Ordonnancement de tâches avec sources d'énergie multiples

Janik RANNOU
Master 1 MApI3

Tuteurs : Christian ARTIGUES
Sandra U. NGUEVEU

Stage effectué lors de l'année 2016-2017 au LAAS-CNRS : 7 Avenue du Colonel Roche, 31400 Toulouse

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements à l'ensemble de l'équipe ROC et plus particulièrement à mes deux tuteurs Christian et Sandra pour m'avoir accueilli chaleureusement et m'avoir donné l'opportunité de travailler avec eux.

Je remercie également le projet NeoCampus qui a financé mon stage.

Enfin je remercie également l'ensemble de mes professeurs de l'université Paul Sabatier pour la formation suivie cette année.

Table des matières

1 Introduction

Ce stage a été effectué au LAAS dans l'équipe ROC(Recherche Opérationnelle, Optimisation combinatoire et Contraintes) du 27 avril 2017 au 25 aout 2017. Il a été encadré par Christian Artigues et Sandra U. Ngueveu et a eu deux objectifs.

Premièrement, implémenter de nouvelles méthodes pour résoudre des problèmes d'ordonnancement de tâches avec un aspect énergétique dans le cadre du projet PGMO (Programme Gaspard Monge pour l'optimisation et la recherche opérationnelle).

Deuxièmement, mettre en œuvre ces méthodes sur le bâtiment ADREAM du LAAS afin de pouvoir déterminer leurs efficacités ainsi que leurs comportements dans le cadre du projet OPA.

Ce stage a été la continuation du stage de Paulin Couzon effectué en 2016 à la même période qui avait modélisé le problème et avait commencé à développer différentes méthodes pour résoudre le problème.

2 Présentation du problème et du modèle général associé

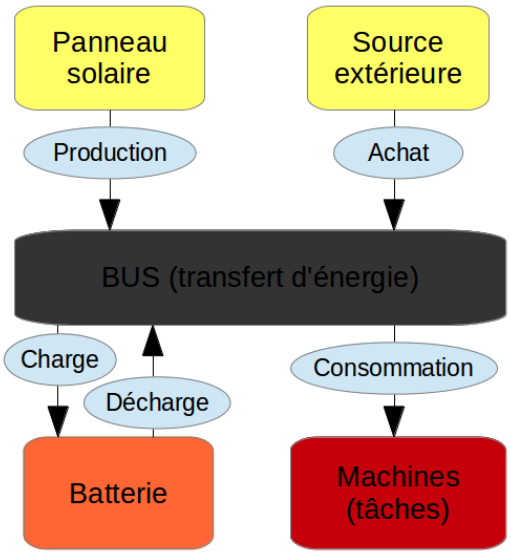
2.1 Présentation du problème

Le problème considéré consiste à affecter des tâches sur des machines de manière à obtenir la quantité d'énergie achetée la plus faible possible. Dans notre problème on va considérer différents types de sources d'énergie ainsi qu'un ensemble de machine qui consommera de l'énergie. On remarquera que l'affection des tâches est soumise à des contraintes liées aux conditions d'utilisation des machines ou des sources d'énergie.

Ici, on considère deux types de sources d'énergie :

- les sources non réversibles qui sont uniquement capables de produire de l'énergie. Dans notre problème, on considérera deux variantes : une gratuite(panneau photovoltaïque) et une payante (source extérieure),
- les sources réversibles qui sont à la fois capables de produire de l'énergie mais aussi de la stocker(batterie).

Voici un exemple de configuration pour le problème :



2.2 Présentation du modèle général associé

Ici, on présentera le modèle où l'on obtient une solution préemptive(avec interruption de tâches possible). Le modèle qui a servi de base est celui issu du stage de Paulin Couzon. C'est aussi ce modèle qui est utilisé pour les tests sur le système du bâtiment ADREAM.

