Programmation Orientée Objet
IG2I, LE3 – 2019-2020

Projet – Planning des coursiers

Diego Cattaruzza, Maxime Ogier

1 Description du contexte

1.1 Livraison de repas à domicile

Deliver2I est une start-up récemment créée qui est spécialisée dans la livraison de repas à Lens. Tous les jours, elle se charge de livrer des repas à des particuliers, à leur bureau ou à leur domicile. Les clients passent leur commande sur une application dédiée. *Deliver2I* possède un entrepôt situé près du centre ville de Lens. Tous les soirs, les repas (entrées, plats, desserts et boissons) sont préparés et conditionnés dans une cuisine centrale. Le lendemain matin, ces repas sont livrés à l'entrepôt par des camions frigorifiques. Ensuite, au cours de la journée, les livraisons des repas à domicile sont réalisées par des coursiers à vélo. Chaque coursier récupère des repas à l'entrepôt, les livre aux clients concernés, puis revient à l'entrepôt pour récupérer de nouveaux repas.

1.2 Planning des coursiers

Le travail d'un coursier consiste à réaliser plusieurs tournées de livraison successives. Toutes ces tournées commencent et terminent au dépôt. Chacune des tournées a un horaire de début et un horaire de fin qu'il convient de respecter au mieux afin que les clients soient livrés à temps. Il est possible qu'il s'écoule un certain temps entre deux tournées successives d'un même coursier. On appelle cet intervalle de temps un *temps mort* car pendant ce temps-là, le coursier reste en attente. L'intervalle de temps pendant lequel un coursier travaille s'appelle un *shift*. À un jour donné, toutes les tournées effectuées par un coursier sont réalisées dans un unique shift, qui commence donc avant la première tournée, et termine après la dernière tournée effectuée par ce coursier.

Deliver2I est une start-up très soucieuse des conditions de travail de ses coursiers. Ainsi, elle s'est engagée auprès de ces derniers :

- à leur donner leur planning de travail une semaine à l'avance ;
- à ce qu'ils soient payés au moins la durée de travail minimale s'il viennent travailler (même s'ils travaillent moins que cette durée) ;
- à ce qu'ils soient payés pendant les temps morts entre deux tournées ;
- à ce qu'un shift ne dure pas plus que la durée maximale.

Par contre, afin de donner de la flexibilité, *Deliver2I* n'est pas obligée de faire appel à tous ses coursiers tous les jours.

Chaque jour J, *Deliver2I* prépare le planning des coursiers pour la journée $J + 7$. Cela est réalisé en plusieurs étapes, décrites par la suite.

1. Demandes prévisionnelles : à partir des données historiques et des caractéristiques du jour $J + 7$ (jour de la semaine, mois, vacances scolaires, météo), des prévisions de demandes sont réalisées pour chaque heure de la journée $J + 7$.
2. Tournées prévisionnelles : à partir des demandes prévisionnelles et des données historiques sur les tournées (heure de début et durée en fonction de la demande), des tournées prévisionnelles sont réalisées. Ces tournées prévisionnelles n'indiquent pas les clients à visiter, mais seulement l'heure de début et l'heure de fin de la tournée.
3. Shifts prévisionnels : les tournées prévisionnelles sont regroupées de façon à obtenir des shifts (créneaux de travail pour les coursiers) qui permettent de minimiser les temps morts.
4. Affectation des coursiers aux shifts prévisionnels : pour chaque shift, on affecte un coursier. Les coursiers affectés à un shift sont ensuite prévenus de l'horaire de début et de fin de leur shift pour le jour $J + 7$.

Actuellement, le service planification de *Deliver2I* est en charge des 4 étapes pour réaliser le planning des coursiers. Les prévisions des demandes et des tournées sont réalisées de manière très performante par un algorithme de machine learning développé par la start-up. Cependant, l'étape de création des shifts prévisionnels ne donne pas entière satisfaction au responsable du service planification. Cette étape est actuellement réalisée "à la main" par un opérateur. Maintenant que la start-up commence à gérer de gros volumes de livraison, l'opérateur passe plusieurs heures par jour à essayer de trouver une bonne solution. En outre, il semble que les solutions qu'il trouve ne sont pas de très grande qualité, car on constate souvent qu'il y a de nombreux coursiers qui ont des temps morts importants dans leur shift.

