avons aussi essayé une deuxième approche, c'est de trier les contraintes par ordre décroissante de LRS/RHS, dont LRS est la somme des coefficients à gauche de l'équation.

Pour la question sur le nombre de coupes à générer, nous avons fait un test sur des instances choisies avec un nombre différent des coupes pour voir c'est combien le seuil pour chaque scénario. Ensuite avec le résultat des scénarios 4, 5 et 6 comme échantillons, nous avons trouvé une fonction qui peut donner un seuil selon le nombre de tâche et le nombre de machine du problème. La recherche de cette fonction est basé sur la procédure de "Multiple Linear Regression". Un outil en ligne [1] a été utilisé pour cette recherche.

Les différents tests effectués sont décrits dans les chapitres suivants.

1.2 Cuts problème-dépendent

1.2.1 Cuts sur les contraintes des ressources

Ressources CPU/GPU

Si la tâche i ne peut pas être affectée au serveur j à cause de la capacité résiduelle de CPU/GPU du serveur, alors pour toute les tâches qui demandent plus de CPU/GPU que la tâche i, cette affectation ne peut pas être effectuée non plus.

Cette contrainte peut être exprimée de façon suivante :

1. http://www.xuru.org/rt/MLR.asp

$Si\ n^c_{i\prime} \geq n^c_i\ alors$

$$x_{i,j,t} + x_{i\prime,j,t} \leq (m^c_j - \sum_{k=1;k\neq i,i\prime;u_{k,t}=q_{k,j}=1}^{N} n^c_k x_{k,j,t})/n^c_i \qquad \forall t = 1,\ldots,T, tq\ u_{i,t} = u_{i\prime,t} = 1;$$
$$\forall j = 1,\ldots,M, tq\ q_{i,j} = q_{i\prime,j} = 1 \qquad (A)$$

$Si\ n^g_{i\prime} \geq n^g_i\ alors$

$$x_{i,j,t} + x_{i\prime,j,t} \leq (m^g_j - \sum_{k=1;k\neq i,i\prime;u_{k,t}=q_{k,j}=1}^{N} n^g_k x_{k,j,t})/n^g_i \qquad \forall t = 1,\ldots,T, tq\ u_{i,t} = u_{i\prime,t} = 1;$$
$$\forall j = 1,\ldots,M, tq\ q_{i,j} = q_{i\prime,j} = 1 \qquad (B)$$

À noter que nous n'avons pas besoin de considérer ici la contraite de préaffectation.

Ressources HDD/RAM

Les cuts sur les ressources HDD/RAM ont le même principe sauf que ces ressources puissent aussi être occupées par l'opération de la migration. Nous pouvons appliquer les mêmes cuts comme pour CPU/GPU mais la prise en compte de la migration peut rendre le cut plus strict.

$Si\ n^h_{i\prime} \geq n^h_i\ alors$

$$x_{i,j,t} + x_{i\prime,j,t} \leq (m^h_j - \sum_{k=1;k\neq i,i\prime;u_{k,t}=q_{k,j}=1}^{N} n^h_k x_{k,j,t} - \sum_{k=1;k\neq i,i\prime}^{N}\sum_{l=1;l\neq j}^{M} n^h_k y^{l,j}_{k,k,t})/n^h_i \qquad \forall t = 1,\ldots,T, tq\ u_{i,t} = u_{i\prime,t} = 1;$$
$$\forall j = 1,\ldots,M, tq\ q_{i,j} = q_{i\prime,j} = 1 \qquad (C)$$

$Si\ n^r_{i\prime} \geq n^r_i\ alors$

$$x_{i,j,t} + x_{i\prime,j,t} \leq (m^r_j - \sum_{k=1;k\neq i,i\prime;u_{k,t}=q_{k,j}=1}^{N} n^r_k x_{k,j,t} - \sum_{k=1;k\neq i,i\prime}^{N}\sum_{l=1;l\neq j}^{M} n^r_k y^{l,j}_{k,k,t})/n^r_i \qquad \forall t = 1,\ldots,T, tq\ u_{i,t} = u_{i\prime,t} = 1;$$
$$\forall j = 1,\ldots,M, tq\ q_{i,j} = q_{i\prime,j} = 1 \qquad (D)$$

1.2.2 Machines équivalentes* (incorrect, à enlever)

Deux serveurs j et j' peut être totalement identique dans une instance de problème si :

1. Les caractéristiques (CPU/GPU/HDD/RAM) de j et j' sont identiques
2. $q_{i,j} = q'_{i,j}$ pour $\forall i$
3. ~~j et j' ont les même voisins dans le réseau~~
4. ~~Pour chaque voisin v de j (et j'), la bande passante entre v et j est la même que celle entre v et j'~~

~~Plus largement, le j et j' ci-dessus peuvent être considérés comme un sous-réseau mais pas simplement un seul serveur. Comme ça, étant donné une solution, on peut très bien construir une autre en échangeant les affectation en j et j'.~~

[Update]Après étude du réseau, il parâit que toutes les machines sont connectées et la bande passante est unique donc on n'a besoin que de considérer les deux premières conditions. [Update]Mais quand même, si l'affectation ne se fait pas au premier instant, c'est possible que l'environnement réseau des machines équivalentes devient différent.

