

Systèmes de Décision Projet

Groupe 21

José Lucas DE MELO COSTA
Fernando KURIKE MATSUMOTO
Victor Felipe DOMINGUES DO AMARAL

Mention Intelligence Artificielle
2021-2022

Université Paris-Saclay
CentraleSupélec

Table des matières

1. Modélisation	2
1.1. Entrées	2
1.2. Variables de décision	3
1.3. Contraintes	3
1.4. Fonctions objectives	5
2. Optimisation sur les trois instances	5
2.1. Implémentation avec Gurobi	5
2.2. Résultats	6
2.2.1. Instance « toy »	6
2.2.2. Instance « medium »	7
2.2.3. Instance « large »	8
3. Modèles de préférence	9
3.1. UTA	10
3.1.1. Résultats UTA	11
3.2. Somme ponderée	14
3.2.1. Résultats - somme ponderée	15
3.3. PROMETHEE I	17
3.3.1. Résultats - PROMETHEE	17
4. Conclusion	18
A. Tableaux des Résultats	20

1. Modélisation

1.1. Entrées

On peut définir N_m le nombre de personnes (membres) à travailler, N_c le nombre de compétences possibles, N_p le nombre de projets, N_j le délai maximal des projets.

L'attribution des compétences est modélisée comme $H \in \mathbb{B}^{N_m \times N_c}$

$$H_{i,k} = \begin{cases} 1, & \text{si la personne } i \text{ a la compétence } k \\ 0, & \text{sinon} \end{cases}$$

Les congés sont modélisés aussi comme une matrice $C \in \mathbb{B}^{N_m \times N_j}$

$$C_{i,\ell} = \begin{cases} 1, & \text{si la personne } i \text{ prends congé au jour } \ell \\ 0, & \text{sinon} \end{cases}$$

La nécessité des projets est modélisée comme $Q \in \mathbb{N}^{N_p \times N_c}$, avec $Q_{j,k} = c_i$, où $c_i \in \{0, \dots, N_c\}$ est le nombre de compétences du type k dont le projet j a besoin.

Le revenu d'un projet j , $Rev_j \in \mathbb{Z}^{0+}$, est le total reçu par la réalisation de ce projet.

La pénalité d'un projet j , $P_j \in \mathbb{Z}^{0+}$, est le coût par jour de retard de ce projet.

La deadline d'un projet j , $Dl_j \in \{1, \dots, N_j\}$, est le nombre de jours accordé pour réaliser ce projet.

1.2. Variables de décision

Compte tenu de la modélisation du problème, nous savons que l'allocation de ressources se fera sur quatre variables : 1) la personne, 2) le projet, 3) le jour et 4) la compétence exercée. Ainsi, on utilise une variable d'optimisation binaire et de dimension égale à quatre :

$$T_{i,j,k,\ell} = \begin{cases} 1, & \text{si la personne } i \text{ travaille dans le projet } j, \\ & \text{avec la compétence } k \text{ au jour } \ell \\ 0, & \text{sinon} \end{cases}$$

avec $i \in \{1, \dots, N_m\}$, $j \in \{1, \dots, N_p\}$, $k \in \{1, \dots, N_c\}$ et $\ell \in \{1, \dots, N_j\}$.

Nous définissons aussi des variables auxiliaires :

- Une variable binaire $R \in \mathbb{B}^{N_p}$ qui indique si un projet est réalisé :

$$R_j = \begin{cases} 1, & \text{si le projet } j \text{ est réalisé} \\ 0, & \text{sinon} \end{cases}$$

- Des variables De , F et Re de dimension N_p qui indiquent les jours de début, de fin et le retard d'un projet : $De, F \in \{1, \dots, N_j\}^{N_p}$ et $Re \in \{0, \dots, N_j - 1\}^{N_p}$.
- Une variable qui indique la durée du projet le plus long : $Dm \in \{0, \dots, N_j\}$.
- Une variable binaire affecté au projet $Af \in \mathbb{B}^{N_m \times N_p}$ qui indique si une personne a travaillée sur un projet.
- Une variable $Mp \in \{0, \dots, N_p\}$ qui indique le nombre maximum des projets auquel une personne est affectée.
- Une variable $W \in \mathbb{B}^{N_p \times N_j}$ qui indique si des travaux ont été réalisés sur un projet j dans un jour l .

1.3. Contraintes

Contrainte de qualification Une personne ne peut exercer une compétence que si elle a cette compétence :

$$\forall i, j, k, \ell \quad T_{i,j,k,\ell} \leq H_{i,k}$$

Contrainte d'unicité de l'affectation Chaque jour, chaque personne est attribué à un seul projet au maximum en ne réalisant qu'une compétence :

$$\forall i, \ell \quad \sum_{j=1}^{N_p} \sum_{k=1}^{N_c} T_{i,j,k,\ell} \leq 1$$

Contrainte de congé Une personne ne peut pas travailler pendant ses jours de congé :

$$\forall i, j, k, \ell \quad T_{i,j,k,\ell} \leq 1 - C_{i,\ell}$$

Contrainte de travail sur projet Si un travail a été réalisé sur un projet j dans le jour ℓ , donc :

$$\forall i, j, k, \ell \quad T_{i,j,k,\ell} \leq W_{j,\ell}$$

Contraintes d'unicité de la réalisation d'un projet et de couverture des qualifications Pour qu'un projet soit considéré comme réalisé, tous les jours de travail de chaque compétence doivent être couverts par des membres du personnel. De plus, un projet ne peut être réalisé qu'une seule fois :

$$\forall j, k \quad \sum_{i=1}^{N_m} \sum_{\ell=1}^{N_j} T_{i,j,k,\ell} \geq R_j \cdot Q_{j,k}$$

Contraintes sur les variables d'affectation (Af et Mp)

$$\begin{aligned} \forall i, j, k, \ell \quad Af_{i,j} &\geq T_{i,j,k,\ell} \\ \forall i \quad Mp &\geq \sum_{j=1}^{N_p} Af_{i,j} \end{aligned}$$

Contraintes sur la durée d'un projet Assure que les variables auxiliaires Début (De), Fin (F) et Retard (Re) et DuréeMax (Dm) soient bonnes. Si un projet j n'est pas réalisé, on a $De_j = N_j$ et $Fin_j = 1$.

$$\begin{aligned} \forall j, \ell \quad De_j &\leq \ell + N_j \cdot (1 - W_{j,\ell}) \\ \forall j, \ell \quad F_j &\geq \ell + N_j \cdot (W_{j,\ell} - 1) \\ \forall j \quad Re_j &\geq F_j - Dl_j \\ \forall j \quad Dm &\geq F_j - De_j + 1 \end{aligned}$$

1.4. Fonctions objectives

Maximisation du résultat financier Une fonction qui compose le total reçu de chaque projet moins les pénalités (retards de chaque projet).

$$f_1(x) = -(\text{Rev}^\top R - \text{Re}^\top P)$$

où $x = (T, R, De, F, Re, Dm, Af)$.

