

UNIVERSITÉ PAUL SABATIER

M1 MAPI3 - ANNÉE SCOLAIRE 2019/2020

STAGE DU 27 AVRIL AU 31 JUILLET 2020 CHEZ TEAMBER

Développement de fonctionnalités pour un logiciel de gestion de projet

Auteur : Antoine PERROT

Maître de stage : Sébastien
PERROT

Enseignant tuteur : Luca
AMODEI

Remerciements

J'aimerais remercier l'ensemble des enseignants qui ont été les miens à l'université Paul Sabatier, tout particulièrement lors de cette année 2019/2020 : messieurs Amodei, Malgouyres et Maréchal. Pour leur écoute et les connaissances qu'ils m'ont apportées je leur suis très reconnaissant.

Dans le contexte de ce stage, j'ai pu échanger régulièrement avec l'ensemble des membres de l'équipe Teamber afin de créer des fonctionnalités. Ils m'ont, par leur expérience, guidé dans mon travail afin de développer des fonctionnalités répondant à de réels besoins clients. Pour l'intérêt qu'ils ont accordé à mon travail, ainsi que pour l'accueil qu'ils m'ont offert dans leur équipe pendant ces 3 mois, je voudrais les remercier sincèrement.

Table des matières

1 Introduction

1.1 Courte présentation de l'entreprise

Le logiciel TEAMBER développé par l'entreprise du même nom est destinée à des entreprises de tailles humaines, relevant principalement des professions intellectuelles (architectes, avocats, bureau d'études...).

Les objectifs sont multiples :

- fluidifier la communication au sein de l'entreprise.
- permettre un meilleur suivi de l'avancement des tâches, des temps, dans l'entreprise.
- avoir une vision à court et long termes de l'état de la situation économique de l'entreprise
- permettre une meilleure organisation et répartition des tâches dans l'entreprise, afin de mieux se focaliser sur les tâches essentielles et éliminer toutes les potentielles perturbations venant dégrader la productivité.

1.2 Visée du stage

L'objectif de ce stage était le développement de fonctionnalités qui pourront être intégrées à la prochaine version de Teamber, qui devrait être achevée fin 2020.

Ces nouvelles fonctionnalités enrichiront le logiciel en répondant à de réels besoins des clients. Dans leur ensemble, elles contribueront pour l'entreprise qui les utilisera, à une plus grande efficacité au travail, en automatisant ou facilitant un certain nombre de tâches qui relèvent de la gestion et de l'organisation.

2 Première mission : aide à la classification de mails

2.1 Présentation de la mission

Le premier objectif qui m'a été confié, a été le classification de mail selon leur catégorie. Explications :

L'utilisateur de TEAMBER, typiquement un membre d'un cabinet d'architecture, reçoit constamment des mails concernant différents projets, répertoriés dans Teamber.

Pour chacun de ces projets, l'on peut connaître ses différents paramètres, son état d'avancement etc. ainsi que l'ensemble des documents (appelés documents mais sont en fait des mails) interagissant avec ce projet.

Teamber n'ayant pas de boite mail intégrée, fait le pont avec le boite mail Outlook du l'utilisateur. Ainsi, quand un mail arrive dans la boite Outlook, quelques instants plus tard il arrive dans Teamber dans le répertoire "A traiter". Lorsque l'utilisateur classe un mail dans Outlook dans un dossier du repértoire Outlook, ce même mail est classé dans le dossier du même nom dans Teamber et à défaut supprimé. Réciproquement, si un mail est classé dans Teamber alors il est classé dans Outlook. L'arborescence des projet de Teamber est contenue dans celle de la boîte de réception Outlook.

L'intérêt principal de ma mission est de supprimer (ou de diminuer au maximum) une tâche chronophage et réberbative, et d'aller dans le sens d'une meilleure organisation générale.

En effet, classer les mails est indispensable pour beaucoup de clients Teamber, afin de rester organisé et ne pas être submergé par le flux constant de mails. Certaines personnes utilisent leur boîte mail comme "To-do list", mais dès que le flux devient trop important ce fonctionnement devient rapidement impossible, certaines études suggèrent même qu'à partir de 7 ou 8 mails, l'on est dépassé.

Cependant, si l'on veut resté organisé, mais que l'on a une cinquantaine de projets en cours, l'on passe alors énormément de temps à faire défiler son écran pour classer chaque mail.

L'idéal serait alors qu'une fois le mail sélectionné, le projet correspondant suggéré par Teamber soit accessible immédiatement. Nous tenterons donc de suggérer le bon projet à chaque nouveau mail.

2.2 Collecte des données

Après être allé collecter les données dans une base de données Drupal, les avoir recoupées dans Microsoft Access, j'ai pu obtenir des données propres afin de pouvoir travailler dans python.

Voici ce à quoi ressemble les données :

	Objet	Expéditeur	Destinataire	Projet
0	Merci d'avoir modifié TEAMBER, Logiciel gestio...	noreply-maps-issues@google.com	alexis.cavaroc@teamber.fr	AGENCE DE COMMUNICATION
1	Vielen Dank, dass Sie Änderungen für TEAMBER, ...	noreply-maps-issues@google.com	alexis.cavaroc@teamber.fr	AGENCE DE COMMUNICATION
2	[TEAMBER] Notes réunion 08 Janvier	b.croquin@labsoft.fr	i.bancel@labsoft.fr	DEVELOPPEMENT
3	[TEAMBER] Notes réunion 08 Janvier	b.croquin@labsoft.fr	teamber@teamber.fr	DEVELOPPEMENT
4	[TEAMBER] Notes réunion 08 Janvier	b.croquin@labsoft.fr	teamber@labsoft.fr	DEVELOPPEMENT

2.3 Formatage des données

Pour transformer les données textuelles en vecteur de nombre, on utilise l'objet **TfidfVectorizer** qui effectue un décompte des mots, calcule leur Inverse Document Frequency et leur attribue un score Tf-idf.

Lors de l'entraînement du modèle, on appellera cet objet avec l'extension **.fit_transform(X_train)** d'abord pour qu'il transforme les données textuelles en vecteurs de \mathbb{R}^n mais aussi pour qu'il se constitue un vocabulaire (ensemble de mots rencontrés). En effet, lorsqu'il s'agira d'effectuer des prédictions sur **X_test**, il faut que celui-ci ait les mêmes dimensions. La taille de ce vecteur est exactement le nombre de mots du vocabulaire. On appellera l'extension **.transform(X_test)** afin de transformer les données de test correctement.

2.4 Choix du modèle

Après avoir transformé nos données textuelles et séparé les données en train/test dans les proportions 80/20, on entraîne différents modèles afin de comparer les scores et choisir le meilleur :

```
models = [RandomForestClassifier(n_estimators=500, max_depth=5, random_state=0),
          LinearSVC(),
          MultinomialNB()]

print("#mails pour training :", X_train_vect.shape[0], '\n#projets : ', y_train.max())
for model in models:
    model_name = model.__class__.__name__
    model.fit(X_train_vect, y_train)
    print('Accuracy', model_name, ':', model.score(X_test_vect, y_test))
```

OUTPUT :

```
#mails pour training : 135806
#projets : 133
Accuracy RandomForestClassifier : 0.3800954288407163
Accuracy LinearSVC : 0.9151154571159283
Accuracy MultinomialNB : 0.8030749293119699
```

On retient donc le modèle LinearSVC qui est de loin le meilleur avec 91% de prédictions correctes.

Rémarque : on a essayé d'augmenter la précision en constituant un top 3 pour un mail donné. On a constaté que le bon projet était dans ce top 3 dans 92% des cas. En prenant en compte le fait qu'il faudra intégrer cette fonctionnalité de prédiction à la boîte mail (menu déroulant pour le top 3 contre simple ligne pour proposition unique) et que le gain de précision n'était pas significatif, il a été décidé de ne pas retenir l'option du top 3.

3 Deuxième mission : optimisation de la répartition des tâches

But : répartir le plus intelligemment possible le travail (les tâches) entre les utilisateurs lors de la planification d'un sprint.

Dans Teamber, une tâche :

- fait référence à une unique compétence.
- fait référence à un unique projet.
- a une durée.

Plus précisément, on va répartir les heures de travail contenues dans les tâches. Ainsi, une première personne pourra réaliser 3h d'une tâche et une seconde 4h de cette même tâche, qui faisait initialement 7h.

Dans Teamber, un utilisateur :

- a des compétences (matrice de compétences).
- dispose d'une quantité maximale (en heure) de travail qu'elle peut réaliser au cours du sprint à venir.
- est impliqué dans un ensemble de projets.

FIGURE 1 – Exemple de matrice de compétences

		Compétences							
		génériques		techniques			humaines		
		A	B	C	D	E	F	G	H
Personnes	Mia L.	1	3	4	3	3	1	1	4
	Joséphine U.	3	3	4	0	3	4	3	2
	Luc D.	3	2	1	0	4	1	0	1

Voici un exemple de matrice de compétences trouvé sur google image, chez Teamber les scores sont compris entre 0 (ne sait pas faire) et 3 (maîtrise parfaite, apte à la formation) mais le principe reste le même.

On a donc un ensemble de tâches avec des durées d_1, \dots, d_n et un ensemble de personnes p_1, \dots, p_m sur lesquels répartir ces durées. Notons u_1, \dots, u_m le nombre d'heures maximale que peuvent effectuer chacune des personnes.

3.1 Politique d'assignation

Notre première politique d'assignation des tâches sera la suivante : une personne P pourra se voir assigner une tâche T si : le score pour la compétence à laquelle fait référence la tâche est d'au moins $1 / 3$, et si P est impliquée dans le projet auquel T fait référence.

Ensuite, si P est en mesure de réaliser T, un bénéfice sera ensuite déterminé quant au fait d'assigner une heure de la tâche T à la personne P. Le choix de ce bénéfice sera expliqué dans les prochains paragraphes.

A l'issue de cette première répartition des tâches, il est possible que certaines heures n'aient pu être assignées.

Si tel est le cas, on proposera à l'utilisateur de tenter des réassigner les tâches restantes en appliquant une politique d'assignation moins restrictive que la précédente.

Dans cette seconde politique, on omettra la composante projet lors de l'assignation des tâches. Ainsi une tâche aura plus de destinataires potentiels.

Bien entendu, cela ne garantit pas que l'on pourra assigner toutes les heures de travail.

3.2 Détermination des bénéfices lors des assignations

Dans cette partie, nous allons expliquer comment pourra être déterminer le bénéfice concernant l'assignation d'une tâche T relevant de la compétence C, à une personne P.

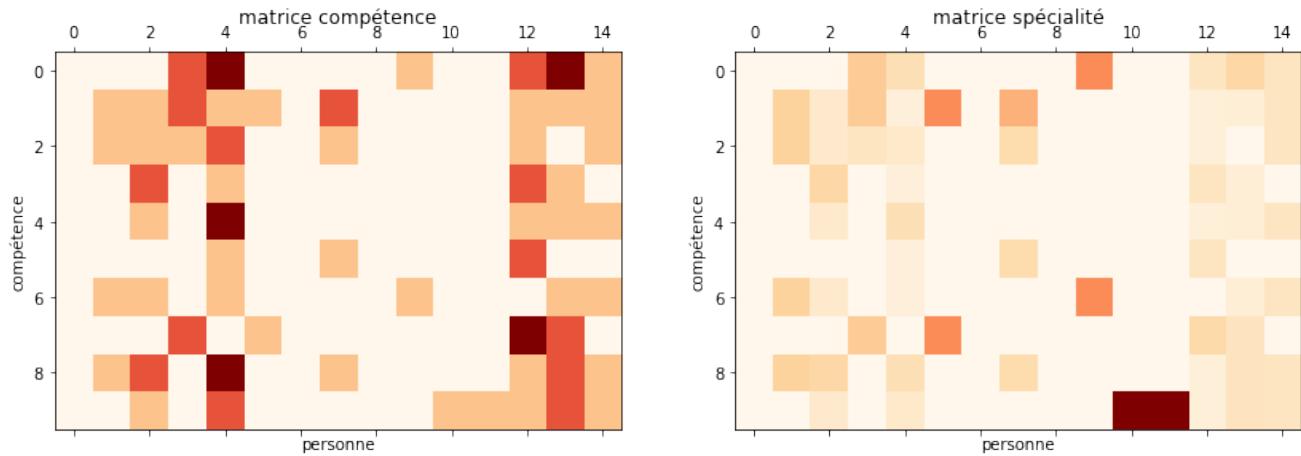
Première approche : la tâche doit être effectuée par la personne la plus compétente possible. Ainsi, aucune difficulté, le bénéfice est égal au niveau de compétence de P pour C.

Deuxième approche : la tâche doit être effectuée par la personne la plus spécialisée possible. Le but de cette approche est de charger d'abord les éléments les plus monovalents de l'entreprise, même s'ils ne sont pas les meilleurs dans leur discipline. Premièrement, ils seront donc occupés, et deuxièmement, les personnes les plus polyvalentes de l'entreprise pourront se consacrer à d'autres tâches.

Par exemple, si le chef de projet est meilleur dessinateur que la nouvelle recrue dessinateur, il est néanmoins préférable de confier un travail de dessin à cette recrue et d'épargner le chef de projet qui lui sait faire beaucoup d'autres choses.

Afin de faire ressortir les spécialités propres à chaque personnes, on va modifier la matrice de compétences.

FIGURE 2 – Normalisation en colonne (par personne) de la matrice de compétence

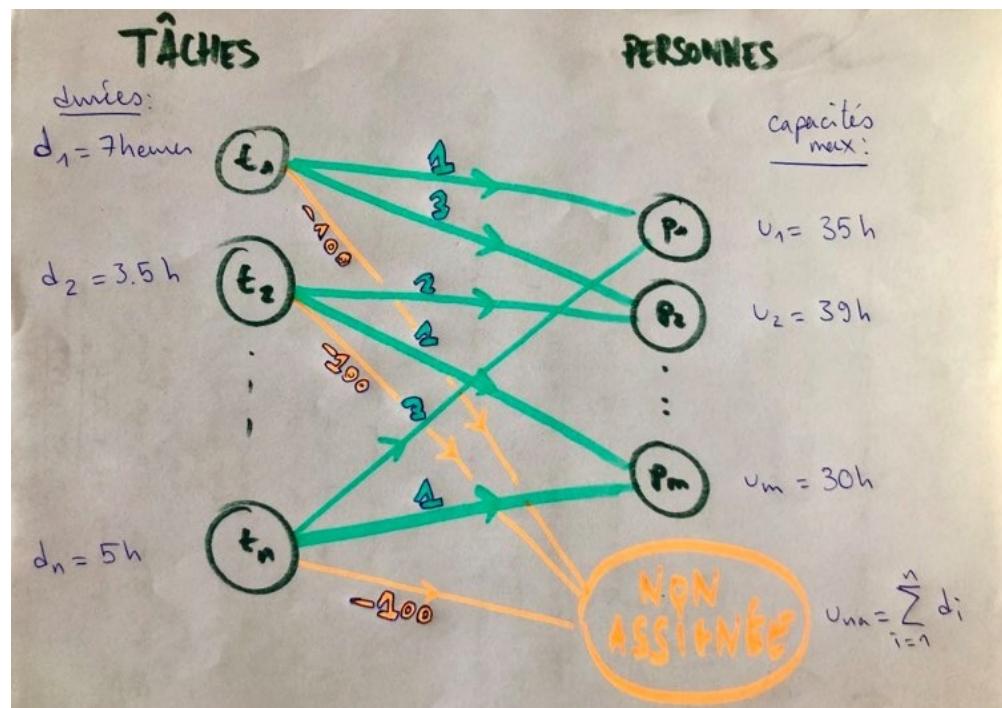


La matrice de droite est en fait la matrice de gauche normalisée (L1) en colonne. Une colonne correspond aux niveaux de compétences d'une personne.