Si j et j' sont deux machines identiques alors pour le premier instant de temps où il y a des tâches qui s'exécutent sur j ou j', alors on peut forcer que le serveur j est plus utilisé que j' pour éliminer les solutions redondantes :

$$\sum_{t_1=1}^{t-1}\sum_{i_1=1}^{N} x_{i_1,j,t_1} + CPUUsed(j,t) - CPUUsed(j',t) \geq 0 \qquad \forall t = 1,\dots,T; \quad \forall j, j' = 1 \dots M, j < j', \; équivalent(j,j'); \qquad \text{(E)}$$

[Update]C'est faux ! On ne doit pas utiliser la différence des CPUUsed dans le formule car cette différence n'est pas binaire, donc même si $\sum_{t_1=1}^{t-1}\sum_{i_1=1}^{N} x_{i_1,j,t_1} \geq 0$, cette contrainte est quand même fonctionnelle.

1.2.3 Dominance des tâches* (à enlever)

Cette fois on va prendre en compte toutes les ressources requises par la tâche. On modélise 2 tâches i et $i\prime$ pour que la tâche $i\prime$ a besoin de plus de ressource que i pour tout type de ressources y compris la partie du réseau.

La formulation courante de ce cut n'est peut-être pas valide, parce que même si nous avons $x_{i,j,t} = 0$ ça ne veut pas dire qu'il n'y a pas de ressources pour la tâche i. Ça peut simplement parce que $x_{i,j,t} = 1$ ne conduit pas à la solution optimale.

Alors si nous voulons créer un Cut sur les ressources du réseau, je pense c'est quand même mieux de reprendre le même principe : si on est sûr que les ressources réseau sont insuffisantes pour la tâche i alors c'est pareil pour $i\prime$. Cependant cette modélisation (état de l'insuffisance de ressources) ne me semble pas évidente.

Test des méthodes de résolution sans Preprocessing

Pour tester le fonctionnement des différentes méthodes de résolution dans ce projet, plusieurs tests ont été effectués sur un ensemble d'instances du problème. Cet ensemble contient 6 scénarios dont chacun est composé par 20 instances du problème. Toutes ces données de test sont générées par le programme Testeur.

2.1 Méthode exacte - solveur Cplex

Dans un premier temps, le test du solveur Cplex est effectué car les solutions trouvées par le solveur peuvent nous servir à évaluer la performance des autres méthodes de résolution.

Le tableau 2.1 représente les statistiques effectuées sur le résultat de test du solveur Cplex. Les significations des colonnes sont :

1. Sc(N/M) : numéro de scénario et le nombre de VM et de serveur physique.
2. #Infeas : le nombre des instances qui sont prouvées comme "Infaisable" par le solveur.
3. #Opt : le nombre des instances qui sont résolues avec la solution optimale trouvée.
4. #Mem : le nombre des instances pour lesquelles le solveur n'a pas pu trouver la solution optimale à cause de la limite de l'espace mémoire.
5. #Tim : le nombre des instances pour lesquelles le solveur n'a pas pu trouver la solution optimale à cause de la limite du temps.
6. T_{min}, T_{avg}, T_{max} : le temps (minimum, moyenne et maximum) de résolution en seconde.

Sc(N/M)	#Infeas	#Opt	#Mem	#Tim	T_{min}	T_{avg}	T_{max}
Sc1(8/2)	2	18	0	0	0.02	0.07	0.15
Sc2(11/3)	9	11	0	0	0.08	0.34	1.08
Sc3(15/4)	1	19	0	0	0.59	4.33	54.57
Sc4(18/5)	0	19	0	1	1.88	239.53	1800.37
Sc5(21/5)	3	6	1	10	1.68	1196.57	1800.74
Sc6(24/6)	2	5	0	13	2.06	1316.75	1820.14

TABLE 2.1 – Résultat de test du solveur Cplex

A partir du scénario 4, on commence à avoir des grosses instances pour lesquelles le solveur n'a pas pu trouver la solution optimale ($T_{max} = 1800$) à cause de la limite de temps.

2.2 Méthode heuristique de liste

Le tableau 2.2 montre le résultat de test de la méthode heuristique de liste (on l'appelle H1 pour raison de simplicité). Dans ce tableau, les colonnes D_{min}, D_{avg} et D_{max} signifient la déviation entre la solution

trouvée par H1 et la solution trouvée par le solveur Cplex. A noter que pour les 3 premiers scénarios le solveur a trouvé la solution optimale pour toutes les instances donc cette déviation peut montrer la qualité de notre méthode H1 par rapport à la solution optimale. A partir du scénario 4 on commence à avoir des instances pour lesquelles le solveur a trouvé une solution faisable mais pas optimale à cause de la limite du temps ou de l'espace mémoire, alors dans ce cas la déviation est aussi affectée.

