Continuous optimization

Samuel Buchet & Dorian Dumez & Brendan Guevel

Mai 2017

1 Introduction

Dans ce projet, la méthode de la relaxation lagrangienne est utilisée afin d'obtenir des bornes pour le problème de bin packing. Dans un premier temps, une formulation du problème en programme linéaire en nombres entiers est proposée. Ensuite, une heuristique de construction de solution est détaillée, suivi du détail du calcul de la relaxation linéaire. Après cela, plusieurs relaxations lagrangiennes sont proposées et les tests sont interprétés. Enfin, des indications sont données sur le programme produit.

2 Problème étudié

Le problème de bin packing consiste à insérer des objets de tailles différentes dans des boîtes. Toutes les boîtes ont une même capacité et le but est de minimiser le nombre de boîte nécessaire pour ranger tous les objets. Dans cette version étudiée, le problème ne considère qu'une seule dimension.

On peut écrire le problème de bin-packing monodimensionnel comme :

$$
\begin{aligned}
\text{(PE)} : z = \min & \sum_{b=1}^{m} u_b \\
\text{s.c.} & \sum_{p=1}^{n} s_p x_{pb} \leqslant u_b c \quad \forall b \in \{1,..,m\} && (1) \\
& \sum_{b=1}^{m} x_{pb} = 1 \quad \forall p \in \{1,..,n\} && (2) \\
& u_b, x_{pb} \in \{0,1\} \quad \forall p \in \{1,..,n\}, \forall b \in \{1,..,m\}
\end{aligned}
$$

Avec pour paramètres :

- n le nombre d'objet
- m le nombre de bin maximal, peut être obtenu à l'aide d'une heuristique ou par défaut n (mais sera coûteux)
- c la capacité d'une boîte
- s le vecteur de la taille des objets

Et pour variables :

$$u_b = \begin{cases} 1 \text{ si on utilise le bin numéro b} \\ 0 \text{ sinon} \end{cases}$$

$$x_{pb} = \begin{cases} 1 \text{ si on met l'objet } p \text{ dans le bin } b \\ 0 \text{ sinon} \end{cases}$$

Bien que juste ce programme linéaire n'est pas très utilisable en pratique, en effet :

- il ne prend pas en compte la redondance des objets, si on a 100 fois le même objet il créera 100 variables
- bien qu'identique de part leur caractéristique tous les bin sont différencié

Il en résultera un nombre considérable de solutions équivalentes due à des symétries. Un algorithme de résolution devrait donc prendre en compte l'interprétation du problème ou y ajouter des contraintes comme :

$$u_b \geqslant u_{b+1} \quad \forall b \in \{1, .., m-1\} \tag{3}$$

$$argmin_{p \in \{i \in \{1,..,n\} | x_{ib}=1\}} p \leqslant argmin_{p \in \{i \in \{1,..,n\} | x_{i,b+1}=1\}} p \quad \forall b \in \{1, .., m-1\} \tag{4}$$

Où la contrainte (3) force les bins ouverts à être ceux avec les plus petits identifiants et la contrainte (4) ordonne les bins par le schéma d'objet qu'il contient. Mais même si cela ne règle pas le problème des variables redondantes, et rend ce défaut nécessaire par leur utilisation.

Si on veut les écrire avec des contraintes linéaires, on peut écrire la contrainte (4) avec :

$$x_{pb} \leqslant \sum_{pp=1}^{p-} x_{pp,b-1} \quad \forall p \in \{2, .., n\} : \forall b \in \{2, .., m\} \tag{4'}$$

$$x(1,1) = 1 \tag{4"}$$

De plus on peut remarquer que sous ces contraintes on a toujours :

$$x_{pb} = 0 \quad \forall p \in \{1, .., n\} \quad \forall b \in \{p+1, .., m\} \tag{5}$$

3 Relaxation linéaire

La relaxation linéaire de (PE) s'écrit :

$$\begin{aligned} (\text{Rl}) : z_{\text{Rl}} = & \min \sum_{b=1}^{n} u_b \\ & \text{s.c} \sum_{p=1}^{n} s_p x_{pb} \leqslant u_b c \quad \forall b \in \{1, .., m\} && (1) \\ & \sum_{b=m}^{n} x_{pb} = 1 \quad \forall p \in \{1, .., n\} && (2) \\ & u_b, x_{pb} \in [0,1] \quad \forall p \in \{1, .., n\} \quad \forall b \in \{1, .., m\} \end{aligned}$$

