Projet SDP : CompuOpti

Thomas Brilland, Guillaume Dugat, Sacha Muller
Mention IA, CentraleSupélec, France
https://github.com/GuillaumeDugat/CompuOpti

I. Introduction

L'objectif est d'optimiser la planification d'une entreprise, CompuOpti, pour la réalisation d'un certain nombre de projets. Chaque projet nécessite des qualifications, un nombre de jours de travail par qualification, et a une date limite de rendu. Il faut ainsi trouver un planning qui permet de maximiser le profit de l'entreprise, tout en respectant les contraintes liées à chaque projet et au personnel disponible.

II. Modèles étudiés

A. Modélisation du problème de constitution du planning

1) Paramètres d'une instance du problème:

Paramètres relatifs aux indices :

- $\mathcal{W}orkers = \{0, \dots, W-1\}$ est l'ensemble du personnel ($W \in \mathbb{N}\backslash\{0\}$).
- $\mathcal{J}obs = \{0, \dots, J-1\}$ est l'ensemble des projets ($J \in \mathbb{N}\backslash\{0\}$).
- $\mathcal{S}kills = \{0, \dots, S-1\}$ est l'ensemble des compétences ($S \in \mathbb{N}\backslash\{0\}$).
- $\mathcal{D}ays = \{0, \dots, D-1\}$ est l'ensemble des jours de travail (horizon $D \in \mathbb{N}\backslash\{0\}$).

Paramètres relatifs aux projets :

- $\forall j \in \mathcal{J}obs, \quad Gains[j] \in \mathbb{N}$ est le gain obtenu si le projet j est réalisé.
- $\forall j \in \mathcal{J}obs, \quad Penalities[j] \in \mathbb{N}$ est la pénalité par jour de retard du projet j.
- $\forall j \in \mathcal{J}obs, \quad Due\_Dates[j] \in \mathcal{D}ays$ est la date de livraison du projet j négociée avec le client.
- $\forall (j,s) \in \mathcal{J}obs \times \mathcal{S}kills, \quad Work\_Days[j,s] \in \mathbb{N}$ est pour une compétence donnée s, le nombre de jours de travail par personne nécessaires pour que le projet j soit réalisé.

Paramètres relatifs au personnel :

- $\forall (w,s) \in \mathcal{W}orkers \times \mathcal{S}kills, \quad Qualifications[w,s] \in \{0,1\}$ vaut 1 si le travailleur w possède la compétence s, 0 sinon.
- $\forall (w,d) \in \mathcal{W}orkers \times \mathcal{D}ays, \quad Vacations[w,d] \in \{0,1\}$ vaut 1 si le travailleur w est en congé le jour d, 0 sinon.

2) Variables de décision:

Variable modélisant un planning complet :

- $\forall (w,j,s,d) \in \mathcal{W}orkers \times \mathcal{J}obs \times \mathcal{S}kills \times \mathcal{D}ays, \quad Works[w,j,s,d] \in \{0,1\}$ vaut 1 si le collaborateur w travaille sur le projet j, le jour d, avec la compétence s, 0 sinon.

Variable pour calculer le gain total :

- $\forall j \in \mathcal{J}obs, \quad Is\_Realized[j] \in \{0,1\}$ vaut 1 si le projet j est réalisé, 0 sinon.

Variables pour calculer la durée maximale (ainsi que les pénalités) :

- $\forall (j,d) \in \mathcal{J}obs \times \mathcal{D}ays, \quad Started\_After[j,d] \in \{0,1\}$ vaut 1 à partir du jour où le travail est commencé pour un projet donné j, 0 sinon.
- $\forall (j,d) \in \mathcal{J}obs \times \mathcal{D}ays, \quad Finished\_Before[j,d] \in \{0,1\}$ vaut 1 à partir du jour où le travail est fini pour un projet donné j, 0 sinon.
- $Max\_Duration \in \mathbb{N}$ représente la durée maximale tout projet confondu.

