

CENTRALESUPÉLEC

SYSTEMES DE TRANSPORT DE PASSAGERS
RAPPORT

Projets de Séquence Thématique : équipe Haikara

Planification quotidienne d'une équipe mobile

L'équipe :

Pierrick BOURNEZ,
Raphaël FOURQUET,
Julien ROSENBERGER,
Emile SAINT-GIRONS,
Antoine VIALLE



Table des matières

I Introduction	1
II Modélisation et résolution de la phase 1	1
II.1 Modélisation du problème	1
II.1.1 Paramètres et ensembles du problème	1
II.1.2 Variables de décision du modèle	2
II.1.3 La fonction objectif	2
II.1.4 Les contraintes	2
II.2 Résultats pour les autres instancesV1 et discussions des résultats	3
II.2.1 Bordeaux	4
II.2.2 Finlande	5
II.2.3 Italy	6
II.2.4 GuineaGolf	8
II.2.5 Poland	9
III Modélisation et résolution de la phase 2	11
III.1 Modélisation du problème	11
III.1.1 Paramètres et ensembles du problème	11
III.1.2 Variables de décision du modèle	11
III.1.3 La fonction objectif	11
III.1.4 Les contraintes	12
III.2 Résultats de la phase 2	12
III.2.1 Bordeaux	13
III.2.2 Pologne	14
III.2.3 Espagne	15
III.2.4 Australie	16
III.2.5 Autriche	17
IV Phase 3 Métaheuristiques	19
IV.1 Clustering des tâches	19
IV.1.1 Détermination des clusters	19
IV.1.2 Résultat du clustering	19
IV.2 L'algorithme Glouton	24
IV.2.1 Sans pauses déjeuner et indisponibilités	24
IV.2.2 Ajout de la pause déjeuner	24
IV.2.3 Ajout des indisponibilités	24
IV.3 Des heuristique parallèle	24
IV.3.1 algorithme fixed_point	24
IV.3.2 Algorithme opti_local	25
IV.4 Conclusion	25

I Introduction

DecisionBrain est une startup qui développe des solutions d'optimisation pour la logistique, l'industrie, la planification de main d'oeuvre et la chaîne d'approvisionnement pour ses clients. Un de ces derniers est l'entreprise *TuttoBene* qui envoie ses techniciens chez ses clients afin de réaliser des tâches variées. Le travail ici présent est de proposer au planificateur de *TuttoBene* un logiciel qui l'aiderait dans son travail en lui indiquant une solution optimale pour la planification de l'équipe de techniciens sur une journée. Son but est de permettre aux techniciens de remplir le maximum de tâches tout en minimisant la distance parcourue.

II Modélisation et résolution de la phase 1

II.1 Modélisation du problème

II.1.1 Paramètres et ensembles du problème

Soit $n \in \mathbb{N}$ le nombre d'employés.

Définissons :

- $E = [\![1, n]\!] \subset \mathbb{N}$ l'ensemble des identifiants des employés ;
- $T_{communes} = \{t_i\}, \subset \mathbb{N}$ l'ensemble des identifiants des tâches communes à tous les employés ;
- $\forall k \in E, T_k \subset \mathbb{N}$ l'ensemble des identifiants des tâches spécifiques à l'employé k (cela peut inclure des temps d'indisponibilité, le départ de chez soi en début de journée, et le retour chez soi en fin de journée). Plus particulièrement, pour tout $k \in E$, on nommera respectivement $Départ_k$ et $Retour_k$ les tâches "partir de chez soi" et "rentrer chez soi" propres à l'employé k . Elles appartiennent toutes les deux à T_k .
- $T = T_{communes} \cup (\bigcup_{k \in E} T_k)$ l'ensemble des identifiants des tâches réalisables.

Par conséquent, $\forall k \in E, T_k \cap T_{communes} = \emptyset$ et $\forall (k, l) \in E^2, k \neq l \implies T_k \cap T_l = \emptyset$.

Pour un employé $k \in E$:

- $ENiveau_k$ son niveau de compétence ;
- $EDébut_k$ l'heure de début de travail et $EFin_k$ l'heure de fin de travail ;

Une tâche $i \in T_{common}$ a :

- une latitude $TLatitude_i$ et une longitude $TLongitude_i$
- une durée de réalisation $TDurée_i$
- un niveau requis pour la réaliser $TNiveau_i$
- un temps possible de début de réalisation $TOuverture_i$ et de fin de réalisation $TFermeture_i$

Quelques traitements particuliers :

- Pour $i \in \bigcup_{k \in E} \{Départ_k, Retour_k\}$, il reste :
 - $TLatitude_i$ et $TLongitude_i$ qui renvoient au logement de l'employé k
 - $TOuverture_i = TFermeture_i$
 - $TDurée_i = 0$.
- Pour $i \in \bigcup_{k \in E} T_k \setminus \{Départ_k, Retour_k\}$, il reste :
 - $TLatitude_i$ et $TLongitude_i$ qui renvoient à la localisation de l'indisponibilité
 - $TOuverture_i, TFermeture_i$ qui sont respectivement le temps de début et de fin possibles de la tâche i
 - $TDurée_i$ qui est la durée de l'indisponibilité.
- Pour tout $(i, j) \in T^2$, on note $TDist_{i,j}$ la distance entre la tâche i et la tâche j .

II.1.2 Variables de décision du modèle

Définissons les variables de décision de notre modèle :

- $(x_{ijk})_{(i,j,k) \in T \times T \times E}$ tel que chaque x_{ijk} vaut 1 si et seulement si la tâche j est faite après la tâche i par l'employé k
- $(t_{ik})_{(i,k) \in T \times E} \in \mathbb{R}_+^{|T| \times |E|}$ tel que chaque t_{ik} représente le temps de début de la tâche i réalisée par l'employé k .

II.1.3 La fonction objectif :

On veut minimiser dans un premier temps les coûts de transport :

$$\sum_{(i,j,k) \in T^2 \times E} x_{ijk} \times TDist_{i,j}.$$

II.1.4 Les contraintes

Les obligations du problème :

- Toutes les tâches doivent être réalisées (permis par l'hypothèse selon laquelle l'équipe de techniciens est suffisante pour réaliser toutes les tâches) :

$$\forall j \in T_{communes}, \sum_{k \in E} \sum_{i \in T} x_{ijk} = 1$$

- Les tâches propres à l'employé doivent être réalisées par celui-ci :

$$\forall k \in E, \sum_{j \in T} x_{D\acute{e}part_{kj}} = 1$$

$$\sum_{k \in E} \sum_{j \in T} x_{D\acute{e}part_{kj}} \leq 1$$

$$\forall k \in E, \forall j \in T_k \setminus \{D\acute{e}part_k\}, \sum_{i \in T} x_{ijk} = 1$$

- Chaque tâche doit être effectuée après le début et après la fin de son créneau.

$$\forall k \in E, \forall j \in T, TOuverture_j \times \sum_{i \in T} x_{ijk} \leq t_{jk} \leq TFermeture_j \times \sum_{i \in T} x_{ijk} - TDurée_j$$

(Puisque $\forall k \in E, \forall j \in T_k \setminus \{D\acute{e}part_k, Retour_k\}, TDurée_j = TFermeture_j - TOuverture_j$, cette contrainte liée à celle de respect des durées force à ce que les indisponibilités tiennent exactement dans leur créneau.)

- Chaque tâche effectuée par un employé doit l'être pendant ses horaires de travail. Cela revient à ce que le temps de démarrage de la première et dernière tâche soient comprises entre son heure d'embauche et l'heure à laquelle il termine le travail :

$$\forall k \in E, \begin{cases} EDébut_k \leq t_{D\acute{e}part_{k,k}} \leq EFin_k \\ EDébut_k \leq t_{Retour_{k,k}} \leq EFin_k \end{cases}$$

Les contraintes des employés :

- Chaque tâche effectuée doit être d'un niveau de compétence inférieur à celui de l'employé qui la réalise :

$$\forall k \in E, \forall j \in T_{communes}, \sum_{i \in T} x_{ijk} \times TNiveau_j \leq ENiveau_k$$

On notera que les indisponibilités ont été représentées par des tâches, dont les temps d'ouverture et de fermeture sont les heures de début et de fin d'indisponibilité, et dont la durée est égale à la différence entre ce début et cette fin.