De plus on peut montrer que la valeur de la relaxation linéaire est $\frac{\sum_{p=1}^{n} s_p}{c}$. La valeur optimale de (PE) étant toujours entière, peut ainsi utiliser comme borne inférieure $\left\lceil \frac{\sum_{p=1}^{n} s_p}{c} \right\rceil$

Proof. Soit u_b^* et x_{bp}^* la valeur des variables dans la solution optimale de la relaxation linéaire. Alors on a $\forall p \in \{1, .., n\} : \sum_{b=1}^{m} x_{pb}^* = 1$ par la contrainte (2). De plus, étant donné que chaque u_b n'apparait que dans une contrainte, on peut dire que $u_b^* = \frac{\sum_{p=1}^{n} s_p x_{pb}^*}{c}$ par la contrainte (1). En effet ce problème est un problème de minimisation donc cette contrainte est limitante. Par propriété de l'algorithme du simplexe elle sera vérifiée à l'égalité dans la solution optimale (car elles sont toutes indépendantes). La valeur cette solution est alors $\sum_{b=1}^{m} \frac{\sum_{p=1}^{n} s_p x_{pb}^*}{c} = \frac{\sum_{p=1}^{n} s_p}{c}$ par la contrainte (2).

□

4 Heuristique de construction

Afin d'obtenir rapidement une borne supérieurs intéressante pour ce problème, une heuristique de construction peut être utilisée. Dans ce projet, l'heuristique best fit a été implémentée. L'aglorithme de best fit consiste à ajouter les objets du plus grand au plus petit. De plus, l'ajout est d'abord tenté dans les bins les plus remplis en premier. Si l'objet ne peut être ajouté dans aucun bin, un nouveau bin est alors ouvert.

5 Relaxations lagrangiennes possibles

Plusieurs choix de contraintes étaient possibles pour la relaxation lagrangienne. Les contraintes dualisées ont été choisies afin d'obtenir des sous-problèmes simples à résoudre et décomposable. Dans cette partie, deux relaxations qui nous semblaient intéressantes sont détaillées.

5.1 Dualisation de la contrainte 1

Premièrement on remarque que l'on peut séparer en deux la contrainte (1) et donc écrire (PE) comme :

$$\text{(PE)} : z = \min \sum_{b=1}^{m} u_b$$

s.c.

$$\forall b \in [\![1, m]\!] : \sum_{p=1}^{n} s_p x_{pb} \leqslant c \ (1')$$

$$\forall p \in [\![1, n]\!] : \forall b \in [\![1, m]\!] : x_{pb} \leqslant u_b \ (1'')$$

$$\forall p \in [\![1, n]\!] : \sum_{b=1}^{m} x_{pb} = 1 \ (2)$$

$$\forall p \in [\![1, n]\!] : \forall b \in [\![1, m]\!] : u_b, x_{pb} \in \{0, 1\}$$

Alors si on dualise la contrainte $(1')$ on obtient :

$$RL_{1'}(u) : z_{1'}(u) = \min \sum_{b=1}^{m} u_b - \mu_b(c - \sum_{p=1}^{n} s_p x_{pb})$$

s.c.

$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,m]\!] : x_{pb} \leqslant u_b \ (1")$$
$$\forall p \in [\![1,n]\!] : \sum_{b=1}^{m} x_{pb} = 1 \ (2)$$
$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,m]\!] : u_b, x_{pb} \in \{0,1\}$$

Où $\mu \geqslant 0$ car la contrainte $(1')$ est une contrainte en inégalité.

En écrivant la fonction objectif comme $z_{1'}(u) = \min \sum_{b=1}^{m}(u_b + \sum_{p=1}^{n}(\mu_b s_p x_{pb})) - \sum_{b=1}^{m} \mu_b c$. On peut alors voir $RL_{1'}(u)$ comme un problème d'UFLP (uncapacited facility location problem) avec des coûts d'ouverture de 1 et des coûts d'associations de $\mu_b s_p$. A noter que l'on ajoute un terme constant à la fonction objectif pour obtenir la vraie valeur de la relaxation lagrangienne.