Variables pour calculer le nombre maximal de projets auxquels un quelconque collaborateur est affecté :

- $\forall (w,j) \in \mathcal{W}orkers \times \mathcal{J}obs, \quad Is\_Assigned[w,j] \in \{0,1\}$ vaut 1 si le collaborateur w est affecté au projet j, 0 sinon.
- $Max\_Assigned \in \mathbb{N}$ représente le nombre maximal de projets auxquels un quelconque collaborateur est affecté.

3) Contraintes:

Contraintes liées au planning :

- Contrainte de qualification du personnel : $\forall(w, j, s, d) \in \mathcal{W}orkers \times \mathcal{J}obs \times \mathcal{S}kills \times \mathcal{D}ays$,

$$Works[w, j, s, d] \leq Qualifications[w, s]$$

- Contrainte d'unicité de l'affectation quotidienne du personnel : *réalisée par la contrainte de congé*
- Contrainte de congé : $\forall(w, d) \in \mathcal{W}orkers \times \mathcal{D}ays$,

$$\sum_{(j,s)\in\mathcal{J}obs\times\mathcal{S}kills} Works[w, j, s, d] \leq 1 - Vacations[w, d]$$

- Contrainte de couverture des qualifications du projet : $\forall(j, s) \in \mathcal{J}obs \times \mathcal{S}kills$,

$$\sum_{(w,d)\in\mathcal{W}orkers\times\mathcal{D}ays} Works[w, j, s, d] = Is\_Realized[j] \times Work\_Days[j, s]$$

- Contrainte d'unicité de la réalisation d'un projet : *réalisée par définition de* $Is\_Realized[j] \in \{0, 1\}$

Contraintes liées à la définition des variables de décision :

- $Is\_Realized$: *réalisée par la contrainte de couverture des qualifications du projet*
- $Started\_After$ / $Finished\_Before$: afin de mieux comprendre ce que représente ces variables et en donner plus facilement les contraintes, voici un exemple fictif des valeurs attendues (en fonction de si un quelconque collaborateur travaille sur le projet) :

Indice jour	0	1	2	3	4	5	6	7	8
Jour travaillé	0	0	1	1	0	1	0	0	0
$Started\_After$	0	0	1	1	1	1	1	1	1
$Finished\_Before$	0	0	0	0	0	0	1	1	1

  - $\forall(w, j, s, d) \in \mathcal{W}orkers \times \mathcal{J}obs \times \mathcal{S}kills \times \mathcal{D}ays$,

$$Works[w, j, s, d] \leq Started\_After[j, d]$$

  - $\forall(j, d) \in \mathcal{J}obs \times \mathcal{D}ays \backslash \{D - 1\}$,

$$Started\_After[j, d] \leq Started\_After[j, d + 1]$$

  - $\forall(j, d) \in \mathcal{J}obs \times \mathcal{D}ays$,

$$1 - Started\_After[j, d] \leq Is\_Realized[j]$$

    (pour des raisons de praticité, on fixe $Started\_After$ à 1 lorsque le projet n'est pas réalisé)

  - $\forall(w, j, s, d) \in \mathcal{W}orkers \times \mathcal{J}obs \times \mathcal{S}kills \times \mathcal{D}ays$,

$$Works[w, j, s, d] \leq 1 - Finished\_Before[j, d]$$

  - $\forall(j, d) \in \mathcal{J}obs \times \mathcal{D}ays \backslash \{D - 1\}$,

$$Finished\_Before[j, d] \leq Finished\_Before[j, d + 1]$$

  - $\forall(j, d) \in \mathcal{J}obs \times \mathcal{D}ays$,

$$1 - Finished\_Before[j, d] \leq Is\_Realized[j]$$

    (pour des raisons de praticité, on fixe $Finished\_Before$ à 1 lorsque le projet n'est pas réalisé)

- $Max\_Duration$: $\forall j \in \mathcal{J}obs$,

$$\sum_{d\in\mathcal{D}ays} Started\_After[j, d] - Finished\_Before[j, d] \leq Max\_Duration$$