Minimisation de la charge des collaborateurs On souhaite que les collaborateurs n'aient pas à changer trop souvent de projet et, pour ce faire on s'attachera à minimiser le nombre de projets sur lesquels un quelconque collaborateur est affecté.

$$f_2(x) = Mp$$

Minimisation d'exécution du projet le plus long Il est important que les projets soient réalisés dans un nombre limités de jours consécutifs, ainsi on cherchera pour cela à exécuter le projet le plus long en un minimum de jours.

$$f_3(x) = Dm$$

2. Optimisation sur les trois instances

2.1. Implémentation avec Gurobi

À l'aide de la librairie d'optimisation Gurobi, on a implémenté le modèle décrit dans la Section 1 (le code est dans le fichier `utils.py`). On a d'abord essayé d'exécuter ce code avec un seul objectif (f_1 , le résultat financier) pour vérifier son bon fonctionnement. Une vérification plus approfondie a été laissée pour plus tard. On a également vérifié que le modèle marchait correctement dans l'optimisation des autres fonctions objectives (f_2 et f_3). Dans ces cas, le code a correctement renvoyé des solutions où personne ne travaille ($T_{i,j,k,\ell} = 0$).

Une fois qu'on avait le modèle prêt pour optimiser une fonction objective (soit f_1 , f_2 ou f_3), on a adopté la méthode suivante pour trouver les solutions non dominées. Comme f_2 est bornée dans l'intervalle $\{0, \dots, N_p\}$ et f_3 dans l'intervalle $\{0, \dots, N_j\}$, l'approche utilisée consiste à parcourir les $(N_p + 1) \cdot (N_j + 1)$ combinaisons de f_2 et f_3 en optimisant f_1 pour chacune d'eux. On a donc :

$$S^* = \{s_{i,j}^* : i \in \{0, \dots, N_p\} \wedge j \in \{0, \dots, N_j\}\}$$

$$s_{i,j}^* = \min_x f_1(x) \quad \text{tq.} \quad \begin{cases} (C) \\ f_2(x) = i \\ f_3(x) = j \end{cases}$$

où (C) sont les contraintes énumérées dans la Section 1.3.

Cependant, cette approche demande un nombre assez large d'optimisations (à l'extrême, 962 pour l'instance « large »), parmi lesquelles il y a plusieurs qui tournent beaucoup plus lentement que l'optimisation mono-objectif. Pour réduire ce problème, on a utilisé la feature multi-scénario fournit par Gurobi, qui nous permet de réaliser tous ces optimisations à la fois.

Après, on a cherché l'ensemble des solutions non-dominées. L'ensemble des solutions non-dominées est un sous ensemble de S^* ($S_{\text{nd}}^* \subseteq S^*$). Il faut donc filtrer quelles sont les solutions non dominées dans cet ensemble. Alors :

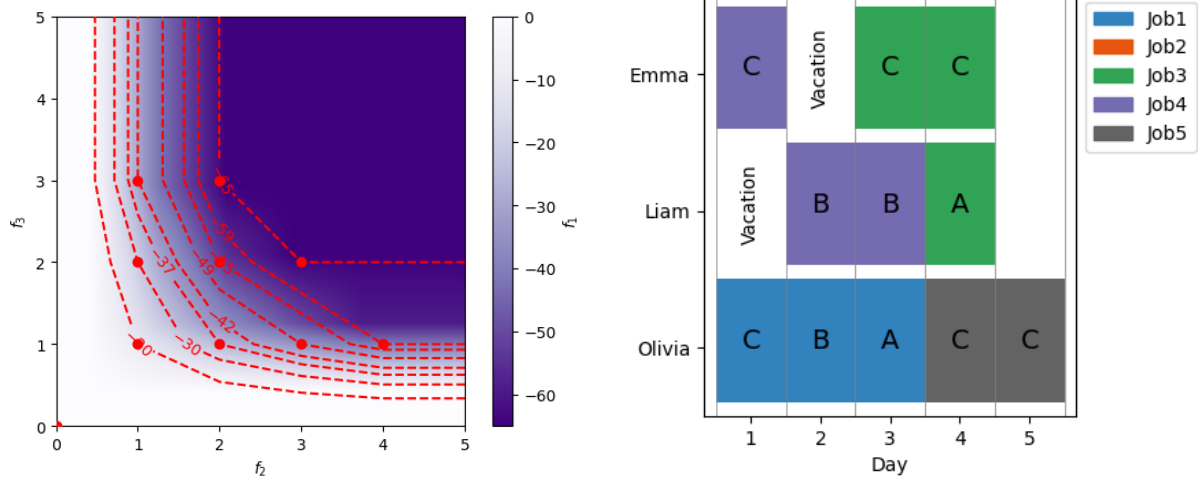
$$S_{\text{nd}}^* = \{s : s \in S^* \wedge \nexists s' \in S^*. s' \succeq s\}$$

Pour afficher les résultats obtenus, on a implémenté un diagramme avec les travaux réalisés par personne par compétence par jour. On a ensuite utilisé ce diagramme pour vérifier si chacune des solutions ND trouvées pour l'instance toy respectaient les contraintes qu'on a défini. Un autre affichage possible est celle des la relation entre les trois fonctions objectives, ainsi que des solutions non-dominées. Ces deux types de figure sont montrés dans la suite.

2.2. Résultats

2.2.1. Instance « toy »

Pour l'instance la plus petite, on a trouvé 10 solutions non-dominées (ND). Les solutions non-dominées sont montrées dans la Figure 1a. Parmi ces solutions, celle qui minimise le f_1 (et le f_2 dans le cas où il y a plusieurs solutions qui minimisent f_1) est affiché dans la Figure 1b.

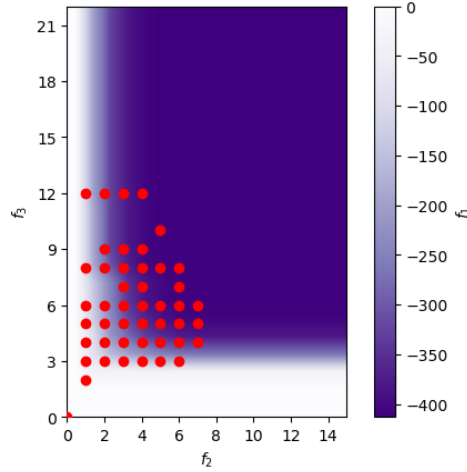


(a) Valeurs optimales de f_1 pour chaque f_2 et f_3 . Les points rouges sont les solutions ND. (b) Planning pour la solution ND qui minimise f_1 ($f_1 = -65, f_2 = 2, f_3 = 3$).