Alors comme on l'a fait en cours on peut dualiser la contrainte (2) pour obtenir :

$RL_{1',2}(u) : z_{1',2}(u) = \min \sum_{b=1}^{m}(u_b + \sum_{p=1}^{n}(\mu_b s_p x_{pb})) - \sum_{b=1}^{m} \mu_b c + \sum_{p=1}^{n} \gamma_p (1 - \sum_{b=1}^{m} x_{pb})$

s.c.

$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,m]\!] : x_{pb} \leqslant u_b \ (1")$$
$$\forall p \in [\![1,n]\!] : \forall b \in [\![1,m]\!] : u_b, x_{pb} \in \{0,1\}$$

Où γ est de signe libre car la contrainte associé est une contrainte d'égalité.
Comme précédemment on peut reformuler la fonction objectif comme :

$z_{1',2}(u) = \min \sum_{b=1}^{m}(u_b + \sum_{p=1}^{n}((\mu_b s_p - \gamma_p)x_{pb})) - \sum_{b=1}^{m} \mu_b c + \sum_{p=1}^{n} \gamma_p$

En pratique on remarque qu'il suffit de résoudre m problème indépendants :

$PI_b(u) : z'_b(u) = \min u_b + \sum_{p=1}^{n}((\mu_b s_p - \gamma_p)x_{pb})$

s.c.

$$\forall p \in [\![1,n]\!] : x_{pb} \leqslant u_b \ (1")$$
$$\forall p \in [\![1,n]\!] u_b, x_{pb} \in \{0,1\}$$

Alors on sait comment calculer la solution optimale de chacun d'entre eux :

1 - $u_b = 1 \Leftrightarrow \sum_{p \in [\![1,n]\!] : \mu_b s_p - \gamma_p < 0} (\mu_b s_p - \gamma_p) < -1$

2 - $\forall p \in [\![1,n]\!] : (\mu_b s_p - \gamma_p < 0) \Rightarrow x_{pb} = u_b$

3 - $\forall p \in [\![1,n]\!] : (\mu_b s_p - \gamma_p \geqslant 0) \Rightarrow x_{pb} = 0$

De plus on peut remarquer que la contrainte anti-symétrie (5) ne nuit pas au lagrangien : les sous-problèmes reste indépendants et la relaxation conserve sa liberté. Cela permet d'améliorer la vitesse de convergence de l'algorithme de sous-gradient ainsi que la vitesse de résolution d'une partie des sous-problèmes.

5.2 Dualisation de la contrainte 2

Une autre possibilité est de dualiser la contrainte (2) : $\sum_{b=1}^{m} x_{pb} = 1$.

On obtient alors :

$RL_2(u) : z_2(u) = \min \sum_{b=1}^{m} u_b + \sum_{p=1}^{n} \mu_p (1 - \sum_{b=1}^{m} x_{pb})$

s.c.

$$\forall b \in [\![1, m]\!] : \sum_{p=1}^{n} s_p x_{pb} \leqslant u_b c \ (1)$$

$$\forall p \in [\![1, n]\!] : \forall b \in [\![1, m]\!] : x_{pb}, u_b \in \{0, 1\}$$

Etant donné que la contrainte (2) est une contrainte d'égalité le signe des μ_i est libre.

On peut réécrire la fonction objectif comme $z_2(u) = \min \sum_{b=1}^{m} \left(u_b - \sum_{p=1}^{n} \mu_p x_{pb} \right) + \sum_{p=1}^{n} \mu_p$.

Le problème est alors décomposable par bin. En effet, le terme de droite est constant et la somme de gauche peut être décomposée par bin. Les variables d'un terme de la somme ne dépendent que d'un seul bin et il y a aussi une unique contrainte par bin. On se retrouve donc avec les m sous problèmes identiques :

$$min \quad z_{2b}(0) = u_b - \sum_{p=1}^{n} \mu_p * x_{pb}$$

$$s.c : \sum_{p=1}^{n} s_p * x_{pb} \leq u_b * c$$

Ce problème peut être décomposé en deux sous problèmes : le problème dans lequel $u_b = 0$ et celui dans lequel $u_b = 1$.

Dans le cas où $u_b = 0$, la contrainte force les variables x_{pb} à être nulles (les s_p étant tous positifs). la valeur optimale de la fonction objectif est alors 0.

Dans le cas où $u_b = 1$ la contrainte (1) devient une contrainte de capacité. Le problème peut alors être vu comme un problème de de sac à dos en inversant le signe de la fonction. Les objets de ce problème ont pour poids s_p et pour valeurs μ_p.