- $Is\_Assigned$:
  - $\forall(w, j, s, d) \in \mathcal{W}orkers \times \mathcal{J}obs \times \mathcal{S}kills \times \mathcal{D}ays$,

$$Works[w, j, s, d] \leq Is\_Assigned[w, j]$$

– $\forall (w, j) \in \mathcal{W}orkers \times \mathcal{J}obs$,

$$Is\_Assigned[w, j] \leq \sum_{(s,d) \in \mathcal{S}kills \times \mathcal{D}ays} Works[w, j, s, d]]$$

- $Max\_Assigned : \forall w \in \mathcal{W}orkers$,

$$\sum_{j \in \mathcal{J}obs} Is\_Assigned[w, j] \leq Max\_Assigned$$

4) Objectifs:

L'objectif principal est de maximiser le résultat financier de l'entreprise :

$$\max f_1 = \sum_{j \in \mathcal{J}obs} \left(Gains[j] \times Is\_Realized[j] - Penalities[j] \times \sum_{d \in \{Due\_Dates[j]+1, \ldots, D-1\}} (1 - Finished\_Before) \right)$$

Comme objectif secondaire, on veut minimiser le nombre de projets sur lesquels un quelconque collaborateur est affecté :

$$\min f_2 = Max\_Assigned$$

Enfin, un autre objectif secondaire est de minimiser la durée du projet le plus long :

$$\min f_3 = Max\_Duration$$

B. Modèle de préférence

Dans la partie modélisation du problème d'instance de planning, nous avons donné un certain nombre de contraintes et d'objectifs afin de calculer un ensemble de plannings possibles qui correspond à l'ensemble des solutions non-dominées du problème multicritère. Afin de choisir un planning en particulier, il est nécessaire de mettre en place un modèle de préférence. Ce dernier fonctionne de la manière suivante : on lui donne en entrée des exemples de plannings inacceptables, corrects et satisfaisants afin de calibrer le modèle, puis en sortie on obtient un scoring des solutions non-dominés afin de les classer.

Dans la suite de cette partie, on supposera que les métriques (f_1, f_2, f_3) sont normalisées entre 0 et 1, et sont toutes les trois à maximiser. Pour ce faire, on pose :

$$f_1 \leftarrow f_1 \; / \sum_{j \in \mathcal{J}obs} Gains[j]$$

$$f_2 \leftarrow 1 - f_2/J$$

$$f_3 \leftarrow 1 - f_3/D$$

Nous avons fait le choix de scorer les solutions avec une somme pondérée des trois critères. L'attribution d'une solution à une catégorie (inacceptable, correct ou satisfaisant) se fait en fonction du score obtenu et de deux seuils. Ainsi, les exemples donnés en entrée servent à trouver les poids de la pondération ainsi que les seuils. Ce calcul se fait par la résolution d'un problème d'optimisation linéaire.

1) Objectifs:

- $\mathcal{ND} = \{(f_1, f_2, f_3), (f'_1, f'_2, f'_3), \ldots\} \in ([0, 1]^3)^N$ est l'ensemble des N solutions non-dominées du problème d'instance de planning.
- $\mathcal{I} \subset \mathcal{ND}$ l'ensemble des solutions inacceptables.
- $\mathcal{C} \subset \mathcal{ND}$ l'ensemble des solutions correctes.
- $\mathcal{S} \subset \mathcal{ND}$ l'ensemble des solutions satisfaisantes.

2) Variables de décision:

- $(\omega_1, \omega_2, \omega_3) \in [0,1]^3$ sont les poids des trois critères.
- $(\theta_1, \theta_2) \in [0,1]^2$ sont les seuils des catégories.
- $\epsilon \in [0,1]$ représente la marge de séparation des catégories : on souhaite que tous les exemples d'une catégorie soient à ϵ près du seuil, avec epsilon le plus grand possible afin que les catégories soient le plus séparées possible.

