Faculté des sciences et génie
Département d'informatique et de génie logiciel

IFT-7020 - Optimisation Combinatoire

Automne 2024

RAPPORT DE PROJET : TRAVAIL 1

Bourez Nicolas — **IDUL**: 537315033
Onciul Alexandra — 537305977
Miralles Alexandre — 537316065

Professeur:
Quimper Claude-Guy

June, 2025

1 Premier problème

1.1 Résumé du problème

L'objectif de ce premier problème est de résoudre le jeu des 4 cubes. Les faces de ces cubes peuvent être colorées en jaune (1), bleu (2), rouge (3) et vert (4). Les patrons des cubes sont donnés dans le problème, c'est-à-dire que chaque face d'un cube a une couleur attribuée.

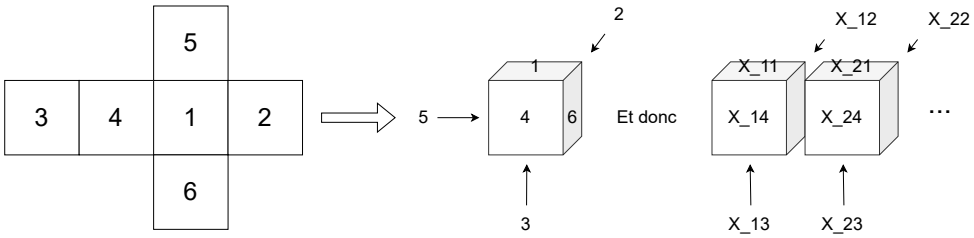
Le but final est d'aligner ces 4 cubes de telle sorte qu'ils forment un prisme dont les 4 couleurs sont représentées sur les 4 faces rectangulaires de ce prisme.

1.2 Modélisation du problème

- **Variables**
  Nous avons choisi de modéliser notre problème à l'aide de 16 variables. Chacune des variables correspondant à l'une des faces visible des cubes. Plus précisement, soit les variables:

$$X_{ij} \quad \forall i, j \in \{1, ..., 4\}$$

  la face X_{ij} représente la j-ème face (visible) du i-ème cube selon la norme (choisie) suivante[1]:



  La couleur, donc la valeur de la variable, sera représentée par un chiffre selon les attributions suivantes:

$$\text{couleurs} = \{1\text{:jaune, } 2\text{:rouge, } 3\text{:bleu, } 4\text{:vert}\}$$

  Chaque variable aura pour domaine:

$$dom(X_{ij}) = \{1, 2, 3, 4\}$$

  On a donc $4 * 4 * 4 = 64$ comme taille de domaine

- **Contraintes**

  Le problème posé peut se traduire en contraintes par le fait que chaque $k-$ème face des cubes doit être d'une couleur différente des $k-$ème face des 3 autres cubes:

$$X_{ik} \neq X_{jk} \qquad \forall i, j, k \in \{1, ..., 4\} \text{ et } i \neq j$$

  On a en tout $4 * (4!)$ contraintes ce qui nous donne un total de 24 contraintes. Cependant, si on laisse le problème sous cette forme, le solveur va nous donner une solution plausible mais qui ne correspondra probablement pas aux patrons donnés dans l'énoncé (sauf coup de chance). On doit donc modéliser le fait que les faces ne peuvent pas prendre n'importe quelle couleur mais qu'elles doivent se limiter aux possibilités que nous offre les patrons.