Ici par exemple, une tâche relevant de la compétence inscrite à la première ligne devrait être attribuée aux personnes 4 ou 13 (très polyvalentes) selon la première approche, tandis qu'avec la seconde approche cela devrait être la personne 9, plus monovalente.

Heures non assignées : il se peut que certaines heures de certaines tâches ne peuvent trouver preneur. Pour modéliser cette situation et garantir l'existence d'une solution mathématique, on pourra assigner une tâche à une personne fictive qui correspondra à une "non assignation". On attribuera à cette action un bénéfice de -100 afin de la pénaliser.

FIGURE 3 – Illustration de la situation



3.3 Formulation du problème

On a donc des heures qui doivent partir des tâches pour aller vers les personnes (fictives ou non) en empruntant des arcs plus ou moins avantageux. Egalement, le flot que reçoit chaque personne ne doit pas excéder une certaine quantité.

Notons :

- x_{ij} le flot sur l'arc $(i, j) \in \mathcal{A}$. Ici l'unité de flot est l'heure.
- f_{ij} le bénéfice sur l'arc $(i, j) \in \mathcal{A}$.

On cherche alors à maximiser la quantité suivante :

$$\sum_{(i,j) \in \mathcal{A}} x_{ij} f_{ij}$$

Sous les contraintes :

$$\underbrace{\forall i = 1, \dots, n \sum_{j|(i,j) \in \mathcal{A}} x_{ij} = d_i}_{\text{Distribution de toutes les heures}}, \quad \underbrace{\forall j = 1, \dots, m \sum_{i|(i,j) \in \mathcal{A}} x_{ij} \leq u_j}_{\text{Respect des capacités de travail}}, \quad \underbrace{\forall (i, j) \in \mathcal{A}, x_{ij} \geq 0}_{\text{Positivité}}$$

Pour mettre le problème sous forme standard c'est à dire uniquement avec des contraintes d'égalités. On doit donc rajouter m variables d'écart. Les contraintes d'inégalités se ré-écrivent donc de la manière suivante :

$$\forall i = 1, \dots, m, \quad y_i + \sum_{i|(i,j) \in \mathcal{A}} x_{ij} = u_j, \quad y_i \geq 0.$$

On a donc ré-écrit le problème sous la forme suivante :

$$\begin{aligned} & \min -f^t x \\ & \text{s.c. } Ax = b \\ & \quad x \geq 0 \end{aligned} \tag{1}$$

avec $x \in \mathbb{R}^{card(\mathcal{A})+m}$, $A \in \mathbb{R}^{(n+m) \times (card(\mathcal{A})+m)}$, et $b \in \mathbb{R}^{n+m}$.

3.4 Résolution dans python

3.4.1 Exemple - Première approche

En plus de résoudre simplement le problème d'optimisation, on permet au futur l'utilisateur du logiciel d'apprécier quelques indicateurs concernant la solution fournie par le programme.

DONNEES :

```
Charge totale de travail : 445 h pour 70 tâches avec 15 compétences possibles.  
(6 h/tâche).  
Disponibilités totales : 526 h réparties sur 16 personnes  
(32 h/personne).  
Approche choisie : 1
```

SORTIE :

```
Total heures non assignées : 66.0  
Niveau moyen de spécialisation pour une heure travaillée : 0.66 / 3  
Niveau moyen de spécialisation général : 0.19 / 3
```

```
Niveau moyen de compétence pour une heure travaillée : 1.25 / 3  
Niveau moyen de compétence général : 0.45 / 3
```

Statut résolution : Optimization terminated successfully.

Méthode de résolution : simplex
fonction objectif : 6128.0

MESSAGE : CERTAINES HEURES N'ONT PU ETRE ASSIGNEES. SOUHAITEZ-VOUS CONTINUER
AVEC UNE POLITIQUE MOINS RESTRICTIVE ?

Remarques et observations :

- 66 heures de travail n'ont pû être assignées, on propose donc à l'utilisateur de les assigner en étant moins restrictif.
- le niveau de spécialisation est un indicateur que l'on créa grâce à la matrice de spécialisation créée précédemment. Il sera intéressant de voir son évolution dans la simulation suivante lorsque l'on aura choisie la deuxième approche pour la conception de la fonction objectif.
- avec la solution fournie, une heure de travail sera réalisée au niveau de compétence 1.25 / 3 en moyenne. Cette valeur dépend d'une multitude de facteurs : nature des tâches, disponibilités et compétences des utilisateurs, etc.

Ces indicateurs permettront par exemple à un chef d'équipe d'apprécier la bonne composition de son équipe.

En effet, cela pourra l'aider à se rendre compte de plusieurs choses : la charge de travail est-elle disproportionnée par rapport à mon effectif ? Manquerais-je d'architectes ? Les bonnes personnes sont-elles sur les bons projets ?

Tout cela contribuera à une vision d'ensemble de qualité sur la santé de l'entreprise, ce qui est exactement la visée du logiciel Teamber.

3.4.2 Exemple - Seconde approche

On reprend le problème précédent en choisissant la seconde approche, afin d'observer comment cela affecte la forme de la solution.

DONNEES :

Charge totale de travail : 445 h pour 70 tâches avec 15 compétences possibles.
(6 h/tâche).

Disponibilités totales : 526 h réparties sur 16 personnes
(32 h/personne).

Approche choisie : 2

SORTIE :

Total heures non assignées : 66.0

Niveau moyen de spécialisation pour une heure travaillée : 0.72 / 3

Niveau moyen de spécialisation général : 0.19 / 3

Niveau moyen de compétence pour une heure travaillée : 1.23 / 3

Niveau moyen de compétence général : 0.45 / 3

Statut résolution : Optimization terminated successfully.

Méthode de résolution : simplex

fonction objectif : 6327.719999999999

PHASE 2 NECESSAIRE

MESSAGE : CERTAINES HEURES N'ONT PU ETRE ASSIGNEES. SOUHAITEZ-VOUS CONTINUER
AVEC UNE POLITIQUE MOINS RESTRICTIVE ?

Observations :

- par rapport à l'approche précédente, le niveau moyen de spécialisation par heure travaillée a augmenté. Difficile de dire si cela est beaucoup mieux en raison de multitudes des paramètres en jeu, mais cela va dans le bon sens.
- à l'inverse, et cela était prévisible, le niveau moyen de compétence par heure travaillée a diminué (assez peu).

[Voir un autre exemple en annexe](#)

Commentaire général : toutes les données utilisées pour la résolution de ce problème sont simulées aléatoirement. J'ai tenté de créer des données qui reflèteraient le mieux possible la réalité, par exemple lors de la création d'une matrice de compétence, où certaines personnes étaient très polyvalentes et d'autres beaucoup moins.

Néanmoins cet effort de représentativité de la réalité a ses limites. Il sera intéressant d'écouter les retours des futurs utilisateurs de cette fonctionnalité, comme pour toutes les fonctionnalités que j'ai développées lors de ce stage.

4 Troisième mission : suggestion d'emplois du temps

4.1 Présentation de la mission

Le deuxième objectif a été le suivant. La 2ème version de TEAMBER sera organisée de telle manière que l'on connaîtra la charge de travail pour un groupe de collaborateurs donné (une équipe). L'enjeu de cette mission est alors de proposer un sprint au chef d'équipe.

Un sprint est une planification très précise des tâches qu'une équipe doit réaliser sur une période courte, généralement une semaine mais pouvant aller jusqu'à la quinzaine de jours. En pratique, de nombreuses entreprises s'organisent déjà en sprint sans le savoir car elles planifient le travail par période d'une ou deux semaines.

Mais cette planification peut être fastidieuse (2 à 3h chaque lundi matin pour un chef d'équipe par exemple), c'est la raison pour laquelle être capable de proposer une organisation suffisamment logique de leurs sprint aux utilisateurs constituerait un véritable atout pour la deuxième version du logiciel TEAMBER.

4.2 Données du problèmes et contraintes

Le sprint lui-même a une date de début et de fin, ainsi qu'une liste de collaborateurs qu'il implique.

Données dont nous disposons pour chaque collaborateur :

- une liste des tâches à programmer dans la période de sprint donnée ; chaque tâche a un niveau de priorité et une durée.
- ses horaires de travail. Exemple :

	A	B	C	D
1	Nom	Jour	Heure début	Heure fin
2		0	0	08:00:00
				12:00:00

Ici le collaborateur indique que le lundi matin, il commence à 8h00 et termine à 12h00. Nous appellerons cela une plage horaire d'un collaborateur.

- les événements déjà planifiés par chacun des collaborateurs sur la période de sprint concernée et qu'ils ne peuvent déplacer, nous les appellerons des "impératifs". Il est préférable pour un collaborateur d'en avoir le moins possible, afin de laisser au mieux le programme construire l'organisation du sprint.
- sur l'ensemble des tâches de tous les collaborateurs, pourra exister des relations de type mère-fille. Cela pour exprimer que dans certaines situations, Monsieur A doit terminer la tâche X avant que Madame B ne commence la tâche Y. Ces tâches ne représenteront qu'une très faible proportion de toutes les tâches. Nous qualifierons ces tâches de "mère-fille".

Il faut aussi penser au fait que les profils des futurs utilisateurs de Teamber pourront être différents : certains préféreront respecter l'ordre des priorités, d'autres tiendront à avoir des emplois du temps les plus remplis possibles. Nous verrons comment nous tiendrons compte de ces préférences dans la section suivante.

[Voir la librairie rédigée pour la résolution du problème](#)

4.3 Impératifs et tâches mère-fille

4.3.1 Prise en compte des impératifs

Cela constituera la première phase de nos calculs : construire la "base" de l'emploi du temps. Cette base est l'ensemble des plages horaires disponibles restantes à un collaborateur une fois ses horaires et ses impératifs pris en compte. Cette première étape consiste à intersecter chaque impératif avec les plages horaires disponibles de chacun des collaborateurs.

4.3.2 Gestion des tâches mère-fille (TMF)

Dans notre problème, la prise en compte de ces tâches mère-fille (TMF) constitue une contrainte raide, qu'il faut absolument respecter ; contrairement aux autres contraintes, plus souples.

Précisément il faut respecter : l'arrangement des tâches, et les disponibilités de chaque utilisateur. De plus, rien n'indique qu'une solution à ce problème existe et il me semble que c'est assez difficile à déterminer rigoureusement.

Le problème est alors le suivant : planifier cette suite de tâche dans les emplois du temps des collaborateurs, sans jamais qu'elles ne se chevauchent dans le temps et qu'elles respectent les horaires de travail des collaborateurs auxquels elles sont assignées. La priorité absolue de ce problème est le respect de l'arrangement des tâches, ainsi nous sacrifierons si nécessaire quelques quarts d'heures de présence en plus pour respecter cette contrainte.

Nous considérerons que le temps zéro $t_0 = 0$ est le début du sprint, et que le temps final t_f est la longueur en heures du sprint. Notons également $(t_i)_{i=1,\dots,n}$ les dates de début (nombres d'heures depuis le début du sprint) des n tâches à planifier, $(d_i)_{i=1,\dots,n}$ leur durée et $(I_i)_{i=1,\dots,n}$ la suite de listes de tous les intervalles candidats à contenir chacune des dates de début.

Méthode de résolution :

- générer aléatoirement (t_1, \dots, t_n) suivant la loi des statistiques d'ordre sur $[t_0, t_f]$
- projeter chaque t_i dans l'intervalle de I_i le plus proche
- projeter (t_1, \dots, t_n) dans l'espace vérifiant $t_i + d_i \leq t_{i+1}, \forall i = 1, \dots, n - 1$.
- calculer un score : le nombre total d'heures séparant les t_i de leur intervalle le plus proche
- répéter ce procédé 100 fois (par exemple) et garder le meilleur tirage que l'on ait eu.

Sous réserve que l'utilisateur du logiciel fournit des données cohérentes : que les disponibilités des utilisateurs, que la charge de travail qui leur ait attribuée et que la période du sprint soit en phase, alors cette méthode de résolution se montre très efficace et rapide.

Ensuite on place ces tâches dans les emplois du temps et on s'attaque à la planification des autres tâches.

4.4 Optimisation d'un emploi du temps avec l'algorithme de recuit simulé

Avant de déterminer quelle tâche doit être faite à quel moment, on va découper ces tâches en tranches d'une durée maximale choisie par l'utilisateur.

Par exemple, une tâche de 3h30 avec une tranche maximale de 1h sera découpée en 3 fois 1h et 1 fois une demi-heure. Ce découpage offre des avantages :

- il se peut que certaines tâches aient une durée trop grande (12h par exemple) et ne puisse être casées à aucun endroit de l'emploi du temps.
- cela permet de mieux remplir l'emploi du temps. En effet, essayer d'arranger des tranches d'1h dans les disponibilités laissera sans doute moins de trous dans l'emploi du temps final que si l'on travaillait avec des durées plus grandes. On perdra donc moins de temps.
- la sensation de continuité dans le travail sera améliorée. Cette étape permettra de créer des situations réalistes comme : "je commence cette tâche avant manger et je continuerai après manger" ou encore "je terminerai cela demain matin", apportant du naturel dans l'organisation du travail.

Le premier inconvénient auquel on s'expose est la dispersion d'une tâche dans le temps. Par dispersion on entend le fait de réaliser une tâche en plusieurs fois, comme la commencer le lundi matin, la reprendre le mercredi et l'achever le vendredi. Ce genre de situation est à éviter, et nous verrons que la plupart du temps nous y parviendrons.

Le second inconvénient est l'augmentation du nombre de tâches (comprendre morceaux de tâches désormais) à planifier.

Cela dit, il s'agit maintenant de mettre pour un collaborateur, la liste de ses tâches bout à bout, et de regarder la qualité de cet arrangement. Laisse-t-il beaucoup de trous dans l'emploi du temps ? Tient-il suffisamment compte de l'ordre des priorités ? Y a-t-il une bonne continuité comme expliqué précédemment ?

Le nombre d'arrangements possibles des tâches étant rapidement trop important, il m'a paru adapté d'utiliser l'algorithme de recuit simulé en définissant une énergie associée à chaque arrangement, pénalisant ceux qui conduisent à de "mauvais" emplois du temps et qui serait minimisée au fur et à mesure des itérations.

4.4.1 L'algorithme de recuit simulé

Définition (wikipedia) : en algorithmique, le recuit simulé est une méthode de programmation empirique (métaheuristique) inspirée d'un processus utilisé en métallurgie. On alterne dans cette dernière des cycles de refroidissement lent et de réchauffage (recuit) qui ont pour effet de minimiser l'énergie du matériau. Cette méthode est transposée en optimisation pour trouver les extrema d'une fonction. Elle a été mise au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983, et indépendamment par V. Černý en 1985.

Principe : partant d'un état initial x , et d'une température initiale T , on génère à chaque itération un état candidat y voisin de x (on considère que la loi de transition est symétrique). Si cet état candidat a une énergie plus faible que x ($\Delta E < 0$), alors il le remplace ; sinon, il le remplace avec probabilité $\exp\left(\frac{-\Delta E}{T}\right)$. A la fin, de cette itération, on fait décroître la température T et si x est le meilleur état que l'on ait rencontré, on le sauvegarde dans une variable x^* . On répète cette opération tant qu'une condition d'arrêt n'est pas vérifiée.