2.2.1 Notations associées au modèle

T est l'ensemble {0,...,T_{max}} qui correspond au temps total du planning créé.
V est l'ensemble des tâches v à exécuter lors de la période T.
P est l'ensemble des machines p sur lesquels vont être effectuées les tâches v

Les données liées au problème sont les suivantes :
- Δt l'unité de temps de T, c'est-à-dire la durée entre deux instants(en heure),
- τ_v : la durée de la tâche v (en unité de temps Δt),
- c_v : nombre de CPU requis pour l'exécution de la tâche v,
- m_v : mémoire requise pour la tâche v (en Mo),
- C_p : nombre maximal de CPU de la machine p,
- M_p : mémoire maximale de la machine p (en Mo),
- $P_{p,min}$: consommation d'énergie de la machine p si aucun cœur n'est utilisé (en Watt),
- $P_{p,max}$: consommation d'énergie de la machine p si tous les cœurs sont utilisés (en Watt),
- q_{min} : charge minimale de la batterie (en Watt-heure),
- q_{max} : charge maximale de la batterie (en Watt-heure),
- q_0 : charge initiale de la batterie, en Watt-heure,
- ρ^{-1} la fonction de rendement de la source non réversible : calcule la quantité d'énergie nécessaire pour une quantité d'énergie demandée (elle est linéaire par morceaux),
- χ la fonction de rendement de la source réversible : calcule la quantité d'énergie nécessaire pour une charge ou une décharge donnée (elle est linéaire par morceaux).

Les variables de décision du modèle sont :
- z_{pt} : variable binaire, elle vaut 1 si la machine p est allumée (ou active) au temps t, définie $\forall p \in P, \forall t \in T$,
- x_{vpt} : variable binaire, elle vaut 1 si la VM v est assignée à la PM p et qu'elle est active en t, définie $\forall v \in V, \forall p \in P, \forall t \in T$,
- c_t : variable positive, indique le cout en énergie de la source EDF, définie $\forall t \in T$.
- r_t : variable réelle, indique de combien d'énergie on charge ou décharge la batterie au temps t (négatif : charge, positif : décharge), définie $\forall t \in T$,
- q_t : variable positive, indique l'état de charge de la batterie, définie $\forall t \in T$ (cette variable est déterminée par la valeur de r_t et par q_0).

2.2.2 Fonction objectif et contraintes

Ici on veut minimiser le cout total en énergie :

$$\min \sum_{t \in T} c_t \tag{1}$$

Sous les contraintes suivantes :

– On fixe les natures des variables de décision :

$$x_{vpt} \in \{0,1\} \ , \forall v \in V, \forall p \in P, \forall t \in T \tag{2}$$

$$z_{pt} \in \{0,1\} \ , \forall p \in P, \forall t \in T \tag{3}$$

$$c_t, d_t \geq 0 \ , \forall t \in T \tag{4}$$

$$q_{min} \geq q_t \geq q_{max} \ , \forall t \in T \tag{5}$$

$$c_t = \rho^{-1}(d_t) \ , \forall t \in T \tag{6}$$

$$r_t \in \mathbb{R} \ , \forall t \in T \tag{7}$$

– On a un bilant énergétique nul à tout instant (i.e. les machines ont assez d'énergie) :

$$d_t + \chi(r_t) - \sum_{v \in V} \sum_{p \in P} \frac{(P_{p,max} - P_{p,min})c_v}{C_p} x_{vpt} - \sum_{p \in P} P_{p,min} z_{pt} = 0 \ , \forall t \in T \tag{8}$$

– Une tâche ne peut tourner que sur une seule machine à la fois :

$$\sum_{p \in P} x_{vpt} \leq 1 \ , \forall v \in V, \forall t \in T \tag{9}$$

– La durée d'exécution de la tache est égale à la durée de la tâche :

$$\sum_{p \in P} \sum_{t \in T} x_{vpt} = \tau_v \ , \forall v \in V \tag{10}$$

– On n'utilise pas plus de cœur que le nombre maximal de chaque machine :

$$\sum_{v \in V} c_v x_{vpt} \leq C_p \ , \forall p \in P, \forall t \in T \tag{11}$$