Les contraintes implicites du problèmes (assurant une cohérence) :

- Chaque tâche est réalisée au plus une fois par un employé donné :

$$\forall j \in T_{communes} \sum_{(i,k) \in T \times E} x_{ijk} \leq 1$$

- Si une tâche est effectuée, alors l'employé qui l'effectue doit en repartir. Cela revient, en modélisant le problème par un graphe de flot, à la conservation des flux entrant et sortant de chaque nœud :

$$\forall j \in T \setminus \left(\bigcup_{k \in E} \{LHome_k, Retour_k\} \right), \sum_{(i,k) \in T \times E} x_{ijk} = \sum_{(i',k) \in T \times E} x_{j_i k}$$

Le lien avec les temps :

- On doit relier temporellement le lien du départ d'une tâche j sachant le temps de la tâche i effectuée avant pour tous les employés :

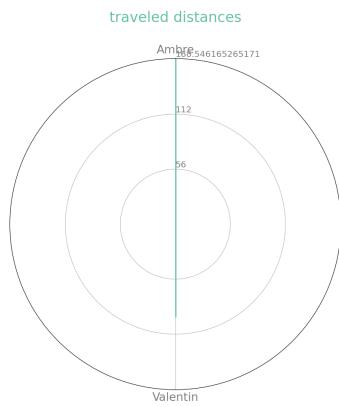
$$\forall (i, j, k) \in T^2 \times E, t_{ik} + x_{ijk} \times (TDist_{i,j} + TDurée_i) < t_{jk} + TFermeture_i \times (1 - x_{ijk})$$

II.2 Résultats pour les autres instancesV1 et discussions des résultats

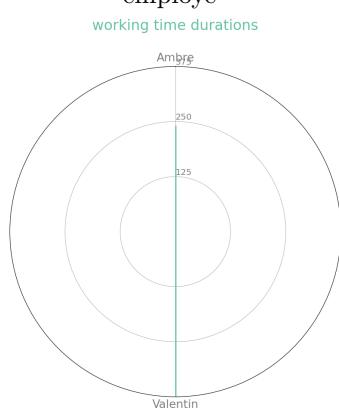
Nous répertorions ici les résultats pour les différentes instances de la phase 1. Nous représentons ici les statistiques intéressantes comme la répartition de durée de travail et de distance de chaque employé, une carte de route de chaque employé,l'évolution de l'optimisation de Gurobi en fonction du temps ainsi qu'un radar chart.

Nos calculs ont été réalisés sur une NVIDIA DGX avec 64 Go de RAM.Pour certains instances, nous avons pus résoudre le problème exactement mais nous avons fixé une limite de temps de 20 minute pour certaines instances comme FinlandV1

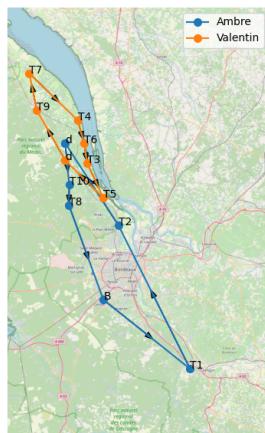
II.2.1 Bordeaux



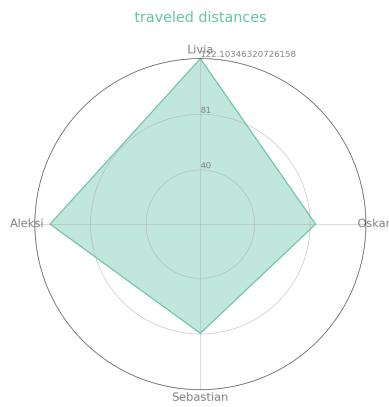
(a) Radar chart de la distance parcourue par employé



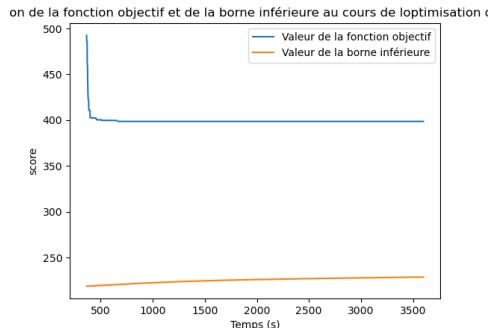
(c) Radar chart de la durée de travail effective par employé



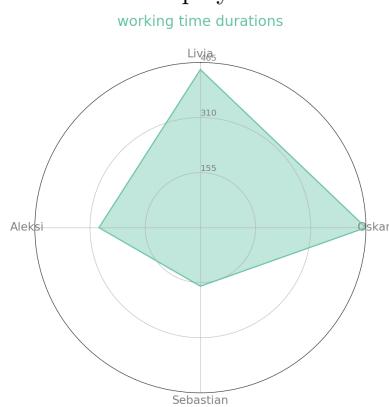
II.2.2 Finlande



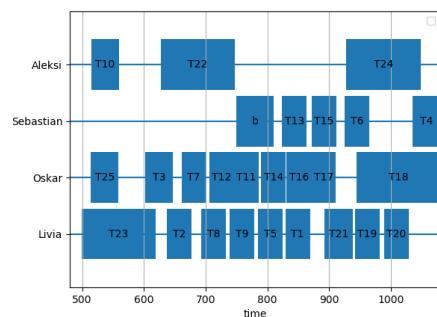
(a) Radar chart de la distance parcourue par employé



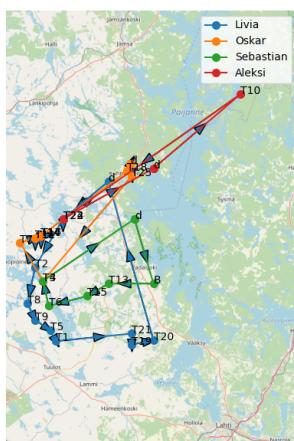
(b) Evolution de l'optimisation de Gurobi en fonction du temps



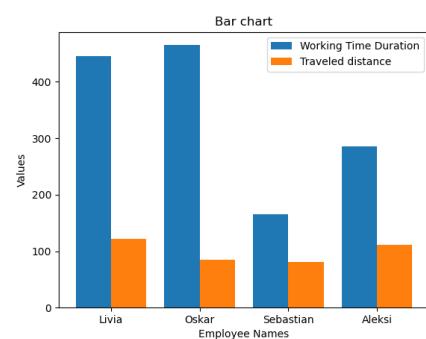
(c) Radar chart de la durée de travail effective par employé



(d) Emploi du temps de chaque employé



(e) RoadMap de chaque employé



(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

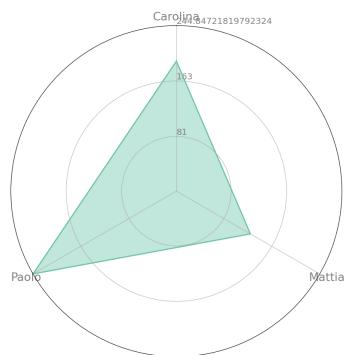
FIGURE 2 – Etude de la solution Finland pour la méthode V1

Les données obtenues sur l'instance finlandaise sont plus intéressantes. Nous avons une répartition relativement uniforme des tâches pour chacun des employés et un employé semble avoir beaucoup plus de distance ou de temps de travail que les autres, sûrement du fait d'une tâche longue. Un autre fait notable est l'évolution de l'optimisation de Gurobi sur cette instance. On voit clairement un saut entre la borne inférieure de travail de Gurobi et celle de la valeur objectif, et cet écart ne

semble pas diminuer avec le temps. Cela est peut-être dû à la difficulté de la recherche d'une solution faisable.

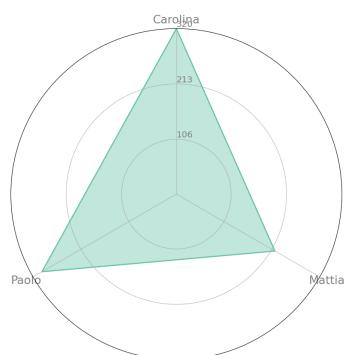
II.2.3 Italy

traveled distances

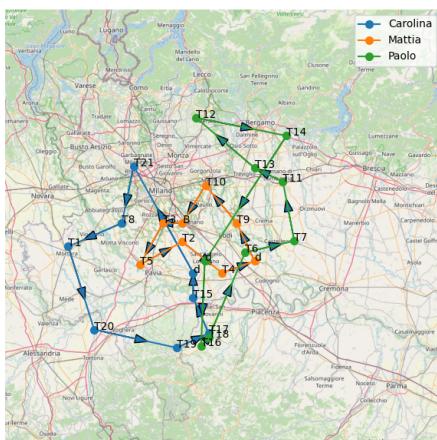


(a) Radar chart de la distance parcourue par employé

working time durations

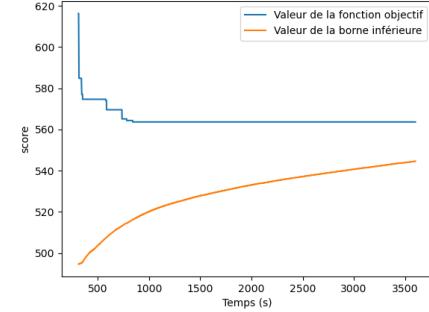


(c) Radar chart de la durée de travail effective par employé

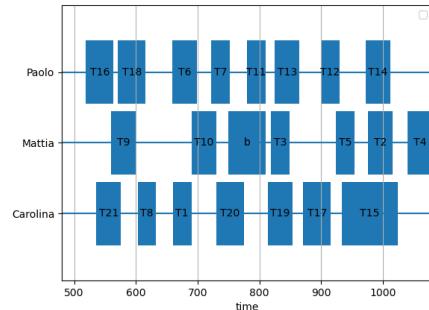


(e) RoadMap de chaque employé

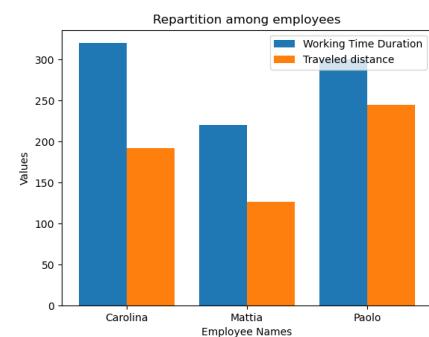
Evolution de la fonction objectif et de la borne inférieure au cours de l'optimisation



(b) Evolution de l'optimisation de Gurobi en fonction du temps



(d) Emploi du temps de chaque employé



(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

FIGURE 3 – Etude de la solution Italy pour la méthode V1

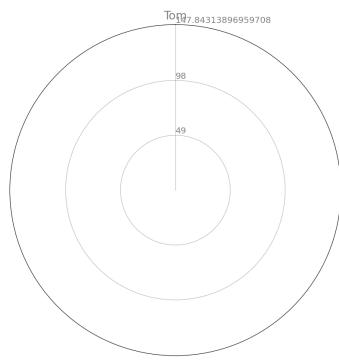
L'algorithme n'a pas pu converger même après 1 heure, mais l'algorithme semblait avoir trouver la solution optimal. On aperçoit ici que Paolo a la majorité des tâche et que c'est aussi celui qui

se déplace le plus. Cette assymétrie de la répartition des tâches peut s'expliquer par la fonction objectif qui ne s'occupe que la distance majoritairement parcourue et non de l'équilibre des distances parcourues.