3) Contraintes:

- Normalisation :
$$\omega_1 + \omega_2 + \omega_3 = 1$$
- Catégorie inacceptable : $\forall (f_1, f_2, f_3) \in \mathcal{I}$,
$$f_1\omega_1 + f_2\omega_2 + f_3\omega_3 \leq \theta_1 - \epsilon$$
- Catégorie correcte :
  - $\forall (f_1, f_2, f_3) \in \mathcal{C}$,
  $$f_1\omega_1 + f_2\omega_2 + f_3\omega_3 \leq \theta_2 - \epsilon$$
  - $\forall (f_1, f_2, f_3) \in \mathcal{C}$,
  $$f_1\omega_1 + f_2\omega_2 + f_3\omega_3 \geq \theta_1 + \epsilon$$
- Catégorie satisfaisante : $\forall (f_1, f_2, f_3) \in \mathcal{S}$,
$$f_1\omega_1 + f_2\omega_2 + f_3\omega_3 \geq \theta_2 + \epsilon$$

4) Objectifs:

Comme objectif de ce problème d'optimisation, nous avons choisi de séparer au mieux les trois catégories :

$$\max \epsilon$$

III. Résultats

A. Exemples de résultats

Afin de vérifier la cohérence de notre travail, nous avons implémenté des fonctions qui permettent d'afficher la solution trouvée. Ainsi, nous avons lancé l'optimisation sur les instances `toy_instance`, `medium_instance` et `large_instance`, avec comme critère, de maximiser d'abord le résultat financier f_1, puis de minimiser le nombre maximal d'affectation f_2 puis la durée maximale f_3 comme seconds objectifs. Les résultats obtenus sont les suivants : voir figures 1, 2 et 3.

	qualifications	vacations
Olivia	[A, B, C]	[]
Liam	[A, B]	[0]
Emma	[C]	[1]

	A	B	C	due date	gain	penalty
Job1	1	1	1	3	20	3
Job2	1	2	0	3	15	3
Job3	1	0	2	4	15	3
Job4	0	2	1	3	20	3
Job5	0	0	2	5	10	3

	0	1	2	3	4
Olivia	B	C	B	C	C
Liam	None	A	B	A	None
Emma	C	None	C	C	None

Fig. 1: Résultats de l'optimisation f_1 puis f_2 puis f_3 sur `toy_instance`.
Objectifs obtenus : $f_1 = 65$, $f_2 = 2$ et $f_3 = 3$.

	qualifications	vacations
Olivia	[A, C, B]	[0, 1]
Liam	[A, D, E]	[0, 1]
Emma	[H, B]	[7, 8]
Noah	[G, D, J, C, H, I]	[]
Amelia	[F, G, J, E]	[14, 15]

	F	G	A	D	J	C	H	E	B	I	due date	gain	penalty
Job1	0	0	4	0	0	0	0	0	4	0	20	15	3
Job2	0	0	4	4	0	1	0	0	2	0	16	30	3
Job3	0	0	4	0	0	1	0	0	0	0	12	30	3
Job4	0	0	6	0	0	0	0	0	0	0	18	30	3
Job5	0	0	0	8	0	4	0	0	3	0	10	70	3
Job6	0	0	0	4	0	0	0	5	0	0	16	40	3
Job7	1	0	0	5	0	0	0	5	0	0	20	20	3
Job8	6	0	0	0	0	0	0	3	0	0	22	10	3
Job9	0	3	0	0	0	0	2	4	0	0	18	25	3
Job10	6	0	0	0	0	0	0	0	0	0	18	20	3
Job11	6	2	0	0	0	0	1	0	0	0	18	25	3
Job12	0	0	0	0	3	0	6	0	0	0	12	45	3
Job13	0	0	0	0	8	0	0	0	0	4	14	40	3
Job14	0	4	0	0	4	0	0	0	0	0	12	60	3
Job15	0	4	0	0	0	0	4	0	0	0	16	50	3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Olivia	None	None	C	C	C	A	A	C	C	A	A	A	A	A	A	A	A	A	A	A	A	None
Liam	None	None	D	D	D	D	D	D	D	D	E	D	D	E	D	D	E	E	E	D	E	E
Emma	B	B	B	H	H	H	H	None	None	H	H	B	H	H	H	H	H	H	B	B	B	None
Noah	I	I	J	J	G	I	I	J	J	G	G	G	G	G	G	G	D	G	G	D	D	D
Amelia	J	J	J	J	J	J	J	J	J	J	J	E	E	E	None	None	E	G	F	E	E	E