– On n'utilise pas pas plus de mémoire que la mémoire maximale de chaque machine :

$$\sum_{v \in V} m_v x_{vpt} \leq M_p \ , \forall p \in P, \forall t \in T \tag{12}$$

– On force z_{pt} à valoir 1 si la machine p est allumée au temps t :

$$z_{pt} \geq x_{vpt} \ , \forall v \in V, \forall p \in P, \forall t \in T \tag{13}$$

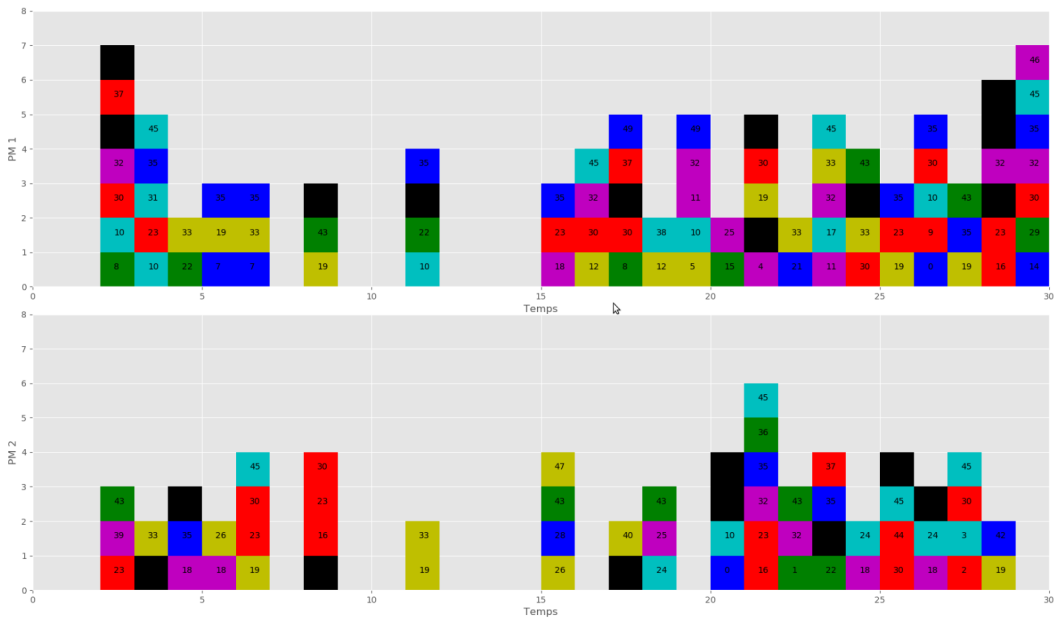
– Le nouveau stock est egal à l'ancien stock moins la variation :

$$q_{t+1} = q_t - r_t \Delta t \ , \forall t \in T \backslash_{\{T_{max}\}} \tag{14}$$

– Le stock final est supérieur ou égal au stock initial (contrainte facultative et pouvant être modifiée, différentes configurations on été utilisées) :

$$q_{T_{max}} \geq q_0 \tag{15}$$

Voici un exemple du type de solution donnée pour cet algorithme avec 2 machines, 50 tâches et un horizon de 30 instants :



2.2.3 Variante non préemptive

Dans le cas où l'on veut une solution non préemptive(pas d'interruption dans les tâches et donc aussi pas de migration d'une machine à l'autre), on modifie le modèle de la façon suivante :