On remarque aussi et c'est aussi ce qui sera à la base d'un développement de notre phase 3 une séparation spatiale des différents trajets de chaque employé, Carolina s'occupe de l'ouest de la ville, Paolo l'est et Mattia est ce lui qui travaille le plus à l'intérieur de la ville.

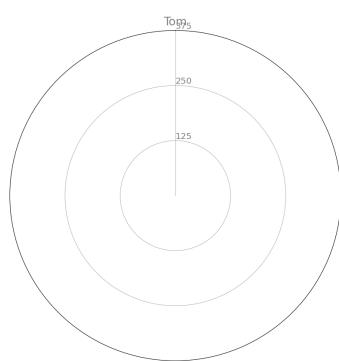
II.2.4 GuineaGolf

traveled distances



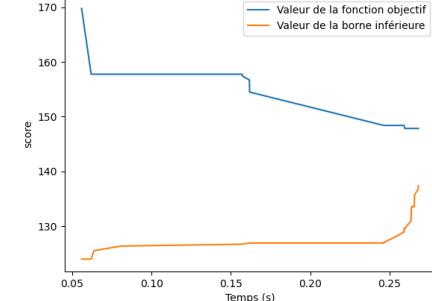
(a) Radar chart de la distance parcourue par employé

working time durations

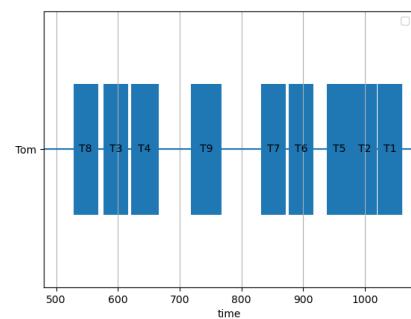


(c) Radar chart de la durée de travail effective par employé

Evolution de la fonction objectif et de la borne inférieure au cours de l'optimisation

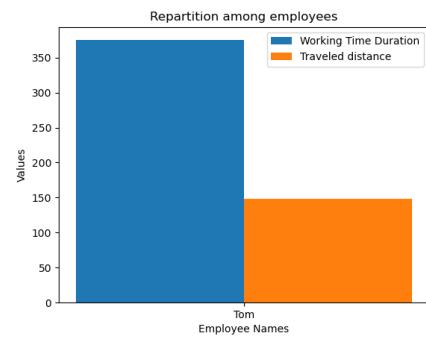
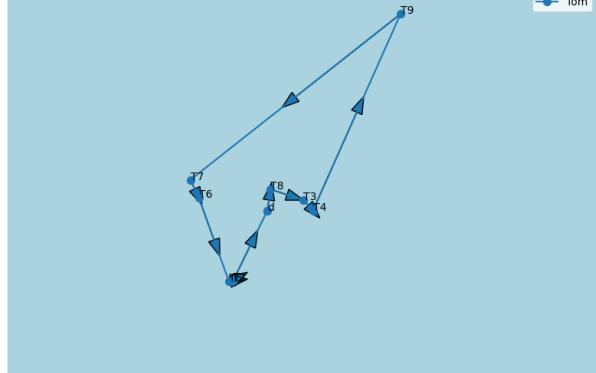


(b) Evolution de l'optimisation de Gurobi en fonction du temps



(d) Emploi du temps de chaque employé

(e) RoadMap de chaque employé

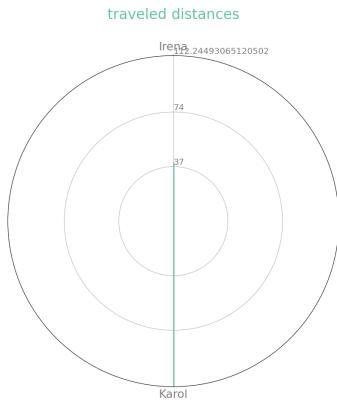


(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

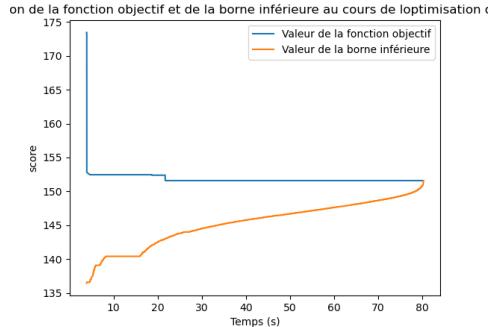
FIGURE 4 – Etude de la solution GuineaGolf pour la méthode V1

L'instance est une instance classique, il s'agit d'un employé qui a des tâches à effectuer dans un ordre, l'algorithme converge bien et on la séparation dans l'emploi du temps de l'employé s'explique par le fait que T9 est très espacé géographiquement des autres tâches

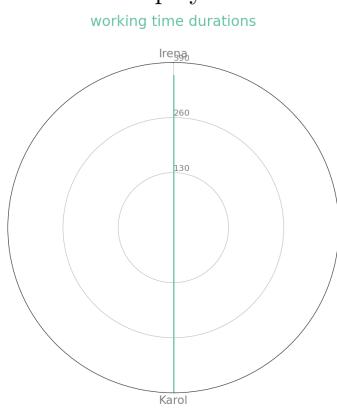
II.2.5 Poland



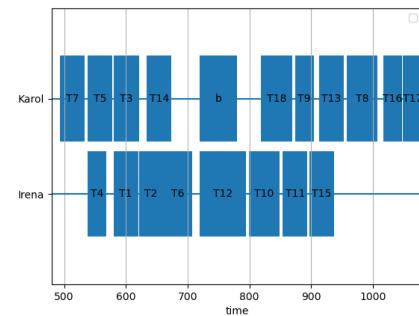
(a) Radar chart de la distance parcourue par employé



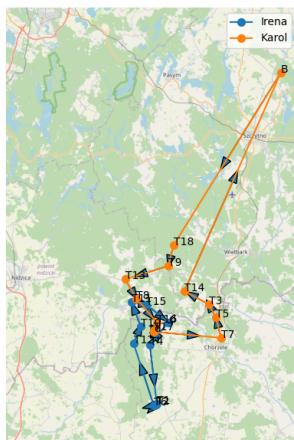
(b) Evolution de l'optimisation de Gurobi en fonction du temps



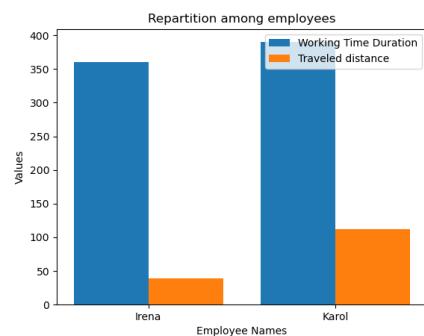
(c) Radar chart de la durée de travail effective par employé



(d) Emploi du temps de chaque employé



(e) RoadMap de chaque employé



(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

FIGURE 5 – Etude de la solution Poland pour la méthode V1

La solution trouvée est encore exacte et nous avons toujours une séparation spatiale entre les différents employés. Les emplois du temps sont relativement égaux. On remarque ici qu'une tâche (T8) est très espacé géographiquement des autres ce qui augmentent drastiquement le temps de trajet de Karol. Une amélioration serait donc d'éliminer cette tâche pour diminuer drastiquement le temps de trajet des employés. C'est pour cela que nous verrons dans une seconde phase le cas où toutes les

tâches ne sont pas nécessairements tous accomplies.

III Modélisation et résolution de la phase 2

III.1 Modélisation du problème

Dans cette nouvelle phase, nous prenons en compte le fait que toutes les tâches ne seront pas nécessairement effectués et nous prenons en compte la pause déjeuner des employés ainsi que des indisponibilités éventuelles des tâches

III.1.1 Paramètres et ensembles du problème

Par rapport à la phase 1, nous ajoutons les paramètres $DDurée$, la durée d'un déjeuner, $DOuverture$, le début du créneau autorisé du déjeuner et $DFermeture$, la fin du créneau autorisé du déjeuner.

Dans cette phase, les tâches peuvent admettre des indisponibilités.