Enfin, la valeur optimal de ces sous problème peut être trouvée en comparant les résultat obtenus avec le problème de sac à dos et le problème avec $u_b = 0$. La valeur optimale du premier sous problème étant 0, le deuxième donne une meilleure valeur si $u_b - \sum_{p=1}^{n} \mu_p * x_{pb} < 0$ ou encore $1 < \sum_{p=1}^{n} \mu_p * x_{pb}$. On peut donc résoudre le problème en résolvant le sous-problème de sac à dos et en comparant le résultat avec 1.

On peut ainsi résoudre $RL_2(u)$ en résolvant 1 problème de sac à dos, en multipliant le résultat par m et en ajoutant la somme des μ.

6 Algorithme de sous-gradient

Afin de trouver les multiplicateurs des contraintes dualisées de la relaxation qui donnent la meilleure relaxation possible, nous avons implémenté un algorithme de sous-gradient avec une longueur de pas dépendant de la satisfaction des contraintes.

- L'initialisation comprend la mise à 0 des variables de calcul ainsi que la fixation des paramètres, le calcul de borne et l'affectation des valeurs initiale des coefficients lagrangiens
- Toutes les résolutions se font selon des procédures spécifiques à la relaxation choisie, détaillé dans les paragraphes correspondant, de même pour le calcul de la valeur de la fonction objectif
- Le calcul du pas se fait en 2 partie :

Algorithm 1 algorithme de sous-gradient

```
1: function RelaxLagrange(instance)
2:     initialisation
3:     première résolution des sous-problèmes
4:     calcul du score et de la première valeur du pas
5:     while ¬ Critères d'arrêts do
6:         calcul des nouveaux multiplicateur
7:         résolution des sous-problèmes
8:         mise à jours de la meilleure solution
9:         calcul de la taille du pas
10:    end while
11:    return best
12: end function
```

1 - Le calcul du pois de ce pas : $\nu = \epsilon.\frac{\omega - val}{||d - Dx(\mu)||^2}$ où ω est un majorant de la valeur de la relaxation lagrangienne (la valeur de best-fit est utilisée en pratique), val est la valeur courante de la fonction objectif dualisé, et $Dx\{\leqslant, =, \geqslant\}d$ sont les contraintes qui ont été dualisé ($x(\mu)$ est la valeur optimale de x dans les sous-problème avec les coef μ).

2 - Ensuite le pas est $\mu = max(\mu + \nu(d - Dx(\mu)), 0)$

ϵ est un facteur qui va déterminer la taille du pas, au départ il est fixé à une valeur intermédiaire puis vas être diminué d'un facteur ρ si aucune amélioration n'a été effectué en t_{max} itérations, avec une ré-augmentation si sa valeur devient vraiment trop faible. De plus, dans le cadre de la première relaxation, on utilise deux ν et ϵ, un pour chaque type de contraintes. En effet la nature des contraintes dualisées étant profondément différente les même valeurs ne leurs conviennent pas. Enfin il faut noter que le signe du multiplicateur est libre (γ de la première est μ de la seconde) il n'y a pas de max.

- On dispose de plusieurs critère d'arrêt, l'activation de l'un d'entre eux suffit :
  - $\nu < \nu_{min} \wedge val > \bar{\omega} - 1$ où ν_{min} est un seuil sur ν et $\bar{\omega}$ la valeur de la relaxation linéaire. Cette formule signifie que l'algorithme a convergé vers une valeur au moins aussi bonne que la relaxation linéaire (la valeur de la relaxation lagrangienne est sensé être au moins aussi bonne).
  - $\omega - val < 1$ qui signifie que la solution construite est optimale
  - $\nu = 0$ qui signifie que les contraintes dualisé sont toutes vérifié
  - no_improve $\geqslant$ max_no_improve qui limite le temps de calcul maximal

7 Réparation

Une fois la relaxation langragienne effectuée, le sous-problème du sac à dos nous donne une solution non admissible qui contient plusieurs fois le même bin (s'il a été ouvert lors de la relaxation). On dispose également des coefficients lagrangiens qui nous donnent une indication de la difficulté à placer les différents objets.