Sc(N/M)	#Infeas	#Solved	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
Sc1(8/2)	2	18	0.00	0.00	0.00	0%	19%	67%
Sc2(11/3)	9	11	0.00	0.00	0.00	0%	17%	46%
Sc3(15/4)	7	13	0.00	0.00	0.00	5%	24%	58%
Sc4(18/5)	11	9	0.00	0.00	0.00	10%	27%	37%
Sc5(21/5)	16	4	0.00	0.00	0.01	30%	36%	43%
Sc6(24/6)	20	0	0.00	0.01	0.02	*	*	*

TABLE 2.2 – Résultat de test de la méthode heuristique de liste (H1)

On trouve souvent 0.00 seconde dans les colonnes de temps, ce qui signifie que le temps de résolution de H1 est très court.

Par rapport à la déviation, la performance de H1 n'est pas très stable : pour certaines instances du problème, H1 a trouvé la solution optimale ($D_{min} = 0\%$) mais on a aussi dans le Sc1 $D_{max} = 67\%$ qui n'est pas très optimiste. Pour le scénario 6, H1 n'a résolu aucune instance donc la déviation n'est pas calculée.

2.3 Méthode heuristique de Cplex

Le tableau 2.3 montre le résulat de test sur la méthode heuristique basée sur le solveur Cplex (H2).

Sc(N/M)	#Infeas	#Solved	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
Sc1(8/2)	2	18	0.02	0.07	0.15	0%	0%	1%
Sc2(11/3)	9	11	0.08	0.29	0.97	0%	0%	1%
Sc3(15/4)	1	19	0.64	1.37	5.21	0%	0%	2%
Sc4(18/5)	0	20	1.53	80.33	400.32	0%	0%	2%
Sc5(21/5)	3	17	1.68	240.31	400.41	0%	2%	4%
Sc6(24/6)	2	18	2.08	286.79	410.43	-1%	1%	7%

TABLE 2.3 – Résultat de test de la méthode heuristique de Cplex (H2)

Puisque la méthode H2 est basée sur le solveur Cplex, pour les petites instances de problème elle a trouvé des solutions qui sont très proches des solutions optimales. Pour les grandes instances (à partir du scénario 4), les solutions qu'elle a trouvées sont aussi très intéressantes avec la déviation maximale égale à 7% qui reste acceptable. Le temps maximum de résolution est vers 200 secondes qui provient de l'implémentation de H2.

Test du Preprocessing

Un autre ensemble de tests a été réalisé pour évaluer la performance de l'approche Preprocessing avec des coupes différentes.

On rappelle ici l'idée du Preprocessing est de fixer autant que possible de variables dans le modèle pour réduire la taille du problème. Pour le faire, il faut avoir une borne supérieure (UB) et une borne inférieure (LB) qui sont assez proches de la solution optimale.

3.1 Preprocessing sans coupes supplémentaires

Nous avons lancé d'abord le Preprocessing sur toutes les variables booléennes sans ajoutant des coupes supplémentaires.

Dans le tableau 3.1 on peut trouver des statistiques sur la qualité des LB et des UB ainsi que la proportion de variables fixées pendant le Preprocessing. Les colonnes s'interprêtent comme le suivant :

1. DevLB : la déviation (minimum, moyenne et maximum) entre la solution optimale et LB.
2. DevUB : la déviation (minimum, moyenne et maximum) entre UB et la solution optimale.
3. Fixed : la proportion du nombre de variables fixées par rapport au nombre de toutes les variables ayant passé le preprocessing.

	DevLB			DevUB			Fixed		
Sc(N/M)	min	avg	max	min	avg	max	min	avg	max
Sc1(8/2)	0.00%	4.83%	17.07%	0.00%	0.04%	0.65%	0.00%	9.25%	51.83%
Sc2(11/3)	0.04%	7.21%	17.13%	0.00%	0.01%	0.15%	0.00%	13.90%	66.67%
Sc3(15/4)	1.10%	4.01%	10.12%	0.00%	0.36%	1.80%	0.00%	0.88%	10.96%
Sc4(18/5)	0.05%	5.28%	12.56%	0.00%	0.53%	1.81%	0.00%	2.86%	55.64%
Sc5(21/5)	1.82%	5.03%	8.94%	0.56%	1.94%	7.62%	0.00%	0.00%	0.00%
Sc6(24/6)	3.94%	5.91%	8.72%	0.30%	1.75%	5.40%	0.00%	0.09%	0.56%

TABLE 3.1 – Résultat de test du Preprocessing sur toute les variables booléannes

Nous constatons que le Preprocessing naïf fonctionne mais pas suffisamment bien. Il ne fixe pas beaucoup de variables donc il n'aide pas la résolution de Cplex. Souvent, ce problème survient quand l'UB ou LB n'est pas assez bonne. En comparant la qualité de LB et de UB, on peut trouver que relativement les UB sont assez proches que la solution optimale, en revanche les LB ne sont pas très bonnes.

