Programmation Orientée Objet
IG2I, LE4-SI – 2018-2019

Projet – Livraison du dernier kilomètre

Diego Cattaruzza, Maxime Ogier

1 Description du contexte

1.1 Livraison du dernier kilomètre

Deliver2I est une start-up récemment créée qui est spécialisée dans la livraison du dernier kilomètre sur la métropole lilloise. Tous les jours, elle se charge de livrer aux particuliers les colis de petite taille qu'ils ont commandé sur internet. *Deliver2I* possède un entrepôt situé près de l'autoroute A1, dans lequel sont livrés les colis en provenance des différents sites marchands (Amazon, Fnac, Darty, Cdiscount, Oscaro, Sarenza). Tous les colis réceptionnés le jour J à l'entrepôt peuvent être livrés au jour J+1 à leur destinataire.

1.2 Modes de livraison

L'originalité de *Deliver2I* est de proposer un ensemble d'alternatives de livraison de telle sorte que le client n'ait pas à se déplacer exprès pour aller chercher son colis (voir Figure 1). Actuellement, la livraison de colis se fait souvent à la maison (auquel cas le client peut devoir rester attendre son colis), ou en point relais (auquel cas le client doit se déplacer exprès pour aller chercher le colis). *Deliver2I* a misé sur le fait qu'il est possible aujourd'hui d'utiliser des moyens de livraison plus innovants. En effet, au lieu des classiques points relais, il existe des *lockers* (consignes automatiques) qui sont installés dans les lieux facilement accessibles au public avec des horaires d'ouverture élargis (gares, centres commerciaux). Lorsque le colis est livré dans un *locker*, le client reçoit alors un code unique qui lui permettra de récupérer son colis quand il le souhaite. Par ailleurs, un nouveau concept de livraison a été proposé il y a trois ans : la livraison dans le coffre de la voiture du client. Volvo a été le premier à proposer ce type de service en Suède en 2016, et Amazon propose aussi ce concept aux États-Unis depuis 2018. Les voitures sont équipées d'un système d'ouverture à distance qui permet au livreur d'ouvrir une seule fois le coffre pour y déposer le colis.

Ainsi, lorsqu'un colis arrive à l'entrepôt de *Deliver2I*, le client est averti, puis il choisit son jour de livraison (à partir du lendemain). Ensuite, pour le jour de livraison choisi, le client renseigne les différent lieux où il se trouvera

Figure 1: Les différents modes de livraison utilisés par *Deliver2I*.

pendant la journée, avec les créneaux horaires associés. *Deliver2I* peut ensuite livrer le client lorsqu'il est chez lui, ou dans le coffre de sa voiture à n'importe quel moment, ou dans un *locker* situé dans un lieu où passera le client, avant le passage du client.

1.3 Problème à résoudre

Deliver2I table sur une forte croissance de son activité dans les mois à venir, et envisage de devoir livrer plus de 100 clients par jour. Il est donc impératif pour *Deliver2I* de disposer dans son système d'informations d'un solveur performant qui considère les différents services de livraison : à la maison, dans le coffre du véhicule, dans un *locker*. Il s'agit d'un problème de tournées de véhicules très original puisqu'un même client peut être livré à des lieux différents, avec des créneaux horaires différents. Un exemple d'instance de ce problème est proposé dans la Figure 2.

Chaque jour, à partir d'un ensemble de colis à livrer, il faut prendre les décisions suivantes :

- choisir le lieu de livraison du colis ;
- déterminer l'horaire de livraison du colis ;
- proposer des tournées qui partent de l'entrepôt pour livrer tous les clients au lieu et à l'heure choisis.

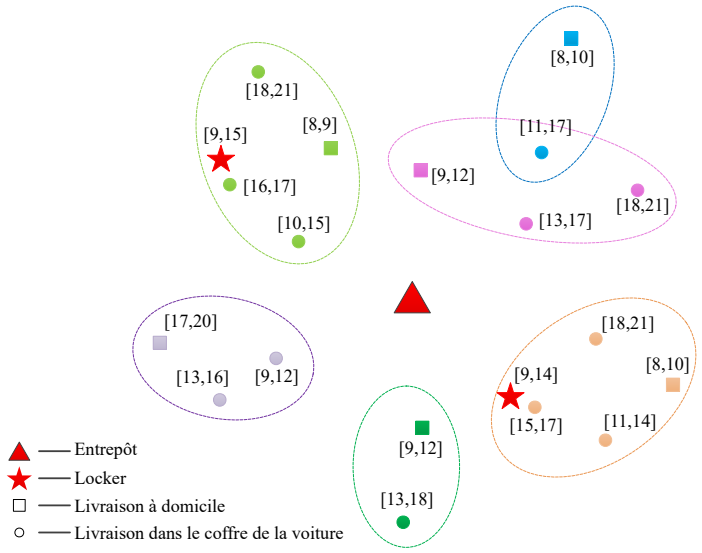

Figure 2: Une instance du problème : chaque client est associé à plusieurs localisations avec des créneaux horaires différents.