Fig. 2: Résultats de l'optimisation f_1 puis f_2 puis f_3 sur `medium_instance`.
Objectifs obtenus : $f_1 = 413$, $f_2 = 5$ et $f_3 = 10$.

	E	I	J	A	C	H	G	D	B	F	due date	gain	penalty
Job1	0	0	0	4	0	0	0	0	4	0	26	15	3
Job2	0	0	0	4	1	0	0	4	2	0	22	30	3
Job3	0	0	0	4	1	0	0	0	0	0	16	30	3
Job4	0	0	0	6	0	0	0	0	0	0	18	30	3
Job6	0	0	0	0	4	0	0	8	3	0	14	70	3
Job7	4	0	0	0	0	7	0	0	3	0	32	50	3
Job8	0	4	0	0	0	0	0	0	3	0	18	50	3
Job9	0	0	0	0	5	0	8	5	0	0	30	80	3
Job10	0	0	0	0	7	8	0	0	0	5	20	80	3
Job11	0	5	0	0	4	0	0	0	0	0	32	20	3
Job12	0	0	4	0	8	0	0	0	0	0	32	20	3
Job13	5	0	0	0	0	0	0	4	0	0	26	40	3
Job14	5	0	0	0	0	0	0	5	0	1	30	20	3
Job15	0	0	3	0	0	0	5	5	0	0	26	20	3
Job16	0	0	2	0	0	5	0	5	0	0	36	20	3
Job17	3	8	0	0	0	0	0	0	0	6	18	80	3
Job18	4	0	0	0	0	2	3	0	0	0	18	25	3
Job20	0	0	0	0	0	0	0	0	0	6	26	20	3
Job21	0	0	0	0	0	1	2	0	0	6	18	25	3
Job22	0	0	0	0	0	4	4	0	0	0	12	60	3
Job23	0	6	4	0	0	0	4	0	0	0	28	50	3
Job24	0	0	4	0	0	0	4	0	0	0	26	50	3
Job25	0	3	0	0	0	6	0	0	0	0	12	45	3
Job27	0	4	8	0	0	0	0	0	0	0	14	40	3
Job28	0	0	10	0	0	0	0	0	0	0	14	50	3

	qualifications	vacations
Olivia	[A, C, B]	[0, 1, 2, 3, 4]
Liam	[E, A, D]	[5, 6, 7, 8, 9]
Emma	[H, B]	[20, 21, 22, 23, 24]
Noah	[C, H, G, D]	[25, 26, 27, 28, 29]
Amelia	[E, I, J, G, F]	[]
Oliver	[J, G, F]	[30, 31, 32]

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Olivia	None	None	None	None	None	C	B	B	A	C	C	C	A	A	C	A	C	A	A
Liam	D	D	D	D	E	None	None	None	None	None	A	A	A	E	A	A	E	A	D
Emma	H	H	H	H	B	H	H	H	H	H	H	H	H	H	B	B	H	B	H
Noah	C	D	G	G	D	D	G	D	C	C	H	G	H	D	C	C	C	C	G
Amelia	I	I	I	F	I	I	I	E	I	F	I	I	I	I	I	I	I	I	I
Oliver	J	J	J	J	J	J	J	J	J	F	J	F	F	F	F	F	F	J	F

19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
B	A	B	A	C	B	C	B	C	C	C	C	B	C	C	C	None
E	D	D	E	E	E	D	E	E	D	E	D	E	D	D	D	D
H	None	None	None	None	None	H	H	H	H	H	H	H	H	H	H	H
D	D	G	D	G	G	None	None	None	None	None	D	D	G	D	D	H
I	G	G	J	I	I	I	I	I	I	I	I	I	G	G	G	J
F	J	J	J	J	G	J	J	G	G	G	None	None	None	G	G	J

Fig. 3: Résultats de l'optimisation f_1 puis f_2 puis f_3 sur `large_instance`.
Objectifs obtenus : $f_1 = 817$, $f_2 = 7$ et $f_3 = 11$.