On doit faire un nouveau traitement : pour chaque tâche $i \in T_{communes}$, on définit T'_i l'ensemble de réalisations de la tâche i pendant des créneaux disjoints deux à deux. Puis $T'_{communes} := \bigcup_{i \in T_{communes}} T'_i$. On réécrit $T' := T'_{communes} \cup \bigcup_{k \in E} T_k$.

Pour tout $i \in T_{communes}$, pour tout $i' \in T'_i$:

- $TLatitude'_i$, $TLongitude'_i$, $TDurée'_i$, $TNiveau'_i$ sont identiques aux paramètres respectifs de la tâche i
- d'où $\forall j \in T, TDist_{i',j} = TDist_{j,i'} = TDist_{i,j}$ et $\forall j \in T, \forall j' \in T'_j, TDist_{i',j'} = TDist_{j',i'} = TDist_{i,j}$
- un temps possible de début de réalisation $TOuverture'_i$ et de fin de réalisation $TFermeture'_i$

Pour rappel, par définition :

$$\forall i \in T_{communes}, \forall (i', i'') \in T'_i, [TOuverture'_i, TFermeture'_i] \cap [TOuverture''_i, TFermeture''_i] = \emptyset$$

III.1.2 Variables de décision du modèle

Nous ajoutons à notre modèle les variables de décision suivantes : $(m_{ijk})_{(i,j,k) \in T'' \times T'' \times E}$ tel que chaque m_{ijk} vaut 1 si et seulement si l'employé k mange entre la tâche i et la tâche suivante j et que la tâche j est bien réalisée après la tâche i par l'employé k .

III.1.3 La fonction objectif

Nous avons choisi de conserver la forme d'un problème d'optimisation mono-objectif, en considérant que notre client cherche à maximiser son bénéfice, qui augmente avec le nombre d'heures réalisées, et diminue avec le temps de trajet. Un "malus" environnemental, modifiable et "neutre" par défaut, lui permet de pénaliser davantage les déplacements.

La fonction objectif à minimiser prend donc la forme suivante :

$$\times CoûtTrajet \sum_{(i,j,k) \in T'^2_{communes} \times E} x_{ijk} TDist_{i,j} - GainTravail \sum_{(i,j,k) \in T'^2_{communes} \times E} x_{ijk} TDurée_j$$

Nous considérons ici une rémunération horaire moyenne. Un modèle plus fin pourrait prendre en compte la difficulté des tâches réalisées, qui influe en principe sur le revenu qui leur est associé. En terme d'optimisation, les deux termes ont des effets très opposés. Plus le Coût du trajet est important, plus l'optimiseur aura tendance à effectué beaucoup de tâche et la résolution sera donc plus lente ou compliquée (l'algorithme pourra même ne pas trouver de solution) alors que plus Gain travail est grand plus certaines tâches seront écartées et plus l'optimiseur sera efficace.

III.1.4 Les contraintes

- Ayant "découpé" les tâches avec des indisponibilités, nous nous assurons qu'une même tâche n'est effectuée qu'une seule fois :

$$\forall j \in T_{communes}, \sum_{(i,j,k) \in T' \times T'_j \times E} x_{ijk} \leq 1$$

Le lien avec les temps :

- On doit relier temporellement le lien du départ d'une tâche j sachant le temps de la tâche i effectuée avant pour tous les employés :

$$\forall (i, j, k) \in T'^2 \times E, t_{ik} + x_{ijk} \times (T\text{Dist}_{i,j} + T\text{Durée}_i) + m_{ijk}DDurée \leq t_{jk} + T\text{Fermeture}_i \times (1 - x_{ijk})$$

- On s'assure que le déjeuner est bien sur le créneau : $\forall (i, j, k) \in T'^2 \times E,$

$$\begin{cases} t_{jk} \geq m_{ijk}(DDurée + DOuverture) \\ t_{ik} + T\text{Durée}_i + DDurée \leq DFermeture + (1 - m_{ijk})(T\text{Fermeture}_i + T\text{Durée}_i + DDurée) \end{cases}$$

La première contrainte de l'accordade assure qu'un minimum de temps est disponible sur l'intervalle [DOuverture, DFermeture]. La deuxième assure que tous les employés finissent de manger avant DFermeture.

Déjeuner :

- On prend une pause entre deux tâches seulement si on enchaîne les deux tâches en question :

$$\forall (i, j, k) \in T'^2 \times E, m_{ijk} \leq x_{ijk}$$

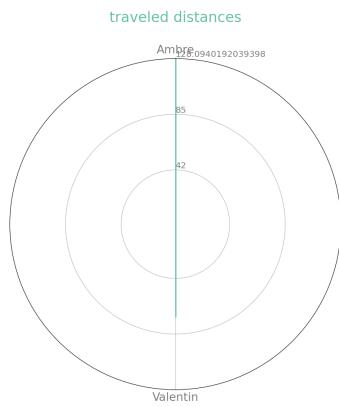
- On s'assure que tous les employés aient un temps de pause déjeuner :

$$\forall k \in E, \sum_{(i,j) \in T^2} m_{ijk} = 1$$

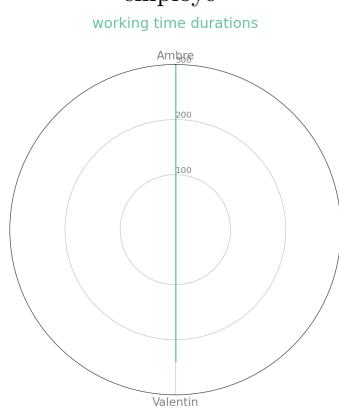
III.2 Résultats de la phase 2

Le RunTime ici était très erratique. Nous l'avons fixé à 20 minute sur la DGX, certaines instances arrivaient à se résoudre presque instantanément comme Bordeaux mais d'autre arrivaient soit à la limite de temps, soit si le coefficient de CoûtTrajet était trop grand, nécessitait 2 fois plus de temps de résolution que le timeLimit donnait

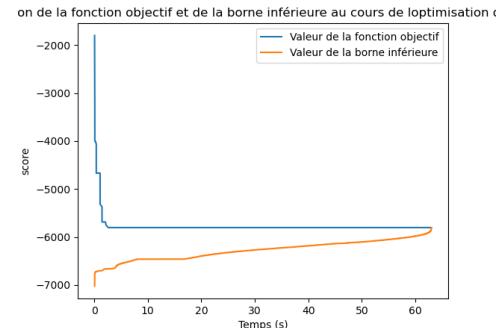
III.2.1 Bordeaux



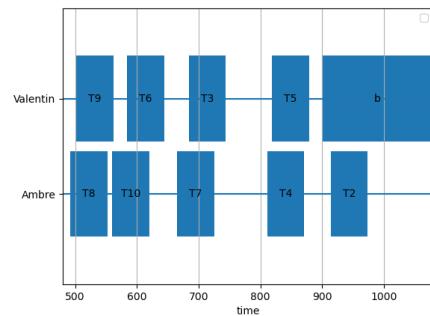
(a) Radar chart de la distance parcourue par employé



(c) Radar chart de la durée de travail effective par employé



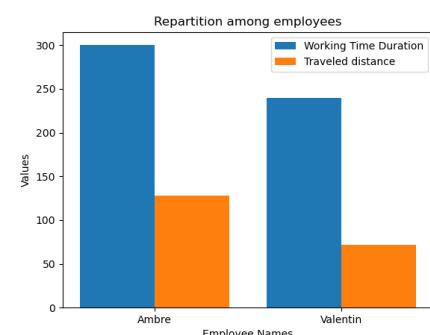
(b) Evolution de l'optimisation de Gurobi en fonction du temps



(d) Emploi du temps de chaque employé



(e) RoadMap de chaque employé

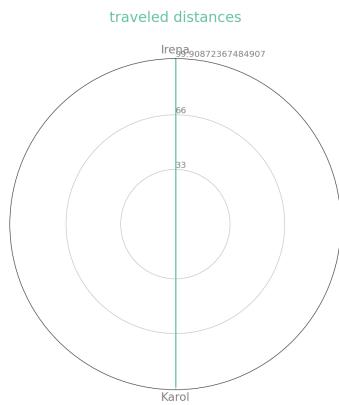


(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

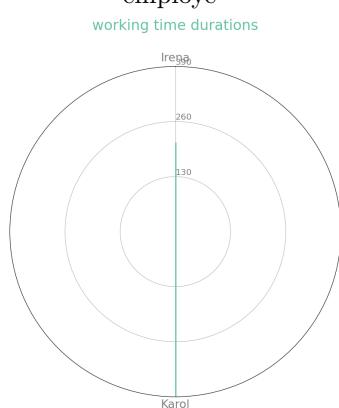
FIGURE 6 – Etude de la solution Bordeaux pour la méthode V2

Ici on voit que le fait qu'on n'impose plus la résolution exacte des tâches laissent la tâche T1 non faite. Elle permet aux employés d'avoir un temps de travail et une distance parcourue raisonnable, alors que la distance parcourue aurait explosé si la tâche T1 était faite.