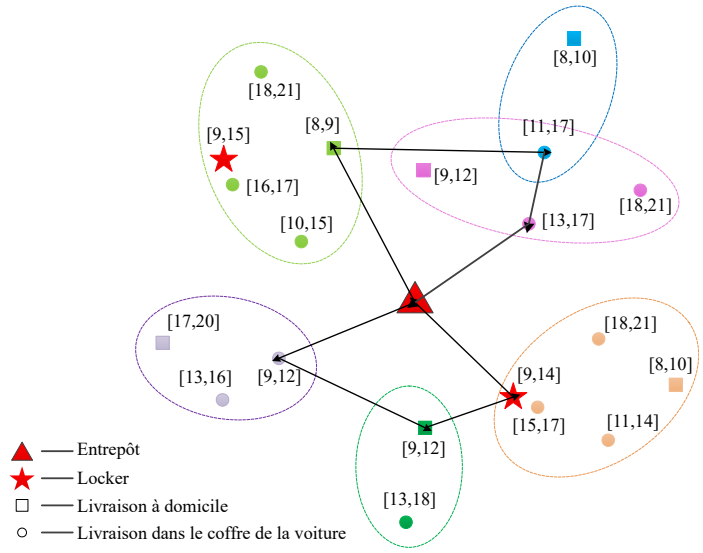

Figure 3: Un exemple de solution avec 2 tournées pour l'instance donnée Figure 2.

L'objectif pour *Deliver2I* est de minimiser ses coûts de livraison : des coûts de fonctionnement liés à la distance parcourue, et des coûts liés à la sous-traitance si son équipe de livreurs n'est pas suffisante et qu'il faut faire appel à des livreurs extérieurs. Notez que le choix des lieux et des horaires de livraison a une très grande influence sur les tournées et donc les coûts de livraison. Un exemple de solution du problème est proposé dans la Figure 3.

2 Description du problème à résoudre

Nous explicitons ici le problème auquel nous nous intéressons : étant donné un ensemble de colis à livrer, il faut déterminer les lieux et horaires de livraison ainsi que l'ensemble des tournées de livraison.

2.1 Les données

Nous récapitulons ici toutes les données nécessaires au problème. Nous supposons qu'avant d'établir le planning des livraisons, les colis sont présents dans l'entrepôt, et tous les clients ont donné les lieux où ils se trouveront avec les créneaux horaires associés.

Nous noterons :

- K le nombre de clients à livrer ;
- T le temps maximum disponible (en minutes) pour effectuer toutes les livraisons ;
- M le nombre de livreurs employés par *Deliver2I* qui sont disponibles à l'entrepôt ;
- C_{EXT} le coût unitaire pour utiliser un véhicule et un livreur d'un prestataire extérieur ;
- Q la capacité des véhicules (volume maximal du chargement) ;
- s l'entrepôt où se trouvent tous les colis ;
- $[E_s, L_s] = [0, T]$ l'intervalle de temps pendant lequel les livreurs peuvent effectuer leur tournée de livraison ;
- $\mathcal{K} = \{1, ..., K\}$ l'ensemble des clients à livrer.

Pour chaque client $k \in \mathcal{K}$, nous noterons :

- Q_k le volume du colis à livrer au client ;
- $\mathcal{V}_k$ l'ensemble des localisations possibles du client ;
- pour chaque localisation $i \in \mathcal{V}_k$, $[E_i, L_i]$ est le créneau horaire pendant lequel le client k se trouve à la localisation i (on appelle $[E_i, L_i]$ une *fenêtre de temps*).