Pour améliorer les LB, nous avons cherché d'ajouter des contraintes supplémentaires (Cuts) au modèle LP du problème, pour enfin améliorer le fonctionnement du Preprocessing.

3.2 Preprocessing avec coupe 1

Nous avons d'abord relancé le test de Preprocessing avec la coupe 1 (cf 1.2.1) qui est problème-dépendant concernant les contraintes de ressources. Malheureusement, l'ajout de cette coupe n'a apporté aucun changement au résulat de test (identique que 3.1). Nous pouvons donc dire que la coupe 1 n'est

pas utile pour le fixage des variables pendant le Preprocessing, mais c'est très probable qu'elle est efficace dans le MIP. Néanmoins, à cause de la limite de temps, nous n'avons pas pu tester la performance de cette coupe dans le MIP.

3.3 Preprocessing avec coupe 2

Cette partie contient le résultat (tableau3.2) de test de Preprocessing avec la coupe 2 (aussi notée 1-Cuts, cf 1.1.1).

	DevLB			DevUB			Fixed		
Sc(N/M)	min	avg	max	min	avg	max	min	avg	max
Sc1(8/2)	0.00 %	2.87%	17.07 %	0.00%	0.04%	0.65%	0.00%	43.91 %	100.00%
Sc2(11/3)	0.04 %	4.54%	11.97 %	0.00%	0.01%	0.15%	0.00%	15.80 %	66.67%
Sc3(15/4)	0.44 %	3.47%	9.40 %	0.00%	0.36%	1.80%	0.00%	2.22 %	10.96%
Sc4(18/5)	0.05 %	4.94%	11.60 %	0.00%	0.53%	1.81%	0.00%	2.91 %	55.64%
Sc5(21/5)	1.82 %	4.72%	8.82 %	0.56%	1.94%	7.62%	0.00%	0.00 %	0.00%
Sc6(24/6)	3.64 %	5.65%	8.45 %	0.30%	1.75%	5.40%	0.00%	0.09 %	0.56%

TABLE 3.2 – Résultat de test du Preprocessing avec la coupe 2

Dans le résultat, les trois colonnes de UB restent les mêmes car le Preprocessing n'affecte que sur les LB. Nous pouvons trouver que avec la coupe 2, le Preprocessing a pu fixé plus de variables qu'avant, surtout pour les premiers 3 scénarios. En conséquence, les LB ont été aussi améliorées (la déviation entre la solution optimale et la LB devient plus petite).

En résumé, la coupe 2 est utile pour renforcer le Preprocessing mais le résultat n'est pas encore assez bon pour bien accélérer la résolution après. A partir de scénario 4, le Preprocessing n'aide quasiment plus.

3.4 Preprocessing avec coupe 1&2* (à enlever, peut-être)

Le résultat du test de Preprocessing avec à la fois la coupe 1 et la coupe 2 reste pareil que le Preprocessing avec la coupe 2 seule (3.2). Ce fait a confirmé que la coupe 1 n'est pas utile pour le fixage des variables.

3.5 Preprocessing avec la coupe 2 renforcée

Selon les tests que nous avons fait, la coupe 2 a des effects positifs pour le Preprocessing. Nous avons donc effectué des démarches pour encore renforcer la coupe 2.

Dans la phase de fixage, le Preprocessing peut fixer des variables en générant des contraintes, mais quand le nombre de contraintes devient plus en plus grand, ça peut au contraire ralentir la résolution de Cplex dans la phase suivante. Dans notre cas, puisque la coupe 2 va générer beaucoup de contraintes, nous essayons donc de trouver un seuil supérieur pour le nombre des contraintes générés. Mise en place de ce paramètre doit diminuler autant que possible le temps utilisé pour la résolution MIP, tout en assurant que le nombre de variables fixées ne soit pas diminué.

Pour trouver la relation entre le nombre de contraites générées à partir de la coupe 2 et le temps dépensé pendant la résolution Cplex, nous avons conçu deux tests d'analyse.

Premier test de l'analyse du nombre de coupes à générer

Les principles du premier test sont :

1. Pour chacun des scénarios 4, 5 et 6, nous choisissons 5 instances de problème qui peuvent être résolues à l'optimalité dans un temps modéré (pas trop court pour être observable ni trop long pour ne pas dépasser la limite de temps).
2. Pour chaque instance, nous faisons un série de tests avec un seuil de nombre de coupes différents. Ce seuil commence à 200 puis s'incrémente d'un pas de 200 jusqu'à le nombre total des coupes qu'on peut générer.
3. Après le test, pour chaque scénario, nous cherchons le seuil qui donne un temps moyen d'exécution minimum pour les 5 instances choisies.
4. Pour que les coupes générées soient les plus contraignantes, nous avons trier avant la génération de coupe 2, les contraintes en ordre croissante de RHS (cf 1.1.1).