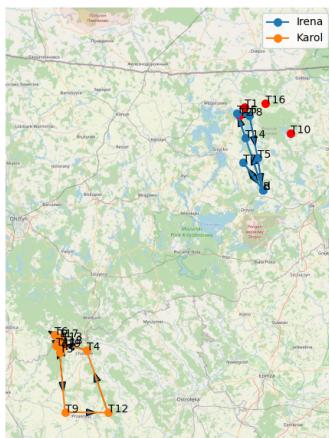
III.2.2 Pologne



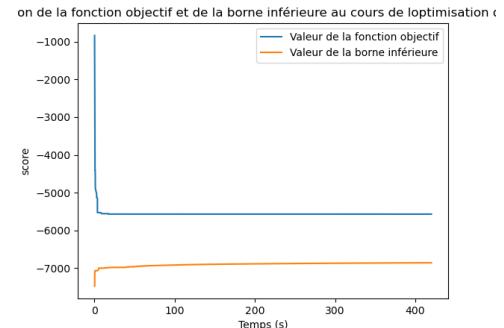
(a) Radar chart de la distance parcourue par employé



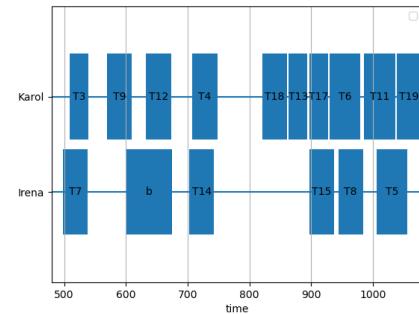
(c) Radar chart de la durée de travail effective par employé



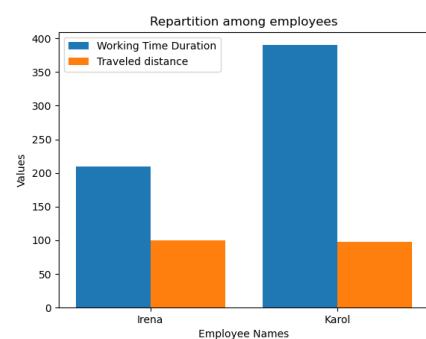
(e) RoadMap de chaque employé



(b) Evolution de l'optimisation de Gurobi en fonction du temps



(d) Emploi du temps de chaque employé

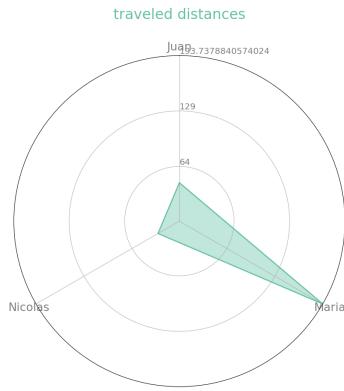


(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

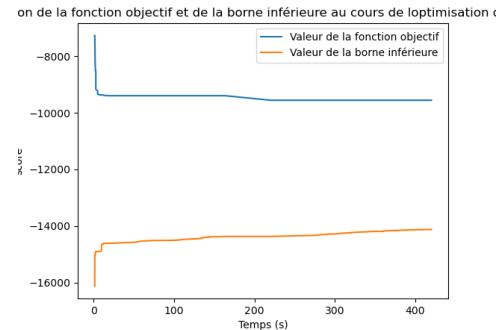
FIGURE 7 – Etude de la solution Poland pour la méthode V2

Ici, les deux zones sont très espacés géographiquement et les deux ouvriers travaillent dans des zones séparées. On voit que les tâches les plus éloignées géographiquement de Irena ne sont pas réalisées et Karol travaille beaucoup plus que Irena (près de 2 fois plus)

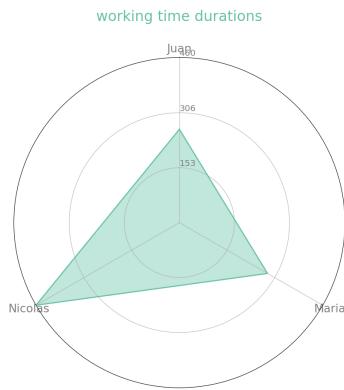
III.2.3 Espagne



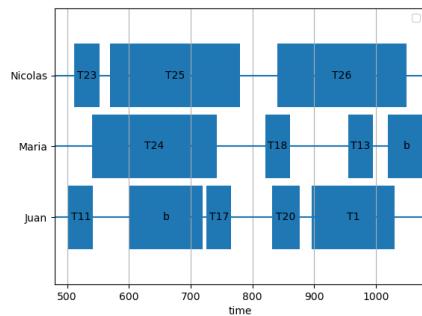
(a) Radar chart de la distance parcourue par employé



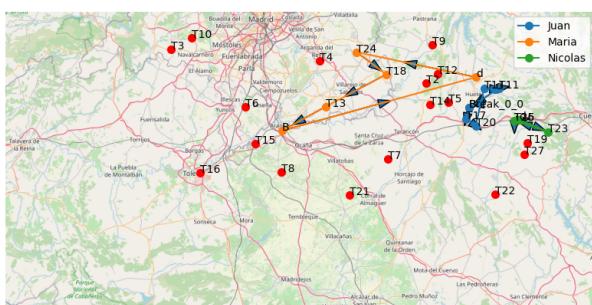
(b) Evolution de l'optimisation de Gurobi en fonction du temps



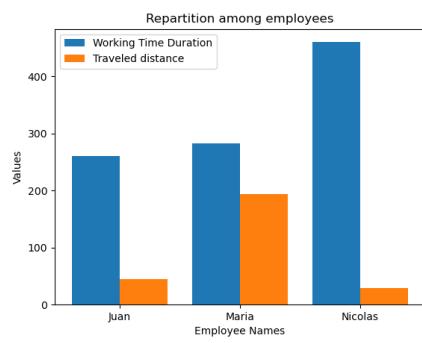
(c) Radar chart de la durée de travail effective par employé



(d) Emploi du temps de chaque employé



(e) RoadMap de chaque employé



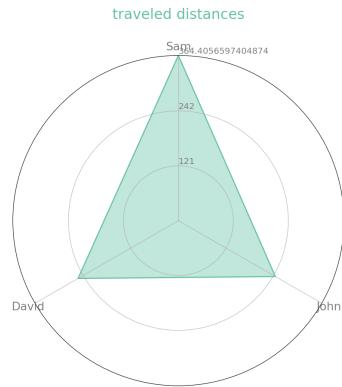
(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

FIGURE 8 – Etude de la solution Spain pour la méthode V2

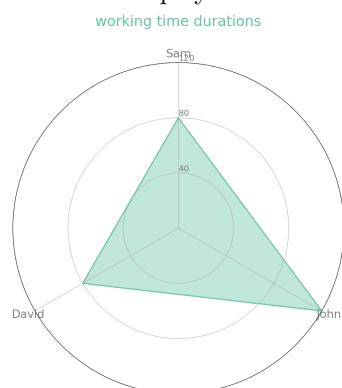
L'interprétation de la figure est ici plus compliquée. Beaucoup de tâches ne semblent pas réalisées mais les employés se concentrent sur les tâches avec le plus de temps de travaux (ce qui est visible par l'emploi du temps des employés). C'est une conséquence de notre fonction objectif, nous n'avons pas pris en compte le nombre de tâche différentes à réaliser et l'optimiseur c'est concentré sur les tâches avec le plus de temps de travail. C'est pour ça que la distance parcourue de Nicolas est aussi

faible malgré le fait qu'il ne fait que 2 tâches : ce sont les tâches qui prennent le plus de temps. Une amélioration aurait donc été de prendre en compte le nombre de tâche différente à réaliser. Maria au contraire fait pas de tâche et c'est pour cela qu'elle fait plus de tâche en globalité.

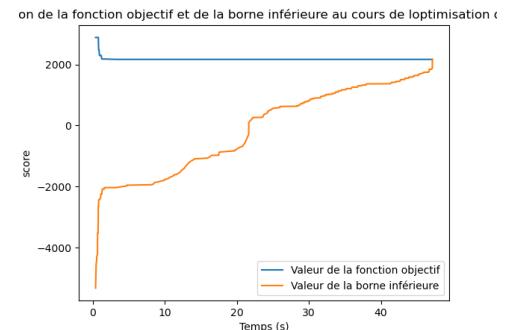
III.2.4 Australie



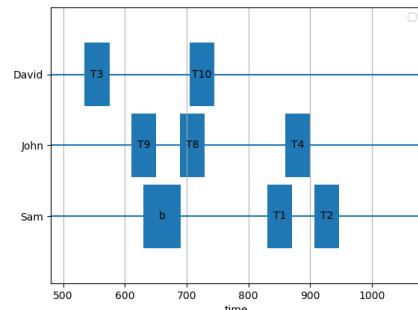
(a) Radar chart de la distance parcourue par employé



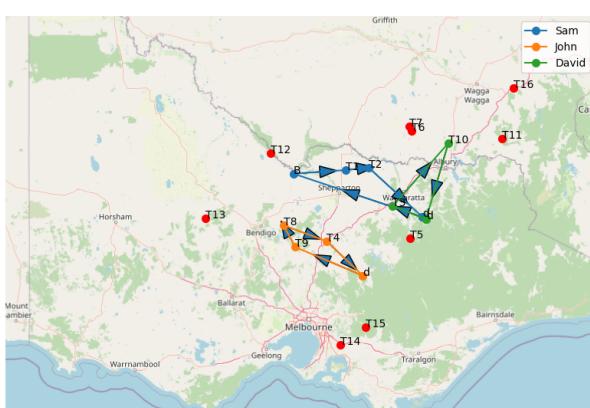
(c) Radar chart de la durée de travail effective par employé