Par ailleurs, afin de modéliser le réseau routier, nous notons :

- $\mathcal{V} = \{s\} \cup \bigcup_{k \in \mathcal{K}} \mathcal{V}_k$ l'ensemble des localisations considérées ;
- pour chaque localisation $i \in \mathcal{V}$, $C(i)$ représente le client associé à la localisation i, de telle sorte que $C(s) = 0$, et $C(i) = k, \forall k \in \mathcal{K}, i \in \mathcal{V}_k$;
- $\mathcal{A} = \{(i,j) | i \in \mathcal{V}, j \in \mathcal{V}\}$ l'ensemble des arcs qui permettent d'aller d'une localisation vers une autre ;
- pour chaque arc $(i,j) \in \mathcal{A}$, T_{ij} est le temps nécessaire (en minutes) pour aller de i vers j, et C_{ij} est le coût lié à l'utilisation du véhicule pour aller de i vers j.

2.2 Les décisions

Pour résoudre le problème de livraison de *Deliver2I*, plusieurs décisions liées entre elles doivent être prises :

- pour chaque client $k \in \mathcal{K}$, il faut choisir une localisation de livraison dans $\mathcal{V}_k$;
- pour chaque client $k \in \mathcal{K}$, il faut déterminer un horaire de livraison t_k ;
- les livraisons des clients doivent être regroupées au sein d'une ou plusieurs tournées, et chaque tournée doit indiquer l'ordre de livraison des clients.

2.3 Solution

Pour donner une solution au problème, il suffit de proposer un ensemble de tournées $\mathcal{R}$. À chaque tournée $r \in \mathcal{R}$, on associe une séquence de $n(r)$ visites $(v_1^r, v_2^r, ..., v_{n(r)}^r)$. Une visite $v_i^r = (u_i^r, t_i^r)$ correspond à arriver à la localisation u_i^r au temps t_i^r.

2.4 Contraintes du problème

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

- chaque client $k \in \mathcal{K}$ est visité dans une tournée, dans une de ses localisations $\mathcal{V}_k$;
- chaque tournée $r \in \mathcal{R}$ part de s et arrive à s, i.e. $u_1^r = s$ et $u_{n(r)}^r = s$;
- le volume des colis livrés dans une tournée n'excède pas la capacité Q des véhicules, i.e. $\sum_{i=2}^{n(r)-1} Q_{C(u_i^r)} \leq Q$;
- dans chaque tournée $r \in \mathcal{R}$, les visites ont lieu pendant le créneau horaire associé à la localisation visitée, i.e. $E_{u_i^r} \leq t_i^r \leq L_{u_i^r}$;
- dans chaque tournée $r \in \mathcal{R}$, le séquencement des visites est réalisable, i.e. $t_{i+1}^r \geq t_i^r + T_{u_i^r, u_{i+1}^r}, \forall i = 1..n(r)-1$.

2.5 Fonction objectif

Deliver2I souhaite payer le moins cher possible pour réaliser ses tournées de livraison. Pour chaque véhicule utilisé, il y a un coût à payer en fonction du trajet utilisé (ce coût dépend de la distance parcourue : utilisation du véhicule, carburant). *Deliver2I* dispose de M livreurs et véhicules. Donc pour eux, les coût fixes (salaires, charges du véhicule) sont déjà payés et ne dépendent pas des tournées effectuées. En revanche, si *Deliver2I* a besoin de faire plus de M tournées pour effectuer ses livraisons, ils doivent alors faire appel en dernière minute à un prestataire extérieur. Chaque tournée effectuée par un prestataire extérieur a un coût de C_{EXT}.

Pour chaque tournée $r \in \mathcal{R}$, on note son coût c_r, et ce coût correspond à la somme des coûts des arcs utilisés pour réaliser la tournée : $c_r = \sum_{i=1}^{n(r)-1} C_{u_i^r, u_{i+1}^r}$.

Ainsi, l'objectif ici est de minimiser le coût total qui s'exprime comme $c = \sum_{r \in \mathcal{R}} c_r + C_{EXT} \times \max\{0, |\mathcal{R}| - M\}$.

2.6 Hypothèses sur les données

On suppose que les données permettent toujours d'obtenir une solution réalisable. Ainsi, pour chaque client $k \in \mathcal{K}$, il existe toujours au moins une localisation qui peut être livrée depuis le dépôt dans sa fenêtre de temps.

Par ailleurs, les coûts et les temps respectent l'inégalité triangulaire, i.e. $C_{ij} + C_{jk} \geq C_{ik}, \forall (i,j,k) \in \mathcal{V}$ et $T_{ij} + T_{jk} \geq T_{ik}, \forall (i,j,k) \in \mathcal{V}$.