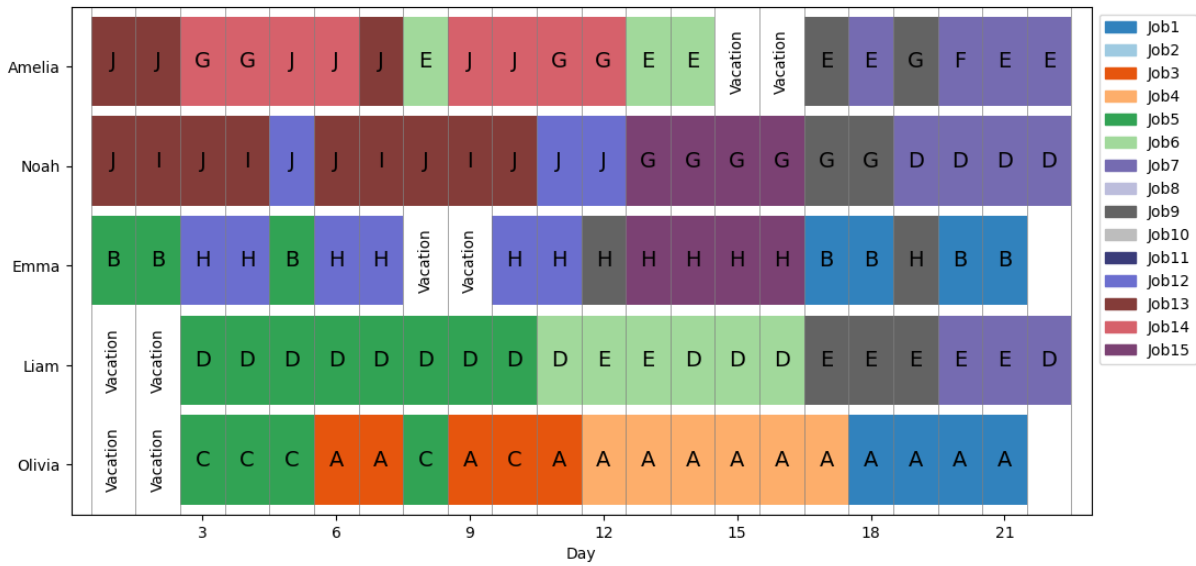
FIGURE 1 – Résultats trouvés pour l'instance « toy »

2.2.2. Instance « medium »

Pour l'instance medium, on a trouvé 46 solutions non-dominées. Des affichages similaires à celles de l'instance précédente sont montrés dans la Figure 2. On remarque dans la Figure 2a que pour la plupart des valeurs de f_2 et f_3 , l'algorithme trouve la solution optimale, ce qui veut dire que si les contraintes sur le nombre maximale des projets par personne et sur la durée maximale des projets ne sont pas trop strictes, on peut toujours maximiser le gain.



(a) Valeurs optimales de f_1 pour chaque f_2 et f_3 . Les points rouges sont les solutions ND.



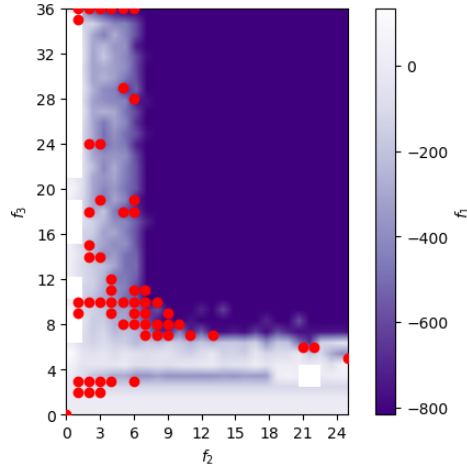
(b) Planning pour la solution ND qui minimise f_1 ($f_1 = -413, f_2 = 5, f_3 = 10$).

FIGURE 2 – Résultats trouvés pour l'instance « medium »

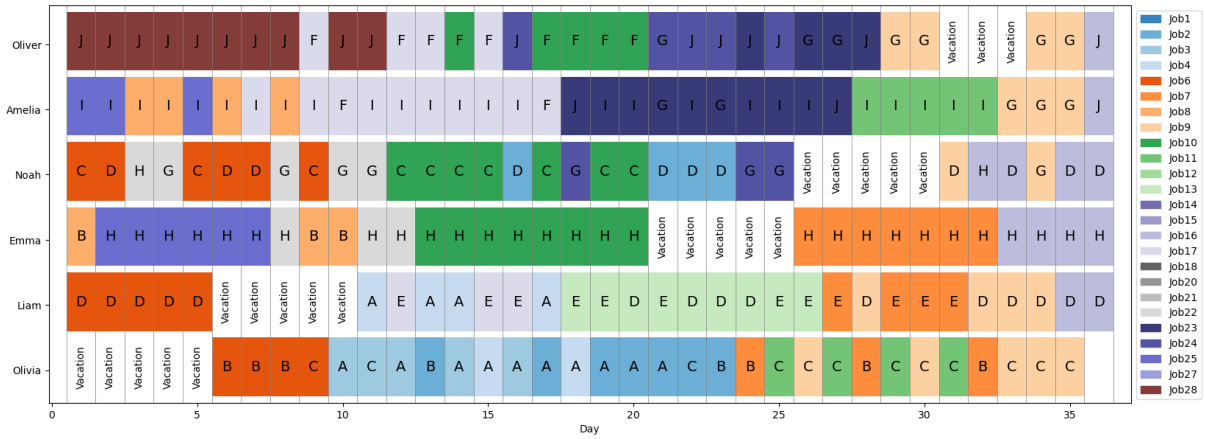
2.2.3. Instance « large »

Pour cette instance, l'optimisation était trop lente et il a fallu l'arrêter avant l'obtention des toutes les solutions optimales. Il y a donc des valeurs de f_2 et f_3 pour lesquelles la solution trouvée n'est pas optimale. Un tableau a été mis à disposition avec la solution trouvée pour chaque valeur de f_2 et f_3 (Annexe A). Pour les cas où l'optimisation n'a pas fini, l'écart relatif entre la meilleure valeur de f_1 trouvée et la borne de f_1 est donnée. Dans 20 cas, aucune solution a été trouvée dans le temps disponible.