Pour notre problème :

- pour générer un candidat y voisin de x , on permutera deux tâches au hasard dans l'arrangement x . Cette manière de transiter d'un état à l'un de ses voisins est bien symétrique.
- notre critère d'arrêt sera un simple nombre d'itérations atteint du type : NOMBRE DE TACHES TOTAL \times 100. 100 est ici arbitraire.
- on fera décroître la température de manière géométrique jusqu'à atteindre une température minimale à la fin des itérations.

4.4.2 Calcul des énergies

En notant a l'arrangement des tâches d'un utilisateur et λ ses préférences, on calcule l'énergie de cet arrangement :

$$E(a, p) = \lambda_p V_p(a) + \lambda_t V_t(a) + \lambda_d V_d(a)$$

Les fonctions V calculent un potentiel entre 0 et 1 de l'arrangement et les abréviations "p", "t" et "d" correspondent respectivement aux non respect des priorités, au temps perdu et au niveau de dispersion qu'engendre l'arrangement a .

Les coefficient de pénalisation sont une combinaison convexe, et permettront aux utilisateurs d'indiquer leurs préférences.

Il est très important que tous les quantités que l'on manipule soit normalisées (comprises entre 0 et 1). Cela permet de donner lieu à des ΔE toujours du même ordre quelque soit la configuration, et d'avoir toujours la même gestion de la température.

Calcul du potentiel "Priorités" : admettons que notre arrangement des tâches est tel que le vecteur des priorités est celui-ci : (5, 2, 1, 3, 4, 6, 4). L'arrangement idéal aurait ce vecteur de priorités : (1, 2, 3, 4, 4, 5, 6). On calcule alors la différence en norme $L1$ de ces deux vecteurs, que l'on divise une première fois par le nombre de niveaux de priorités différents (6 ici) puis par le nombre de tâches au total (7 ici).

Calcul du potentiel "Temps perdu" : admettons que les tâches aient ces durées en heures (1.75, 0.5, 1.25, 0.75, 2, 1.5...) et que les plages horaires, aient ces durées : (4, 3.5, 2, 2...). Il s'agit alors de remplir la plage horaire courante tant qu'on le peut, autrement on passe au remplissage de la plage horaire suivante. Une fois une plage remplie, on compte le temps non utilisé de cette plage. On s'arrête une fois que toutes les tâches ont été comptabilisées (on ne compte pas alors le temps restant jusqu'à la fin du sprint comme du temps perdu) ou dès qu'on a épousé toutes les plages. On retourne alors :

- un potentiel pour le temps perdu : la somme des temps perdus sur chaque plage horaire utilisée divisée par le temps total disponible sur ces mêmes plages.
- un vecteur contenant la correspondance entre l'identifiant de la tâche et l'identifiant de la plage horaire dans laquelle elle tombe.

Calcul du potentiel "Dispersion" : considérons une tâche de 3h30 qui peut être réalisée en au maximum 4 fois ($3 * 1\text{h} + 1 * 0.5\text{h}$). Si elle est réalisée en 4 fois, alors son taux de dispersion est de 100%, en 3 fois : 66%, 2 fois : 33% et 1 fois : 0%. On fait la moyenne de ces taux de dispersion pour les tâches qui peuvent être réalisées en plusieurs fois. On considère ici qu'une interruption est simplement le fait de se consacrer à une autre tâche avant de reprendre la première, ainsi le fait de reprendre une tâche après manger, le lendemain en revenant au travail, ou après un impératif ne constitue pas une interruption.

4.5 Simulation - Planification de tâches mère-fille

Dans cet exemple, tous les utilisateurs travaillent de 8h à 12h et de 13h à 17h chaque jour de la semaine. Ils ont tous les mêmes impératifs que l'on peut apercevoir plus bas.

Dans un premier temps, on indique l'équipe à concernée par les tâches que l'on souhaite affecter et les tâches elle même.

Dès lors l'utilisateur n'intervient plus dans le programme.

```
from Planning import *          #import librairie
group_file = 'exemple_groupe_de_travail.xlsx'  #choix de l'équipe
TMF_file   = 'exemple_TMF.xlsx'    #fichier contenant les tâches à planifier

DATE_DEBUT = '2020-05-11 08:30:00'      #date de début du sprint
DATE_FIN   = '2020-05-16 18:00:00'      #date de fin du sprint
```

On importe le groupe dont on va planifier les tâches mère-fille :

```
group = pd.read_excel(group_file).drop('Unnamed: 0',axis=1)
group
```

	ID	Nom	Prénom	Fichier Horaires	Fichier Impératifs
0	0	Dumont	David	exemple_horaires.xlsx	exemple_impératifs.xlsx
1	1	Bertrand	David	exemple_horaires.xlsx	exemple_impératifs.xlsx
2	2	Gerard	Anne	exemple_horaires.xlsx	exemple_impératifs.xlsx
3	3	Mathieu	Christophe	exemple_horaires.xlsx	exemple_impératifs.xlsx
4	4	Durand	Eric	exemple_horaires.xlsx	exemple_impératifs.xlsx

On crée des objets de type planning pour travailler avec :

```
plannings = list()
for i in range(len(group)):
    P = Planning(group['Nom'][i],
                 group['ID'][i], group['Fichier Horaires'][i])
    P.initialise(DATE_DEBUT,DATE_FIN)
    P.addImperatifs(pd.read_excel(group['Fichier Impératifs'][i]))
    plannings.append(P)
```

Visualisation des tâches à planifier :

```
TMF = pd.read_excel(TMF_file).drop('Unnamed: 0', axis=1)  
TMF
```

	id tache	Objet	Durée	id utilisateur	Next tache id	Prev tache id
0	0	tache 0	0.25	2	1	-1
1	1	tache 1	3.00	3	2	0
2	2	tache 2	3.00	3	3	1
3	3	tache 3	1.00	2	4	2
4	4	tache 4	2.50	4	5	3
5	5	tache 5	1.00	0	6	4
6	6	tache 6	1.75	3	7	5
7	7	tache 7	1.00	1	8	6
8	8	tache 8	1.75	0	9	7
9	9	tache 9	3.00	0	10	8
10	10	tache 10	1.00	2	11	9
11	11	tache 11	3.00	0	12	10
12	12	tache 12	1.00	2	-1	11

Visualisation impératifs (identiques pour tous les utilisateurs) :

```
pd.read_excel('exemple_impératifs.xlsx')
```

	Objet	Priorité	Date début	Date fin
0	Rdv DENTISTE	IMPERATIF	2020-05-12 10:00:00	2020-05-12 11:00:00
1	Accueil nouvelle recrue	IMPERATIF	2020-05-13 09:30:00	2020-05-13 11:30:00
2	Rdv Expert Comptable	IMPERATIF	2020-05-14 09:00:00	2020-05-14 10:00:00
3	Rdv Banque	IMPERATIF	2020-05-15 14:00:00	2020-05-15 16:00:00
4	Réserver les vacances d'été	IMPERATIF	2020-05-17 16:00:00	2020-05-17 18:00:00
5	Restaurant avec l'équipe	IMPERATIF	2020-05-20 12:00:00	2020-05-20 12:50:00
6	Footing	IMPERATIF	2020-05-22 08:00:00	2020-05-22 09:00:00

On observe que la meilleure solution que l'on ait trouvé retire une heure de temps libre au total au groupe de travail pour de réussir à planifier ces tâches.

Enfin on peut constater que les tâches ont bien été placées dans les emplois du temps.

Planification des tâches :

```
planned_TMF = planifieTMF(TMF, DATE_DEBUT, DATE_FIN, plannings, maxiter = 1500)
for i in np.unique(planned_TMF['id utilisateur'].values):
    plannings[i].addImperatifs(planned_TMF.loc[planned_TMF['id utilisateur']==i,
                                                ['Objet','Priorité','Date début',"Date fin"]])
planned_TMF
```

Non respect des horaires en heures pour la planification des tâches mère-fille : 0.0

	Objet	Priorité	Date début	Date fin	id utilisateur
0	tache 0	TMF	2020-05-11 08:30:00	2020-05-11 08:45:00	2
1	tache 1	TMF	2020-05-11 09:00:00	2020-05-11 12:00:00	3
2	tache 2	TMF	2020-05-11 14:00:00	2020-05-11 17:00:00	3
3	tache 3	TMF	2020-05-12 08:00:00	2020-05-12 09:00:00	2
4	tache 4	TMF	2020-05-12 14:30:00	2020-05-12 17:00:00	4
5	tache 5	TMF	2020-05-13 08:00:00	2020-05-13 09:00:00	0
6	tache 6	TMF	2020-05-13 13:00:00	2020-05-13 14:45:00	3
7	tache 7	TMF	2020-05-13 14:45:00	2020-05-13 15:45:00	1
8	tache 8	TMF	2020-05-14 08:00:00	2020-05-14 09:45:00	0
9	tache 9	TMF	2020-05-14 14:00:00	2020-05-14 17:00:00	0
10	tache 10	TMF	2020-05-15 08:00:00	2020-05-15 09:00:00	2
11	tache 11	TMF	2020-05-15 09:00:00	2020-05-15 12:00:00	0
12	tache 12	TMF	2020-05-15 13:00:00	2020-05-15 14:00:00	2

Les tâches sont planifiées :

```
plannings[3].makePlanning(to_file=False)
```

	Objet	Priorité	Date début	Date fin
0	tache 1	TMF	2020-05-11 09:00:00	2020-05-11 12:00:00
1	tache 2	TMF	2020-05-11 14:00:00	2020-05-11 17:00:00
2	Rdv DENTISTE	IMPERATIF	2020-05-12 10:00:00	2020-05-12 11:00:00
3	Accueil nouvelle recrue	IMPERATIF	2020-05-13 09:30:00	2020-05-13 11:30:00
4	tache 6	TMF	2020-05-13 13:00:00	2020-05-13 14:45:00
5	Rdv Expert Comptable	IMPERATIF	2020-05-14 09:00:00	2020-05-14 10:00:00
6	Rdv Banque	IMPERATIF	2020-05-15 14:00:00	2020-05-15 16:00:00
7	Réserver les vacances d'été	IMPERATIF	2020-05-17 16:00:00	2020-05-17 18:00:00
8	Restaurant avec l'équipe	IMPERATIF	2020-05-20 12:00:00	2020-05-20 12:50:00
9	Footing	IMPERATIF	2020-05-22 08:00:00	2020-05-22 09:00:00

Ci-dessus l'emploi du temps l'utilisateur n°3.

On voit qu'on a trouvé une solution respectant toutes les contraintes.

4.6 Simulations - Optimisation d'un emploi du temps

D'abord on crée un objet planning auquel on ajoute tous les paramètres de la planification :

```
P = Planning(nom_utilisateur=nom_utilisateur,
             id_utilisateur=id_utilisateur,
             plagehoraire_filename=plagehoraire_filename)

t = pd.read_excel(tasks_filename)
tasks = split_tasks(t,1).reset_index(drop=True)
P.addTasks(tasks)
P.initialise(DATE_DEBUT,DATE_FIN)
P.addImperatifs(pd.read_excel(imperatifs_filename))
```

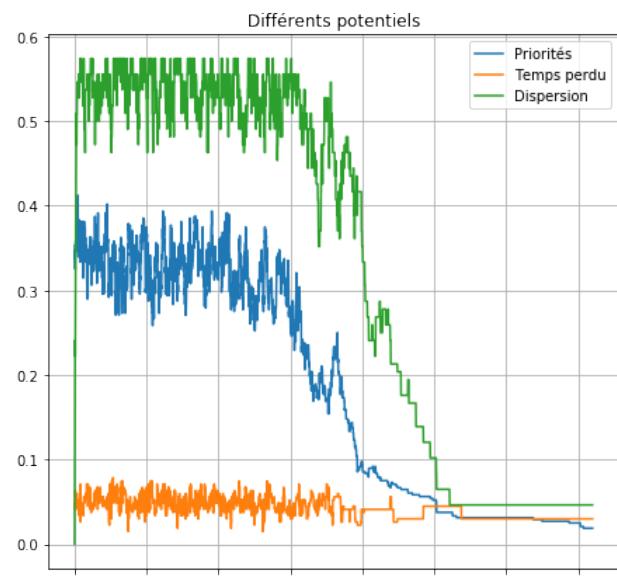
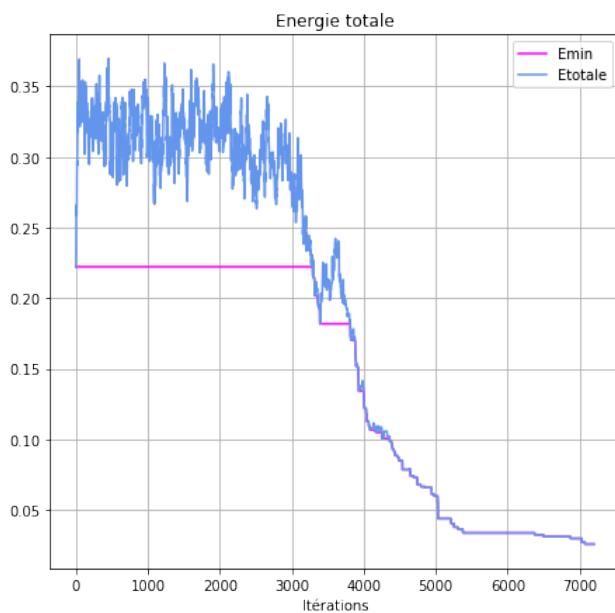
Ensuite on peut changer les préférences (qui se normalisent automatiquement) lors de l'optimisation :

```
P.setPenalties([1,4,1])
```

OUTPUT :
array([0.16666667, 0.66666667, 0.16666667])

On peut à présent lancer l'optimisation :

```
P.Optimise(show=True)
```



On voit que l'on arrive à minimiser toutes les composantes de notre critère.
Page suivante, obeservons l'emploi du temps résultant de cette optimisation.

Les horaires sont 8h-12h et 13h-17h tous les jours de la semaine ; la période de planification des tâches commence le 11 mai 2020 8h30 et se termine le 15 mai 2020 18h00.