3 Solution triviale

Vous pouvez dans un premier temps considérer la solution triviale suivante :

- pour chaque client $k \in \mathcal{K}$, on cherche une localisation dans $\mathcal{V}_k$ qui peut être livrée dans sa fenêtre de temps depuis le dépôt ;
- on crée une tournée pour livrer seulement cette localisation.

Cette solution est réalisable mais probablement avec un coût très élevé. On peut ensuite chercher à faire mieux en regroupant des localisations dans les tournées.

4 Fichiers d'instance et fichiers de solution

4.1 Fichiers d'instance

Un ensemble de 40 instances est fourni sur Moodle dans le dossier `instances`. Vous devrez faire vos tests sur tous ces fichiers, plus éventuellement vos propres instances. Lors de l'évaluation du projet, nous testerons les programmes sur ce jeu d'instances, plus d'autres instances.

Chaque instance est sous forme d'un fichier texte, avec comme séparateur des tabulations. Voici les données que vous allez trouver, dans l'ordre, dans un fichier d'instance.

- `NB_CUSTOMERS` : donne le nombre de clients, donc c'est égal à K, la cardinalité de $\mathcal{K}$.
- `NB_LOCATIONS` : donne le nombre de localisations, donc c'est égal à la cardinalité de $\mathcal{V}$.
- `NB_VEHICLES` : donne le nombre de véhicules et livreurs disponibles à l'entrepôt, donc c'est M.
- `VEHICLE_CAPACITY` : donne la capacité de chaque véhicule, il s'agit d'un volume en dm^3.
- `EXTERNAL_VEHICLE_COST` : donne le coût pour utiliser un véhicule et un livreur d'un prestataire extérieur, donc c'est C_{EXT}.
- `DEPOT` : donne les informations sur l'entrepôt s :
  - son identifiant (0) ;
  - sa demande (0) ;
  - son nombre de localisations (1) ;
  - l'identifiant de la localisation (0).
- `CUSTOMERS` : donne les informations sur les clients. Pour chaque client $k \in \mathcal{K}$, une ligne donne les informations suivantes :
  - l'identifiant du client (entre 1 et K) ;
  - le volume Q_k du colis (en dm^3) ;
  - le nombre de localisations possibles, i.e. $|\mathcal{V}_k|$;
  - les identifiants des localisations du client.
- `LOCATIONS` : donne les informations sur les localisations. Pour chaque localisation $v \in \mathcal{V}$, une ligne donne les informations suivantes :
  - l'identifiant de la localisation (entre 0 et $|\mathcal{V}| - 1$) ;
  - les coordonnées (abscisse et ordonnée) de la localisation dans un repère orthonormé centré sur l'entrepôt s ;
  - la fenêtre de temps $[E_v, L_v]$ pendant laquelle la localisation v peut être visitée (E_v et L_v sont des temps en minute depuis le début de l'horizon de planification qui est fixé à 0).
- `TRAVEL_COSTS_TIMES` : donne les informations sur les arcs. Pour chaque arc $(i, j) \in \mathcal{A}$, une ligne donne les informations suivantes :
  - les identifiants i et j de l'arc (i, j) ;
  - le coût C_{ij} pour aller de la localisation i vers la localisation j ;
  - le temps T_{ij} (en minutes) pour aller de la localisation i vers la localisation j.

4.2 Fichier de solution

Si le nom du fichier d'instance est `nomInstance.txt`, alors le nom du fichier de solution associé devra être `nomInstance_sol.txt`. Ceci est notamment nécessaire pour que la solution puisse être contrôlée par le *checker* proposé (voir Section 5).

Chaque fichier solution est sous forme d'un fichier texte, avec comme séparateurs des tabulations ou espaces. Le fichier de solution contient juste la description des localisations visitées, de la manière suivante :

- chaque ligne correspond à une tournée ;
- sur chaque ligne, on trouve les identifiants des localisations visitées, dans l'ordre où elles sont visitées.

Notez qu'un fichier de solution doit donc contenir K identifiants entre 1 et $|\mathcal{V}|-1$. Le dépôt dont l'identifiant est 0 n'apparait pas dans le fichier de solution.

Un exemple de fichier de solution est fourni dans le dossier `solutions`. Un programme dont les fichiers solutions ne respectent pas le format demandé n'est pas considéré valide.