On a ensuite utilisé cet ensemble des solutions approximatives trouvées pour obtenir les 59 solutions approximatives non dominées. Les résultats sont affichés dans la Figure 3



(a) Valeurs optimales de f_1 pour chaque f_2 et f_3 . Les points rouges sont les solutions ND.



(b) Planning pour la solution ND qui minimise f_1 ($f_1 = -817, f_2 = 7, f_3 = 11$). Dans ce cas, la solution est exacte (il n'est pas possible d'avoir $f_1 < -817$)

FIGURE 3 – Résultats approximatifs trouvés pour l'instance « large »

3. Modèles de préférence

Cette seconde partie vise à développer un modèle de préférence permettant de discriminer entre les solutions de la surface des solutions non-dominées.

Une fois que toutes les solutions non maîtrisées ont été trouvées, on entre dans un autre problème : déterminer un planning qui plaise au chef de projet et à l'administrateur. Il convient de mentionner que tous les modèles de préférences présentés ici nécessitent une sorte de préférences a priori.



Bienvenue chez CompuOpti

Vous êtes ingénieur-développeur chez CompuOpti, une entreprise qui aide ses clients à optimiser leurs décisions grâce à ses solutions.

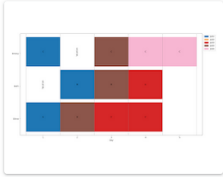
Notre PDG, Margaux Dourtille, cherche à maximiser le bénéfice total des projets réalisés. Et il y a plusieurs critères à prendre en compte:

- Maximiser le résultat financier de l'entreprise en maximisant le bénéfice, incluant les pénalités éventuelles
- Minimiser le nombre de projets sur lesquels un collaborateur est affecté, afin que les employés n'aient pas à changer de projet trop souvent.

Alors, laquelle des deux options est la meilleure selon vous ?

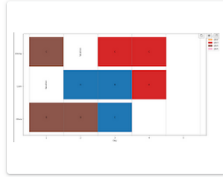
 jmcosta@usp.br (non partagé)
 [Changer de compte](#)


Quel est la meilleure option?



Un bénéfice de 65 euros avec un maximum de 3 projets par personne, et avec une durée maximale de 2 jours pour chaque projet ?

☐



Un bénéfice de 55 euros avec un maximum de 2 projets par personne, et avec une durée maximale de 2 jours pour chaque projet ?

☐

Quel est la meilleure option?

FIGURE 4 – Formulaires utilisés pour recueillir les préférences

Compte tenu de ce qui a été présenté dans le cours, trois types de modèles ont été testés : les deux premiers sont des modèles d'agrégation, le premier considère la somme pondérée des alternatives, tandis que le second traite des fonctions de valeur additives. Enfin, une méthode de surclassement a également été testée, à travers l'implémentation de l'algorithme Promethee.

Pour obtenir les préférences, un formulaire a été créé, comme présenté para la Figure 4 pour recueillir les préférences des utilisateurs, principalement pour le petit jeu d'instances (jouets). Cela a été fait uniquement pour obtenir des données sur les classements partiels ou les paires de préférences.

3.1. UTA

Pour ce premier problème, l'ensemble des options sont les alternatives qui constituent l'ensemble des solutions non dominées, appelé A . En outre, nous avons trois critères à considérer, représentés par $c = (c_1, c_2, c_3)$. Dans un premier temps, nous avons décidé d'utiliser une autre méthode classique d'agrégation pour déterminer les préférences : la combinaison de tous les critères en un seul critère appelé fonction d'utilité, $U(c) = U(c_1, c_2, c_3)$.

L'objectif de la méthode UTA est d'estimer les fonctions d'utilité en utilisant la programmation linéaire, en supposants que ces fonctions sont linéaire par morceaux.

Les entrées du programme incluent les choix à faire et une liste indiquant les préférences. Plus précisément, dans ce cas, il est nécessaire d'avoir un ranking R de taille J . Pour chaque critère, nous avons défini des limites et une subdivision, ce seront les valeurs minimales et maximales des données et une subdivision de 3, également espacée. De cette manière, pour chaque critère il est possible de considérer les points $c_i^k = \min_i + \frac{k}{L}(\max_i - \min_i)$, avec $k \in \{1, 2, 3\}$. Ainsi, si une valeur est dans un intervalle on considère :

$$u_i(c_i^j) = u_i(c_i^k) + \frac{c_i^j - c_i^k}{c_i^{k+1} - c_i^k} (u_i(c_i^{k+1}) - u_i(c_i^k))$$

En plus, pour un indice i , on considère $u(i) = \sum_d^3 s_d(c_d^i) + \sigma_i$. Cette erreur est décomposée en une erreur de sous-estimation et une erreur de surestimation.

De cette manière, le problème d'optimisation était :

$$\min(\sum_j^J \sigma_j^+ + \sigma_j^-)$$

Contrainte des préférences :

$$u(i) - \sigma_i^+ + \sigma_i^- \geq s(j) - \sigma_j^+ + \sigma_j^- + \epsilon \quad \forall \text{pairs } i, j$$

Contrainte de monotonie des fonctions :

$$u_i(c_i^{k+1}) - u_i(c_i^k) \geq \epsilon \quad \forall i, k = 0..L_i - 1$$

Normalisation :

$$u_i(c_i^0) = 0 \quad \forall i$$

$$\sum_{i=1}^n \omega_i = 1$$

3.1.1. Résultats UTA

Pour l'instance *large*, en utilisant le ranking $27 > 8 > 11 > 7 > 40 > 12 > 17 > 53 > 33 > 34$, les résultats de cinq premiers résultats sont présentés dans le tableau 2 et dans la Figure ??

Pour l'instance *medium*, en utilisant le ranking $32 > 41 > 36 > 6 > 27 > 3 > 23 > 22 > 40 > 37$, les résultats de cinq premiers résultats sont présentés dans le tableau 2 et dans la Figure 7