Le résultat du test (tableau3.3) nous a donné des échantillons pour étudier la corrélation entre le meilleur seuil et les caractéristiques de scénarios.

Sc	N	M	BestSeuil
4	18	5	600
5	21	5	400
6	24	6	1600

TABLE 3.3 – Résultat du premier test d'analyse du seuil de la coupe 2

Ce résultat montre que quand N (le nombre de tâches) augmente, BestSeuil diminue et quand M (le nombre de machines) augmente, BestSeuil augmente. Empiriquement, nous avons fait une "Multiple Linear Regression" [1] sur ces données de test pour trouver une fonction qui peut donner le meilleur seuil pour un scénario. Le fonction trouvée (notée Seuil1) est :

$$BestSeuil = -66.67N + 1400M - 5200 \quad \text{(A)}$$

Elle s'applique pour les scénarios après le 4, car les 3 premiers sont trop facile.

Néanmoins, la corrélation entre le temps écoulé et le nombre de contraintes n'est pas très évident dans le cas de Cplex. Ça concerne le mécanisme au sein de Cplex que nous ne pouvons pas savoir donc cet approche est avant tout empirique.

Deuxième test de l'analyse du nombre de coupes à générer

Une deuxième analyse (tableau3.4) a été faite pour chercher le meilleur seuil de la coupe 2. Cet analyse est presque pareil que la première sauf que cette fois au lieu de trier les contraintes en ordre croissante de RHS, nous les trions en ordre décroissante de LHS/RHS, dont LHS signifie le somme de tous les coefficients à gauche de l'équation.

Sc	N	M	BestSeuil
4	18	5	1800
5	21	5	1200
6	24	6	2600

TABLE 3.4 – Résultat du deuxième test d'analyse du seuil de la coupe 2

La fonction trouvée (notée Seuil2) cette fois est :

$$BestSeuil = -200N + 2000M - 4600 \quad \text{(B)}$$

1. http://www.xuru.org/mlr

Test de la coupe 2 avec seuil

Après mettre en place la fonction de seuil, nous avons relancé encore 2 fois le test de la coupe 2 avec la fonction de seuil différente.

D'abord sur le fixage des variables, avec la mise en oeuvre du seuil, le nombre de variables fixées reste le même pour tous les 6 scénarios, sauf 3 instances (tableau3.5) qui sont affectées par la première fonction de seuil (Seuil1) :

Sc-id	NoSeuil	Seuil1	Seuil2
3-11	485	484	485
3-19	677	666	677
4-8	101	0	101

TABLE 3.5 – Nombre des variables fixées des 3 instances particulières

Ensuite sur le temps total de la résolution (Preprocessing + MIP), voici (tableau3.6) la déviation des 2 cas avec seuil par rapport au cas sans seuil. Seulement les instances dont les solutions optimales sont trouvées par toutes les trois méthodes (NoSeuil, Seuil1, Seuil2) sont considérées.

	Seuil1			Seuil2		
Sc	DevMin	DevAvg	DevMax	DevMin	DevAvg	DevMax
1	-40,00%	-6,93%	12,50%	-40,00%	-6,09%	0,00 %
2	-8,64 %	2,41 %	15,52%	-4,69 %	3,12 %	19,75%
3	-14,03%	-2,16%	12,84%	-13,59%	1,46 %	36,63%
4	-42,53%	-6,54 %	39,40%	-42,75 %	-4,16 %	23,02%
5	-56,47%	-13,39 %	22,21%	-34,76 %	-12,49 %	23,78%
6	-50,45%	-22,79 %	51,32%	-66,12 %	-23,16 %	13,95%

TABLE 3.6 – Déviation de temps avec les 2 fonctions de seuil par rapport au cas sans seuil

Nous pouvons trouver qu'avec l'ajout du seuil, par rapport au cas sans seuil, nous pouvons gagner beaucoup de temps en moyenne. Le Seuil1 est meilleur que le seuil2 pour les scenarios 4 et 5 mais pas 6. Donc aucun seuil peut dominer l'autre.

De plus, nous voulons maintenant faire un autre tableau3.7 de déviation mais cette fois par rapport au MIP naïf (cf 2.1) sans coups 2 ajouté.