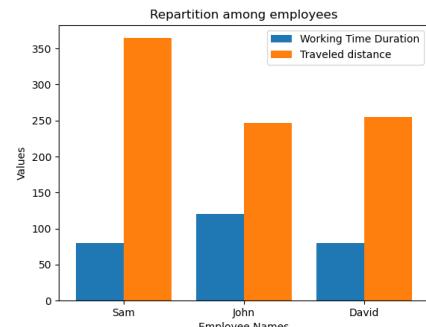
(b) Evolution de l'optimisation de Gurobi en fonction du temps



(d) Emploi du temps de chaque employé



(e) RoadMap de chaque employé

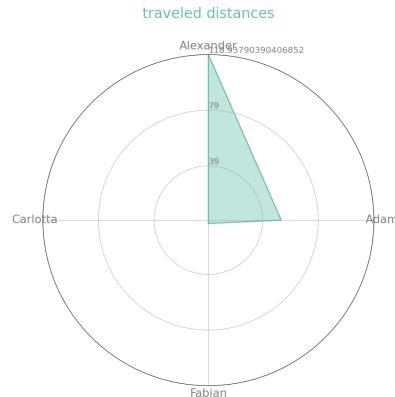


(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

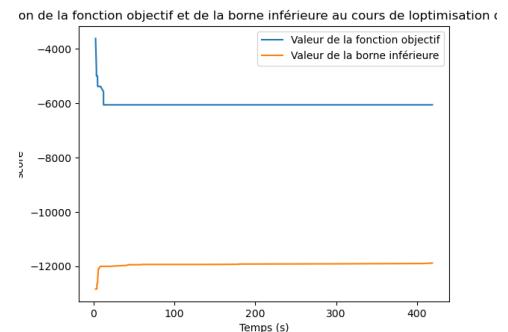
FIGURE 9 – Etude de la solution Australia pour la méthode V2

Les données ici sont plus équilibrés, les tâches sont globalement à une distance très grande et les employés passent la majorité du temps à se déplacer. Le facteur étonnant est l'absence de tâche de David l'après-midi, cela s'explique par le coefficient sûrement trop grand dans les distances. Après T10 les tâches sont sûrement trop loin

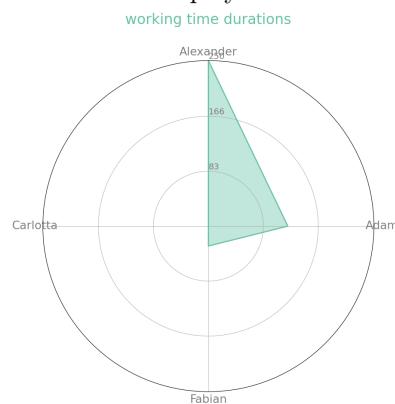
III.2.5 Autriche



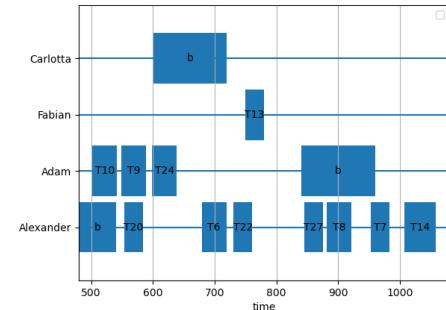
(a) Radar chart de la distance parcourue par employé



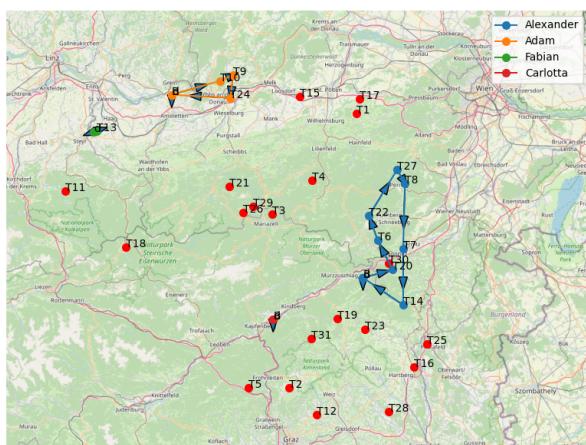
(b) Evolution de l'optimisation de Gurobi en fonction du temps



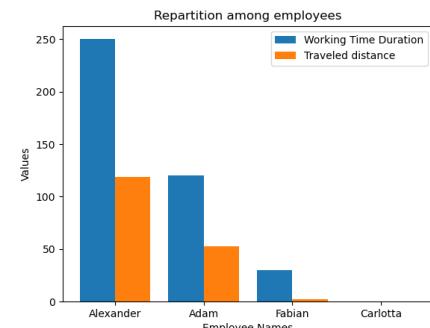
(c) Radar chart de la durée de travail effective par employé



(d) Emploi du temps de chaque employé



(e) RoadMap de chaque employé



(f) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

FIGURE 10 – Etude de la solution Austria pour la méthode V2

Les défauts trouvés dans l'instance Australie sont encore plus visible sur la RoadMap de l'Autriche. En effet, nous suspectons ici que le coefficient devant le coût de travail était beaucoup trop faible et les ouvriers effectuent très peu de tâches. Fabian commence près de T13 et la distance a toutes les autres tâches étant trop grandes, il n'a pas pu effectuer une quelconque tâche alors qu'Alexander commence proche de beaucoup de tâche et peut en réaliser plusieurs. Nous nous étonnons quand même pourquoi Fabian n'aurait pas réaliser la tâche T11 qui semble très proche de T13, et les créneaux de temps ne se chevauchent pas. Cela peut s'expliquer par le temps de recherche de la solution de 7 minutes qui n'a pas pu améliorer la solution notablement.

IV Phase 3 Métaheuristiques

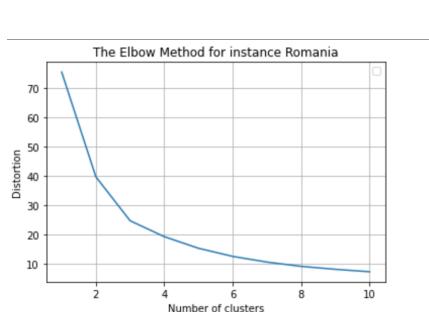
Nous allons dans cette phase étudier des métaheuristiques. La première est construite au dessus de guroby avec un clustering des tâches au préalable, puis un algorithme glouton

IV.1 Clustering des tâches

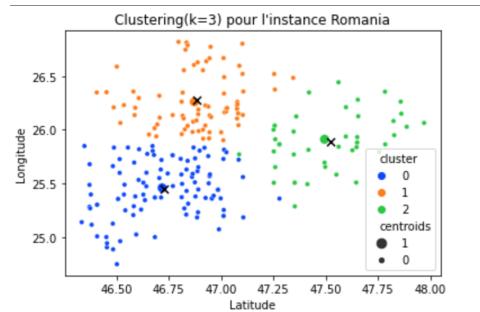
Comme les instances PolandV2 semblaient suggérer, les ouvriers semblent privilégier des tâches proches géographiquement. Cette idée issue de la phase 2 nous a poussé à regrouper les tâches proches géographiquement, puis de résoudre le sous-problème pour chaque cluster avant de proposer la solution finale.

IV.1.1 Détermination des clusters

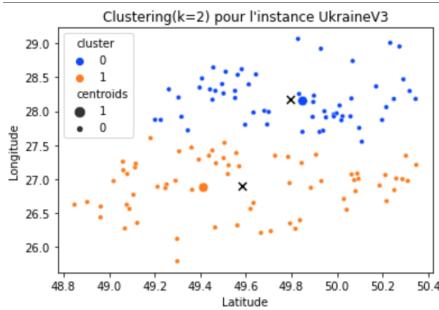
Pour déterminer les clusters, nous avons utilisé la méthode "elbow" qui consiste à appliquer une méthode des plus proches voisins avec plusieurs nombres de clusters, de visualiser la distortion en fonction du nombre de cluster et de prendre la valeur où il y a une cassure dans la courbe. Nous ne donnerons ici que le résultat pour Ukraine car les graphiques issus de cette méthode pour les 3 villes sont quasiment semblables



(a) Méthode de Elbow pour l'instance ColombieV3



(b) Clusters obtenu pour k=3 pour l'instance ColombieV3

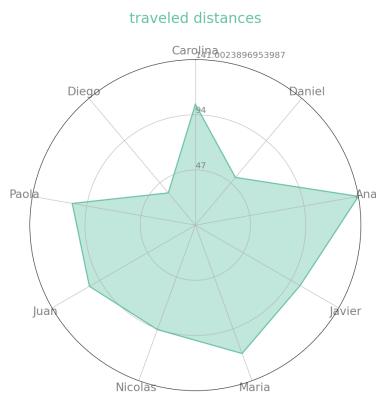


(c) Clusters obtenu pour k=3 pour l'instance UkraineV3

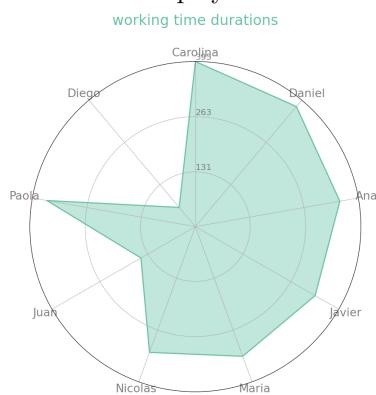
La cassure a lieu pour 2 ou 3 clusters et la répartition des clusters pour les instances Ukraine et Roumanie semblent valides. Un choix de cluster de 2 semblait aussi judicieux mais nous avons tranché pour 3.