  Pour chaque branchement d'une variable à une valeur, on doit vérifier:

  – si on respecte les contraintes des patrons des cubes

  – si oui, alors comment ce branchement va réduire les domaines des autres variables

[1] les faces 5 et 6 sont simplement notifiées par soucis de compréhension mais elles n'interviennent pas dans notre modèle

Pour modéliser cela, on va attribuer à chaque **première** face une contrainte pour chaque branchement possible (1,2,3 ou 4). Pour la première face on aura:

$$
\begin{aligned}
X_{11}=1 \implies & (X_{13}=4 \wedge ((X_{12}=4 \wedge X_{14}=2) \vee (X_{12}=2 \wedge X_{14}=3) \\
& \vee (X_{12}=2 \wedge X_{14}=4) \vee (X_{12}=3 \wedge X_{14}=2))) \\
X_{11}=2 \implies & (X_{13}=4 \wedge ((X_{14}=1 \wedge X_{12}=4) \vee (X_{14}=4 \wedge X_{12}=1) \\
& \vee (X_{14}=2 \wedge X_{12}=3) \vee (X_{14}=3 \wedge X_{12}=2))) \vee \\
& (X_{13}=3 \wedge ((X_{14}=2 \wedge X_{12}=4) \vee (X_{14}=4 \wedge X_{12}=2) \\
& \vee (X_{14}=1 \wedge X_{12}=4) \vee (X_{14}=4 \wedge X_{12}=1))) \\
X_{11}=3 \implies & (X_{13}=2 \wedge ((X_{14}=4 \wedge X_{12}=2) \vee (X_{14}=2 \wedge X_{12}=4) \\
& \vee (X_{14}=1 \wedge X_{12}=4) \vee (X_{14}=4 \wedge X_{12}=1))) \\
X_{11}=4 \implies & (X_{13}=1 \wedge ((X_{12}=4 \wedge X_{14}=2) \vee (X_{12}=2 \wedge X_{14}=3) \\
& \vee (X_{12}=2 \wedge X_{14}=4) \vee (X_{12}=3 \wedge X_{14}=2))) \vee \\
& (X_{13}=2 \wedge ((X_{14}=4 \wedge X_{12}=2) \vee (X_{14}=2 \wedge X_{12}=4) \\
& \vee (X_{14}=1 \wedge X_{12}=4) \vee (X_{14}=4 \wedge X_{12}=1)))
\end{aligned}
$$

La première contrainte traduit le fait que si on branche la face X_{11} sur la couleur jaune, alors la face opposée sera verte (d'office)(en bleu) tandis que les 2 autres faces pourront effectuer une rotation de 90° en gardant toujours la même couleur en face d'elle. Cette rotation se traduit par les 4 disjonctions (en rouge). Il nous faut également les contraintes pour les faces X_{21}, X_{31} et X_{41}:

$$
\begin{aligned}
X_{21}=1 \implies & (X_{23}=3 \wedge ((X_{22}=4 \wedge X_{24}=4) \vee (X_{22}=2 \wedge X_{24}=3) \\
& \vee (X_{22}=3 \wedge X_{24}=2))) \\
X_{21}=2 \implies & (X_{23}=3 \wedge ((X_{22}=4 \wedge X_{24}=4) \vee (X_{22}=1 \wedge X_{24}=3) \\
& \vee (X_{22}=3 \wedge X_{24}=1))) \\
X_{21}=3 \implies & (X_{23}=2 \wedge ((X_{22}=4 \wedge X_{24}=4) \vee (X_{22}=1 \wedge X_{24}=3) \\
& \vee (X_{22}=3 \wedge X_{24}=1))) \vee \\
& (X_{23}=1 \wedge ((X_{22}=4 \wedge X_{24}=4) \vee (X_{22}=2 \wedge X_{24}=3) \\
& \vee (X_{22}=3 \wedge X_{24}=2))) \\
X_{21}=4 \implies & (X_{23}=4 \wedge ((X_{22}=2 \wedge X_{24}=3) \vee (X_{22}=3 \wedge X_{24}=2) \\
& \vee (X_{22}=1 \wedge X_{24}=3) \vee (X_{22}=3 \wedge X_{24}=1)))
\end{aligned}
$$