En outre, nous avons implémenté un générateur aléatoire d'instances. Les détails de son fonctionnement sont fournis en annexe. Les résultats sont les suivants : voir figure 4.

	qualifications	vacations
Cpb	[W, D]	[]
Bdcxqtylwd	[W, D, E, F, H]	[]
Huuz	[H]	[]
Syisntxbtg	[D, E, F, H]	[0]

	W	D	E	F	H	due date	gain	penalty
Job1	0	2	1	0	0	5	40	3
Job2	0	0	0	1	0	4	23	3
Job3	1	0	9	0	0	5	52	3
Job4	0	2	0	0	3	5	31	3
Job5	0	0	7	0	2	5	28	3

	0	1	2	3	4
Cpb	D	D	None	D	D
Bdcxqtylwd	F	E	E	E	E
Huuz	H	H	H	H	H
Syisntxbtg	None	E	E	E	E

Fig. 4: Résultats de l'optimisation f_1 puis f_2 puis f_3 sur `random_instance` (issu du générateur aléatoire d'instances). Objectifs obtenus : $f_1 = 122$, $f_2 = 2$ et $f_3 = 5$.

B. Surface des solutions non dominées

La méthode des ε-contraintes a été adaptée pour un problème tri-objectif. La première étape est de redéfinir l'objectif en sommant à l'objectif principal les deux objectifs secondaires, pondérés par ϵ_2 et ϵ_3, tels que $1 \gg \epsilon_2 > \epsilon_3$.

$$f = f_1 - \epsilon_2 f_2 - \epsilon_3 f_3 \tag{1}$$

En pratique, nous avons choisi $\epsilon_2 = 0.005$ et $\epsilon_3 = 0.001$. Il faut ensuite rajouter des contraintes sur les deux objectifs secondaires et itérativement décrémenter les valeurs de ces contraintes de sorte à calculer toute la surface. Étant donné que les valeurs que peuvent prendre nos deux objectifs sont entières et bornées, les contraintes sont initialisées à leur borne supérieure respective, puis décrémentées avec deux boucles imbriquées en récupérant les valeurs de la solution précédente, moins un. Pour trouver la valeur de la contrainte de la boucle exterieur, on prend la valeur maximum que cet objectif a atteint dans les solutions du tour de boucle précédent. Le pseudo-code est lisible ci-dessous.

```
ε_duration = horizon
ε_assigned = len(jobs)
while ε_duration ≥ 0 :
    add constraint(max duration <= ε_duration)
    next_ε_duration = 0
    while ε_assigned ≥ 0 :
        add constraint(max assigned <= ε_assigned)
        solution ← model.optimize()
        save(solution)
        ε_assigned = solution's assigned - 1
        next_ε_duration = max(next_ε_duration, solution's assigned)
        remove previous constraint on assigned
    ε_duration = next_ε_duration - 1
    remove previous constraint on duration
```

Une fois l'ensemble des solutions non-dominées obtenu, nous pouvons visualiser la surface associée (voir figures 5 et 6). En raison du temps de calcul extrêmement long, nous n'avons pas pu obtenir les résultats sur le jeu de données **large_instance**.

C. Optimisation du modèle de préférence

Afin de tester le modèle de préférence, nous avons choisi 6 solutions non-dominées obtenues avec `toy_instance`, que nous avons séparées en inacceptable, correct et satisfaisant. Le choix de ces instances ainsi que le score obtenu après calcul du modèle est présenté figure 7. Pour cet exemple, nous avons considéré comme inacceptable une solution avec un trop faible résultat financier ou un trop grand nombre maximum d'affectation. Pour la séparation entre correct et satisfaisant, nous avons privilégié le gain. Ces choix arbitraires se traduisent par un faible coefficient pour la durée maximale, et un coefficient légérement plus important pour le gain. Ainsi, les coefficients obtenus sont cohérents avec les préférences données en entrée, ce qui appuie la pertinence de notre modèle de préférence.