1.3 Problème à résoudre

Le responsable du service planification de *Deliver2I* souhaite donc faire développer un outil informatique performant afin d'aider l'opérateur en charge de la création des shifts prévisionnels. Pour ce faire, il fait appel aux meilleurs élèves ingénieurs en informatique des Hauts-de-France.

Le problème auquel nous nous intéressons ici est dénommé *Problème de constitution de shifts*. Une instance de ce problème représente les données d'entrées

qui sont nécessaires pour résoudre le problème. Un exemple d'instance est donné dans la Figure 1. De manière simplifiée, une instance consiste en un ensemble de tournées, chaque tournée étant définie par une date de début et une date de fin.

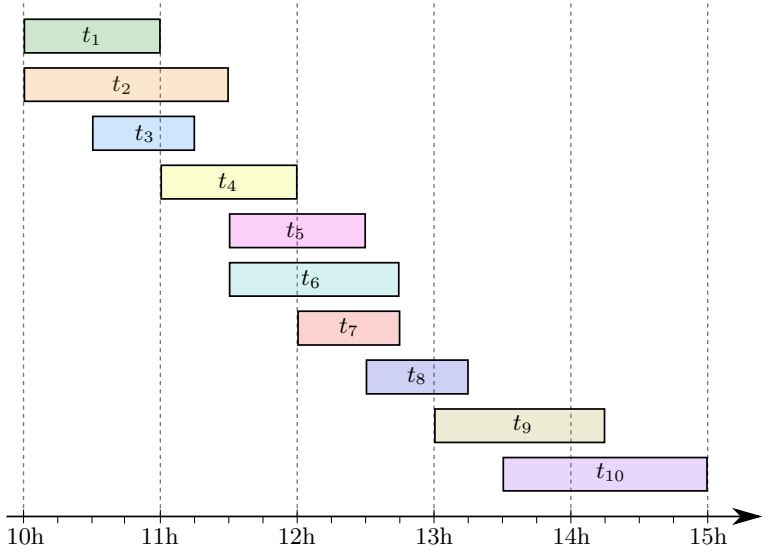

Figure 1: Une instance du problème : chaque tournée est nommée t_1 à t_{10}.

Chaque jour, à partir de cet ensemble de tournées à réaliser, il faut prendre la décision suivante :

- regrouper les tournées dans un ensemble de shifts.

Les contraintes à respecter sont les suivantes :

- toutes les tournées doivent être dans un shift ;
- un shift ne peut pas contenir des tournées qui ont lieu en même temps ;
- un shift doit durer moins que la durée maximale ;
- si un shift dure moins que la durée minimale, il est tout de même payé pour la durée minimale (en ajoutant du temps mort).

L'objectif pour *Deliver2I* est de minimiser les temps morts dans les shifts, c'est-à-dire la somme des périodes de temps pendant lesquels les coursiers ne réalisent pas une tournée pendant leur shift. Un exemple de solution du problème est proposé dans la Figure 2. On suppose ici que la durée minimale est de 2 heures et que la durée maximale est de 3 heures et 30 minutes. Sur cette figure, chaque ligne représente un shift avec ses tournées. Les temps morts sont matérialisés en couleur rouge vif. Notez qu'il peut y avoir des temps morts à cause de l'attente entre deux tournées (par exemple entre t_3 et t_6) ou pour que le shift ait la durée minimale de 2 heures (par exemple pour le shift contenant la tournée t_9).

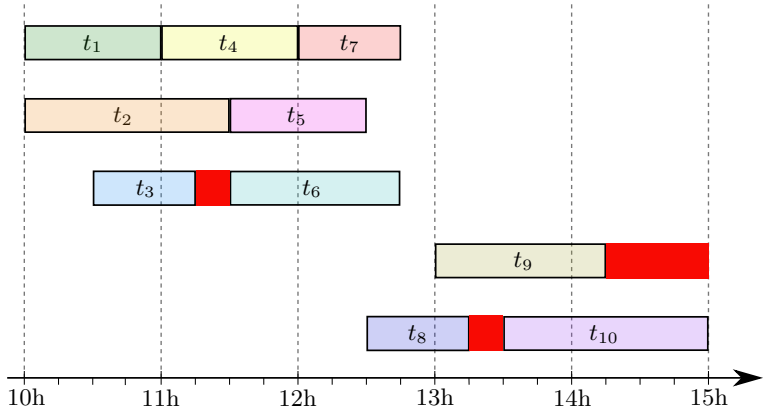

Figure 2: Un exemple de solution pour l'instance de la Figure 1.