IV.1.2 Résultat du clustering

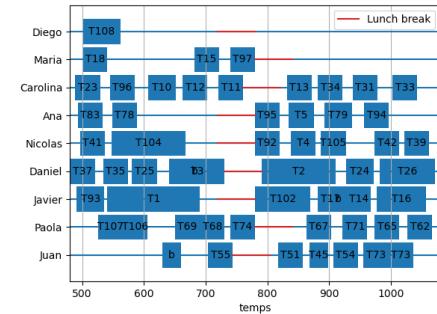
Nous donnons les résultats du clustering ici Colombie



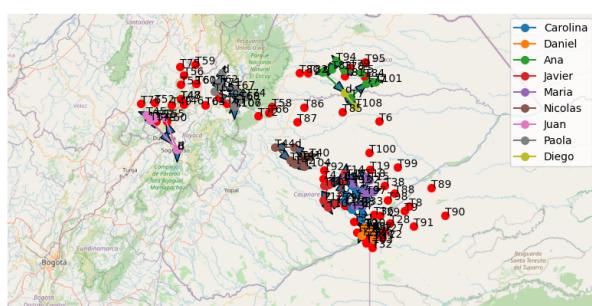
(a) Radar chart de la distance parcourue par employé



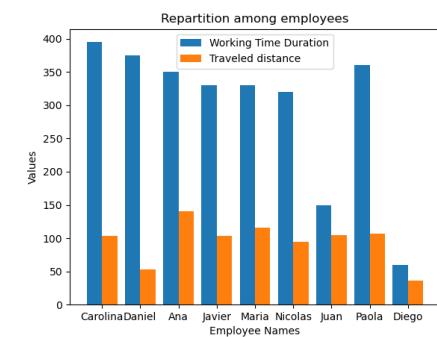
(b) Radar chart de la durée de travail effective par employé



(c) Emploi du temps de chaque employé



(d) RoadMap de chaque employé



(e) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

FIGURE 12 – Etude de la solution Colombia pour la méthode des clusters

Nous donnons à titre indicatif les résultats dans le cas de la phase2

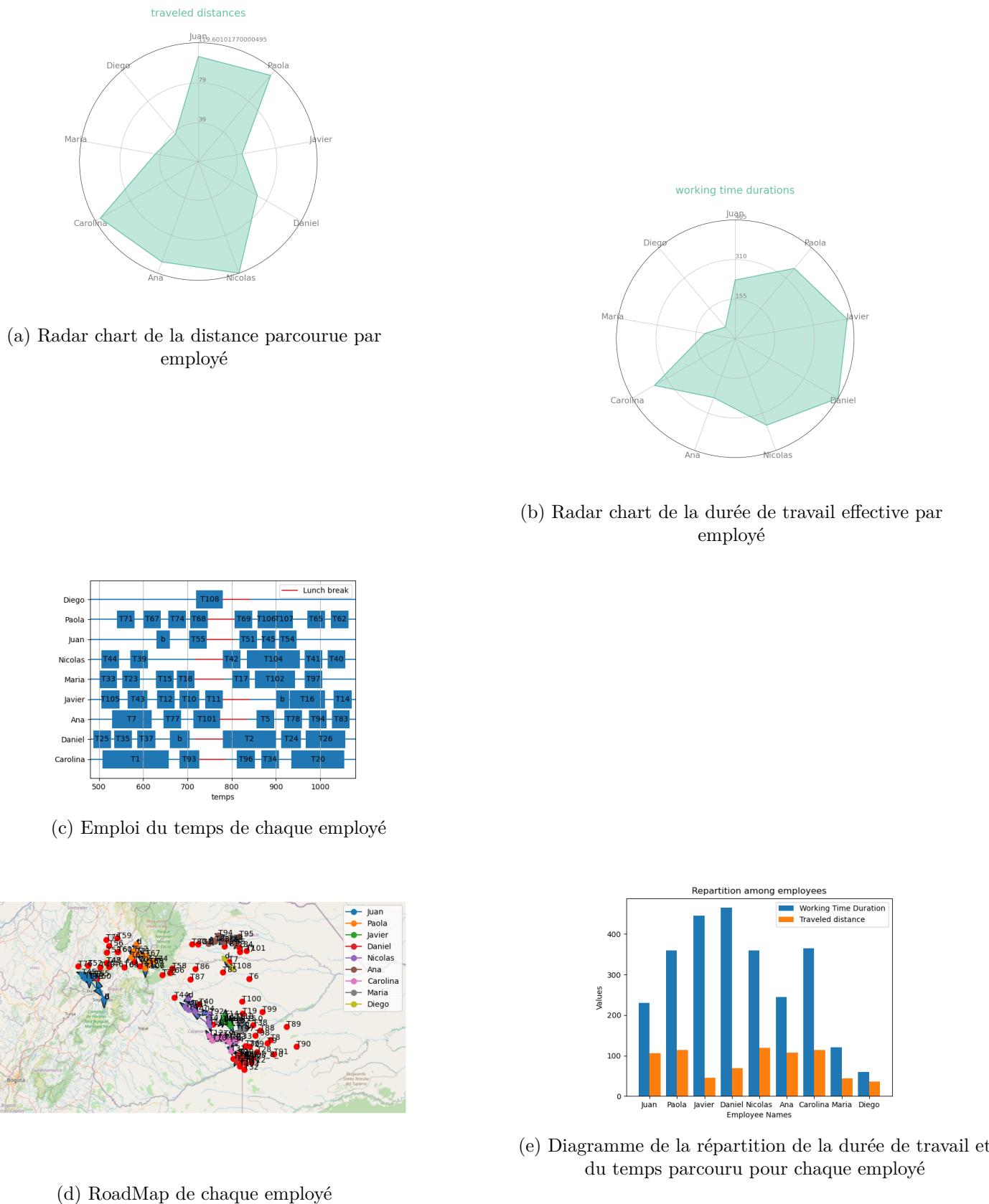
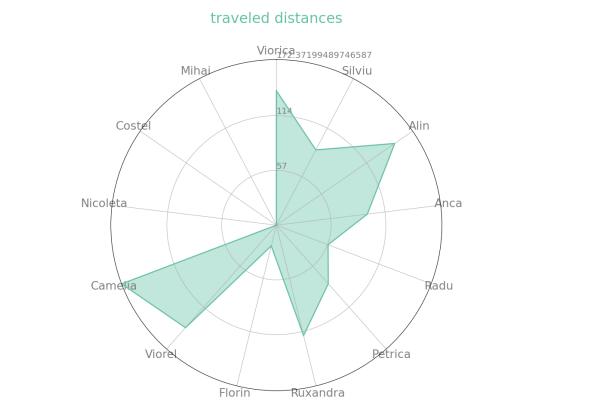


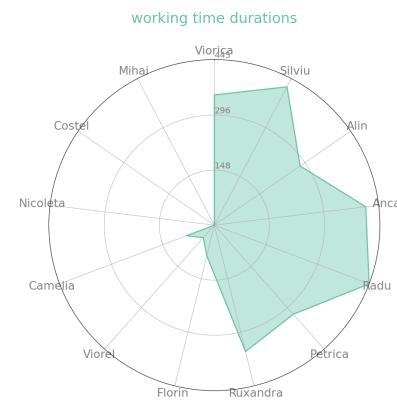
FIGURE 13 – Etude de la solution Colombia pour la méthode des clustersv3

Les résultats semblent probants des deux côtés et l'unique argument pour la métaheuristique est le runtime qui est légèrement inférieur. Nous donnons les résultats sur les autres instances. Roumanie

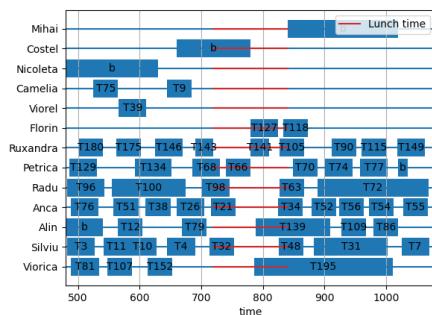
On voit une forte dyssymétrie du travail des employés, qui est sûrement dû à la dyssymétrie de l'affectation des employés (peu d'employé peuvent être affecté sur un grand cluster)



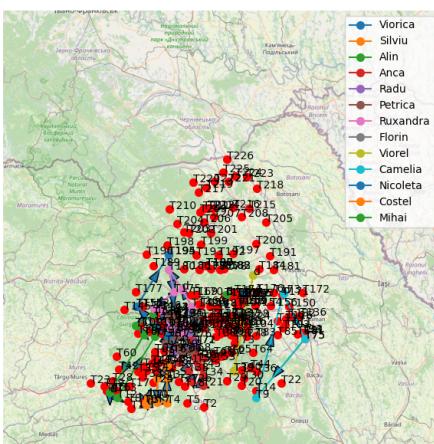
(a) Radar chart de la distance parcourue par employé