	Objet	Priorité	Date début	Date fin
0	LABSOFT test DEVELOPPEMENT (2015-1)	0	2020-05-11 08:30:00	2020-05-11 11:00:00
1	GARCIA INGENIERIE RE: TABLEAU DE BORD FACTU ...	4	2020-05-11 11:00:00	2020-05-11 12:00:00
2	GARCIA INGENIERIE RE: TABLEAU DE BORD FACTU ...	4	2020-05-11 13:00:00	2020-05-11 13:45:00
3	R2M Préparer les imports R2M (2018-90)	5	2020-05-11 13:45:00	2020-05-11 14:30:00
4	Perrot Sébastien Vacances TEAMBER (2018-51)	3	2020-05-11 14:30:00	2020-05-11 16:15:00
5	TEAMBER Spécifications TEAMBER V2 (2019-00...)	6	2020-05-11 16:15:00	2020-05-11 17:00:00
6	TEAMBER Spécifications TEAMBER V2 (2019-00...)	6	2020-05-12 08:00:00	2020-05-12 09:00:00
7	R2M Préparer les imports R2M (2018-90)	5	2020-05-12 09:00:00	2020-05-12 10:00:00
8	Rdv DENTISTE	IMPERATIF	2020-05-12 10:00:00	2020-05-12 11:00:00
9	BADETS Claude RE: Teamber OTCE GROUPE (201...	7	2020-05-12 11:00:00	2020-05-12 12:00:00
10	BADETS Claude RE: Teamber OTCE GROUPE (201...	7	2020-05-12 13:00:00	2020-05-12 15:00:00
11	Perrot Sébastien Vacances TEAMBER (2018-51)	8	2020-05-12 15:00:00	2020-05-12 15:15:00
12	BADETS Claude Bilan_Financier.xls OTCE GRO...	10	2020-05-12 15:15:00	2020-05-12 16:15:00
13	Perrot Sébastien Rdv cardiologue SNCF (201...	9	2020-05-13 08:00:00	2020-05-13 09:30:00
14	Accueil nouvelle recrue	IMPERATIF	2020-05-13 09:30:00	2020-05-13 11:30:00
15	Perrot Sébastien Rdv cardiologue SNCF (201...	9	2020-05-13 13:00:00	2020-05-13 14:00:00
16	PELISSOU Loïc RE: Kick Off - Phase Archi (fo...	11	2020-05-13 14:00:00	2020-05-13 15:00:00
17	ROTARY CLUB LAMASQUERE PAYS DE MURET Faire m...	12	2020-05-13 15:00:00	2020-05-13 17:00:00
18	ROTARY CLUB LAMASQUERE PAYS DE MURET Faire m...	12	2020-05-14 08:00:00	2020-05-14 09:00:00
19	Rdv Expert Comptable	IMPERATIF	2020-05-14 09:00:00	2020-05-14 10:00:00
20	SAULIERES Vincent TR: EIFFAGE ROUTE / Cahier...	13	2020-05-14 10:00:00	2020-05-14 11:00:00
21	Cavaroc Alexis RE: teamber VERDI INGENIERI...	13	2020-05-14 11:00:00	2020-05-14 12:00:00
22	Cavaroc Alexis RE: teamber VERDI INGENIERI...	13	2020-05-14 13:00:00	2020-05-14 13:45:00
23	ZANON Stéphane Formation dernière demi-journ...	14	2020-05-14 13:45:00	2020-05-14 15:45:00
24	PELISSOU Loïc Labsoft in Teamber LABSOFT (...)	14	2020-05-14 15:45:00	2020-05-14 16:30:00
25	PELISSOU Loïc Labsoft in Teamber LABSOFT (...)	14	2020-05-15 08:00:00	2020-05-15 09:00:00
26	ZANON Stéphane Formation dernière demi-journ...	14	2020-05-15 09:00:00	2020-05-15 10:00:00
27	Semetey Cécile RE: Cloud Ectare CABINET E...	15	2020-05-15 10:00:00	2020-05-15 11:45:00
28	TEAMBER rdv chez elle avec Jean-Luc PROPRI...	16	2020-05-15 13:00:00	2020-05-15 14:00:00
29	Rdv Banque	IMPERATIF	2020-05-15 14:00:00	2020-05-15 16:00:00
30	TEAMBER rdv chez elle avec Jean-Luc PROPRI...	16	2020-05-15 16:00:00	2020-05-15 17:00:00
31	Réserver les vacances d'été	IMPERATIF	2020-05-17 16:00:00	2020-05-17 18:00:00
32	Restaurant avec l'équipe	IMPERATIF	2020-05-20 12:00:00	2020-05-20 12:50:00
33	Footing	IMPERATIF	2020-05-22 08:00:00	2020-05-22 09:00:00

Rémerques :

- si certaines tâches durent plus d'une heure alors qu'elles ont été découpées en tranche d'une heure maximum, c'est parce qu'on les a recollées lorsque cela était possible. Cela offre une meilleure lisibilité.
- les tâches sont quasiment classées par ordre de priorité, hormis ligne 12-13 ou cela a permis de caser une tâche d'1h30 dans un créneau d'exactement 1h30, idéal !
- les calculs ont pris environ de 5 secondes au total, ce qui est raisonnable.
- les tâches sont quasiment toujours réalisées en une fois.

Observons à présent les tâches qui n'ont pu être planifiées :

P.notScheduled()					
id morceau	Objet	Durée	Priorité	id tache	
0	17 TEAMBER rdv chez elle avec Jean-Luc PROPRI...	1.00	16	7	
1	24 LABSOFT Confirmer le nom du répertoire de ma...	0.75	17	10	
2	23 LABSOFT Confirmer le nom du répertoire de ma...	1.00	17	10	
3	12 TEAMBER Travail sur structure BD TEAMBER V...	1.00	18	6	
4	14 TEAMBER Travail sur structure BD TEAMBER V...	0.50	18	6	
5	13 TEAMBER Travail sur structure BD TEAMBER V...	1.00	18	6	

En ayant connaissance de cette information, un chef d'équipe pourra donc choisir de relancer les calculs en changeant les horaires des membres de son équipe afin d'atteindre les objectifs dans les temps impartis.

4.7 Conclusion

L'outil créé arrive à générer en des temps raisonnables des emplois du temps respectant dans l'ensemble plutôt bien les contraintes décrites en introduction du problème. Pour rappel, il s'agit plus de suggérer que d'imposer les emplois du temps aux utilisateurs, libre ensuite à eux d'appliquer des corrections. Pour la deuxième version du logiciel Teamber, cet outil offrira, on l'espère plusieurs avantages :

- aider un chef de projet dans la planification de la semaine pour son équipe en accomplissant le gros du travail. Pour la prochaine version du logiciel Teamber, s'agissant de ce module de planification, on imagine que l'utilisateur pourra lancer plusieurs fois les calculs en jouant sur les paramètres et retenir la proposition de sprint qui lui conviendra le mieux.
- une utilisation personnelle sur du très court terme : un utilisateur qui n'a pas réussi à terminer tout ce qu'il s'était planifié dans la journée, cliquerait à ce moment là sur un bouton "Replanifier les tâches inachevées" et le programme lui ré-organiserait son emploi du temps à partir du lendemain.
A l'inverse, si à un certain moment l'utilisateur est avance dans son travail, il pourra reprogrammer son emploi du temps à partir de cet instant.
- des agendas lisibles pour tous les utilisateurs. L'idée étant d'éviter à tout prix les agendas qui renvoient une sentiment de débordement, dans lesquels les événements se chevauchent, témoignant d'un travail dans l'urgence souvent peu efficace.
- contribuer à une meilleure organisation de l'entreprise, le leitmotiv du logiciel Teamber.

5 Quatrième mission : complétion intelligente d'une tournée commerciale

L'idée est la suivante : un commercial dispose d'une succession de rendez-vous datés et localisés pour sa tournée. Il va donc faire un certain nombre de trajets.

D'autre part, existera dans la seconde version de TEAMBER une liste de **clients** et une liste de **prospects**. Seront renseignés dans ces listes la date de dernière visite, et pour les prospects, une espérance de concréétisation allant de zéro à cent pourcents.

Au moment d'utiliser le logiciel pour compléter sa tournée, l'utilisateur indiquera une sélection de lieux qu'il accepte d'éventuellement ajouter à sa tournée initiale. Trois sélections seront possibles :

- Sélection 1 : prospects tels que Date dernière visite > 3 mois et espérance concréétisation > 70%
- Sélection 2 : prospects tels que Date dernière visite > 3 mois
- Sélection 3 : prospects et clients tels que Date dernière visite > 4 mois.

A noter que rien n'empêche de personnaliser davantage la sélection, mais c'est ce qui m'a été demandé pour ce programme.

5.1 Données et contraintes du problème

Données : Sur la base de la tournée initiale du commercial (voir Figure 4) on va tâcher d'ajouter intelligemment et dans la mesure du possible des points de passage sur sa route.

FIGURE 4 – Exemple de tournée initiale

	Nom	Date début	Date fin	Adresse	Type
0	Départ de Muret	2020-06-15 08:30:00	2020-06-15 08:30:00	113 Boulevard de Lamasquère 31600 Muret	RDV
1	Foire MARSEILLE	2020-06-15 14:00:00	2020-06-15 19:00:00	Marseille Chanot	RDV
2	Foire LYON	2020-06-16 14:00:00	2020-06-17 12:00:00	Eurexpo Lyon	RDV
3	Expo Brive la gaillarde	2020-06-18 14:00:00	2020-06-18 16:00:00	gare de Brive la Gaillarde	RDV
4	Retour Muret	2020-06-19 12:00:00	2020-06-19 12:00:00	113 Boulevard de Lamasquère 31600 Muret	RDV

Contraintes :

- on doit fournir une charge de travail cohérente au commercial, pour cela il pourra indiquer qu'il est disponible de telle heure à telle heure chaque jour.
- il s'agit de le faire dévier le moins possible de son itinéraire initial afin de minimiser le temps de trajet.
- selon qu'on aille visiter un prospect ou un client, le temps passé sur place varie, on pourra indiquer des temps référents selon la catégorie de l'endroit visité. On considérera dans notre cas qu'on accorde en moyenne 2h à un client et 30min à un prospect.
- notre programme doit être simple d'utilisation, le plus rapide possible, et son utilisation la moins coûteuse possible (frais Google maps).

5.2 Méthode de résolution

Assez simplement, entre chaque paire de rendez-vous, on va ajouter des visites tant que faire se peut.

A l'aide des horaires de référence du commercial et ceux des deux rendez-vous consécutifs, on a alors une idée du temps restant disponible dont il dispose entre les deux rendez-vous, on va alors chercher la meilleure visite possible.

La meilleure visite possible doit être réalisable (temps passé sur place + temps déviation trajet < temps restant disponible) et telle que le temps de déviation est minimal. S'il y a une visite candidate, alors on l'ajoutera à notre tournée, et on actualisera le temps restant disponible et l'on cherchera une nouvelle candidate.

L'utilisateur indiquera également une durée qui correspondra à une marge de sécurité (1h par exemple). Une fois que le temps restant disponible tombera en dessous de ce seuil, l'on s'arrêtera d'ajouter des points d'arrêts sur le trajet.

5.2.1 Raffinage géographique de la sélection

Afin de mettre en place ce programme, il est nécessaire de connaître les durées de trajet (Distance Matrix) entre toutes les adresses postales impliquées dans ce programme. Cela se fait au travers de google maps développer à l'aide d'une clé API.

Ce service est payant, j'ai pu utiliser la période d'essai pour mettre en place le programme.

Les calculs prennent un certain temps, (N adresses donnent N^2 calculs d'itinéraires), et le coût étant proportionnel aux nombre de demandes effectuées, il faut donc ne garder dans notre sélection d'adresses candidates des adresses qui sont cohérentes vis à vis de notre tournée.

Dans un premier temps, j'avais pensé à inviter l'utilisateur à préciser les régions/départements concernés par sa tournée pour filtrer les adresses par leur code postal. Mais cela contribuerait à la pénibilité d'utilisation du programme.

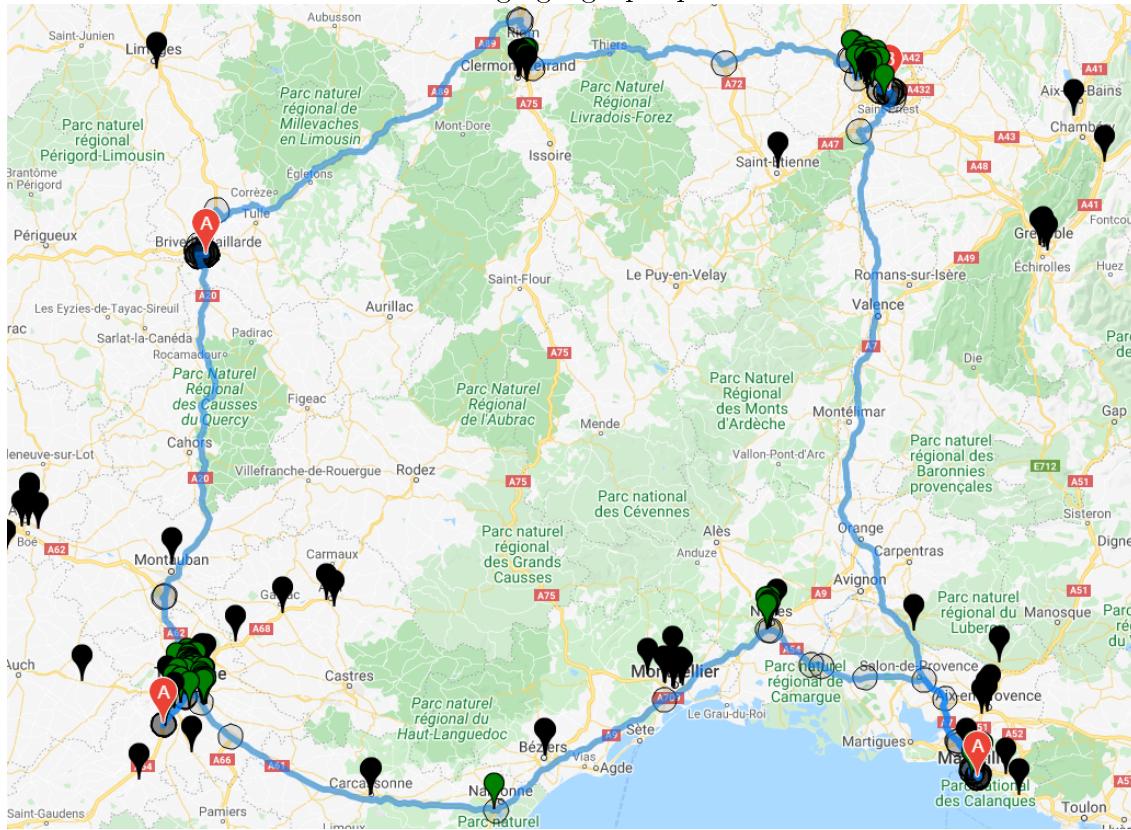
Des adresses cohérentes sont en réalité des adresses qui ne nous font pas tellement dévier l'itinéraire original : celui qui mène de rendez-vous ferme en rendez-vous ferme. Ainsi il faut réussir à attraper les adresses se situant suffisamment proche de cet itinéraire.

Pour cela, en utilisant le module "Directions" de la google maps API, on va pouvoir obtenir une succession de points (latitude-longitude) pour un trajet donné. Ces points correspondent à des instructions GPS (tourner à gauche, prendre la sortie 37 etc.).

A partir de ces points, on va pouvoir dessiner des cercles d'un rayon fixé en préambule du programme et ne conserver que les adresses tombant dans au moins un de ces cercles.

En vert : les adresses retenues, en noir : les adresses écartées, en rouge : les étapes de notre itinéraire initial.

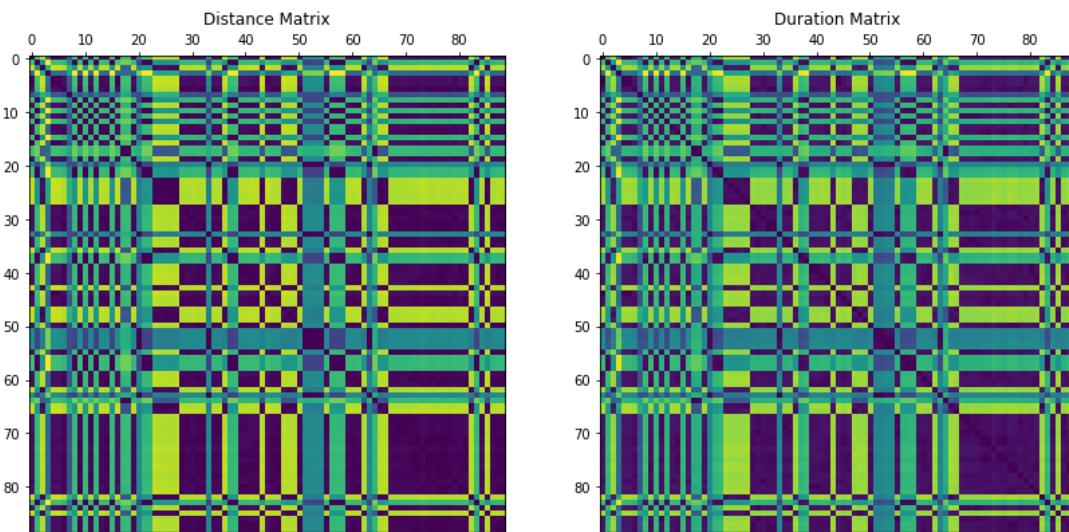
FIGURE 5 – Raffinement géographique de la sélection



Remarques :

- les cercles étant centrés sur des intersections, des bifurcations ou autre nous empêche parfois de capter les adresses à proximité des autoroutes (voire Carcassonne).
- ce filtrage nous a permis de passer de 326 adresses à 86 pour cet exemple.