2 Description détaillée du problème à résoudre

Nous explicitons ici le problème de constitution des shifts.

2.1 Les données

Nous récapitulons ici toutes les données nécessaires au problème.
Nous noterons :

- $\mathcal{T}$ l'ensemble des tournées ;
- T^{min} la durée minimale pour laquelle le livreur qui effectue le shift est payé (en minutes) ;
- T^{max} la durée maximale d'un shift (en minutes).

Pour chaque tournée $t \in \mathcal{T}$, nous noterons :

- d_t la date de démarrage de la tournée (à la minute près) ;
- f_t la date de fin de la tournée (à la minute près).

2.2 Solution

Pour donner une solution au problème, il suffit de proposer un ensemble de shifts $\mathcal{S}$. À chaque shift $s \in \mathcal{S}$, on associe un ensemble de tournées $\mathcal{T}(s)$.

La durée d'un shift s est une fonction, notée $dur(s)$, et définie par :

$$dur(s) = \max_{t \in \mathcal{T}(s)} \{f_t\} - \min_{t \in \mathcal{T}(s)} \{d_t\}.$$

Ceci signifie simplement que la durée d'un shift est calculée comme la différence entre la date de fin de la dernière tournée et la date de début de la première tournée.

Le temps morts d'un shift s est une fonction, notée $tm(s)$, et définie par :

$$tm(s) = \max\{T^{min}; dur(s)\} - \sum_{t \in \mathcal{T}(s)} (f_t - d_t).$$

Le temps morts d'un shift est donc la différence entre la durée du shift et la somme des durées des tournées qui constituent ce shift. Notez que si la durée d'un shift est inférieure à T^{min}, alors il faut également compter un temps mort d'une durée de $T^{min} - dur(s)$.

2.3 Contraintes du problème

Pour qu'une solution soit réalisable, il faut respecter les conditions suivantes :

- chaque tournée $t \in \mathcal{T}$ doit être présente dans exactement un shift $s \in \mathcal{S}$;
- dans chaque shift $s \in \mathcal{S}$, les tournées n'ont pas lieu en même temps : $\forall (t_1, t_2) \in \mathcal{T}(s), (f_{t_1} \leq d_{t_2}) \vee (f_{t_2} \leq d_{t_1})$;
- chaque shift $s \in \mathcal{S}$ doit durer moins que la durée maximale : $dur(s) \leq T^{max}$.

2.4 Fonction objectif

Deliver2I souhaite payer le moins cher possible pour réaliser ses livraisons. Ainsi, il convient de proposer une solution qui permette de minimiser la somme des temps morts des shifts. En effet, les temps morts sont très préjudiciables pour *Deliver2I* puisque les coursiers sont payés alors qu'on ne leur donne aucun travail à faire.

Ainsi, l'objectif ici est de minimiser les temps morts, ce qui s'exprime comme $tm = \sum_{s \in \mathcal{S}} tm(s)$.

2.5 Hypothèses sur les données

On suppose que les données sont correctes, c'est-à-dire, que pour toute tournée $t \in \mathcal{T}$, on a $d_t < f_t$.

Par ailleurs, vous noterez qu'il n'y a pas de limite sur le nombre maximum de shifts que peut contenir une solution. En effet, *Deliver2I* estime qu'elle dispose de suffisamment de coursiers potentiels pour effectuer tous les shifts nécessaires.

3 Solution triviale

Vous pouvez dans un premier temps considérer la solution triviale suivante : pour chaque tournée $t \in \mathcal{T}$, on crée un shift qui contient uniquement cette tournée. Cette solution est réalisable, très facile à mettre en place, mais probablement avec des temps morts très élevés.

4 Interface graphique

Afin d'aider l'opérateur qui s'occupe de créer des shifts prévisionnels, *Deliver2I* souhaite que vous développiez une interface graphique de qualité, qui soit agréable au visuel et simple à utiliser par l'opérateur.