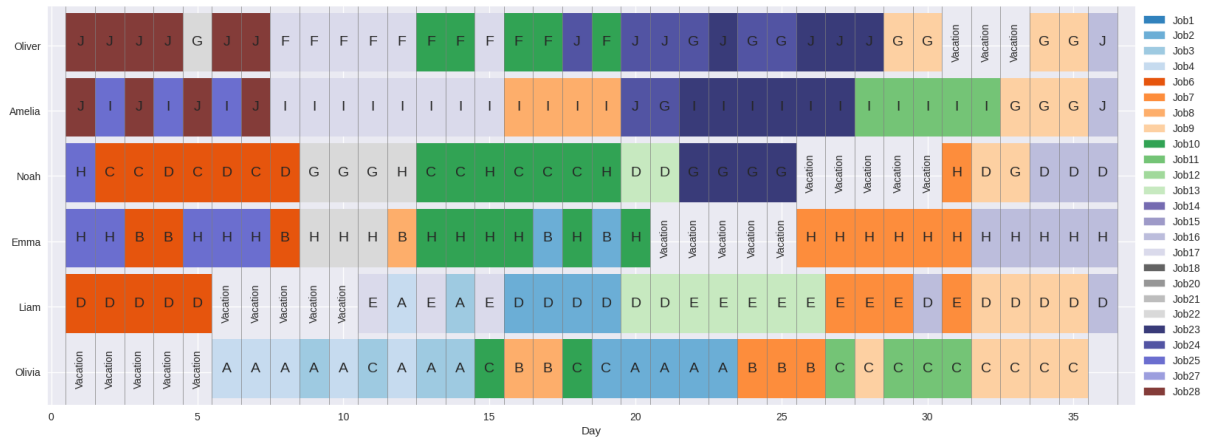


FIGURE 5 – Meilleure solution UTA - instance *LARGE*

index	g_1	g_2	g_3	Valeur UTA
21	814	-9	-8	0.108670
22	817	-10	-8	0.108090
34	814	-7	-10	0.107700
127	817	-9	-9	0.107605
35	817	-8	-10	0.1071216

TABLE 1 – Valeurs classées pour l'ensemble des données *medium*

Pour l'instance *small*, en utilisant le ranking $9 > 3 > 1$, les résultats sont présentés dans le tableau 3 et dans la Figure 9

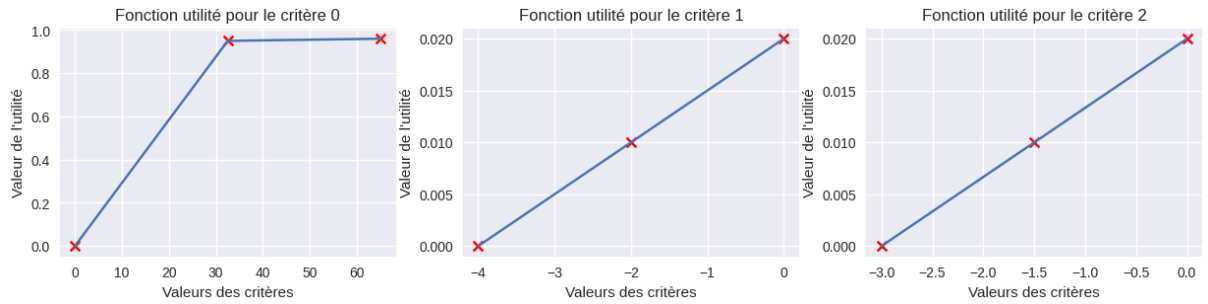


FIGURE 8 – Fonction utilité pour chaque critère - instance *small*

index	g_1	g_2	g_3	Valeur UTA
13	374	-6	-4	0.112768
12	356	-5	-4	0.112758
19	389	-5	-5	0.111723
11	331	-4	-4	0.111632
20	403	-6	-5	0.111096