Pour reconstruire la solution, on regarde d'abord si la solution de la relaxation a ouvert au moins un bin ou pas (si c'est le cas ce seront tous les mêmes), et on prend ce premier bin dans notre solution réparée. On essaie ensuite de "cloner" ce bin en prenant des objets de même taille taille, si c'est possible. Une fois que l'on a fait cette première étape, on prends les objets restants à placer un par un dans l'ordre décroissant de leur coefficient de lagrange : si on peut le placer dans un bin déjà existant on le fait, sinon on ouvre un nouveau bin et on le met dedans.

Algorithm 2 algorithme de réparation

```
1: function RÉPARATION(instance, bin, ouvert, coef)
2:     if ouvert then
3:         ouvrir un nouveau bin qui contient les objets de "bin"
4:         si possible, cloner ce bin avec d'autres objets de même taille
5:     end if
6:     trier les objets dans l'ordre décroissant des coefficients de lagrange
7:     while il reste des objets à mettre do
8:         prendre un objet dans l'ordre décroissant trié précédemment
9:         placer l'objet dans un bin existant si possible
10:        sinon créer un nouveau bin et le mettre dedans
11:    end while
12:    return solutionRéparée
13: end function
```

8 Tests

Les tests ont été réalisé sur chacunes des instances disponibles. C'est la deuxième relaxation lagrangienne qui a été utilisée pour fournir la borne car la première ne donnait pas de bons résultats (l'algorithme des sous-gradients ne parvenait pas à converger). Etant donné que l'algorithme des sous-gradients peut avoir des difficultés pour converger, une limite de temps de 15 minutes a été utilisée pour la relaxation lagrangienne. Les résultats obtenus sont présents en annexe.

On remarque que pour un bon nombre d'instances, le temps pour effectuer la relaxation lagrangienne est supérieur à la limite autorisée. Pour certaines instances, le temps est dépassé de manière importante, ce qui indique que le sous problème à résoudre (sac à dos) peut être assez difficile à résoudre. Dans le cas où la limite de temps est dépassée, la relaxation lagrangienne reste proche et est parfois meilleure que la relaxation linéaire. La recontruction donne une borne proche de celle de best fit mais est rarement meilleures (y compris pour les instances où lalgorithme des sous gradients converge).

On remarque également que pour les instances de taille 120 dont les boîtes ont pour taille 150, l'algorithme converge. Cependant, cele n'est pas le cas pour les instances avec comme taille de boîte 1000. Il est donc probable que ces dernières instances soient plus difficiles. Cela peut s'expliquer avec le plus grand nombre de combinaisons d'objets possible dans les bins grâce à la plus grande capacité (voir formulation du problème pour la génération de colonne).

Dans les cas où l'algorithme des sous gradients converge, la relaxation lagrangienne est généralement meilleure que la relaxation linéaire. On peut également voir que dans le cas des instances dont la taille des boîtes est 1000, la relaxation lagrangienne et la reconstruction améliorent moins, ce qui conforte l'idée que ces instances sont plus difficiles.

9 Détails sur le programme

Tous les algorithmes présentés dans ce rapport ont été implémenté en python. Pour exécuter le programme, il faut se placer dans le dossier *code* et utiliser la commande : *python3 bin_packing.py nom_de_l'instance*. Le programme fourni alors les différentes bornes: la borne de la relaxation linéaire, la borne de best fit, la borne de la deuxième relaxation lagrangienne et la borne de l'algorithme de reconstruction basé sur la deuxième relaxation lagrangienne.

Pour les besoin de la deuxième relaxation lagrangienne, un petit solver de problème sac à dos (branch and

bound) a été implémenté dans le fichier *knapsack.py*. Le fichier *instance.py* contient les fonctions nécessaires pour charger l'instance du problème et pour calculer la valeur de la relaxation linéaire. L'algorithme best fit est implémenté dans le fichier *heuristique.py*. La première relaxation lagrangienne est calculée dans le fichier *lagrange.py* et la deuxième est présente dans *lagrange_kp2.py*. Enfin, le fichier *reparation.py* contient l'algorithme de réparation basé sur la relaxation lagrangienne.

10 Conclusion

Pour conclure, on peut voir que la relaxation lagrangienne donne des résultats intéressants. Lorsque l'algorithme parvient à converger, les bornes obtenues sont généralement meilleures que celles de la relaxation linéaire. Cependant, l'algorithme prend un temps considérable en comparaison à la relaxation linéaire. Utiliser cette méthode dans un algorithme de branch and bound n'est donc pas forcément la meilleure stratégie. Son utilisation pourrait cependant être combinée à d'autres bornes inférieures plus rapides à calculer. La relaxation lagrangienne peut également être utile pour vérifier le résultat d'un algorithme inexacte, type métaheuristique.