- la variable x_{vpt} change de signification au lieu d'indiquer si la tâche v est active sur la machine p à l'instant t, elle indique désormais si la tâche v a commencé sur la machine p à l'instant t,
- les contraintes (9) (non parallélisme) et (10) (durée d'éxecution) sont remplacées par :
  - Une tâche ne peut être lancé que une seule fois :

$$\sum_{p \in P} \sum_{t \in T} x_{vpt} = 1 \ , \forall v \in V \tag{16}$$

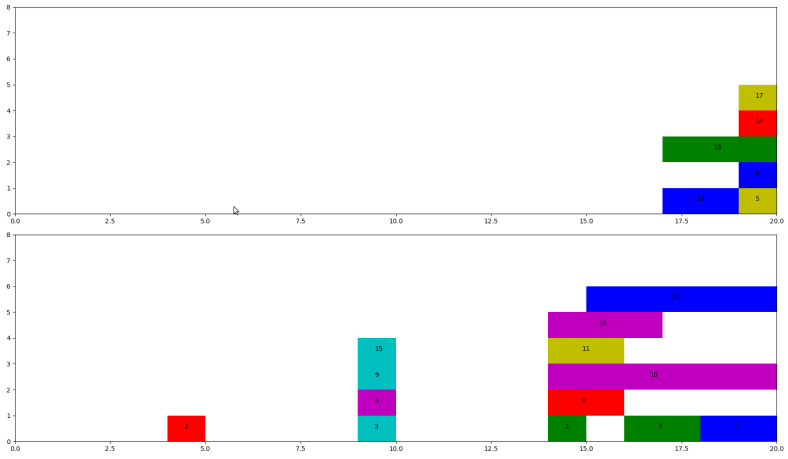
  - On ne lance pas de tâche si on ne peut pas la terminer :

$$\sum_{p \in P} \sum_{t \in T} x_{vpt}(t + \tau_v) \leq T_{max} \ , \forall v \in V \tag{17}$$

- Dans les contraintes restantes, on remplace les termes x_{vpt} par le terme $\sum_{s=t-\tau_v}^{t} x_{vps}$ qui détermine si la tâche v est active sur la machine p à l'instant t (ancienne signification de x_{vpt}), par exemple la contrainte d'utilisation des cœurs (11) devient :

$$\sum_{v \in V} c_v \sum_{s=t-\tau_v}^{t} x_{vps} \leq C_p \ , \forall p \in P, \forall t \in T \tag{18}$$

Voici un exemple de solution donnée par ce modèle pour 2 machines, 19 tâches et un horizon de 20 instants :



2.3 Liste des modifications apportées au modèle fourni au début du stage

2.3.1 Modification des données connues

Dans le modèle précédent, on possédait des informations sur les tâches et les machines concernant le nombre d'instruction ou le nombre d'instruction par seconde, toutes ces données ont été remplacées par le nombre de cœur utilisé par les tâches ou disponible sur la machine pour mieux coller aux instructions que l'on peut avoir sur les tâches. On connait aussi désormais la durée des tâches alors qu'avant on devait effectuer un calcul pour les connaitre. Un autre changement apporté concerne les calculs de consommations et de quantité d'énergie, on utilise maintenant les consommations(vitesse d'utilisation d'énergie) ainsi que l'unité de temps Δt afin de déterminer alors qu'avant on ne considérait uniquement que des consommations en quantité d'énergie ce qui impliquait une moins bonne modularité du code.

2.3.2 Fonction de rendement de la source non réversible payante (fournisseur d'énergie)

Dans la nouvelle version du modèle, on prend en compte le fait que la fonction de rendement du fournisseur d'énergie peut dépendre de l'instant t. Cela permet par exemple de considérer deux types de rendement un pour la journée et un pour la nuit.

Un changement dans le type d'implémentation de ces fonctions a aussi été effectué, la dernière solution qui avait été choisie était l'utilisation des contraintes SOS type 2. Cette implémentation a été remplacée par l'utilisation des contraintes linéaires par morceaux de CPLEX qui utilise elle aussi indirectement les contraintes de type SOS type 2 mais permet une initialisation plus rapide ainsi que la prise en compte de cas particuliers comme les fonctions linéaires par morceaux discontinues.