FIGURE 6 – Distance Matrix et Duration Matrix



5.2.2 Résultats

Import librairies

Import contacts

Sélection

Mode sélection : 1

88 adresses candidates

Obtention de points le long de la route

Raffinage des adresses grâce aux cercles

Avant raffinement : 326

Après raffinement : 86

pour des rayons de : 5000 m.

Obtention de la distance/duration matrix

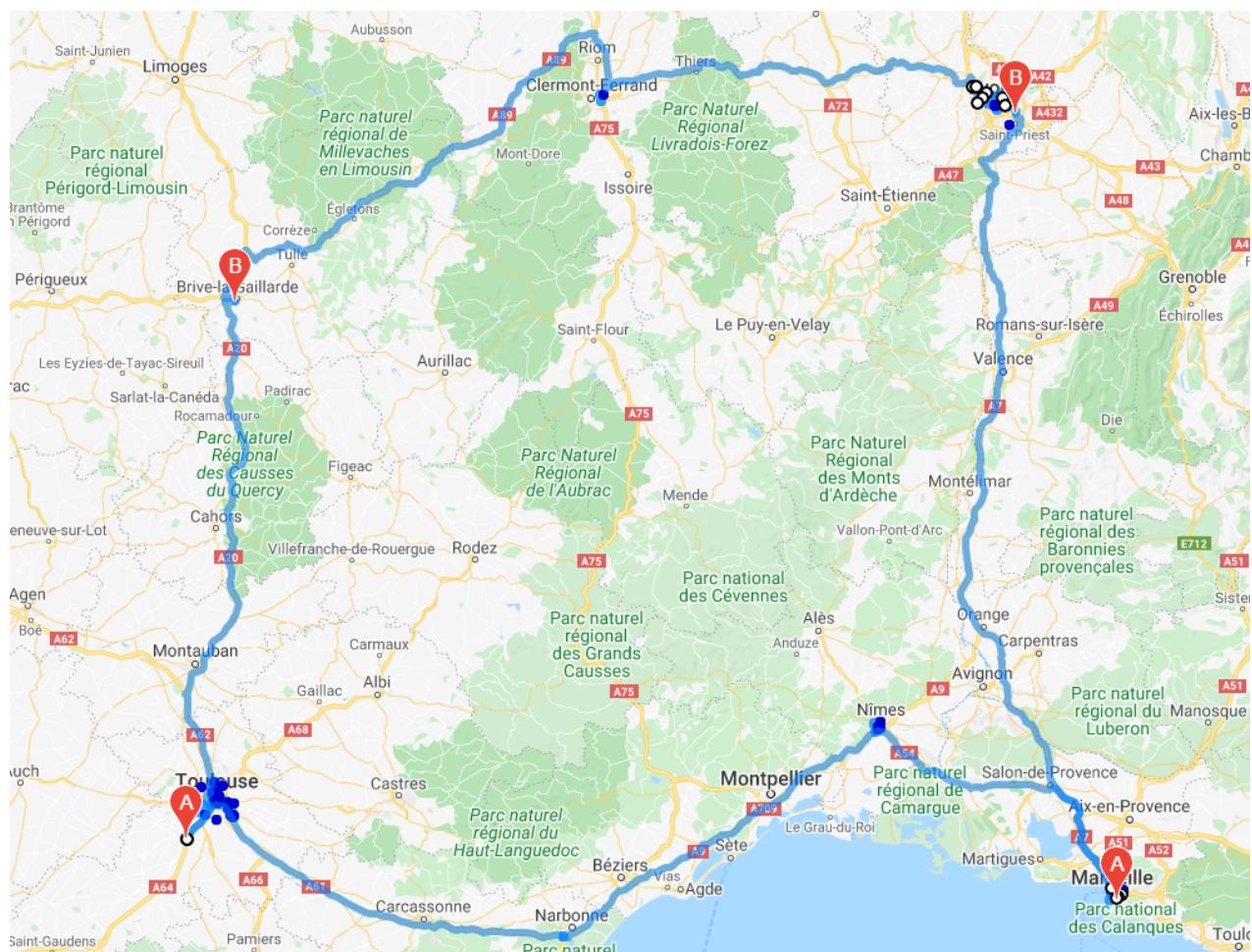
Optimisation de la tournée

Calcul des horaires de passages pour la tournée

Temps total : 22.08952021598816

Dessin de la tournée

FIGURE 7 – Résultat



En bleu clair : les prospects, en bleu foncé : les clients, et en noir et blanc : les étapes ajoutées à la tournée initiale. [Voir le zoom de la carte](#)

On peut ensuite établir un planning de la tournée, afin de connaître les horaires de passages dans les étapes ajoutées et être en mesure de prévenir les intéressés de notre prochain passage.

FIGURE 8 – Résultat

	Nom	Adresse	Type	Date début	Date fin
0	Départ de Muret	113 Boulevard de Lamasquère 31600 Muret	RDV	2020-06-15 08:30:00	2020-06-15 08:30:00
1	Bureau d'étude AXIOLIS	210 avenue de Toulon - 13010 Marseille	Prospect	2020-06-15 12:35:00	2020-06-15 13:05:00
2	Agence d'architecture SARL SOCIETE D'ARCHITEC...	30 Rue Louis Rège - 13008 Marseille	Prospect	2020-06-15 13:10:00	2020-06-15 13:40:00
3	Foire MARSEILLE	Marseille Chanot	RDV	2020-06-15 14:00:00	2020-06-15 19:00:00
4	SAS GARCIA INGENIERIE	164 Chemin Saint Jean du désert - 13005 Marseille	Prospect	2020-06-16 08:10:00	2020-06-16 08:40:00
5	Société TPFI	Immeuble Le Balthazar 2 quai d'Arenc - 13002 ...	Client	2020-06-16 08:54:00	2020-06-16 10:54:00
6	Foire LYON	Eurexpo Lyon	RDV	2020-06-16 14:00:00	2020-06-17 12:00:00
7	Bureau d'étude ELEYS	51 Rue Emile Decors - 69100 Villeurbanne	Prospect	2020-06-17 12:14:00	2020-06-17 12:44:00
8	Agence d'architecture AAMCO	20 Rue Octavie - 69100 Villeurbanne	Prospect	2020-06-17 12:52:00	2020-06-17 13:22:00
9	M. Remy GAILLARD	25 rue Joannes Carret - 69009 Lyon	Client	2020-06-17 13:32:00	2020-06-17 15:32:00
10	Agence d'architecture H.T.V.S Architecture	2 rue de la Gare - 69009 Lyon	Client	2020-06-17 15:39:00	2020-06-17 17:39:00
11	Bureau d'étude KORELL	17 Rue Georges Perret - 69160 Tassin-la-Demi...	Prospect	2020-06-17 17:46:00	2020-06-17 18:16:00
12	M. Anthony SIMON	Mini Parc Bat 3 3 chemin du Jubin - 69570 Dard...	Client	2020-06-17 18:27:00	2020-06-17 20:27:00
13	Agence d'architecture ARCHIGROUP	411 allée des noisetiers - 69760 Limonest	Prospect	2020-06-18 08:05:00	2020-06-18 08:35:00
14	Expo Brive la gaillarde	gare de Brive la Gaillarde	RDV	2020-06-18 14:00:00	2020-06-18 16:00:00
15	M. Stéphane ZANON	113 BD DE LAMASQUERE - 31600 MURET	Client	2020-06-18 18:14:00	2020-06-18 20:14:00
16	SARL ENZO & ROSSO	113 boulevard de Lamasquère - 31600 Muret	Client	2020-06-19 08:00:00	2020-06-19 10:00:00
17	Retour Muret	113 Boulevard de Lamasquère 31600 Muret	RDV	2020-06-19 12:00:00	2020-06-19 12:00:00

FIGURE 9 – Détail des étapes ajoutés entre les RDV

	Origine	Destination	Date fin rdv Origine	Date début rdv Destination	Temps dispo interRDV	Temps trajet initial	Temps trajet supplémentaire	Distance trajet initial (km)	km ajoutés	Prospects visités	Clients visités	Temps prospects	Temps clients	Temps restant
0	Départ de Muret	Foire MARSEILLE	2020-06-15 08:30:00	2020-06-15 14:00:00	05:30:00	04:09:00	00:07:00	423.0	2	2	0	01:00:00	00:00:00	00:14:00
1	Foire MARSEILLE	Foire LYON	2020-06-15 19:00:00	2020-06-16 14:00:00	06:00:00	03:10:00	00:14:00	322.0	4	1	1	00:30:00	02:00:00	00:06:00
2	Foire LYON	Expo Brive la gaillarde	2020-06-17 12:00:00	2020-06-18 14:00:00	12:30:00	03:46:00	00:38:00	380.0	10	4	3	02:00:00	06:00:00	00:06:00
3	Expo Brive la gaillarde	Retour Muret	2020-06-18 16:00:00	2020-06-19 12:00:00	06:30:00	02:14:00	00:00:00	219.0	0	0	2	00:00:00	04:00:00	00:16:00

6 Cinquième mission : programmation d'une tournée optimale

Dans cette section, nous devons résoudre le problème du voyageur de commerce. Il consiste en trouver le plus court trajet reliant un point de départ, des adresses de clients fournies par l'utilisateur et un point d'arrivée.

Ces adresses doivent pouvoir être choisies de différentes manières :

- une liste explicite fournie par l'utilisateur.
- en filtrant par région/département les clients recensés dans la base de données TEAMBER.
- en choisissant l'ensemble des clients contenus dans un cercle de rayon et de centre donnés.

Une fois les adresses connues, il s'agira alors de déterminer un trajet optimal ou quasi-optimal.

Nous verrons dans la sous-section suivante que des difficultés nous conduiront à trouver une alternative à l'algorithme du recuit simulé, qui ne fonctionne pas parfaitement dans un certain nombre de situations.

6.1 Trajet optimal - Recuit simulé

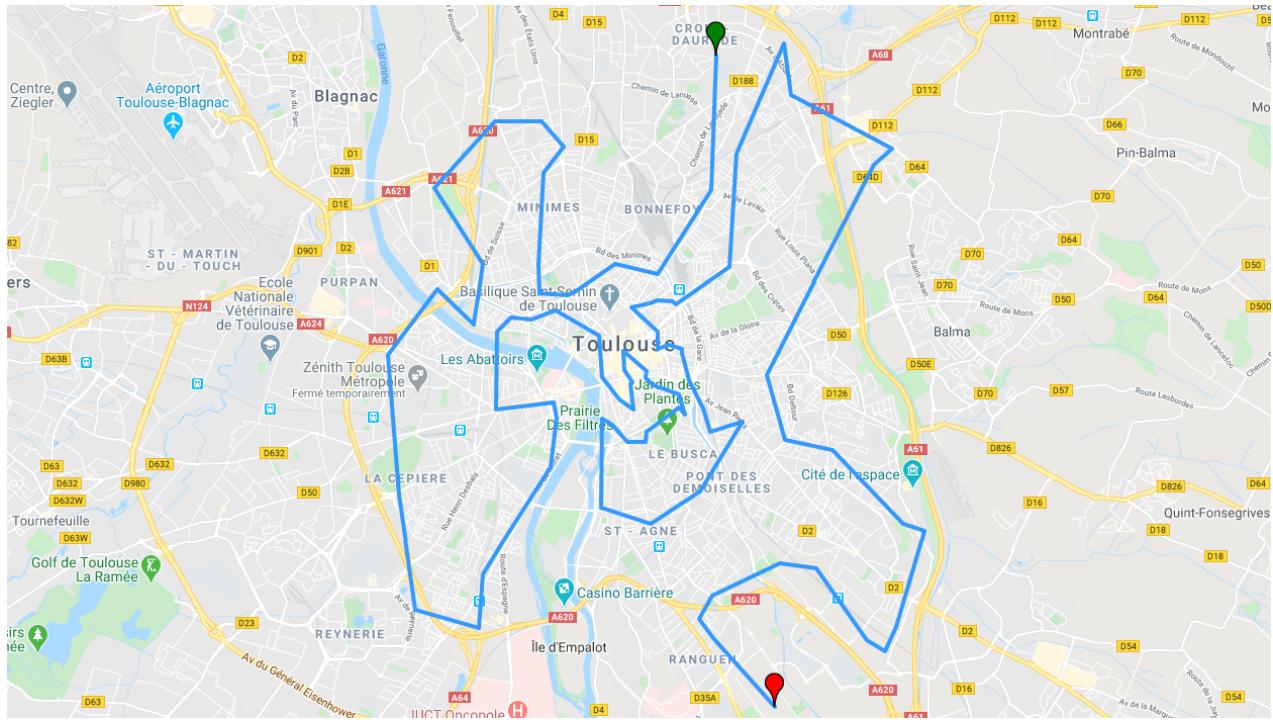
FIGURE 10 – Adresses à Toulouse



Remarques :

- dans l'algorithme de recuit simulé, on considérera qu'un état voisin y de l'état x est tel que l'on peut passer de x à y en retournant une chaîne d'adresses dans x . Cette manière de transiter d'un état à l'autre est symétrique.
- dans ces exemples on considérera la distance à vol d'oiseau entre les adresses, afin d'économiser des crédits google maps et de s'assurer visuellement de la qualité des résultats.
- d'autres modes de déplacement seront disponibles : en voiture, à pied, en vélo, en transports.