La reconstruction n'améliore pas vraiment l'heuristique best fit. Cependant on peut voir que best fit est parfois très proche de la solution optimale, ce qui en fait une heuristique très intéressante. Une autre idée pour la reconstruction qui n'a pas été testée pourrait être reconstruire une solution à partir du début en ajoutant les objets dont les multiplicateurs de la relaxation lagrangienne sont les plus grands (positifs ou négatifs) en premiers car ils correspondent aux objets difficiles à placer.

11 Annexe

Résultats sur les instances avec des boîtes de taille 150 :

instance	best fit	reconstruction	relaxation linéaire	relaxation lagrangienne	temps (s)
Falkenauer_u120_00	49	49	46	48	284
Falkenauer_u120_01	49	49	47	49	39
Falkenauer_u120_02	47	47	44	46	35
Falkenauer_u120_03	50	50	48	49	337
Falkenauer_u120_04	50	50	48	50	77
Falkenauer_u120_05	49	49	47	48	184
Falkenauer_u120_06	49	49	46	48	137
Falkenauer_u120_07	50	50	48	49	311
Falkenauer_u120_08	51	51	48	50	257
Falkenauer_u120_09	47	47	45	46	111
Falkenauer_u250_00	100	100	97	99	240
Falkenauer_u250_01	101	101	98	99	126
Falkenauer_u250_02	104	104	100	102	767
Falkenauer_u250_03	101	101	96	100	187
Falkenauer_u250_04	102	102	100	101	257
Falkenauer_u250_05	104	104	100	101	418
Falkenauer_u250_06	103	103	99	101	137
Falkenauer_u250_07	105	105	99	103	900
Falkenauer_u250_08	107	107	102	105	900
Falkenauer_u250_09	102	102	97	101	562
Falkenauer_u500_00	201	201	194	192	900
Falkenauer_u500_01	204	204	195	196	900
Falkenauer_u500_02	205	205	199	196	900
Falkenauer_u500_03	207	207	198	199	900
Falkenauer_u500_04	209	208	199	201	900
Falkenauer_u500_05	207	207	199	201	900
Falkenauer_u500_06	210	210	203	199	901
Falkenauer_u500_07	207	207	198	200	900
Falkenauer_u500_08	199	198	192	192	900
Falkenauer_u500_09	204	204	198	194	900

Résultats sur les instances avec des boîtes de taille 1000 :

instance	best fit	reconstruction	relaxation linéaire	relaxation lagrangienne	temps (s)
Falkenauer_t120_00	45	45	40	40	904
Falkenauer_t120_01	45	45	40	40	900
Falkenauer_t120_02	45	45	40	40	903
Falkenauer_t120_03	46	46	40	40	903
Falkenauer_t120_04	46	46	40	40	901
Falkenauer_t120_05	46	46	40	40	905
Falkenauer_t120_06	45	46	40	40	901
Falkenauer_t120_07	46	46	40	40	900
Falkenauer_t120_08	46	46	40	40	904
Falkenauer_t120_09	45	45	40	40	903
Falkenauer_t249_00	94	94	83	81	993
Falkenauer_t249_01	95	95	83	83	1000
Falkenauer_t249_02	94	94	83	83	910
Falkenauer_t249_03	95	95	83	83	914
Falkenauer_t249_04	95	95	83	78	1000
Falkenauer_t249_05	95	95	83	83	935
Falkenauer_t249_06	95	95	83	82	933
Falkenauer_t249_07	96	96	83	83	901
Falkenauer_t249_08	95	95	83	83	998
Falkenauer_t249_09	95	95	83	83	934
Falkenauer_t60_00	23	23	20	20	178
Falkenauer_t60_01	23	23	20	20	177
Falkenauer_t60_02	23	23	20	20	200
Falkenauer_t60_03	23	23	20	20	181
Falkenauer_t60_04	24	24	20	20	176
Falkenauer_t60_05	23	23	20	20	179
Falkenauer_t60_06	23	23	20	20	179
Falkenauer_t60_07	23	23	20	20	180
Falkenauer_t60_08	23	23	20	20	181
Falkenauer_t60_09	23	23	20	20	180