$$
\begin{aligned}
X_{31}=1 \implies & (X_{33}=4 \wedge ((X_{32}=1 \wedge X_{34}=1) \vee (X_{32}=3 \wedge X_{34}=2) \\
& \vee (X_{32}=2 \wedge X_{34}=3))) \vee \\
& (X_{33}=1 \wedge ((X_{32}=1 \wedge X_{34}=4) \vee (X_{32}=4 \wedge X_{34}=1) \\
& \vee (X_{32}=2 \wedge X_{34}=3) \vee (X_{32}=3 \wedge X_{34}=2))) \\
X_{31}=2 \implies & (X_{33}=3 \wedge ((X_{32}=1 \wedge X_{34}=1) \vee (X_{32}=1 \wedge X_{34}=4) \\
& \vee (X_{32}=4 \wedge X_{34}=1))) \\
X_{31}=3 \implies & (X_{33}=2 \wedge ((X_{32}=1 \wedge X_{34}=1) \vee (X_{32}=1 \wedge X_{34}=4) \\
& \vee (X_{32}=4 \wedge X_{34}=1))) \\
X_{31}=4 \implies & (X_{33}=1 \wedge ((X_{32}=1 \wedge X_{34}=1) \vee (X_{32}=2 \wedge X_{34}=3) \\
& \vee (X_{32}=3 \wedge X_{34}=2)))
\end{aligned}
$$

$$
\begin{aligned}
X_{41}=1 \implies & (X_{43}=2 \wedge ((X_{42}=1 \wedge X_{44}=3) \vee (X_{42}=3 \wedge X_{44}=1) \\
& \vee (X_{42}=2 \wedge X_{44}=4) \vee (X_{42}=4 \wedge X_{44}=2))) \vee \\
& (X_{43}=3 \wedge ((X_{42}=1 \wedge X_{44}=2) \vee (X_{42}=2 \wedge X_{44}=1) \\
& \vee (X_{42}=2 \wedge X_{44}=4) \vee (X_{42}=4 \wedge X_{44}=2))) \\
X_{41}=2 \implies & (X_{43}=4 \wedge ((X_{42}=1 \wedge X_{44}=3) \vee (X_{42}=3 \wedge X_{44}=1) \\
& \vee (X_{42}=1 \wedge X_{44}=2) \vee (X_{42}=2 \wedge X_{44}=1))) \vee \\
& (X_{43}=1 \wedge ((X_{42}=1 \wedge X_{44}=3) \vee (X_{42}=3 \wedge X_{44}=1) \\
& \vee (X_{42}=2 \wedge X_{44}=4) \vee (X_{42}=4 \wedge X_{44}=2))) \\
X_{41}=3 \implies & (X_{43}=1 \wedge ((X_{42}=1 \wedge X_{44}=2) \vee (X_{42}=2 \wedge X_{44}=1) \\
& \vee (X_{42}=2 \wedge X_{44}=4) \vee (X_{42}=4 \wedge X_{44}=2))) \\
X_{41}=4 \implies & (X_{43}=2 \wedge ((X_{42}=1 \wedge X_{44}=3) \vee (X_{42}=3 \wedge X_{44}=1) \\
& \vee (X_{42}=1 \wedge X_{44}=2) \vee (X_{42}=2 \wedge X_{44}=1)))
\end{aligned}
$$

Cela nous fait 16 contraintes supplémentaires, ce qui nous donne un total de 40 contraintes.

1.3 Results

Voici les différents résultats obtenus en utilisant les solveurs Geocode (6.3.0*) et Chuffed (0.13.2*):

Solveur	Geocode	Chuffed

Output	`Running untitled_model.mzn`
	`face11 = 1;`
	`face12 = 4;`
	`face13 = 4;`
	`face14 = 2;`
	`face21 = 3;`
	`face22 = 2;`
	`face23 = 1;`
	`face24 = 3;`
	`face31 = 2;`
	`face32 = 1;`
	`face33 = 3;`
	`face34 = 4;`
	`face41 = 4;`
	`face42 = 3;`
	`face43 = 2;`
	`face44 = 1;`
	`----------`
	`Finished in 93msec.`
		`Running untitled_model.mzn`
		`face11 = 4;`
		`face12 = 4;`
		`face13 = 2;`
		`face14 = 1;`
		`face21 = 2;`
		`face22 = 1;`
		`face23 = 3;`
		`face24 = 3;`
		`face31 = 1;`
		`face32 = 3;`
		`face33 = 4;`
		`face34 = 2;`
		`face41 = 3;`
		`face42 = 2;`
		`face43 = 1;`
		`face44 = 4;`
		`----------`
		`Finished in 301msec.`