	Seuil1			Seuil2		
Sc	DevMin	DevAvg	DevMax	DevMin	DevAvg	DevMax
1	-50,00%	110,65%	350,00%	-50,00%	108,80%	300,00%
2	-9,09 %	68,03 %	160,00%	-9,09 %	69,02 %	150,00%
3	-13,43%	64,99 %	156,41%	-12,40%	69,33 %	143,59%
4	-59,14%	14,56 %	75,40 %	-71,09%	20,22 %	91,65 %
5	-43,79%	5,06 %	74,50 %	-33,12%	9,00 %	88,87 %
6	-62,88%	-5,71 %	79,70 %	-57,16%	-8,30 %	35,32 %

TABLE 3.7 – Déviation de temps avec les 2 fonctions de seuil par rapport au MIP sans Preprocessing

A partir du tableau3.7 nous pouvons observer l'amélioration que nous avons apportée jusqu'à présent, c'est-à-dire le Preprocessing plus la coupe 2 avec seuil. Le temps que nous avons gagné est considérable.

Les trois tableaux ci-après3.8 sont toujours pour comparer les trois tests (NoSeuil, Seuil1 et Seuil2), mais d'une autre façon. Dans ces tableau la déviation est calculée par

$$Dev = (Sol - Min(Sol_{NoSeuil}, Sol_{Seuil1}, Sol_{Seuil2}))/Min(Sol_{NoSeuil}, Sol_{Seuil1}, Sol_{Seuil2})$$

C'est-à-dire la déviation entre la solution donnée et la meilleure solution des trois cas. Cette fois toutes les instances testées sont considérées.

NoSeuil												
Sc	n	m	#Fea	#Opt	#Tim	#Mem	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
1	8	2	18	18	0	0	0,01	0,07	0,11	0,00%	0,00%	0,00%
2	11	3	11	11	0	0	0,20	0,40	0,81	0,00%	0,00%	0,00%
3	15	4	19	19	0	0	0,75	2,97	27,94	0,00%	0,00%	0,00%
4	18	5	20	20	0	0	1,40	112,68	1100,71	0,00%	0,00%	0,00%
5	21	5	17	10	6	1	60,48	1084,11	1811,32	0,00%	0,56%	4,02%
6	24	6	18	6	6	6	162,41	1149,89	1813,04	0,00%	0,50%	1,71%

Seuil1												
Sc	n	m	#Fea	#Opt	#Tim	#Mem	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
1	8	2	18	18	0	0	0,01	0,06	0,09	0,00%	0,00%	0,00%
2	11	3	11	11	0	0	0,20	0,40	0,74	0,00%	0,00%	0,00%
3	15	4	19	19	0	0	0,73	2,83	26,05	0,00%	0,00%	0,00%
4	18	5	20	20	0	0	1,42	106,37	1102,95	0,00%	0,00%	0,00%
5	21	5	17	10	7	0	40,31	1032,84	1803,47	0,00%	0,21%	1,23%
6	24	6	18	6	8	4	155,72	1138,01	1813,67	0,00%	0,45%	3,39%

Seuil2												
Sc	n	m	#Fea	#Opt	#Tim	#Mem	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
1	8	2	18	18	0	0	0,01	0,06	0,09	0,00%	0,00%	0,00%
2	11	3	11	11	0	0	0,20	0,41	0,97	0,00%	0,00%	0,00%
3	15	4	19	19	0	0	0,75	2,92	26,36	0,00%	0,00%	0,00%
4	18	5	20	20	0	0	1,39	106,86	1066,17	0,00%	0,00%	0,00%
5	21	5	17	9	6	2	43,63	1041,35	1808,87	0,00%	0,67%	3,98%
6	24	6	18	6	8	4	92,53	1188,52	1816,31	0,00%	1,01%	5,39%

TABLE 3.8 – NoSeuil vs Seuil1 vs Seuil2

A partir de la colonne D_{avg} nous pouvons constater que si nous prenons compte de toutes les instances alors le seuil 1 peut donner en général une solution qui est la meilleure entre les trois cas.

Test avec AddMIPStart* (pas mieux)

En plus de la mise en place des seuils, nous avons aussi testé le fonctionnement de AddMIPStart [2]. Avec AddMIPStart, nous pouvons fournir une solution comme le point de départ de la résolution MIP. Ça peut donc peut-être gagner du temps pour Cplex. Mais le souci est que l'application de cette fonction va peut-être changer la startégie de branchement du Cplex donc nous ne sommes pas sûr que ça marche toujours.

Le tableau3.9 montre la déviation du temps et du nombre de nodes quand on fait AddMIPStart sur une fonction de seuil. $Deviation = (Val_{Seuil+AddMIPStart} - Val_{Seuil})/Val_{Seuil}$.

Dans les tableaux3.10 et 3.11, $Deviation = (Sol - V)/V$ dont $V = Min(Sol_{Seuil+AddMIPStart}, Sol_{Seuil})$.