Le fait de considérer des fonctions de rendement dépendantes du temps permet aussi d'effectuer le changement abordé dans le prochain paragraphe.

2.3.3 Suppression du terme de production photovoltaïque

Dans la présentation du problème, la production d'énergie par une source non réversible gratuite (panneau photovoltaïque) a été abordée or cette production n'apparait pas dans la contrainte du bilan énergétique (8). En effet, on modifie maintenant la fonction de rendement de la source extérieure pour prendre en compte que l'on dispose d'une certaine quantité d'énergie qui n'aura pas de cout. On effectue la modification de la façon suivante :

- si la production photovoltaïque est non nulle égale a pv alors on rajoute un morceau à la fonction de rendement : $\rho^{-1}(x) = 0$ si $x \in [0, pv]$ sinon $\rho^{-1}(x - pv) = \rho^{-1}(x)$ si $x \geq pv$
- si la production est nulle, on n'effectue pas de changement

Par exemple si la fonction de rendement de base était représenté par la courbe en noir et que la production photovoltaïque est de 2 on obtient la courbe rouge.

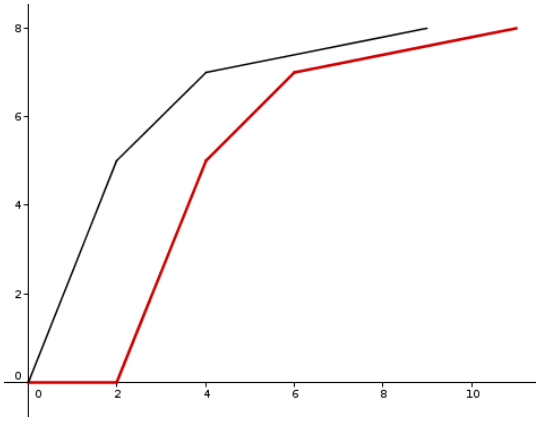


FIGURE 1 – Rendement avant et après modification avec la production photovoltaïque

3 Modèle alternatif : une décomposition du problème

3.1 Principe de la décomposition

Le modèle général associé au problème n'est pas très performant quand il s'agit de traiter des instances considérées comme grandes. Il a donc été décidé d'essayer de décomposer le problème afin d'essayer de pouvoir traiter de "grandes" instances.

Dans cette décomposition on découpe le problème en deux : tout d'abord en effectuant l'ordonnancement des tâches sans stockage afin d'obtenir la consommation des machines à chaque instant puis en déterminant comment utiliser la batterie afin de réduire le plus possible la consommation totale. Une fois les variations de la batterie déterminée, on peut réeffectuer l'ordonnancement sans stockage en ayant préalablement modifié la fonction de rendement de la source extérieure (principe abordé plus en détail dans la suite du rapport). Cela nous permet d'avoir un processus itératif qui va converger vers la solution.

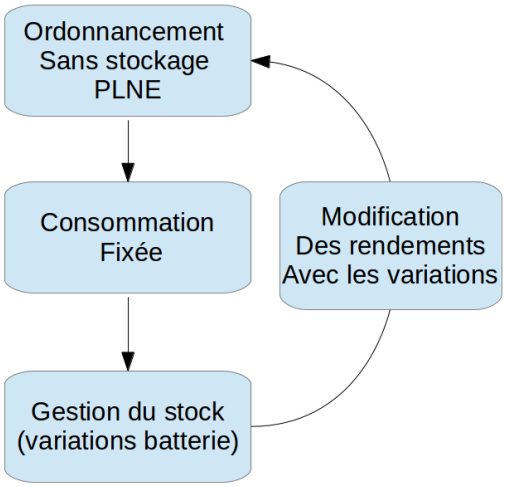


FIGURE 2 – Schéma de la décomposition

3.2 Première étape de la décomposition : l'ordonnancement sans stockage

Ici on reprend le modèle général en enlevant les contraintes liées à la batterie, (14) et (15), et en modifiant la contrainte du bilan énergétique (8). On obtient donc le modèle suivant :