TABLE 2 – Valeurs classées pour l'ensemble des données *medium*

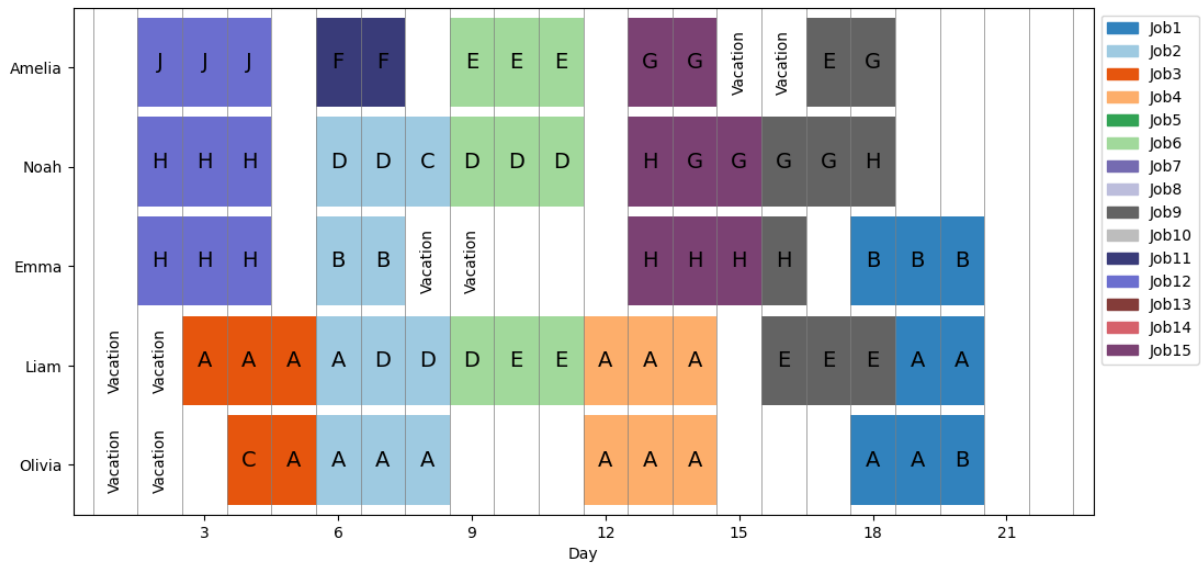


FIGURE 6 – Meilleure solution UTA - instance *medium*

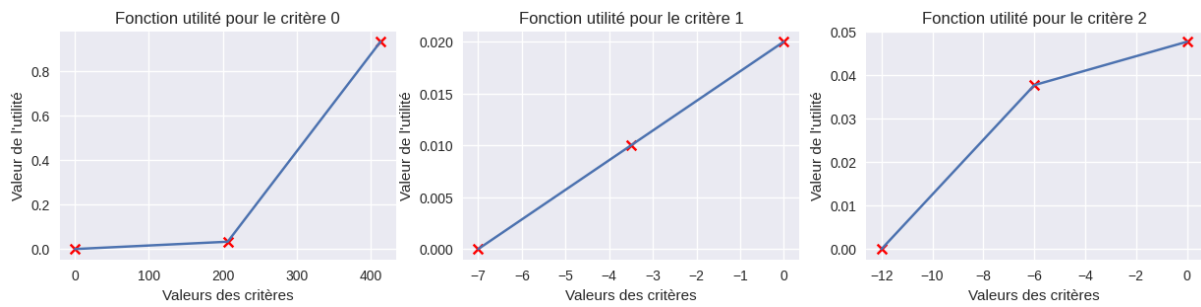


FIGURE 7 – Fonction utilité pour chaque critère - instance *medium*

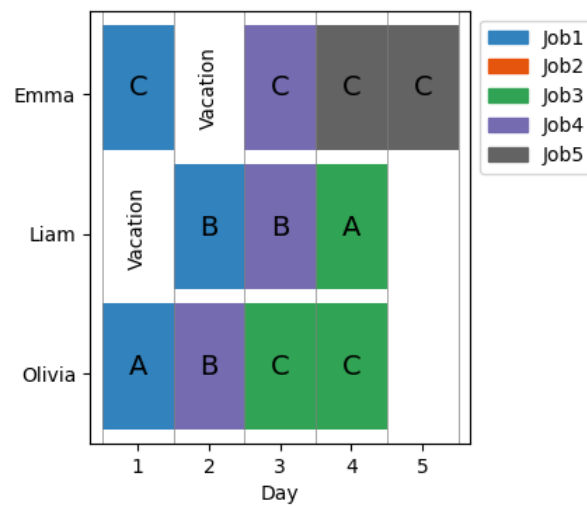


FIGURE 9 – Meilleure solution UTA - instance *small*

index	g_1	g_2	g_3	Valeur UTA
7	65.0	-3.0	-2.0	1.911667
9	65.0	-2.0	-3.0	1.910000
4	59.0	-4.0	-1.0	1.737949
6	55.0	-2.0	-2.0	1.624359
3	49.0	-3.0	-1.0	1.450641
8	42.0	-1.0	-3.0	1.242692
2	37.0	-2.0	-1.0	1.104872
5	30.0	-1.0	-2.0	0.898590
1	20.0	-1.0	-1.0	0.612949
0	-0.0	-0.0	-0.0	0.040000

TABLE 3 – Valeurs classées pour l’ensemble des données *small*

3.2. Somme pondérée

La méthode de la somme pondérée est une approche couramment utilisée dans l’analyse décisionnelle multicritères (MCDA), probablement la plus simple à conceptualiser. Comme dans la méthode précédente, dans cette méthode, l’ensemble d’alternatives (solutions non dominées) est évalué sur la base de trois critères, qui sont combinés en une unique valeur.

Dans ce cas, les critères sont pondérés pour refléter leur importance relative, et les valeurs des alternatives pour chaque critère sont multipliées par leurs poids respectifs pour calculer un score total pondéré, comme présenté par l’équation 3.2. Ce score représente l’évaluation globale de chaque alternative, et le décideur peut choisir l’alternative ayant le score le plus élevé comme option préférée. La méthode de la somme pondérée est simple, intuitive et facile à utiliser, mais elle a ses limites car elle suppose que les critères sont indépendants et de même poids.

$$U(g^k) = \sum_{i=1}^3 \omega_i * g_i^k$$

En outre, certaines contraintes de normalisation ont été fixées, commenté ci-dessous.

Pour chaque paire :

$$i > j \rightarrow \omega_1(g_1^i - g_1^j) + \omega_2(g_2^i - g_2^j) + \omega_3(g_3^i - g_3^j) \geq 0$$

Normalisation :

$$\sum_{i=1}^n \omega_i = 1$$

Le problème d’optimisation à résoudre consiste à rechercher des vecteurs de poids dans l’espace défini par les contraintes. Une fois les relations de dominance trouvées, il est possible de créer une pré-commande, comme le montre le graphe 14.

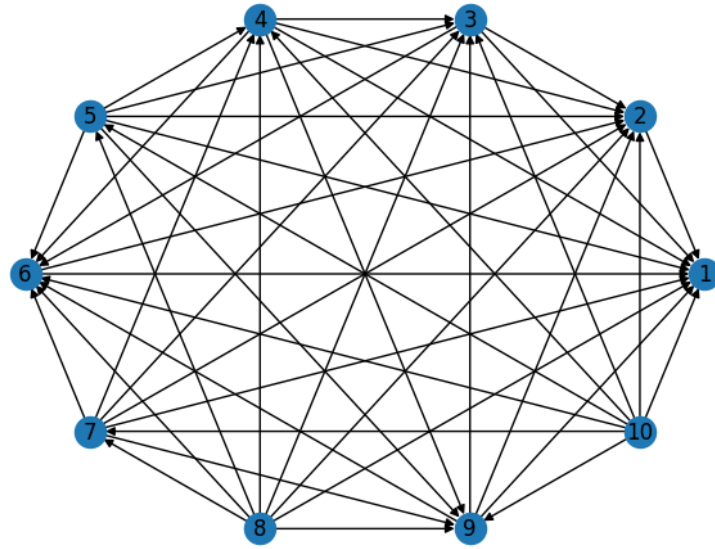


FIGURE 10 – Graphe Somme ponderée - instance *small*

La valeur permettant de classer les alternatives, cependant, a été calculée en tenant compte du degré de sortie moins le degré d'entrée de chaque nœud du graphe.

3.2.1. Résultats - somme ponderée

Pour l'instance *large*, les préférences utilisées étaient $(27 > 3)$, $(27 > 4)$, $(27 > 55)$, $(57 > 56)$, $(58 > 52)$, $(58 > 3)$, $(57 > 8)$, $(58 > 2)$, $(23 > 28)$, $(4 > 5)$, $(21 > 27)$, $(35 > 27)$, $(35 > 38)$, $(21 > 34)$. Les résultats des cinq premiers éléments sont présentés dans le tableau 4 et le planning final est présenté dans la Figure 11.

index	g_1	g_2	g_3	degree
22	817	-10	-8	56
27	817	-9	-9	54
31	814	-9	-8	54
35	817	-8	-10	52
34	814	-7	-10	52