2. http://pic.dhe.ibm.com?topic=%2Filog.odms.cplex.help%2FContent%2FOptimization%2FDocumentation%2FOptimization_Studio%2F_pubskel%2Fps_usrmancplex1850.html

Déviation du Seuil1+AddMIPStart par rapport au Seuil1						
	#Nodes			Duration		
sc	min	avg	max	min	avg	max
1	0,00%	0,00%	0,00%	-40,00%	22,73%	111,11%
2	0,00%	0,00%	0,00%	-13,43%	5,25%	15,00%
3	-100,00%	-35,98%	8,97%	-23,63%	1,85%	25,79%
4	-100,00%	-7,79%	120,89%	-45,79%	1,75%	46,62%
5	-33,62 %	27,18 %	135,76 %	-37,11 %	14,80 %	123,79%
6	-50,28 %	27,58 %	76,42 %	-40,81 %	12,48 %	57,68 %

Déviation du Seuil2+AddMIPStart par rapport au Seuil2						
	#Nodes			Duration		
sc	min	avg	max	min	avg	max
1	0,00%	0,00%	0,00%	-40,00%	137,35%	1120,00%
2	-42,20%	-9,99%	22,22%	-17,74%	2,12%	15,00%
3	-100,00%	-19,27%	45,79%	-30,12%	-0,22%	28,30%
4	-68,49 %	38,52 %	294,06 %	-52,93 %	11,12 %	94,88%
5	-22,45 %	21,39 %	159,75 %	-40,80 %	3,53 %	47,60%
6	-40,97 %	5,63 %	58,17 %	-34,07 %	-6,12 %	24,97%

TABLE 3.9 – Déviation suivant l'ajout de AddMIPStart sur les fonctions de Seuil

Comme ce que dont nous nous sommes inquiétés, AddMIPStart a bien changer la stratégie de branchement du Cplex. Le nombre des instances qui ont atteint la limite du temps et de la mémoire deviennent différent que sans AddMIPStart. Sur D_{avg} il n'a pas toujours aidé non plus.

Cette fois avec AddMIPStart sur Seuil2, nous avons une meilleure déviation moyenne des solutions mais le nombre des instances qui sont résolues optimalement a décrémenté.

En conclusion, pour bien optimiser la résolution exacte de ce problème, il vaut mieux de faire d'abord le Preprocessing avec la coupe 2 et le seuil 2, puis donner une solution de départ avec le lancement de MIP. La solution de départ corresponds à la UB du Preprocessing.

Tableaux supplémentaires

Les tableaux suivants concernent des comparaisons entre MIP seul, Seuil1 et Seuil2+AddMIPStart.

Tableau 3.12 contient des statistiques sur les résultats de résolution pour les 3 méthodes. Il s'agit des nombres d'instances dans chaque état de résolution : optimal, limite de temps, limite de mémoire.

Tableau 3.13 compare les temps d'exécution pour les 3 méthodes. L'analyse est faite sur les instances pour lesquelles toutes les 3 méthodes ont trvoué la solution optimale ou ont prouvé qu'elles sont infaisables.

Tableau 3.14 peut montrer la qualité des solutions trouvées par les 3 méthodes. Dans le tableau, $Dev = (Sol - V)/V$ dont $V = Min(Sol_{MIP}, Sol_{Seuil1}, Sol_{Seuil2+AddMIPStart})$. Cet analyse est faite sur les instances pour lesquelles :

- Aucunne méthode a atteint la limite de mémoire
- Toutes les méthodes ont trouvé une solution faisable

Tableau 3.15 compare le pourcentage des variables fixées pour Seuil1 et Seuil2+AddMIPStart. Il n'y a pas beaucoup de différence puisque comme nous avons expliqué, il n'y a que 3 instances pour lesquelles le nombre des variables fixées est différent pour ces 2 méthodes.

Seuil1												
Sc	n	m	#Fea	#Opt	#Tim	#Mem	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
1	8	2	18	18	0	0	0,01	0,06	0,09	0,00%	0,00%	0,00%
2	11	3	11	11	0	0	0,20	0,40	0,74	0,00%	0,00%	0,00%
3	15	4	19	19	0	0	0,73	2,83	26,05	0,00%	0,00%	0,00%
4	18	5	20	20	0	0	1,42	106,37	1102,95	0,00 %	0,00%	0,00%
5	21	5	17	10	7	0	40,31	1032,84	1803,47	0,00 %	0,29%	3,15%
6	24	6	18	6	8	4	155,72	1138,01	1813,67	0,00 %	0,43%	2,81%

Seuil1+AddMIPStart												
Sc	n	m	#Fea	#Opt	#Tim	#Mem	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
1	8	2	18	18	0	0	0,01	0,08	0,19	0,00%	0,00%	0,00%
2	11	3	11	11	0	0	0,22	0,41	0,83	0,00%	0,00%	0,00%
3	15	4	19	19	0	0	0,83	2,87	26,74	0,00%	0,00%	0,00%
4	18	5	20	20	0	0	1,59	104,40	852,77	0,00%	0,00%	0,00%
5	21	5	17	9	5	3	25,57	989,14	1803,49	0,00%	0,44%	5,61%
6	24	6	18	6	6	6	101,76	1180,56	1809,59	0,00%	0,25%	1,31%