FIGURE 11 – Solution après recuit simulé



La solution est satisfaisante. Voir trajet optimal dans Toulouse à pied

6.1.1 Limites

A présent, un deuxième exemple qui va mettre en difficulté notre méthode de résolution :

FIGURE 12 – Tournée région Rhônes-Alpes

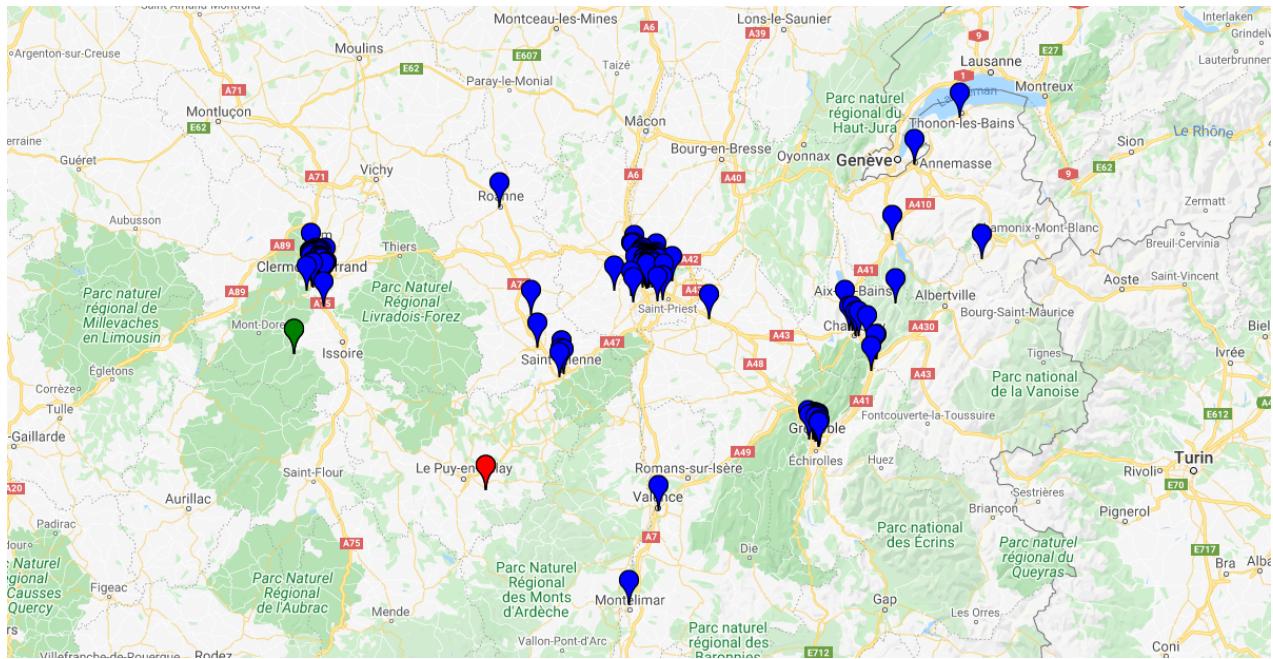
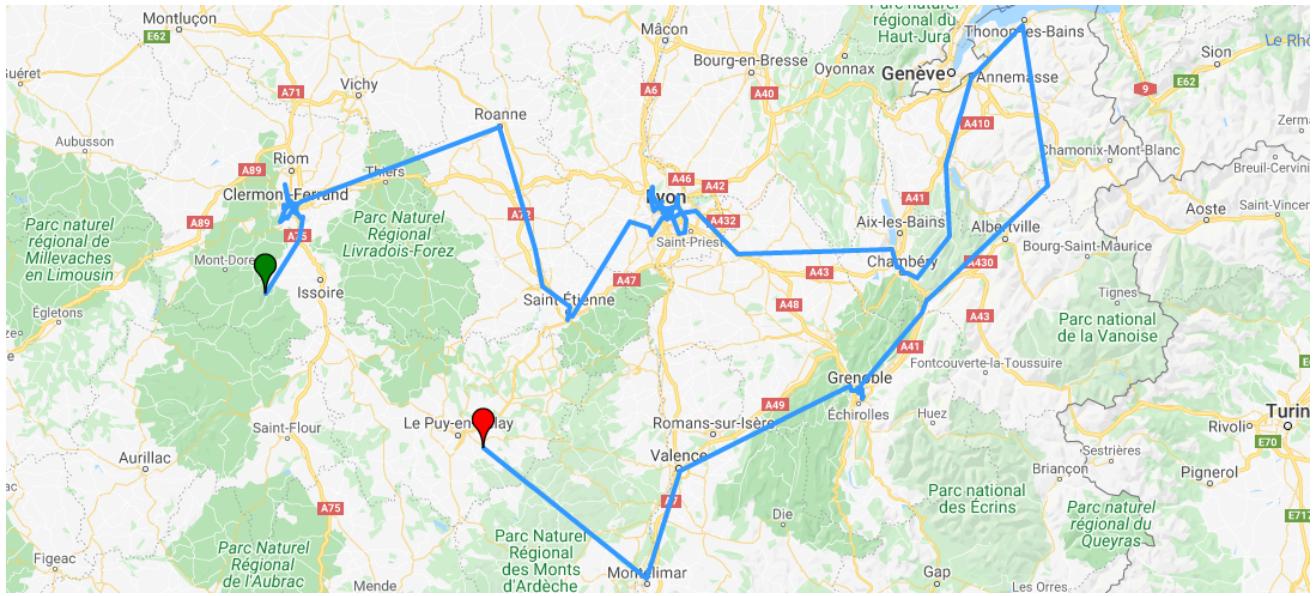
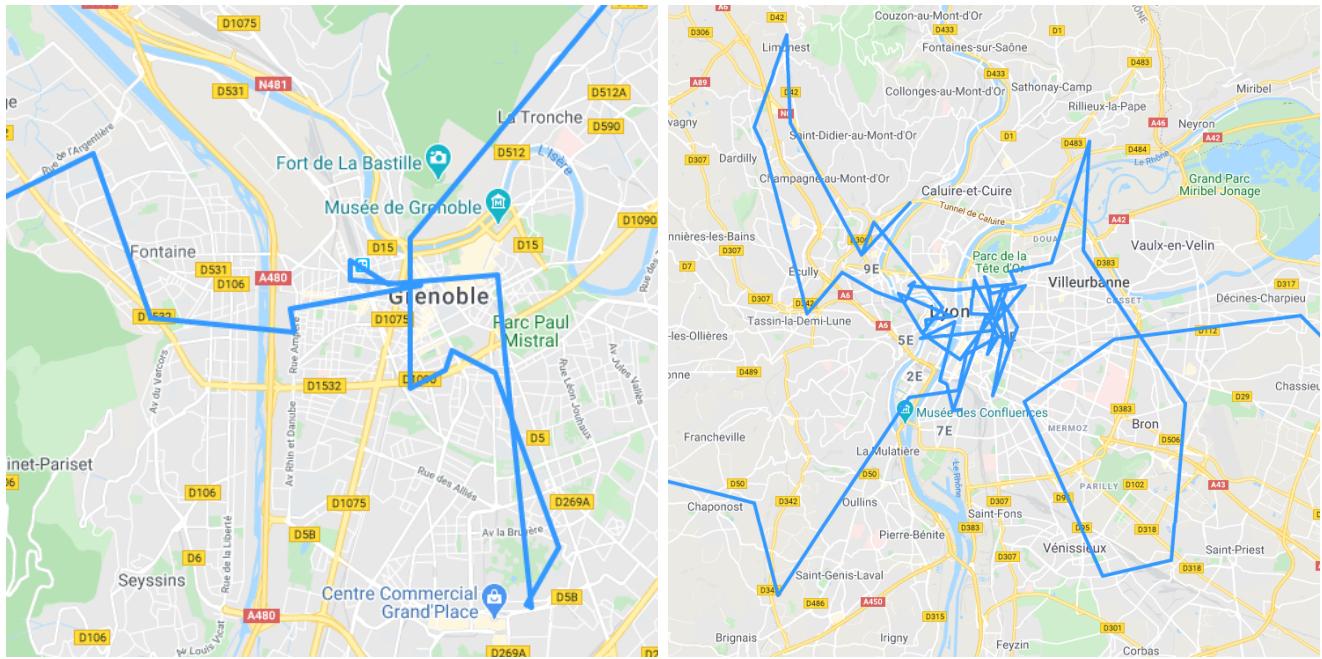


FIGURE 13 – Solution recuit simulé - Rhônes-Alpes



De loin, le trajet est cohérent, mais à l'intérieur des villes, les trajets ne sont pas optimaux.

FIGURE 14 – Erreurs dans les villes



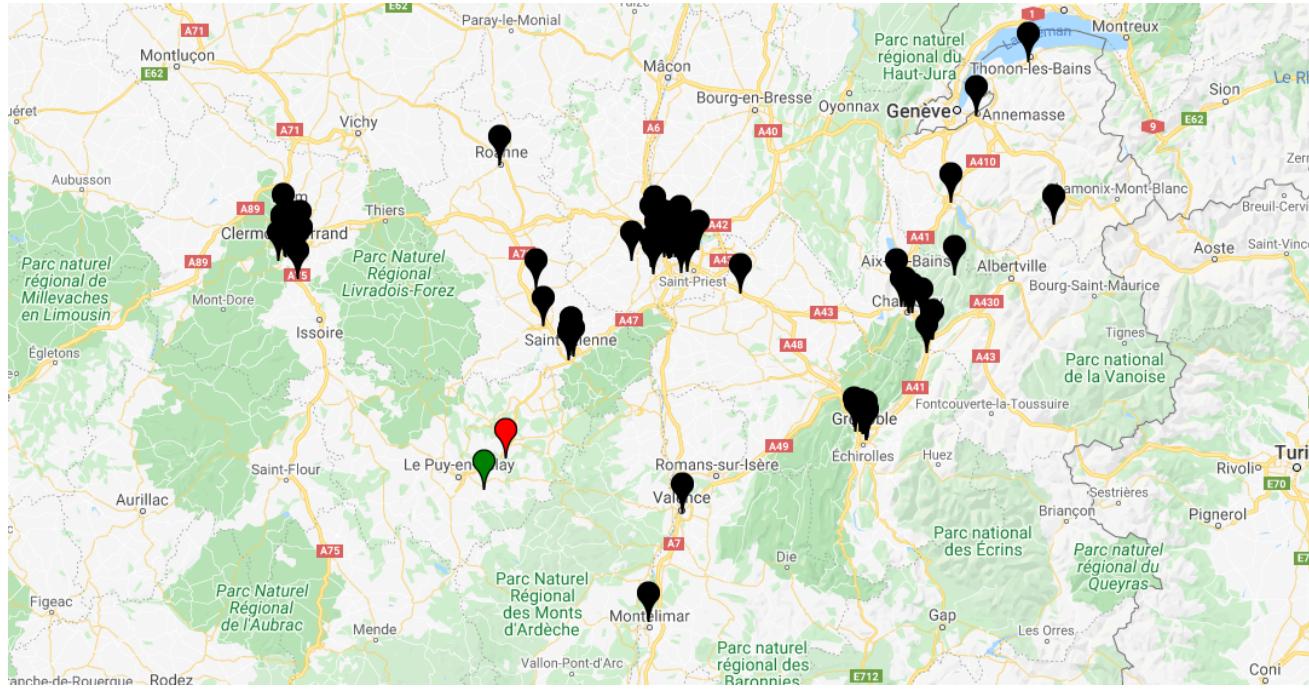
Cela vient du fait que les distances entre les adresses à l'intérieur des villes sont négligeables devant d'autres distances. Les distances ne sont pas suffisamment homogènes.

Nous remédions à ce problème dans la section suivante.

6.2 Clustering d'adresses

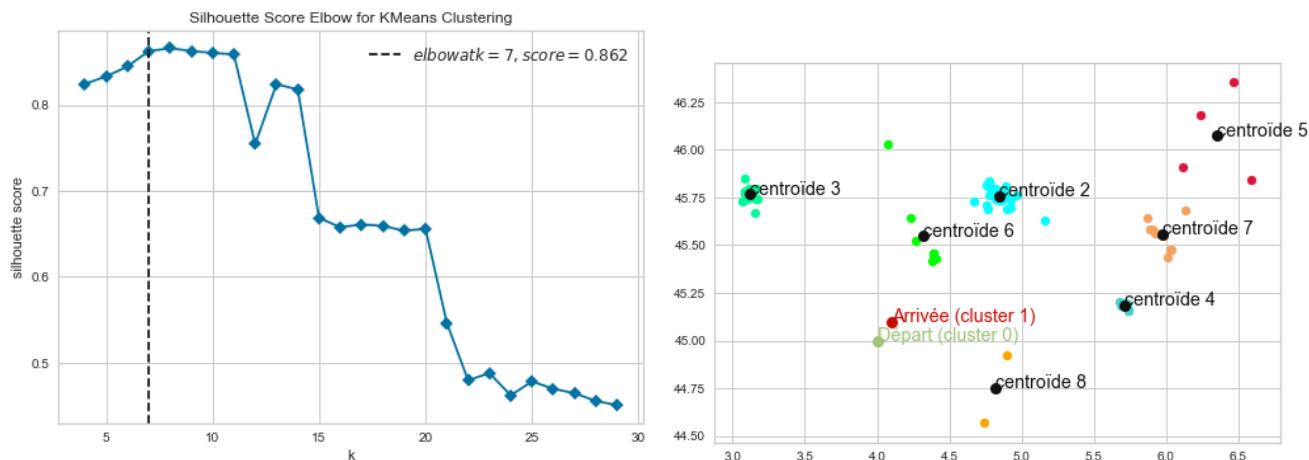
Afin de remédier au problème rencontré dans la section précédente, on choisit de regrouper les adresses dans des clusters.

FIGURE 15 – 151 adresses dans la région Auvergne Rhônes-Alpes



Nous utiliserons l'algorithme des k-moyennes afin d'identifier formellement des clusters dans cet ensemble de positions géographiques.

FIGURE 16 – Clustering avec k-means



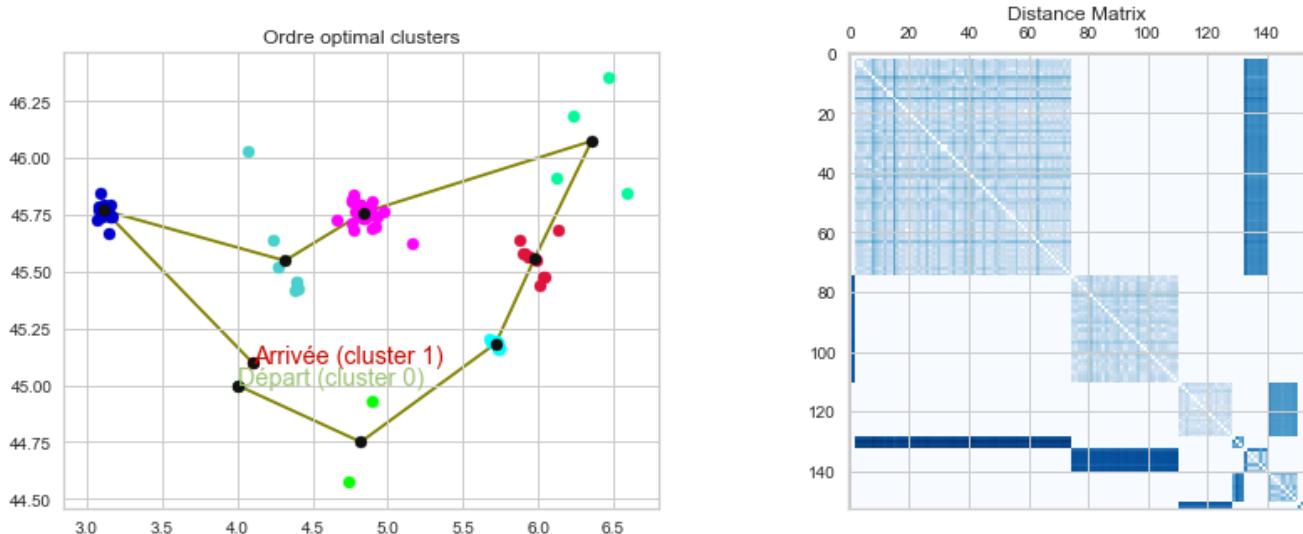
On s'appuie sur l'indice de Silhouette afin de déterminer un nombre de cluster optimal. On peut donc fournir une suggestion de regroupement d'adresses à l'utilisateur.

Remarques :

- si le nombre de clusters paraît incohérent, l'utilisateur peut toujours le modifier.
- on considère (de force) que les adresses de départ et d'arrivée comme deux clusters d'une seule adresse. Cela rajoute augmente donc le nombre de cluster de 2.

Une fois que l'on a déterminé les clusters, nous allons déterminer un ordre de passage optimal dans ces clusters à l'aide des coordonnées GPS des centroïdes.