Nombre de noeuds visités	26	14

Comme on peut facilement le vérifier, les deux solutions sont valides et le temps d'exécution est assez rapide.

2 Deuxième problème

2.1 Résumé du problème

Le problème consiste à organiser des horaires de travail pour plusieurs employés avec différentes activités. Il s'agit d'un problème de minimisation afin d'utliser au mieux les ressources avec des surcouts et souscout qui interviennent quand il y a trop ou trop peu d'employés présents.

2.2 Modélisation du problème

- **Constantes:**
  - $N_e = \mathbb{N}^*$: Nombre d'employés
  - $N_a = \mathbb{N}^*$: Nombre de différentes activités à faire dans la journée
  - $N = 96$: Nombre de périodes de 15 minutes dans une journée
  - $t = \{33, \ldots, 80\}$
- **Paramètres à récuperer dans les bench:**
  - $d_{t,a}$: Demande pour l'activité a à la période t
  - $C_{t,a}$: Surcoût pour l'activité a à la période t
  - $c_{t,a}$: Sous-coût pour l'activité a à la période t

$$\text{dom}(d_{t,a}) = \{1, \ldots, N_e\} \quad \forall t \in \{1, \ldots, N\},\ a \in \{1, \ldots, N_a\}$$

$$\text{dom}(C_{t,a}) = \{1, \ldots, N_e\} \quad \forall t \in \{1, \ldots, N\},\ a \in \{1, \ldots, N_a\}$$

$$\text{dom}(c_{t,a}) = \{1, \ldots, N_e\} \quad \forall t \in \{1, \ldots, N\},\ a \in \{1, \ldots, N_a\}$$

- **Variables à optimiser:**

  Après avoir reformulé le problème de manière concrète et analysé les différentes contraintes, il semble que la journée de travail d'un employé peut être découpée en plusieurs phases, dont l'agencement relatif reste constant.

  En effet, la journée de travail peut être modélisée comme une succession de neuf phases distinctes :

  1. la phase avant le début des activités,
  2. une première période d'activité d'au moins 4 périodes,
  3. une courte pause d'une période,
  4. une deuxième période d'activité d'au moins 4 périodes,
  5. la pause repas de 4 périodes,
  6. une troisième période d'activité,
  7. une seconde courte pause d'une période,
  8. une quatrième période d'activité d'au moins 4 périodes,
  9. la phase suivant la fin des activités.

  Cela permet de tirer une conclusion utile pour la création des variables : un seul employé ne peut faire que quatre activités par jour vu qu'il ne peut pas changer d'activité entre les pauses. Ce qui nous permet d'avoir une matrice $P_{e,p}$ avec N_e lignes et que 4 colonnes. Une ligne est associée à un employé pour laquelle on retrouve 4 valeurs chacune associée à l'activité faite à chaque crénau.

  De plus, cette structure rigide nous permet de devoir trouver uniquement 8 positions par employé dans $P_{e,p}$. Nous avons décidé de mettre d'indiquer les débuts et fin d'activités. C'est à dire un index pour la première période de l'activité et un index pour la dernière période d'une activité.