TABLE 3.10 – Seuil1 vs Seuil1+AddMIPStart

Seuil2												
Sc	n	m	#Fea	#Opt	#Tim	#Mem	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
1	8	2	18	18	0	0	0,01	0,06	0,09	0,00%	0,00%	0,00%
2	11	3	11	11	0	0	0,20	0,41	0,97	0,00%	0,00%	0,00%
3	15	4	19	19	0	0	0,75	2,92	26,36	0,00%	0,00%	0,00%
4	18	5	20	20	0	0	1,39	106,86	1066,17	0,00%	0,00%	0,00%
5	21	5	17	9	6	2	43,63	1041,35	1808,87	0,00%	0,76%	8,62%
6	24	6	18	6	8	4	92,53	1188,52	1816,31	0,00%	1,01%	9,03%

Seuil2+AddMIPStart												
Sc	n	m	#Fea	#Opt	#Tim	#Mem	T_{min}	T_{avg}	T_{max}	D_{min}	D_{avg}	D_{max}
1	8	2	18	18	0	0	0,01	0,15	0,66	0,00%	0,00%	0,00%
2	11	3	11	11	0	0	0,22	0,40	0,81	0,00%	0,00%	0,00%
3	15	4	19	19	0	0	0,81	2,80	24,85	0,00%	0,00%	0,00%
4	18	5	20	20	0	0	1,59	105,66	996,1	0,00%	0,00 %	0,00%
5	21	5	17	8	6	3	25,83	969,67	1809,21	0,00%	0,42 %	2,92%
6	24	6	18	6	6	6	89,18	1106,45	1814,34	0,00%	0,26 %	1,16%

TABLE 3.11 – Seuil2 vs Seuil2+AddMIPStart

				MIP			Seuil1			Seuil2+AddMIPStart		
Sc	n	m	Infea	Opt	Tim	Mem	Opt	Tim	Mem	Opt	Tim	Mem
1	8	2	2	18	0	0	18	0	0	18	0	0
2	11	3	9	11	0	0	11	0	0	11	0	0
3	15	4	1	19	0	0	19	0	0	19	0	0
4	18	5	0	19	1	0	20	0	0	20	0	0
5	21	5	3	6	10	1	10	7	0	8	6	3
6	24	6	2	5	13	0	6	8	4	6	6	6

TABLE 3.12 – Statsistiques sur les résultats de résolution

	MIP			Seuil1			Seuil2+AddMIPStart		
Sc	T_{min}	T_{avg}	T_{max}	T_{min}	T_{avg}	T_{max}	T_{min}	T_{avg}	T_{max}
1	0,01	0,03	0,04	0,01	0,07	0,20	0,01	0,15	0,66
2	0,04	0,17	0,59	0,03	0,24	0,74	0,03	0,24	0,81
3	0,31	2,45	30,09	0,22	2,70	26,05	0,22	2,68	24,85
4	1,12	99,69	783,99	1,42	106,37	1102,95	1,59	105,66	996,10
5	0,95	248,50	769,13	0,67	222,59	919,01	0,66	265,22	1022,32
6	1,19	365,32	1788,43	2,26	205,93	663,83	2,26	200,79	658,59

TABLE 3.13 – Temps de résolution pour les instances résolues

	MIP			Seuil1			Seuil2+AddMIPStart		
Sc	Dev_{min}	Dev_{avg}	Dev_{max}	Dev_{min}	Dev_{avg}	Dev_{max}	Dev_{min}	Dev_{avg}	Dev_{max}
1	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
2	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
3	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
4	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
5	0,00%	0,49%	5,38%	0,00%	0,42%	5,44%	0,00%	0,26%	1,68%
6	0,00%	0,07%	0,36%	0,00%	0,21%	1,87%	0,00%	0,43%	3,91%

TABLE 3.14 – Déviation de solutions

	Seuil1			Seuil2+AddMIPStart		
Sc	Fix_{min}	Fix_{avg}	Fix_{max}	Fix_{min}	Fix_{avg}	Fix_{max}
1	0,00%	43,92%	100,00%	0,00%	43,92%	100,00%
2	0,00%	15,80%	66,67%	0,00%	15,80%	66,67%
3	0,00%	**2,21%**	**15,77%**	0,00%	**2,22%**	**16,03%**
4	0,00%	**2,86%**	55,64%	0,00%	**2,91%**	55,64%
5	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
6	0,00%	0,09%	0,56%	0,00%	0,09%	0,56%

TABLE 3.15 – Pourcentage des variables fixées

Rapport des tests du problème SCP (hors-ligne)

Département Informatique
5e année
2013 - 2014

Rapport des Tests

Résumé :

Mots clefs :

Abstract:

Keywords:

Encadrants
Vincent T'Kindt
vincent.tkindt@univ-tours.fr

Étudiants
Lei SHANG
lei.shang@etu.univ-tours.fr

Université François-Rabelais, Tours

DI5 2013 - 2014