On veut minimiser le cout total en énergie :

$$\min \sum_{t \in T} c_t \tag{19}$$

Sous les contraintes suivantes :

– On fixe les natures des variables de décision :

$$x_{vpt} \in \{0,1\} \ , \ \forall v \in V, \forall p \in P, \forall t \in T \tag{20}$$

$$z_{pt} \in \{0,1\} \ , \ \forall p \in P, \forall t \in T \tag{21}$$

$$c_t, d_t \geq 0 \ , \ \forall t \in T \tag{22}$$

$$c_t = \rho^{-1}(d_t) \ , \ \forall t \in T \tag{23}$$

– On a un bilant énergétique nul à tout instant :

$$d_t - \sum_{v \in V} \sum_{p \in P} \frac{(P_{p,max} - P_{p,min})c_v}{C_p} x_{vpt} - \sum_{p \in P} P_{p,min} z_{pt} = 0 \ , \ \forall t \in T \tag{24}$$

– Une tâche ne peut tourner que sur une seule machine à la fois :

$$\sum_{p \in P} x_{vpt} \leq 1 \ , \ \forall v \in V, \forall t \in T \tag{25}$$

– La durée d'exécution de la tache est égale à la durée de la tâche :

$$\sum_{p \in P} \sum_{t \in T} x_{vpt} = \tau_v \ , \ \forall v \in V \tag{26}$$

– On n'utilise pas plus de cœur que le nombre maximal de chaque machine :

$$\sum_{v \in V} c_v x_{vpt} \leq C_p \ , \ \forall p \in P, \forall t \in T \tag{27}$$

– On n'utilise pas pas plus de mémoire que la mémoire maximale de chaque machine :

$$\sum_{v \in V} m_v x_{vpt} \leq M_p \ , \ \forall p \in P, \forall t \in T \tag{28}$$

– On force z_{pt} à valoir 1 si la machine p est allumée au temps t :

$$z_{pt} \geq x_{vpt} \ , \forall v \in V, \forall p \in P, \forall t \in T \tag{29}$$

3.3 Deuxième étape de la décomposition : détermination de la gestion du stock

On note u_t la consommation des machines déterminée par l'ordonnancement sans stockage.

3.3.1 Avec un algorithme de lot-sizing

Cet algorithme a été formulé par Omar Saadi pendant son stage(2017) dans le cadre du projet PGMO.

Avant de pouvoir utiliser l'algorithme, certaines hypothèses doivent être imposées sur le modèle. Tout d'abord, on impose que certaines valeurs soient entières : les stocks, les productions, les abscisses des breakpoints de la fonction de rendement de la source extérieur. De plus on impose aussi le fait que la batterie ait un rendement idéal, que le stock final soit nul, que l'unité de temps soit égale à 1 et enfin on ne limite pas les variations sur le stock.

Le principe de l'algorithme est de déterminer pour tout $k \in \{1, ..., T_{max} - 1\}$ et pour tout $s \in \{0, ..., b_{k-1}\}$ (les variables b_k représentent la limite du stock au moment k), le cout minimum, pour la période entre les instants k et T_{max} sachant que l'on dispose d'un stock s à l'instant k. On note ce cout $F_k(s)$. En plus de cela, on détermine au même moment la production optimale correspondante notée x_{ks}.

Ces différents couts et production sont déterminés par un algorithme de programmation dynamique de façon backward(on calcul d'abord pour $k = T_{max} - 1$ puis $k = T_{max} - 2$, etc).

Une fois tous les couts et productions calculées on reconstruit la solution en calculant pour chaque instant le stock disponible puis en déduisant la production pour celui-ci.