FIGURE 17 – Ordre de passage optimal dans les clusters et distance matrix



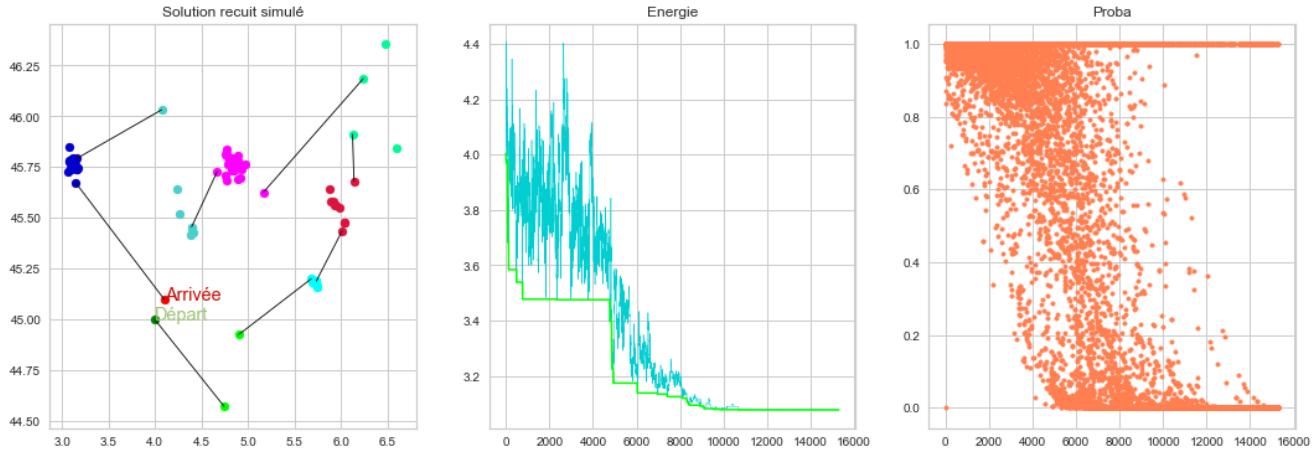
Remarque Nous n'avons besoin que des distances sein des clusters eux-mêmes et d'un cluster au suivant (voir Figure 13 graphe de droite). Ici par exemple on fera 8625 requêtes d'itinéraires à Google maps API en fonctionnant par clusters au lieu de 22801 sinon.

Entrées et sorties des clusters : nous allons maintenant déterminer des adresses de sortie et d'entrée pour chacun de ces clusters. Pour les paires de clusters successifs, nous allons devoir trouver la meilleure sortie et la meilleure entrée. Si possible, (c'est à dire que le cluster contient au moins deux adresses) nous ne passerons pas deux fois par la même adresse.

Encore une fois nous utiliserons le recuit simulé pour résoudre ce problème : étant donnée une liste d'entrées/sorties pour chacun des clusters, nous choisirons un cluster aléatoirement selon son poids (son nombre d'adresses par rapport au nombre d'adresses total) et changerons aléatoirement (une chance sur deux) son entrée ou sa sortie par une adresse qui n'est pas utilisée si possible. Sinon : pour 2 adresses, échange entrée/sortie, sinon pour 1 adresse : rien. Cette manière de procéder est bien symétrique d'une configuration à la configuration voisine.

Nous tentons alors de minimiser la somme des distances d'une sortie à l'entrée suivante.

FIGURE 18 – Calcul des entrées/sorties optimales des clusters



A présent il ne reste plus qu'à déterminer les trajet optimaux au sein de chaque cluster entre l'entrée et la sortie (recuit simulé), et à mettre tous les trajets bout à bout.

FIGURE 19 – Trajet final

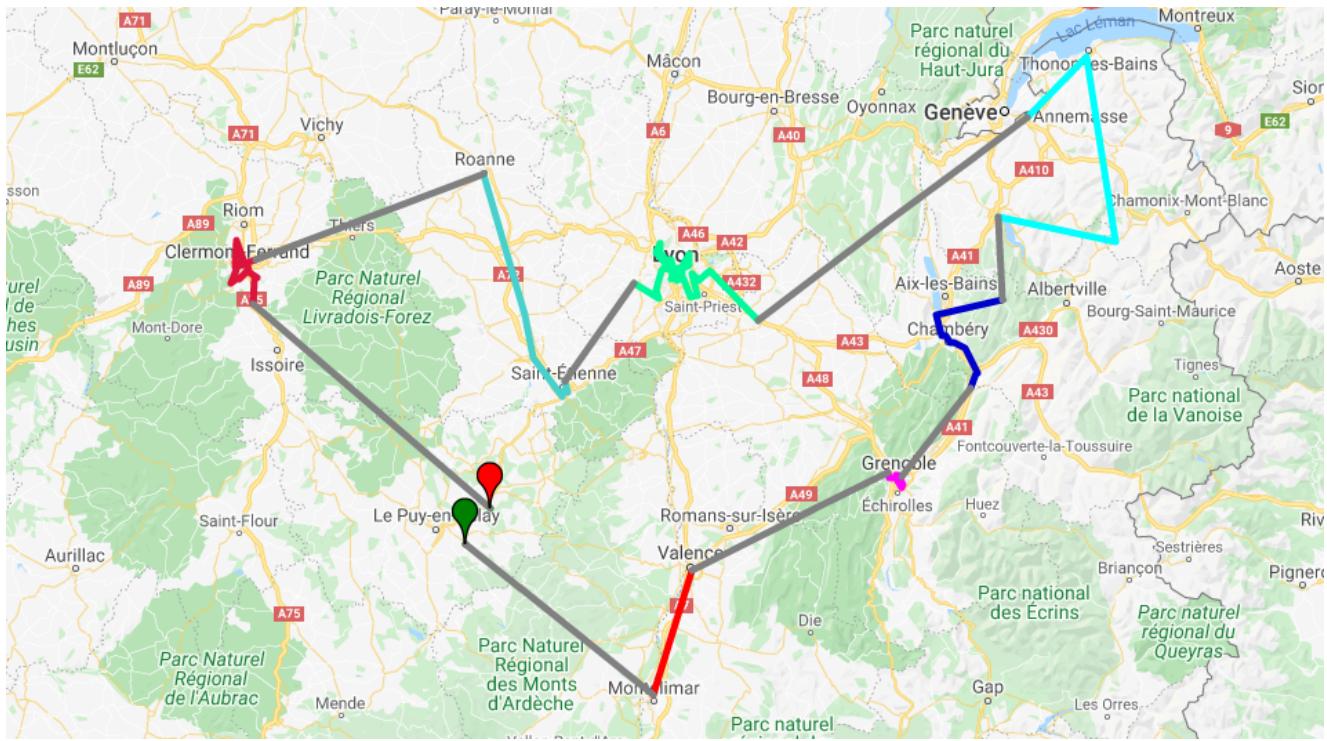


FIGURE 20 – Zoom sur les clusters importants



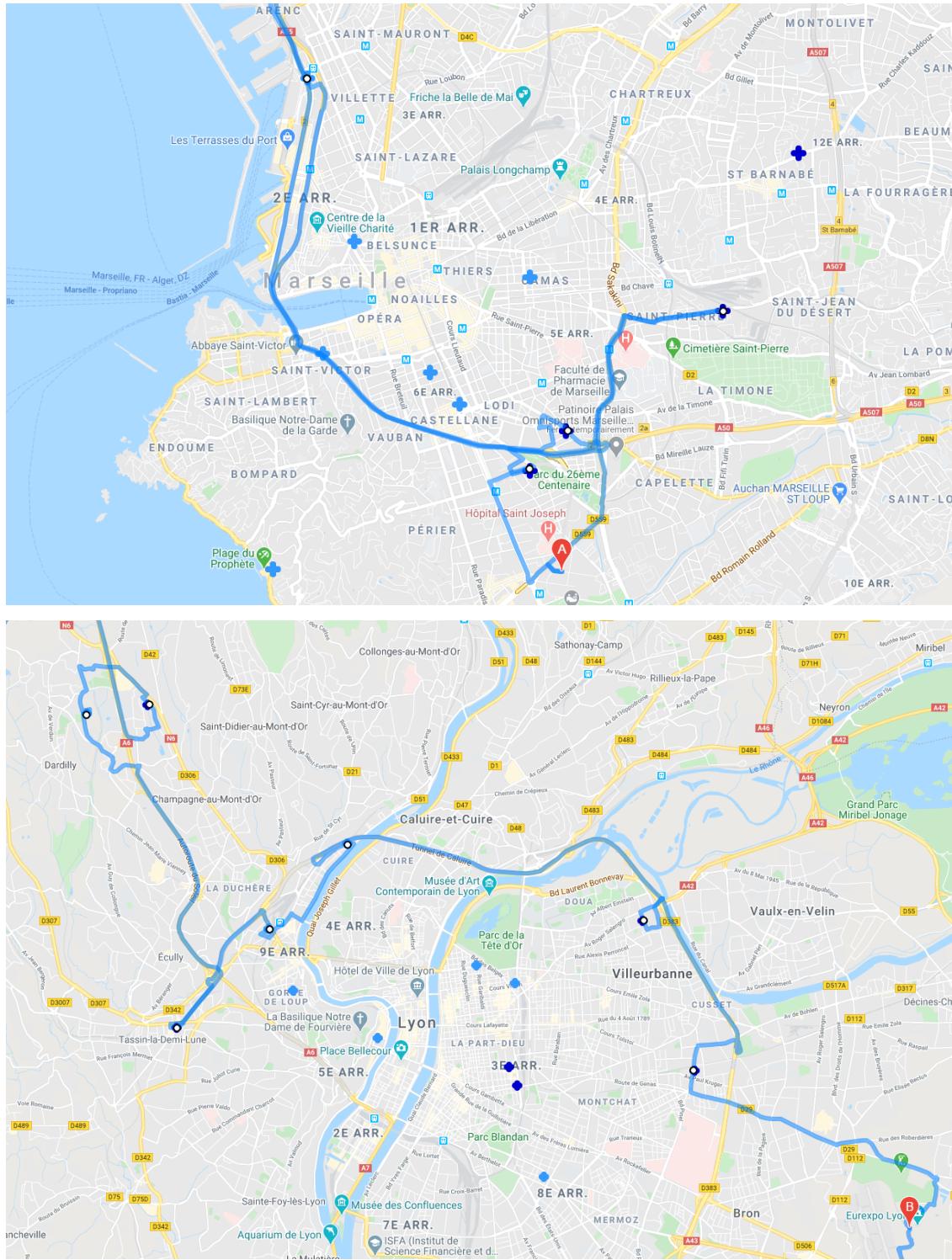
Quelques observations :

- Nous avons ici optimisé la distance à vol d'oiseau entre nos adresses afin de nous assurer visuellement du bon fonctionnement du programme. Mais par la suite, au sein des clusters ainsi que d'un cluster au suivant, l'utilisateur choisira d'optimiser le temps de trajet (ce qui sera sans doute le cas la plupart du temps) ou le nombre de km parcourus.
- Il sera également possible de déterminer la zone de tournée commerciale en indiquant un centre et un rayon, ainsi on cherchera le meilleur trajet entre les adresses de ce disque. C'est même la plupart du temps que cela fonctionnera. De ce fait, connaître les coordonnées GPS des adresses est indispensable.
- D'autres exemples d'optimisations de tournées sont visibles en [annexe](#) (toujours avec la distance à vol d'oiseau).
- Lorsque nous cherchons un ordre de passage entre les clusters, nous faisons l'hypothèse que la distance à vol d'oiseau et temps de trajet sont suffisamment corrélées. Cette hypothèse pourra s'avérer incohérente dans certains cas.
- K-means ne s'applique que dans un espace euclidien, or la terre est ronde et notre façon de déterminer les clusters n'est donc pas réaliste. Mais pour l'instant, Teamber se destine majoritairement à des entreprises françaises où cette approximation fournira des résultats d'une qualité suffisante. Si jamais le logiciel commence à s'exporter alors il faudra réfléchir davantage (clustering avec distance matrix par exemple). L'avantage de K-means est qu'il fournit facilement un représentant de son cluster par son centreïde.

7 Annexe

7.1 Compléction intelligente

FIGURE 21 – Compléction intelligente tournée - Zoom carte



Retour lecture

7.2 Tournée optimale - D'autres exemples

FIGURE 22 – Tournée optimale dans Toulouse, à pied



Remarque : impossible d'afficher tous les itinéraires à pied sur une même page, on voit donc les adresses reliées par des lignes, mais l'ordre optimal de passage a été calculé en utilisant les durées de trajets à pied fournies par google maps. [Retour lecture](#)