  Nous avons également remarqué que les journées de travail ne commencent pas avant 8h30 et ne se terminent pas après 20h, donc nous avons limité les valeurs de t de 33 à 80 (de 8h15 à 20h). Nous avons fait ces choix d'implémentation afin de limiter un maximum l'espace de recherche.
  Voici nos variables à optimiser et leur domaines qui suivent:

  - $P_{e,p}$: Planification de l'employé e pour la période p
  - $A_{e,a}$: Activité assignée à l'employé e pour l'activité a
  - $X_{i,a}$: Nombre d'employés assignés à l'activité a à la période i, vous retrouverez ci-dessous la manière dont nous la remplissons:

$$\forall t \in \{33, \ldots, 80\},\ \forall a \in \{1, \ldots, N_a\},$$

$$X_{t,a} = \sum_{e=1}^{N_e} ((P_{e,1} \leq t \leq P_{e,2} \wedge A_{e,1} = a) + (P_{e,3} \leq t \leq P_{e,4} \wedge A_{e,2} = a) +$$

$$(\mathrm{P}_{e,5} \leq t \leq P_{e,6} \wedge A_{e,3} = a) + (P_{e,7} \leq t \leq P_{e,8} \wedge A_{e,4} = a)$$

$$\mathrm{dom}(P_{e,p}) = \{33, \ldots, 80\} \quad \forall e \in \{1, \ldots, N_e\},\ p \in \{1, \ldots, 8\}$$

$$\mathrm{dom}(A_{e,a}) = \{1, \ldots, N_a\} \quad \forall e \in \{1, \ldots, N_e\},\ a \in \{1, \ldots, 4\}$$

$$\mathrm{dom}(X_{i,a}) = \{0, \ldots, N_e\} \quad \forall i \in \{33, \ldots, 80\},\ a \in \{1, \ldots, N_a\}$$

- **Contraintes**

  Soit $i \in \{1, ..., Ne\}$. Alors on a la contrainte :

  $$P_{i,2} - P_{i,1} \geq 3 \wedge P_{i,4} - P_{i,3} \geq 3 \wedge P_{i,6} - P_{i,5} \geq 3 \wedge P_{i,8} - P_{i,7} \geq 3$$

  Qui traduit le fait que lorsqu'un employé commence une activité, il doit y passer au moins une heure (nous répétons la contrainte avec une conjonction pour chaque nouvelle phase d'activité).

  $\forall i \in \{1, ..., Ne\}$:
  $$P_{i,3} = P_{i,2} + 2 \wedge P_{i,5} = P_{i,4} + 5 \wedge P_{i,7} = P_{i,6} + 2$$

  Qui est la contrainte qui définit la durée des pauses (1 créneau pour les pauses courtes et 4 créneaux pour les pauses longues dans cet ordre précis).

  $\forall i \in \{1, ..., Ne\}$:
  $$22 \leq \sum_{k \in \{1,3,5,7\}} (P_{i,k+1} - P_{i,k} + 1) \leq 30$$

  Cette contrainte assure que chaque employé travaille entre 6 et 8 heures par jour avec les petites pauses incluses. Nous devons

  $\forall i \in t$.
  $$\forall a \in \{1, ..., A\} \quad (d_{i,a} = 0 \wedge c_{i,a} = 0 \wedge C_{i,a} = 0) \implies X_{i,a} = 0$$

  Cette contrainte évoque que s'il n'y a ni demande ni sous-coût ni sur-coût pour toutes les activités à un créneau i alors la boutique est fermée.

  $\forall i \in t$ et $a \in \{1, ..., N_a\}$
  $$d_{i,a} > 0 \implies X_{i,a} \geq 1$$

  Cela contraint à avoir au moins un employé travaillant sur cette activité si on a une demande supérieure à 1 sur celle-ci.

  Afin d'optimiser notre recheche, nous avons ajouté une contrainte suplémentaire:

  $$\text{seuil} = 100000$$
  $$\text{Si } c_{t,a} \geq \text{seuil, alors } \begin{cases} x_{t,a} \geq d_{t,a} \\ \sum_{e=1}^{N_e} \sum_{k=1,3,5,7} \left(P_{e,k} \leq t \leq P_{e,k+1} \wedge A_{e,\frac{k+1}{2}} = a \right) \geq d_{t,a} \end{cases}$$
  Sinon, la contrainte est toujours vérifiée.