IV. Conclusion

Pour conclure, nous avons donc réalisé un outil d'optimisation pour le problème posé par la société CompuOpti. Nous avons d'abord modélisé le problème sous forme de problème d'optimisation linéaire sous contraintes, puis déterminé les solutions non-dominées sur les exemples donnés, et enfin nous avons établi un modèle de préférences pour classifier ces solutions.

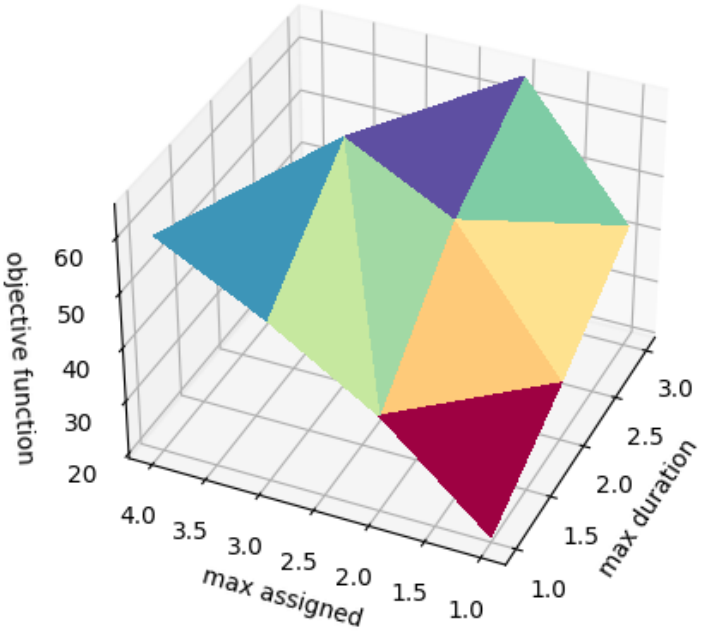

Fig. 5: Surface des solutions non-dominées sur `toy_instance`.

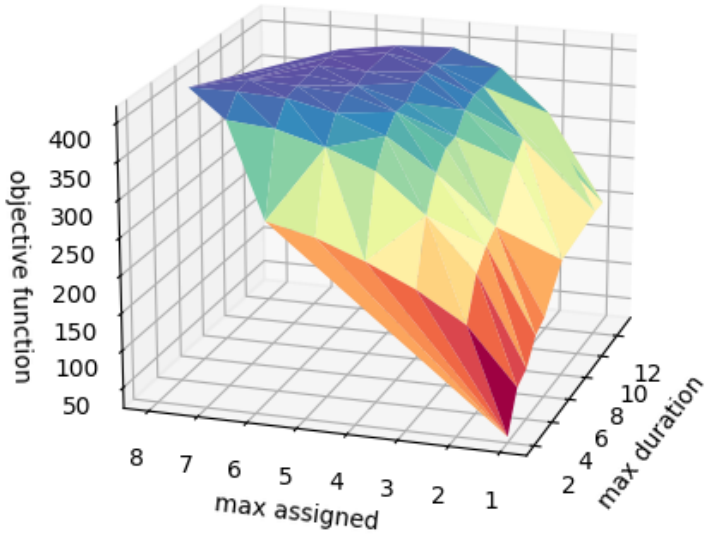

Fig. 6: Surface des solutions non-dominées sur `medium_instance`.

	objVal	max_assigned	max_duration	status
0	59	4	1	unacceptable
1	20	1	1	unacceptable
2	42	1	3	correct
3	49	3	1	correct
4	65	2	3	satisfactory
5	55	2	2	satisfactory