Voici les fonctionnalités que doit offrir cette interface :

- il faut pouvoir charger une nouvelle instance (voir la Section 6 pour plus d'informations sur le format des fichiers d'instance) ;
- il faut pouvoir visualiser la liste de toutes les instances ;
- il faut pouvoir sélectionner une instance et résoudre cette instance (proposer une solution pour cette instance) ;
- il faut pouvoir sélectionner une instance, et visualiser une ou plusieurs solutions de cette instance; en particulier, on souhaite connaître la somme des temps morts et avoir l'ensemble des shifts de la solution, avec pour chaque shift les tournées dont il est composé dans l'ordre chronologique.

5 Base de données

Afin de garder une trace de cette étape de création des shifts prévisionnels, le responsable du service planification de *Deliver2I* souhaite que vous enregistriez systématiquement en base de données toute instance qui est chargée depuis l'interface, ainsi que toutes les solutions qui sont associées à chacune des instances.

6 Fichiers d'instance

Deliver2I souhaite que l'outil que vous allez développer puisse être capable de charger des fichiers qui contiennent une instance du problème de constitution de shifts. Ce fichier est généré lors de l'étape de création des tournées prévisionnelles, et il contient donc les données nécessaires pour résoudre le problème de constitution des shifts.

Deliver2I vous a préparé un ensemble de 11 instances, qui sont disponibles sur Moodle dans le dossier `instances`. Vous devrez faire vos tests sur tous ces fichiers, plus éventuellement vos propres instances. L'instance `instance_test.csv` est une toute petite instance utile pour faire vos premiers tests. Les autres instances correspondent à des données réelles fournies par *Deliver2I*.

Chaque instance est sous forme d'un fichier `.csv`, avec comme séparateur des points-virgules. Voici les données que vous allez trouver, dans l'ordre, dans un fichier d'instance.

- `Nom` : donne un nom pour l'instance, sous forme d'une chaîne de caractères.
- `Duree min` : donne la durée minimale d'un shift T^{min} en minutes.
- `Duree max` : donne la durée maximale d'un shift T^{max} en minutes.
- `Date` : la date correspondant à cette instance, au format `jj/mm/aaaa`.
- Une ligne vide.
- Une ligne de noms de colonnes, qui contient les valeurs : `Debut` et `Fin`.
- Une ligne pour chaque tournée $t \in \mathcal{T}$: chaque ligne contient deux valeurs correspondant respectivement au début d_t et à la fin f_t de la tournée t. Les horaires de début et de fin sont exprimés au format `hh:mm`.

Notez que vous trouverez sur Moodle un paquetage **io** dans lequel une classe **InstanceReader** vous permet de réaliser la lecture des instances. Bien sûr, il faut apporter quelques modifications à cette classe pour qu'elle soit adaptée à votre modèle de données.

7 Directives informatiques

7.1 Développement de l'application

Le programme doit être écrit en Java. Les interactions avec la base de données doivent être réalisées avec JPA (*Java Persistence API*).

Pour le développement de votre programme, nous vous conseillons de le faire de manière incrémentale en suivant les étapes décrites ci-dessous.

7.1.1 Première version

Dans cette première version, il faut pouvoir :

- lire les fichiers d'instance (au format `csv`) ;
- enregistrer les instances dans la base de données ;
- visualiser sur une interface graphique les instances enregistrées dans la base de données.

Ici, l'objectif est de prendre en main le problème, en se concentrant sur les instances. L'idée est de mettre en place la base de données et l'interface dès le début.

7.1.2 Deuxième version

Dans cette deuxième version, il faut pouvoir :

- réaliser les traitements métiers afin de fournir une solution réalisable au problème ;
- enregistrer les solutions liées à une instance dans la base de données ;
- utiliser l'interface graphique pour donner une solution à une instance, et pouvoir visualiser les solutions d'une instance.

Ici, l'objectif est de valider que vous arrivez à produire des solutions réalisables (mêmes si elles ne sont pas de bonne qualité). Par ailleurs, vous continuez à développer le modèle objet et l'interface graphique en ajoutant la partie liée aux solutions.

7.1.3 Version finale

Pour aboutir à la version finale, vous pouvez vous focaliser sur l'un et/ou l'autre des points suivants.

- Améliorer le traitement algorithmique afin d'être capable de fournir des solutions de très bonne qualité, tout en restant sur des temps de résolution raisonnables (de l'ordre de la minute).
- Proposer une interface graphique de grande qualité. En particulier, un effort est attendu sur la visualisation des solutions : il faut un visuel simple et qui permette de voir l'ensemble de la solution.