  Nous obligeons à avoir au moins assez d'employé lorsque le souscout est supérieur à un seuil, nous avons ici choisi 100000. Nous savons que cet agencement fera partie de la solution optimale vu la pénalisation élevée associée.

  Nous avons également ajouté une contrainte de symmetrie étant donné que l'employé assigné ne modifie pas le score seulement le nombre d'employés présents à un certain moment.

  $$\forall i_1, i_2 \in \{1, \ldots, N_e\} \text{ avec } i_1 < i_2, \quad P_{i_1,1} \leq P_{i_2,1}$$

  Cette dernière impose que la première période de planification de l'employé i_1 soit inférieure ou égale à celle de l'employé i_2 afin de ne pas considérer 2 fois la même solution juste répartie differemment parmis les employés.

2.3 Fonction objectif

Ces contraintes nous amènent à définir la fonction objectif qui est le calcul de la pénalité totale que l'on doit minimiser, et dans ce cas là on aura la solution optimale :

$$\sum_{i=33}^{80} \sum_{a=1}^{A} (C_{i,a}(X_{i,a} - d_{i,a}) + c_{i,a}(d_{i,a} - X_{i,a}))$$

2.4 Heuristique et solveur

Après de multiples essais avec différents solveurs et différentes heuristiques de recherche, nous utilisons le solveur OrTools avec 8 threads qui offre le meilleur temps d'execution. L'heuristique de recherche que nous avons choisi est celle par défaut de MiniZinc. Nous avons activé le free search qui semble améliorer notre temps d'execution.Malheureusment ORTools ne fournit pas le nombre de noeuds visités mais la propagation et les conflits peuvent nous donner une idée de l'espace de recherche visité. Nos résultats vont donc décrire les résultats avec Highs également pour les 2 premières instances.

Nous avons également ajouté des contraintes sur les domaines possibles de périodes et fixé certains employés à certaines périodes en raison de leur cout momumental afin de réduire l'espace de recherche comme mentionné précédemment.

2.5 Résultats

Pour rappel, le P de chaque employé décrit les premières et dernières périodes de chaque bloc d'activité $(2 * 4 = 8)$ et le A de chaque employé correspondant à chaque activité associée à chaque créneau d'activité.

2.5.1 Bench 1

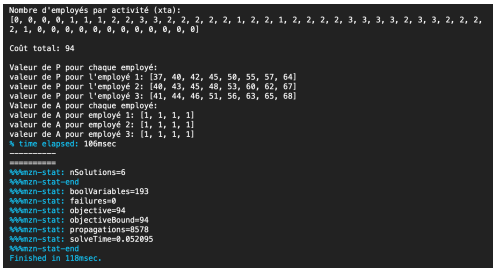


Figure 1: Bench 1: Ortools avec 8 threads et free search

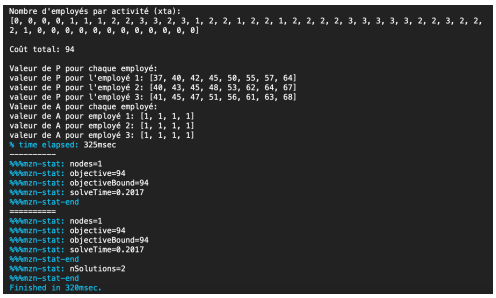


Figure 2: Bench 1: Highs

Nous pouvons voir que nous avons atteint un score de 94 pour le score optimal en 118 ms avec ORTools et en 325ms avec Highs. Le nombre de noeuds visités avec Highs est de 1.

2.5.2 Bench 2

Nous pouvons voir que nous avons atteint un score de 38 pour le score optimal en 3,312 s avec ORTools et en 15,84ms avec Highs. Le nombre de noeuds visités avec Highs est de 1516.