TABLE 4 – Valeurs classées pour l'ensemble des données *medium*

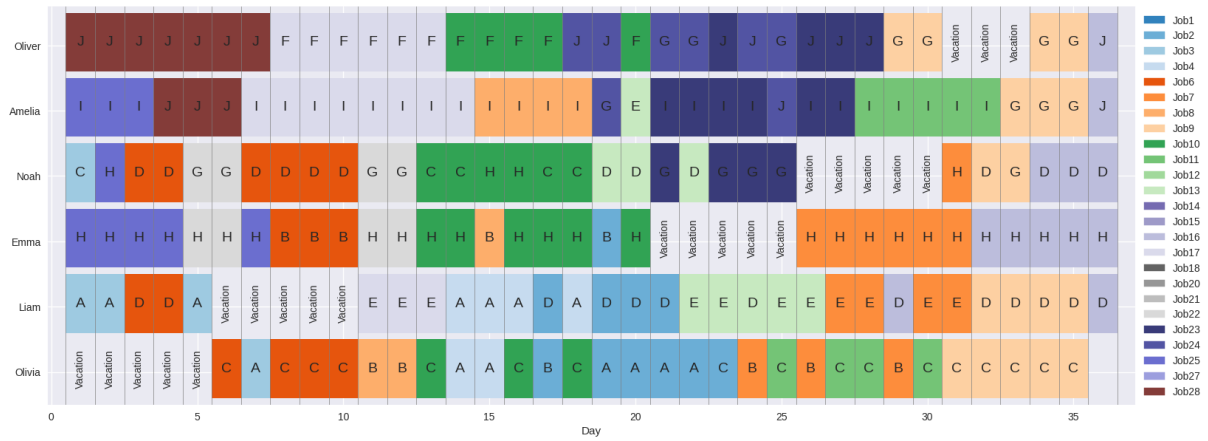


FIGURE 11 – Meilleure solution somme ponderee - instance *large*

Pour l'instance *medium*, les préférences utilisées étaient $(9 > 3)$, $(9 > 1)$, $(12 > 7)$, $(30 > 6)$, $(31 > 35)$, $(3 > 1)$, $(40 > 30)$. Les résultats des cinq premiers éléments sont présentés dans le tableau 5.

index	g_1	g_2	g_3	degree
27	408	-6	-6	37
26	403	-5	-6	35
36	410	-5	-8	34
20	403	-6	-5	34
28	411	-7	-6	34

TABLE 5 – Valeurs classées pour l'ensemble des données *medium*

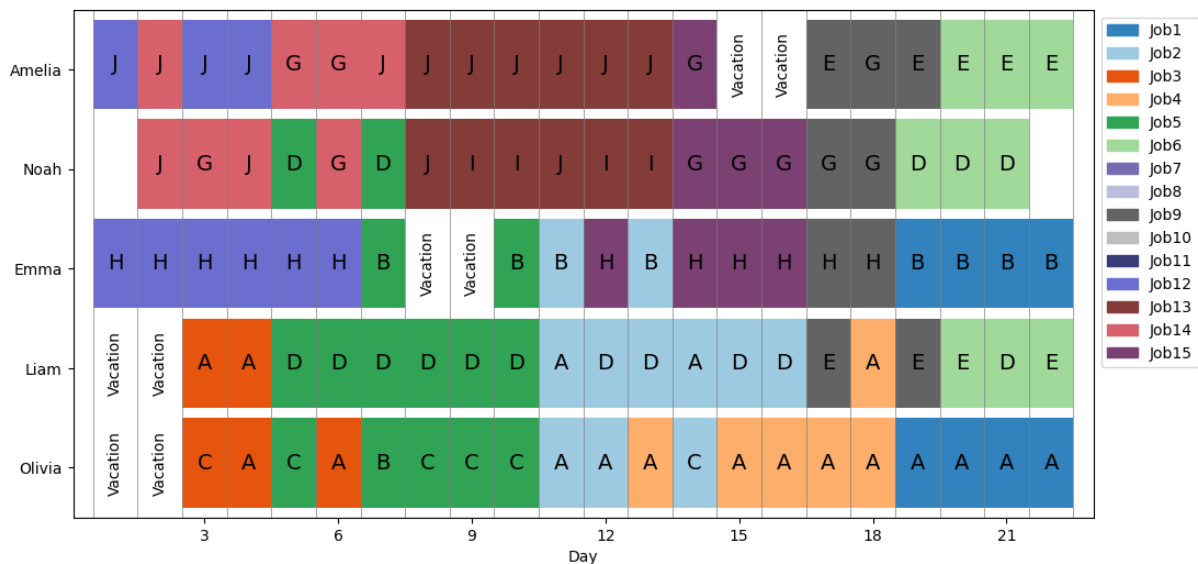


FIGURE 12 – Meilleure solution somme ponderee - instance *medium*

Pour l'instance *small*, en utilisant les préférences $9 > 3$, $9 > 1$ et $3 > 1$, les résultats sont présentés dans le tableau 6.

index	g_1	g_2	g_3	degree
8	65.0	-3.0	-2.0	8
10	65.0	-2.0	-3.0	8
5	59.0	-4.0	-1.0	4
7	55.0	-2.0	-2.0	4
4	49.0	-3.0	-1.0	1
9	42.0	-1.0	-3.0	-1
3	37.0	-2.0	-1.0	-3
6	30.0	-1.0	-2.0	-5
2	20.0	-1.0	-1.0	-7
1	-0.0	-0.0	-0.0	-9

TABLE 6 – Valeurs classées pour l’ensemble des données *small*

3.3. PROMETHEE I

En outre des méthodes d’aggrégation, un autre experiment était l’utilisation d’une méthode de surclassement. Ces méthodes, comme Promethee, sont utilisées pour l’enrichissement des évaluations et son complément descriptif analyse géométrique pour l’aide interactive. Fondamentalement, ces méthodes peuvent nous aider à prendre des décisions en déterminant quelle option est la plus adaptée à nos objectifs et à la compréhension du problème. Au lieu de nous dire la “bonne” réponse, la méthode Promethee nous offre un cadre complet et rationnel pour prendre une décision.

Au début, il est nécessaire de fournir un vecteur de poids, défini par le décideur, et des paramètres pour les fonctions qui modelisent les critères. En ce qui concerne les choix faites, les poids utilisées étaient ceux sortis de la méthode de somme pondéré. Les préférences étaient modélisés avec une fonction gaussienne, c’est-à-dire :

$$P(a, b) = f(d) = 1 - \exp\left(-\frac{d^2}{2\sigma^2}\right) \quad (1)$$

où, $d = a - b$ et σ sont les paramètres fournis pour chaque critère.

La méthode consiste en créer une matrice \mathbb{R}^{N_s, N_f} , où N_s est le nombre de solutions et N_f est le nombre de critères. Après la méthode calcule la distance de chaque solution par chaque critère : $D = \{d \in \mathbb{R}^{N_s, N_s, N_f} : d_k(s_i, s_j) = f_k(s_i) - f_k(s_j)\}$. Ensuite, appliquer la fonction gaussienne et multiplier pour les poids. Finalement, les scores sont calculées en fonction du flux de sortie et d’entrée.

3.3.1. Résultats - PROMETHEE

Pour l’instance *small*, les paramètres utilisés étaient $W = [0.070.50.43]$ pour les poids, $S = [20, 1, 2]$ pour les fonctions gaussiennes. Les résultats sont présentés dans la Figure 14.