3.3.2 Avec la PLNE

Voici le modèle à résoudre pour déterminer le stockage si l'on considère la même configuration que pour le lot-sizing(rendement de batterie idéal, stock final nul, unité de temps égale à 1, pas de stock minimum, pas de limite sur les variations de charge).

Ici, on veut toujours minimiser le cout total en énergie :

$$\min \sum_{t \in T} c_t \tag{30}$$

Sous les contraintes suivantes :

– On fixe les natures des variables de décision :

$$c_t, d_t, q_t \geq 0 \ , \ \forall t \in T \tag{31}$$

$$c_t = \rho^{-1}(d_t) \ , \ \forall t \in T \tag{32}$$

$$r_t \in \mathbb{R} \ , \ \forall t \in T \tag{33}$$

– On a un bilan énergétique nul à tout instant :

$$d_t + r_t - u_t = 0 \ , \ \forall t \in T \tag{34}$$

– Le nouveau stock est égal à l'ancien stock moins la variation :

$$q_{t+1} = q_t - r_t \ , \ \forall t \in T \tag{35}$$

– Le stock est toujours inférieur ou égal au stock maximum :

$$q_t \leq q_{max} \ , \ \forall t \in T \tag{36}$$

– Le stock final est égal à 0 :

$$q_{T_{max}} = 0 \tag{37}$$

3.4 Modification de la fonction de rendement

À chaque itération, après avoir déterminé les variations du stock, on modifie la fonction de rendement de la façon suivante : pour chaque instant on prend la fonction de rendement initiale associée (fonction de rendement avec production photovoltaïque) puis on la modifie avec la variation de charge. Si la variation correspond à une décharge on effectue le même type de changement que pour la production photovoltaïque : on translate la fonction vers la droite (cela correspond à créer des morceaux de la fonction où l'énergie est gratuite). Si la variation correspond à une charge on translate la fonction vers la gauche (on force une certaine consommation).

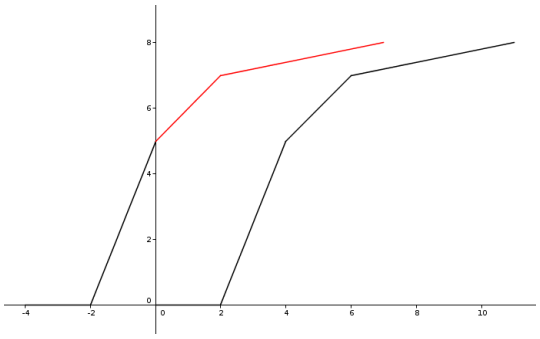


FIGURE 3 – Exemple de modification lors d'une charge

Ici, la fonction en noir pour les termes positifs devient la fonction en rouge si l'on considère une charge de 4 unités.

4 Comparaison numérique des différentes méthodes

4.1 Modèle général et modèle décomposé PLNE

Voici un tableau de résultat récapitulant les différents temps d'exécution ainsi que les valeurs objectives entre le modèle PLNE classique et celui décomposé. On a utilisé ici les variantes préemptives des modèles. La répartition pour le modèle de la décomposition est la suivante :

- Ordonnancement préemptif sans stockage : 90% de la time limit
- Détermination de l'utilisation du stockage : 10$ de la taille limit

Les différents tests ont été réalisés avec une fonction linéaire par morceaux qui contient 7 morceaux pre-modification et un rendement idéal sur la batterie.

INSTANCE T P V	COMPLET : OBJECTIF TPS/LIMIT	ITERATIF : OBJECTIF TPS/LIMIT*ITERMAX
20 2 20	4114 450/450	4114 154/150 $\times$ 3
30 2 50	11280 450/450	11280 174/150 $\times$ 3
30 4 100	22805 72/450	22805 281/150 $\times$ 3
60 2 100	12535 450/450	12881 450/150 $\times$ 3
60 4 150	25966 2000/2000	26183 1893/1000 $\times$ 2
120 2 150	5944 2000/2000	6972 1855/1000 $\times$ 2 (NO SOLUTION 200 $\times$ 10)
120 1 150	11255 1353/2000	11265 228/200 $\times$ 10

TABLE 1 – Comparaison du modèle classique avec le modèle itératif

On peut remarquer que le modèle décomposé est plus performant sur les petites instances car le modèle général a du mal à démontrer qu'une solution est l'optimum. Sur les instances plus grandes, de façon générale les deux modèles ont des performances similaires.