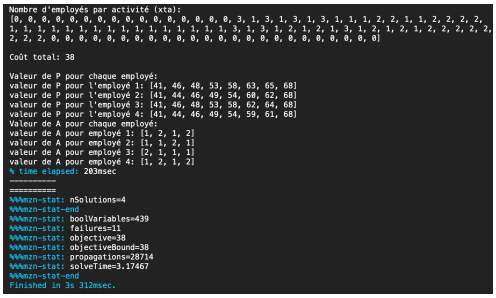


Figure 3: Bench 2: Ortools avec 8 threads et free search

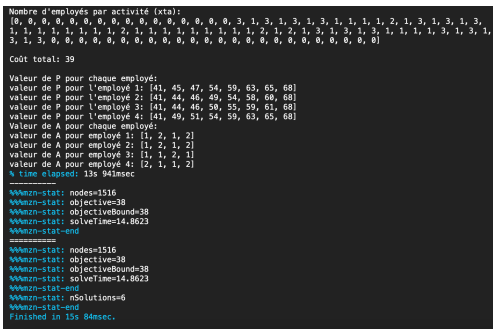


Figure 4: Bench 2: Highs

2.5.3 Bench 3

Concernant le Bench 3 et 4, avec Ortools, le modèle semble converger particulièrement vite (score de 16 en 3,844 s) comme vous pouvez l'observer ci-dessous. Mais le problème est qu'il ne se termine pas. Nous avons essayé de diminuer l'espace de recherche au plus pour que le solveur puisse atteindre l'optimalité au plus vite mais maheureusement sans succès.

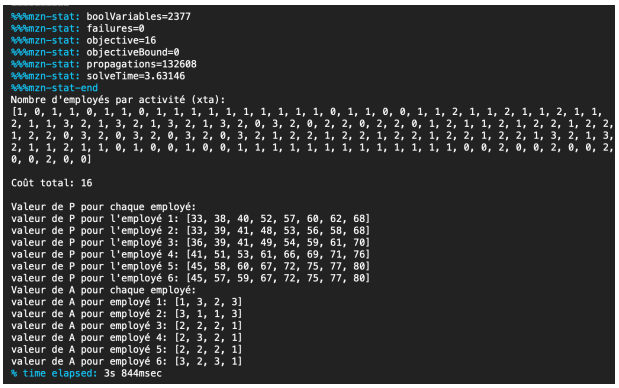


Figure 5: Bench 3: Ortools avec 8 threads et free search

2.5.4 Bench 4

Le même phénomène semble se passer pour le bench 4 qui lui converge à un score de 10 après seulement 6,528s et stagne par la suite.

2.5.5 Tentative pour bench 3 et 4

Nous avons essayé une multitude d'heuristiques différentes avec le seq_search et le search, des solveurs et de contraintes fixatrices mais tout cela en vain.

Par exemple pour des contraintes fixatrice, nous avions pensé à appliquer la même méthodologie appliquée

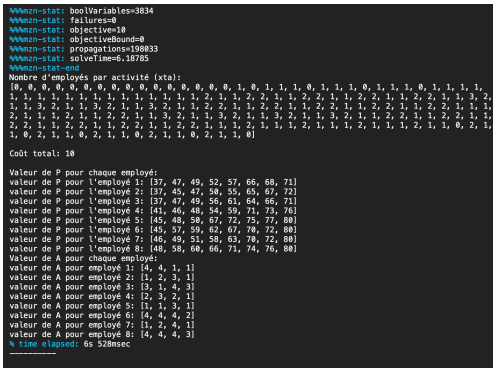


Figure 6: Bench 4: Ortools avec 8 threads et free search

au souscout mais pour le surcoutavec un seuil différent, malheureusment cela rendait un problème insolvable pour certaines instances, comme pour de nombreuses autres contraintes fixatrices essayées. Nous ne sommes pas parvenus à atteindre la fin pour l'instance 3 et 4. Nous pensons tout de même que nous avons trouvé les solutions à l'optimum pour le bench 3 et 4 vu que les scores ne bougent absolument plus après.