7.2.1 Autres exemples - Clustering d'adresses

FIGURE 23 – Nouvelle Aquitaine - 91 adresses



FIGURE 24 – Nouvelle Aquitaine - Clustering

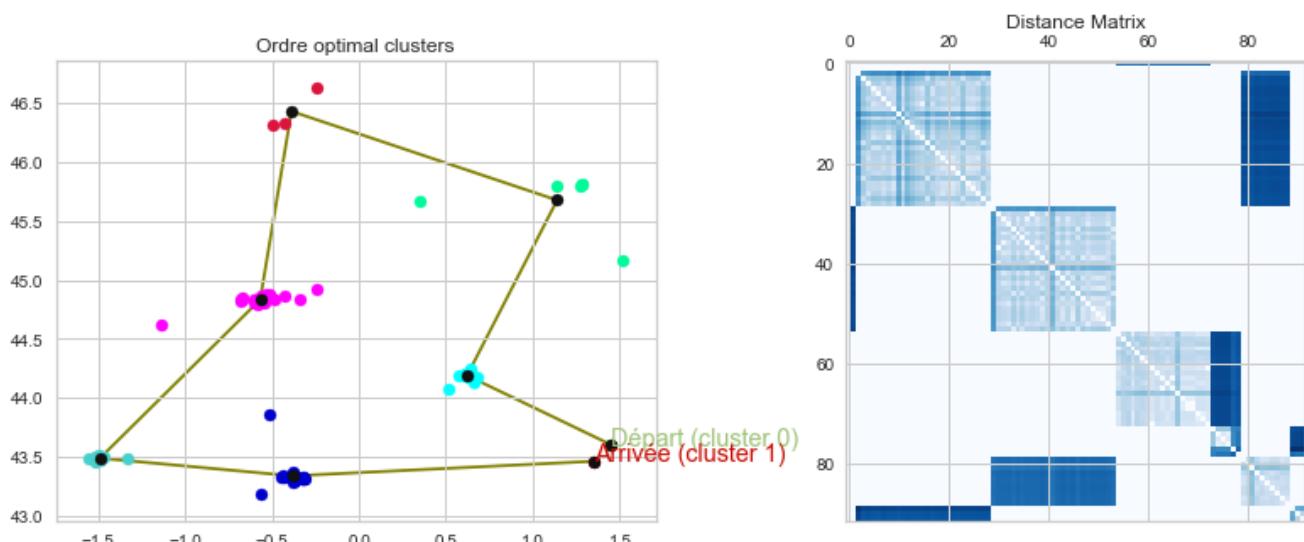


FIGURE 25 – Nouvelle Aquitaine - Calcul des entrées/sorties

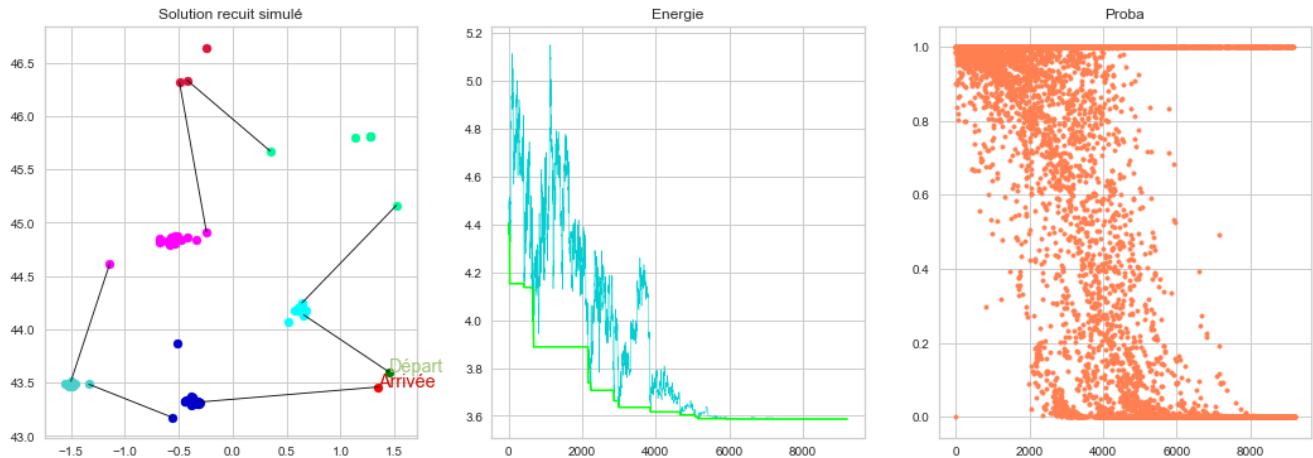


FIGURE 26 – Nouvelle Aquitaine - Trajet final - 90 adresses

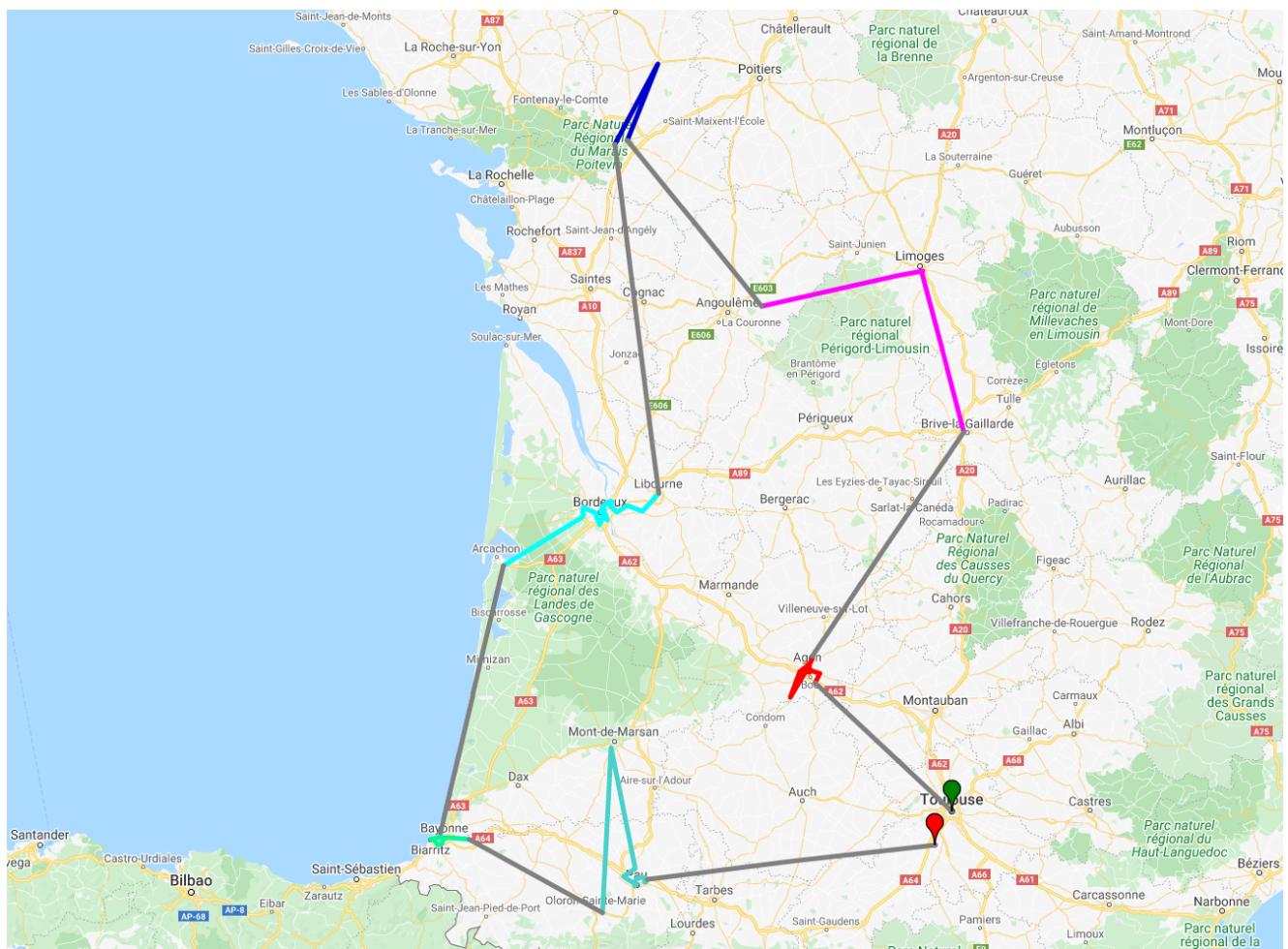
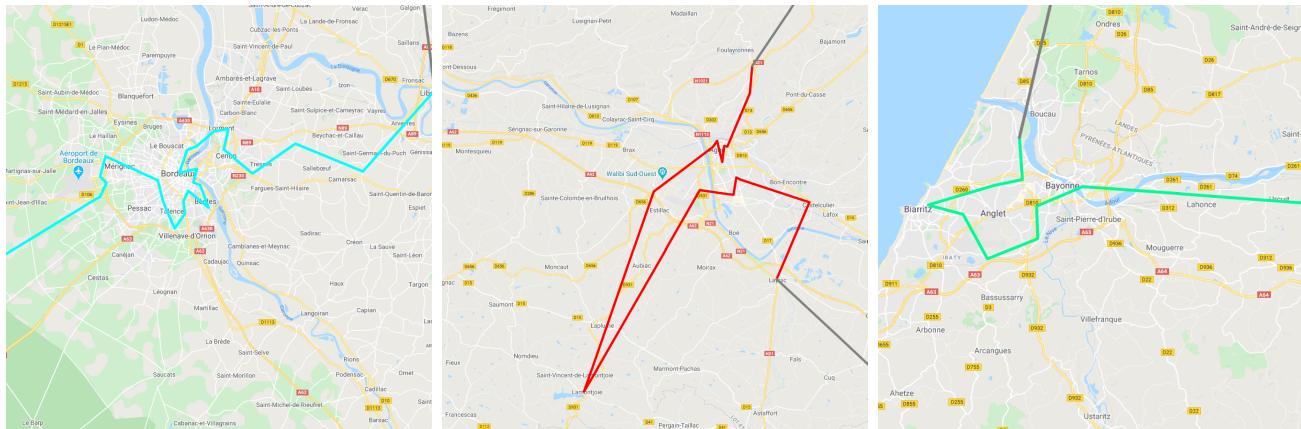


FIGURE 27 – Nouvelle Aquitaine - Zoom clusters importants



[Retour lecture](#)

7.3 Optimisation répartition tâches

DONNEES :

Charge totale de travail : 1307 h pour 210 tâches avec 15 compétences possibles.
(6 h/tâche).

Disponibilités totales : 1398 h réparties sur 40 personnes
(34 h/personne).

Approche choisie : 1

SORTIE :

Total heures non assignées : 0.0

Niveau moyen de spécialisation pour une heure travaillée : 0.98 / 3

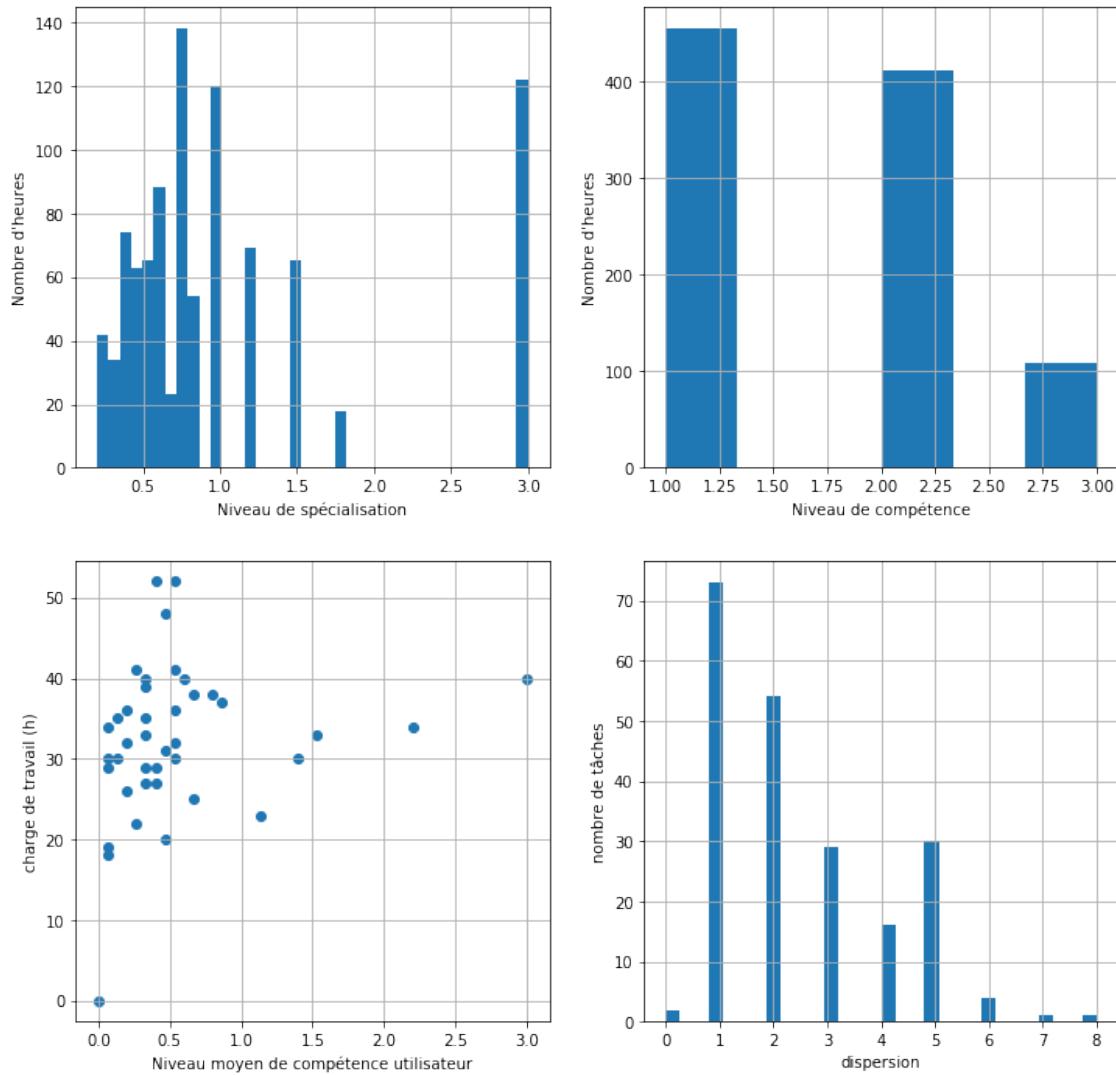
Niveau moyen de spécialisation général : 0.2 / 3

Niveau moyen de compétence pour une heure travaillée : 1.64 / 3

Niveau moyen de compétence général : 0.55 / 3

Statut résolution : Optimization terminated successfully.

Méthode de résolution : interior-point
fonction objectif : -2141.9999996335127



Dans cet exemple, on a simulé un problème de "grande taille" : beaucoup d'utilisateurs, de tâches, de projets. A priori ce programme devrait être utilisé principalement dans des dimensions plus petites.

Lorsque la dimension devient trop grande, l'algorithme du simplexe fournit dans la librairie scipy échoue à la résolution du problème, on relance donc en utilisant la méthode des points intérieurs. Cette méthode a l'avantage de fonctionner même en grande dimension, mais alors la forme de la solution est différente. Cela nous d'ailleurs indique que la solution n'est pas unique.

L'inconvénient majeur de la solution fournie par les points intérieurs est qu'elle a tendance à attribuer les heures d'une même tâche à plusieurs utilisateurs (histogramme en bas à droite), ce qui dans la réalité n'est pas une bonne chose. Il est préférable qu'une seule personne se charge de dessiner les plans plutôt que 3.

Cela dépend bien sûr aussi des données, mais pour tenter de fournir la meilleure solution à l'utilisateur, j'ai choisi de résoudre le problème en utilisant le simplexe, et à défaut de succès, les points-intérieurs.

[Retour lecture](#)

7.4 Emploi du temps

```
class Planning:
    # Création de l'objet
    def __init__(self, nom_utilisateur, id_utilisateur, plagehoraire_filename):
        # Pour changer les paramètres d'optimisation :
        def setPenalties(self, new_penalties):

            # Ajout des tâches à planifier
            def addTasks(self, df_tasks):

                # Calcul de la base de planning et des potentiels initiaux
                def initialise(self, DATE_DEBUT, DATE_FIN):
                    # Ajout d'impératifs à prendre en compte
                    def addImperatifs(self, df_imperatifs):

                        # (maths) Met à jour les potentiels s'il y eu des changements
                        def UpdateScores(self):

                            # Propose un arrangement voisin des tâches et stocke ses potentiels
                            # Affiche les scores pour les différents indicateurs :
                            def showScores(self):

                                # (maths) Propose un arrangement voisin des tâches et stocke ses potentiels
                                def buildVoisin(self):

                                    # (maths) Remplace l'arrangement courant par l'arrangement voisin
                                    def replace(self):
                                        # (maths) Stocke l'arrangement conduisant à l'énergie totale la plus faible
                                        def replaceMin(self):

                                            # (maths) Remplace l'arrangement courant par l'arrangement "minimal" rencontré
                                            def applyMin(self):

                                                # (maths) Renvoie les potentiels de l'arrangement courant ou voisin selon la valeur du
                                                def getPotentiels(self, voisin=False):

                                                    # Exporte le planning en fichier excel et retourne le dataframe correspondant
                                                    def makePlanning(self,to_file=False):

                                                        # Fait l'inventaire des tâches non planifiées
                                                        def notScheduled(self):

# TACHES MULTI-UTILISATEUR
# (maths)
def proj1(t,d):
```

```

# (maths)
def proj2(t,Is):

# (maths)
def h(t,Is):

#Planifie les tâches multi-utilisateurs à l'aide des 3 fonctions ci-dessus
def planifieTMU(TC, DATE_DEBUT, DATE_FIN, plannings, maxiter = 100):

#AUTRES FONCTIONS :

#Calcule la base du planning, supprime les plages horaires < LONGUEUR MIN
def CalculBasePlanning(plagehoraire_filename, DATE_DEBUT, DATE_FIN, LONGUEUR_MIN = 0.5):

#Reconstruit la base en prenant en compte des impératifs
def addImperatifs(base, imp, LONGUEUR_MIN = 0.5):

# (maths) Permute deux taches dans une liste de tache
def permuteTasks(tasks):

# Découpe des tâches en morceaux n'excédant pas une durée choisie (mod_lenght en heure)
# ----> Avec mod_lenght = 1, une tâche de 3h30 est découpée
#       en 3 morceaux d'une heure et 1 d'une demi-heure
def split_tasks(Tasks, mod_lenght = 1):

```

[Retour lecture](#)