(a) Labélisation de modèles donnée en entrée

	objVal	max_assigned	max_duration	score	status
0	64.987000	2	3	0.687419	satisfactory
3	54.988000	2	2	0.643675	satisfactory
1	41.992000	1	3	0.624950	correct
2	64.983001	3	2	0.624894	correct
4	29.993000	1	2	0.568706	correct
7	36.989000	2	1	0.549931	correct
6	48.984000	3	1	0.543650	correct
8	19.994000	1	1	0.524963	unacceptable
5	58.979000	4	1	0.524869	unacceptable

(b) Scoring obtenu en sortie

Fig. 7: Exemple de résultat du modèle de préférence obtenu sur `toy_instance`.
Poids obtenus : $\omega_1 = 0.50$, $\omega_2 = 0.41$ et $\omega_3 = 0.09$. Seuils obtenus : $\theta_1 = 0.53$ et $\theta_2 = 0.63$.

ANNEXE : CRÉATION D'INSTANCES ALÉATOIRES

Le code comporte une section pour créer des instances aléatoirement. La fonction peut prendre en entrée un horizon, un nombre de personnes, de jobs et de skills, pour permettre à l'utilisateur·ice de contrôler la taille de l'instance. Si rien n'est renseigné l'horizon est choisi grâce à une loi exponentielle privilégiant les horizons courts, et les autres variables sont tirées aléatoirement entre 1 et 10. Les qualifications des personnes sont tirées aléatoirement parmi les compétences disponibles avec une loi uniforme. Pour la création des jobs, le gain est tiré uniformément entre 10 et 80, puis le nombre total de travail requis pour valider le job est déduit avec une loi identifiée empiriquement sur les trois instances fournies en exemple (Figure 8). Le nombre de compétences sur lesquelles répartir ce temps de travail est également obtenu avec une loi empirique (Figure 9).

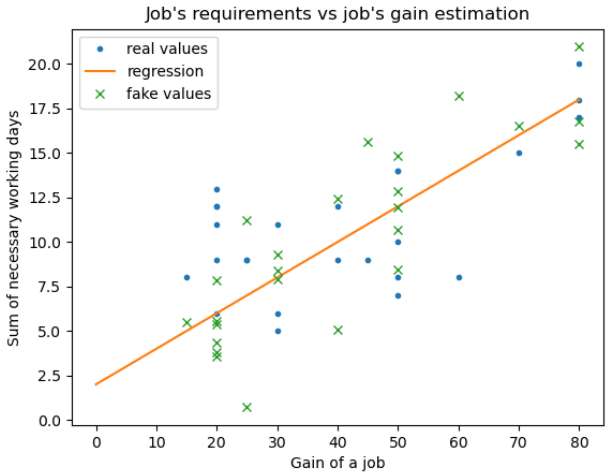

Fig. 8: Évolution du nombre de jours de travail nécessaires cumulés d'un job en fonction de son gain et son approximation linéaire.

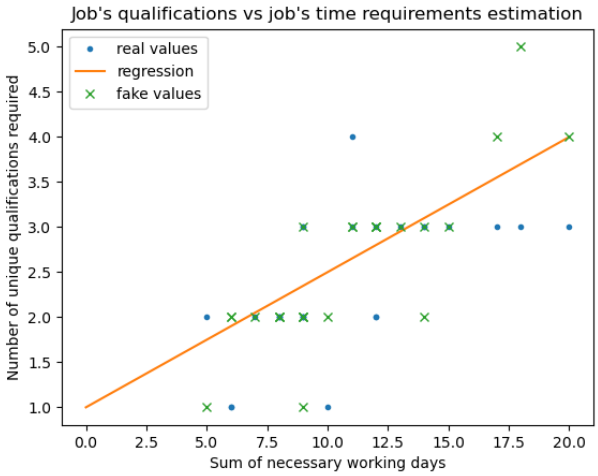

Fig. 9: Évolution du nombre de compétences requises pour un job en fonction de son nombre de jours de travail cumulés.