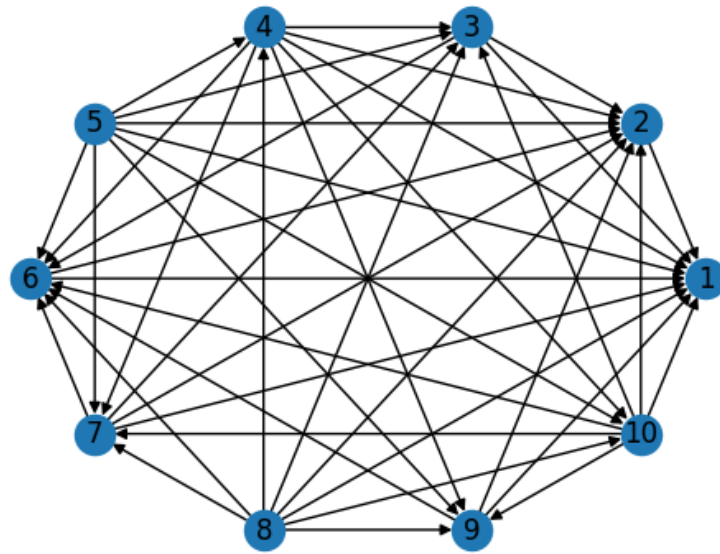


FIGURE 13 – Graphe Promethee - instance *small*

	0		2		4		6		8	
0	I	R	R	R	R	R	R	R	R	R
	P	I	R	R	R	R	R	R	R	R
2	P	P	I	R	R	P	R	R	R	R
	P	P	P	I	R	P	P	R	P	P
4	P	P	P	P	I	P	P	R	P	P
	P	P	R	R	R	I	R	R	R	R
6	P	P	P	R	R	P	I	R	R	R
	P	P	P	P	R	P	P	I	P	P
8	P	P	R	R	R	P	R	R	I	R
	P	P	P	R	R	P	P	R	P	I

FIGURE 14 – Matrice de préférences - instance *small*

4. Conclusion

Ce document montre l'implémentation de l'optimisation pour trois instances en utilisant la librairie Gurobi. La méthode utilisée consiste à parcourir les combinaisons de f_2 et f_3 et à optimiser f_1 pour chacune d'entre elles pour trouver les solutions non dominées. La feature multi-scénario de Gurobi a été utilisée pour réduire les délais d'optimisation. Des solutions non-dominées ont été trouvées pour l'instance la plus petite, avec un total de 10 solutions. Les résultats sont affichés sous forme de diagrammes pour montrer

les travaux effectués par personne, compétence et jour, ainsi que la relation entre les trois fonctions objectives et les solutions non-dominées. Les prochains pas pourraient inclure une vérification plus approfondie des résultats et une évaluation des performances pour les deux autres instances.

Finalement, la seconde partie de ce document a cherché à développer un modèle de préférence pour discriminer les solutions de la surface des solutions non-dominées. Pour cela, trois types de modèles ont été testés, à savoir les modèles d'agrégation pondérés et les fonctions de valeur additives, ainsi qu'une méthode de surclassement à travers l'algorithme Promethee. Pour obtenir les préférences, un formulaire a été créé pour recueillir les données des utilisateurs. Le premier modèle UTA testé implique une estimation des fonctions d'utilité à l'aide de la programmation linéaire en supposant que ces fonctions sont linéaires par morceaux. Bien que les résultats semblent prometteurs, il est important de noter que la méthode UTA nécessite des données préliminaires sur les préférences des utilisateurs. Les prochains pas incluent la continuation de l'application de ces modèles à des jeux de données plus importants et plus complexes, ainsi que l'analyse des résultats pour évaluer leur pertinence pour résoudre ce type de problème.

A. Tableaux des Résultats

f_1	f_2	f_3
0	0	0
-20	1	1
-37	2	1
-49	3	1
-59	4	1
-30	1	2
-55	2	2
-65	3	2
-42	1	3
-65	2	3

TABLE 7 – Solutions non-dominées pour l’instance « toy »

f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
0	0	0	-356	5	4	-343	3	6	-401	4	8
-30	1	2	-374	6	4	-385	4	6	-410	5	8
-80	1	3	-380	7	4	-403	5	6	-413	6	8
-155	2	3	-120	1	5	-408	6	6	-325	2	9
-195	3	3	-232	2	5	-411	7	6	-379	3	9
-225	4	3	-322	3	5	-346	3	7	-404	4	9
-250	5	3	-365	4	5	-386	4	7	-413	5	10
-265	6	3	-389	5	5	-411	6	7	-194	1	12
-90	1	4	-403	6	5	-180	1	8	-326	2	12
-210	2	4	-406	7	5	-305	2	8	-382	3	12
-279	3	4	-130	1	6	-370	3	8	-405	4	12
-331	4	4	-245	2	6						

TABLE 8 – Solutions non-dominées pour l’instance « medium »

f_1	f_2	f_3	écart (%)	f_1	f_2	f_3	écart (%)
0	0	0		-303	3	10	169.6%
-60	1	2		-359	4	10	127.6%
-90	2	2	53.3%	-514	5	10	58.9%
-115	3	2	173.9%	-534	6	10	53.0%
-90	1	3		-814	7	10	0.4%
-185	2	3	77.3%	-817	8	10	
-235	3	3	107.2%	-372	4	11	119.6%
-290	4	3	106.6%	-597	6	11	36.9%
-370	6	3	93.2%	-817	7	11	
-399	25	5	104.8%	-463	4	12	76.5%
-482	21	6	69.5%	-221	2	14	269.7%
-674	22	6	21.2%	-392	3	14	108.4%
-373	7	7	119.0%	-237	2	15	244.7%
-687	8	7	18.9%	-297	2	18	175.1%
-707	9	7	15.6%	-587	5	18	39.2%
-737	11	7	10.9%	-667	6	18	22.5%
-782	13	7	4.5%	-472	3	19	73.1%
-304	5	8	168.8%	-723	6	19	13.0%
-415	6	8	96.9%	-327	2	24	149.8%
-571	7	8	43.1%	-496	3	24	64.7%
-751	8	8	8.8%	-747	6	28	9.4%
-814	9	8	0.4%	-676	5	29	20.9%
-817	10	8		-143	1	35	471.3%
-112	1	9	629.5%	-290	1	36	
-332	4	9	146.1%	-510	2	36	19.0%
-433	6	9	88.7%	-626	3	36	16.8%
-681	7	9	20.0%	-708	4	36	11.2%
-817	9	9		-764	5	36	6.9%
-132	1	10	518.9%	-779	6	36	4.9%
-207	2	10	294.7%				

TABLE 9 – Solutions non-dominées pour l’instance « large ». Pour les solutions qui ne sont pas forcément optimales l’écart relatif d’optimalité à la fin de l’optimisation est indiqué (c.f. Section 2.2.3).