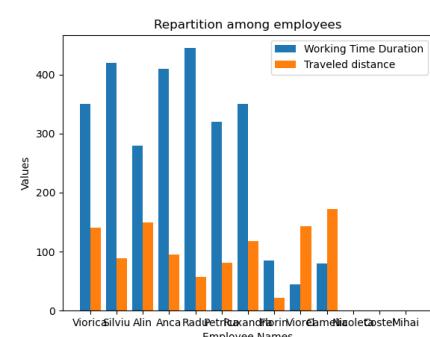
(b) Radar chart de la durée de travail effective par employé



(c) Emploi du temps de chaque employé



(d) RoadMap de chaque employé



(e) Diagramme de la répartition de la durée de travail et du temps parcouru pour chaque employé

FIGURE 14 – Etude de la solution Romania pour la méthode des clustersv3

Ukraine

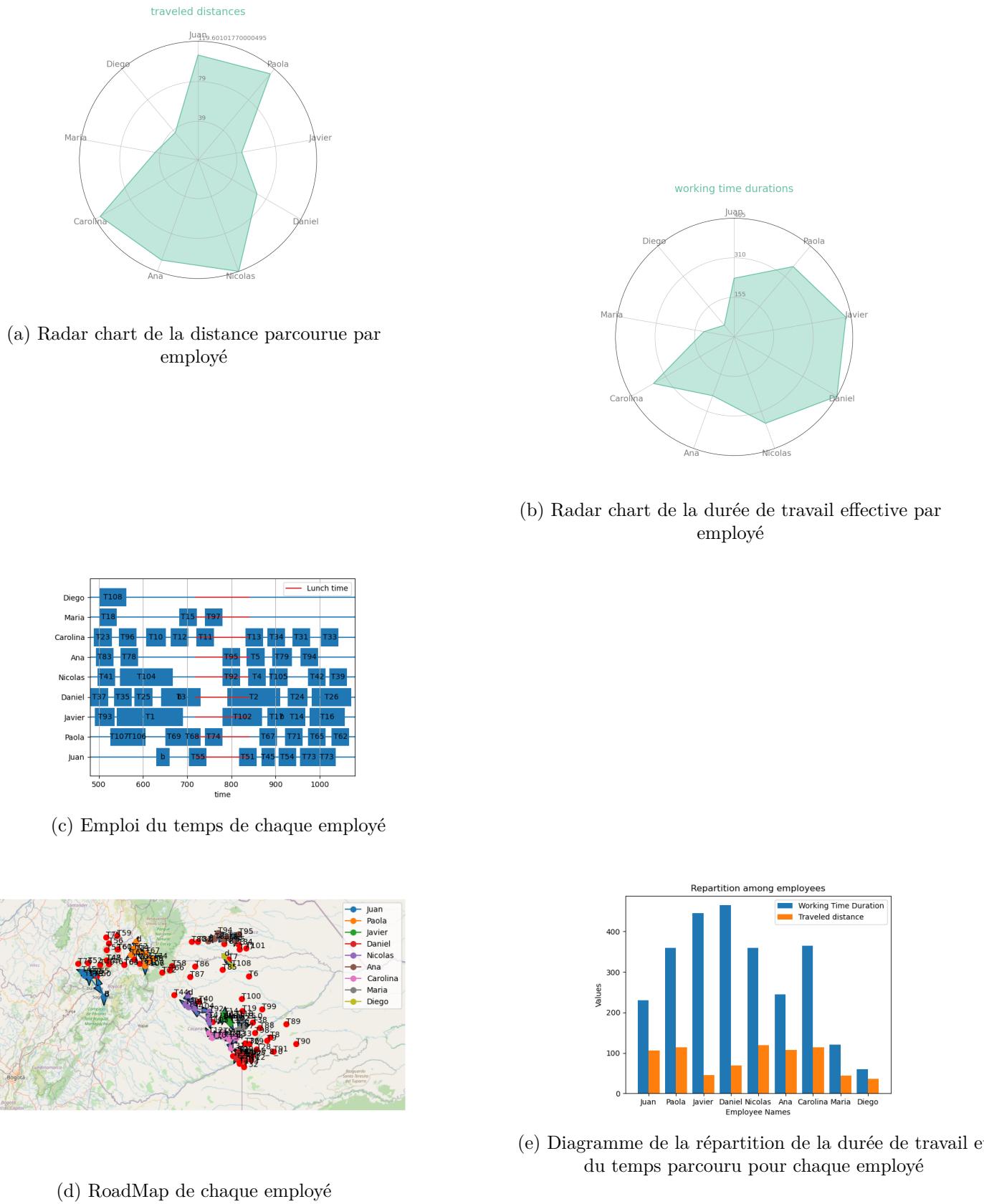


FIGURE 15 – Etude de la solution Ukraine pour la méthode des clustersv3

Une instance correcte mais nous n'avons plus le temps pour analyser les instances.

IV.2 L'algorithme Glouton

IV.2.1 Sans pauses déjeuner et indisponibilités

On notera l'ensemble des Tâches qui reste à faire : E

Voici le fonctionnement de notre code :

1. Si il reste des employées à affecter, on choisit un employée (on commence par les employées de bas niveau) et on passe à l'étape 2.
Sinon, on finit l'algorithme
2. On construit son emploi du temps à partir des tâches de E :
 - (a) On tri les tâches selon une loss (en premier lieu, le temps qu'il faut pour que l'employé puisse commencer la tâche), on choisit la Tâche qu'il peut commencer le plus tôt (on prend en compte les unavailability des tâches durant ce processus de tri) et on l'enlève de E .
Si on ne peut pas choisir de tâche qui nous permette de revenir chez soi à l'heure, on déclare que l'emploi du temps est terminé, et on revient à 1. Sinon, on passe à 2.(b)
 - (b) On met à jour le temps de début de travail et la position de l'employee et on revient à 2.(a)

L'intérêt de l'algorithme glouton est que nous sommes sûr de la faisabilité de la solution à chaque étape et donc de la faisabilité de la solution finale.

L'inconvénient de cette méthode est que les solutions proposées ne sont pas optimales. Nous verrons néanmoins comment améliorer l'efficacité de cette heuristique en utilisant des critères à notre loss pour orienter l'algorithme.

IV.2.2 Ajout de la pause déjeuner

Pour ajouter le lunch break :

On ajoute une étape entre 2.(a) et 2.(b)

2.(a – b) Si la tâche choisie en 2.(a) finit après 13h, alors on mange plutôt que de la faire. On stocke l'endroit dans l'emploi du temps où on mange et on relance la fonction en ajoutant 1h à la suite de la création de l'emploi du temps. Puis on revient à 2.(a)

IV.2.3 Ajout des indisponibilités

Pour ajouter les indisponibilités des employés : Cette fois on modifie l'étape 1 :

1'. On choisit un employé puis on considère chaque séquence de travail comme une journée de travail pour l'employé. Ainsi,

on itère l'action 2 sur ses différentes périodes de travail avant de revenir à 1'

IV.3 Des heuristique parallèle

Nous avons cherché à améliorer les résultats de l'algorithme Glouton en s'aider d'heuristique.

Voici les deux heuristiques que nous avons implémentés

IV.3.1 algorithme fixed_point

L'algorithme fixed point est une méta-heuristique qui va lancer ar itérations des algorithmes gloutons.

Il va donner des récompenses à choisir des Tâches non par les résultats de glouton précédents (et ce de manière en partie aléatoire).

Ainsi, cet algorithme va forcer les algorithmes gloutons a explorer plus de possibilités.

On se rend compte ne faisant tourner cet algorithme qu'il permet d'améliorer les solutions de l'algorithme glouton.

IV.3.2 Algorithme opti_local

Cet algorithme prend une solution en entrer et va chercher à supprimer les agendas des plus mauvaises employées (en ajoutant une part d'aleatoire dans le processus) et les réaffecter.

On se rend compte que l'algorithme n'arrive pas à améliorer les résultats des solutions.

On peut l'expliquer du fait que les employées ont accès qu'à un pool de tâches très faible. Il est donc très difficile d'améliorer leur solution (comme spatialement les tâches sont très éloignées)

IV.4 Conclusion

L'algorithme de clustering ne semble pas donner de meilleurs résultats que la résolution brute de gurobi. Nous pensons que cette méthode peut être meilleur dans le cas où la fonction objectif est plus subtil. Nous pensons aussi que un clustering selon le temps de chaque tâche serait une possibilité pour obtenir de meilleurs résultats. L'algorithme glouton est assez performant pour donner des solutions faisables. Seulement, il peut ne pas donner de solution optimale pour de petites instances. Par exemple, pour Bordeaux, on ne trouve pas la meilleure instance (qu'il s'agisse d'optimiser le nombre d'heure de travail ou le nombre de tâche réalisé).

L'algorithme fixed_point nous permet de palier ce problème, en effet, par son côté aléatoire et sa tendance à tester des solutions qui ne semblerait pas optimal, on arrive à trouver les meilleures solutions sur des petites instances.

De même, sur des grosses instances, fixed_point améliore de manière non négligeable les solutions obtenues.

L'algorithme glouton s'exécute très rapidement sur toutes les instances données (moins de 2 secondes pour les plus grosses instances).

On arrive donc en moins de 3 minutes à faire tourner l'algorithme fixed point sur 200 gloutons.