5 Validation de la solution

Un *checker* est à votre disposition dans le dossier `checker` qui se trouve sur Moodle. Il est nommé `CheckerLastMileDelivery.jar`. Pour utiliser le *checker* vous devez le mettre dans le même dossier que les fichiers suivants :

- le fichier d'instance ;
- le fichier de solution associé.

Vous devez impérativement respecter le nommage des fichiers et leur format. Le *checker* s'utilise en ouvrant une invite de commande, pour tester l'instance nommée `nomInstance.txt`, il suffit de taper la commande suivante : `java -jar CheckerLastMileDelivery.jar nomInstance`.

6 Directives informatiques

6.1 Développement de l'application

Le programme doit être écrit en Java.

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

6.1.1 Première version

Dans cette première version, il faut pouvoir :

- lire les fichiers d'instance (en format `txt`) ;
- réaliser les traitements métiers afin de fournir une solution réalisable au problème dans un temps raisonnable (de l'ordre d'une minute au maximum) ;
- écrire la solution selon le format spécifié dans un fichier `txt`.

Ici, l'objectif est de prendre en main le problème, et de valider que vous arrivez à produire des solutions réalisables (mêmes si elles ne sont pas de bonne qualité). Par ailleurs, dans cette première version, il s'agit d'un programme Java sans interface et sans base de données. Mais cela vous permet d'entamer la réflexion sur le modèle objet à utiliser.

6.1.2 Deuxième version

Dans cette deuxième version, il faut pouvoir :

- stocker les données de l'instance lue dans une base de données ;
- stocker la solution associée dans une base de données.

Après avoir réfléchi au modèle objet, il devrait être assez simple de mettre en place la base de données.

6.1.3 Troisième version

Dans cette troisième version, il faut travailler sur la partie algorithmique pour améliorer la qualité des solutions proposées. Vous pouvez réutiliser des éléments algorithmiques vus dans les séances de TD, ou bien proposer d'autres approches, qu'il est préférable de valider au préalable avec les enseignants. Par ailleurs, faites attention à ce que le temps de résolution reste raisonnable (quelques minutes au maximum).

6.1.4 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.
- Proposer une interface graphique qui permet notamment de visualiser les solutions. Il peut s'agir d'une interface graphique développée en Java avec Swing, ou bien d'une interface web. Dans le cas d'une interface web, on peut même envisager un autre programme qui vient se connecter directement sur la base de données.

7 Consignes générales

7.1 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.

7.2 Travail en groupe

Le travail est à réaliser par groupes de 3 ou 4 élèves, que vous pouvez constituer comme vous le souhaitez. À la fin de la séance du 2 avril 2019, les personnes non encore affectées à un groupe seront affectées d'office par les enseignants.

8 Rendu

Vous déposez sur Moodle, dans la zone de dépôt **Dépôt du Projet 2018/2019** une archive compressée de votre répertoire de projet. Cette archive doit être

nommée **Projet_Nom1_Nom2_Nom3_Nom4.zip** avec les noms des membres de l'équipe (- 2 points si ce nommage n'est pas respecté). Cette archive doit comprendre tous vos fichiers sources, les fichiers d'instances et les fichiers de solutions.

La date limite de rendu est le **13 juin 2019 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 **14 juin 2019 après-midi**. 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 groupe, les résultats obtenus et une démonstration si vous avez une interface graphique. Il s'en suivra des questions posées par les enseignants.

9 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.

Seuls les projets qui proposent les éléments des trois premières versions (décrites dans les Sections 6.1.1–6.1.3) peuvent aspirer à obtenir une note qui permet de valider le projet. 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.

9.1 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.

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, mise en place des *design pattern* vus en cours) ;
- 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).

9.2 Coefficients individuels

Par ailleurs, chaque groupe doit proposer un coefficient par membre de son groupe, reflétant l'investissement de chacun des membres sur le projet. Ces coefficients seront ensuite multipliés par la note globale que le groupe a obtenue, ce qui donnera une note individuelle sur ce projet. La moyenne de ces coefficients doit être de 1, et la différence entre le plus grand et le plus petit coefficient doit être d'au moins 0,05 et d'au plus 0,4.

Ces coefficients doivent être **envoyés par mail aux enseignants**, avec tous les membres du groupe en copie, avant le **13 juin 2019 à 23h59**.

9.3 Participation active

Les enseignants se réservent la possibilité d'exclure de l'évaluation du travail d'un groupe les étudiants qui n'ont pas participé activement au développement du programme. Leur note sera automatiquement zéro.