4.2 Modèle décomposé PLNE et Modèle décomposé lot-sizing

Comparaison entre determination de la strategie de stockage par cplex(limité à 300 sec) et par un algorithme de lotsizing. On peut remarquer que CPLEX s'adapte bien au changement de stock maximal(tableau1) mais mal à l'augmentation des montants de la demande(tableau 2). Le lot-sizing a le comportement réciproque.

T=nombre d'instant Q=nombre moyen de break-point S=stock maximal

INSTANCE T Q S	CPLEX : OBJECTIF TPS	LOTSIZING : OBJECTIF TPS
20 4 100	898 0.009	900 0.013
20 4 100	898 0.009	900 0.013
100 4 100	3823 0.023	3853 0.079
100 10 100	3983 0.025	4071 0.13
100 4 1000	3808 0.145	3857 0.854
100 10 1000	3822 0.334	3924 1.405
100 4 10000	3595 0.097	3692 14.037
100 10 10000	3617 0.045	3715 19.035
500 4 100	4647 0.125	4687 0.372
500 10 100	4841 0.126	4945 0.632
500 4 1000	4626 0.47	4676 4.529
500 10 1000	4345 1.783	4465 133
500 10 1000	4654 8.821	4780 7.067
500 4 10000	4316 1.225	4435 113.044
1000 4 100	8613 0.309	8692 0.745
1000 10 100	8972 0.306	9211 1.268
1000 4 1000	6612 3.296	6739 9.003
1000 10 1000	6661 300	6916 14.164
1000 4 10000	4396 3.325	4508 279
1000 10 10000	4396 2.908	4508 279

TABLE 2 – Comparaison enter CPLEX et lotsizing ($max(dt) = 100$)

INSTANCE T Q S	CPLEX : OBJECTIF TPS	LOTSIZING : OBJECTIF TPS
20 4 100	22040 0.011	22041 0.013
20 10 100	22363 0.013	22363 0.023
100 4 100	126914 0.035	126921 0.072
100 10 100	128690 0.053	128697 0.127
100 4 1000	124685 300	124719 0.863
100 10 1000	125843 300	125865 1.449
100 4 10000	113475 300	113558 20.977
100 10 10000	113377 9.381	113471 26.996
500 4 100	478405 0.21	478436 0.374
500 10 100	485370 0.241	485413 0.645
500 4 1000	464550 300	463547 4.507
500 10 1000	469451 300	468353 7.539
500 4 10000	424397 300	420624 104
500 10 10000	424506 300	420767 138
1000 4 100	961865.6 0.492	961932.5 0.758
1000 10 100	975523 0.568	975617 1.308
1000 4 1000	931375 300	927841 9
1000 10 1000	940165 300	937886 15

TABLE 3 – Comparaison enter CPLEX et lotsizing ($max(dt) = 2000$)

5 Conclusion

Tout d'abord je tiens encore à remercier l'ensemble de l'équipe ROC pour cette opportunité et le temps passé avec eux. Ce stage m'aura permis de découvrir le domaine de la recherche opérationnelle et plus particulièrement la programmation linéaire. Il m'a aussi permis de découvrir le monde de la recherche dans un laboratoire tel que le LAAS. J'aurais aimé avoir eu le temps d'implémenter les nouvelles variantes de l'algorithme de lot-sizing qui imposaient moins de conditions sur le modèle cependant cela n'a pas été possible.