8 Consignes générales

8.1 Plagia

Il n'est pas interdit d'utiliser des algorithmes *sur papier* provenant d'autres groupes, voire des codes trouvés sur internet, **à condition d'en citer les sources** et de ne constituer qu'une **partie minoritaire** du code global (inférieur à 10%). Toute infraction à cette règle sera considérée comme du plagia et sanctionnée comme tel, ce qui implique le risque de passer devant le conseil de discipline de Centrale Lille et d'en assumer les conséquences.

8.2 Travail en groupe

Le travail est à réaliser en binômes, que vous pouvez constituer comme vous le souhaitez dans chaque demi-groupe de TP. Au début de la séance du 3 décembre 2019, les élèves non encore affectés à un binôme seront affectés d'office par les enseignants.

9 Rendu

Vous déposez sur Moodle, dans la zone de dépôt **Dépôt du Projet 2019/2020** une archive compressée de votre répertoire de projet. Cette archive doit être nommée **POO3_projet_Nom1_Nom2.zip** avec les noms des membres du binôme (- 2 points si ce nommage n'est pas respecté). Cette archive doit comprendre tous vos fichiers sources, ainsi qu'un fichier qui indique, pour chaque instance, la valeur de la meilleure solution que vous avez trouvée.

La date limite de rendu est le **24 janvier 2020 à 23h59** (-2 points par heure de retard). Le retard sera calculé de la façon suivante : $\lceil \frac{nbMinutesRetard}{60} \rceil$ où $nbMinutesRetard$ est le nombre de minutes de retard. Donc une minute de retard comptera pour une heure et donnera lieu à 2 points de pénalisation.

Par ailleurs, une soutenance aura lieu le **27 janvier 2020**. Cette soutenance comportera une présentation de votre projet durant laquelle vous présenterez l'architecture de votre application, l'algorithme de résolution, l'organisation de votre travail en binôme, les résultats obtenus et une démonstration avec l'interface graphique. Puis, chaque membre du binôme sera interrogé individuellement sur le projet ainsi que sur les notions qui ont été abordées dans le cours de POO3. Lors de cette partie individuelle, il peut vous être demandé de réaliser du code afin d'évaluer vos capacités de programmation.

10 Notation

Une note globale sera donnée sur la base du travail fourni par le groupe, mais une note individuelle sera ensuite déterminée pour chaque étudiant. Chaque membre du groupe doit **participer activement** au développement de l'application et connaître son fonctionnement général (même s'il n'a pas tout implémenté). Durant la soutenance, chaque membre du groupe est sensé être capable de répondre aux questions sur toutes les différentes composantes de l'application, même s'il n'a pas développé la partie du code associée. Il ne faut donc pas compter sur l'autre membre de votre binôme pour réaliser tout le travail.

Seuls les projets qui proposent les éléments des deux premières versions (décrites dans les Sections 7.1.1–7.1.2) peuvent aspirer à obtenir une note positive. Bien sûr, la qualité de la solution proposée sera aussi importante pour le calcul de la note. Il n'est pas suffisant de *faire* les choses, mais il faut veiller à ce qu'elles soient *bien* faites.

Notez bien que la note du projet ne sera pas la note finale donnée à chaque élève. La note du projet est ensuite modulée (à la hausse ou à la baisse) en fonction de la partie individuelle lors de l'épreuve orale.

10.1 Critères de notation

Ce projet est évalué sur 20 points. La moitié des points est attribuée sur les critères suivants :

- la quantité de travail réalisé ;
- l'originalité du travail réalisé ;
- la qualité des solutions obtenues ;
- la qualité de l'interface graphique.

L'autre moitié des points est attribuée en fonction de la qualité du travail réalisé. Ceci englobe :

- la qualité du code (respect des principes de la programmation orientée objet) ;
- la qualité d'implémentation des algorithmes (utilisation des bonnes structures de données, attention portée à la complexité algorithmique des traitements) ;
- la présentation du code (indentation correcte, méthodes courtes, respects des conventions Java dans les noms de classe, attributs, méthodes et variables, commentaires pertinents au format Javadoc) ;
- l'avancement progressif de votre travail (vous devez avoir des choses à montrer à chaque séance et pas uniquement à la soutenance finale).

10.2 Participation active

Les enseignants se réservent la possibilité d'exclure de l'évaluation du travail d'un binôme un étudiant qui n'aurait pas participé activement au développement du programme. Sa note sera automatiquement zéro.