# Programmation Orientée Objet
# IG2I, LE4-SI – 2020-2021

## Projet – Livraison et installation de machines

*Diego Cattaruzza, Maxime Ogier*

### 1 Description du contexte

*Machines2I* est une entreprise spécialisée dans la livraison et dans l'installation de différents types de machines. Les clients de *Machines2I* demandent la livraison d'un ensemble de machines ainsi que leur installation à faire sur place par des techniciens spécialisés. Les deux opérations doivent se faire pendant un horizon de temps, appelé horizon de planification. Chaque client fournit une fenêtre de temps (un intervalle horaire) pendant laquelle la livraison doit se faire. Cette fenêtre de temps consiste en un ou plusieurs jours de l'horizon de planification. Une fois la livraison effectuée, un technicien, habilité à l'installation de ces machines, viendra chez le client pour l'installation.

*Machines2I* possède un entrepôt central qui contient toutes les machines à installer. *Machines2I* fait appel à des livreurs en externe pour apporter les machines chez les clients. De même, *Machines2I* fait appel à des techniciens en externe pour les opérations d'installation. *Machines2I* s'intéresse à déterminer le planning de livraison des machines et le planning des installations par les techniciens sur une période de plusieurs jours. Ce planning vise à minimiser le coût opérationnel total.

### 2 Description du problème à résoudre

**Livraison des machines.** Toutes les machines sont situées dans l'entrepôt de *Machines2I* au début de l'horizon de planification. L'entrepôt contient un nombre suffisant de machines de chaque type pour satisfaire toutes les demandes. Des camions sont loués pour transporter les machines du dépôt aux clients. On suppose que la flotte de véhicules n'est pas limitée. On suppose aussi que tous les camions sont identiques.

Un camion peut accueillir n'importe quelle combinaison de machines du moment que la taille totale ne dépasse pas la capacité du camion. La capacité d'un camion est exprimée dans la même unité que la dimension des machines.

La livraison d'une demande d'un client ne peut pas être fractionnée, c'est-à-dire que toutes les machines d'une même demande doivent être livrées simultanément par un seul camion. On suppose ainsi qu'aucune des tailles de demandes d'un client ne dépasse la capacité du véhicule.

L'itinéraire quotidien d'un camion commence et termine au dépôt. Un camion peut revenir au dépôt plusieurs fois dans la journée pour récupérer des machines. La distance totale parcourue par un camion chaque jour est limitée à une valeur maximale. On considère que les temps de chargement des machines au dépôt et les temps de déchargement des machines chez les clients sont nuls.

**Installation des machines.** Après la livraison, chaque machine doit être installée chez le client par un technicien habilité. L'installation d'une machine ne peut pas avoir lieu le jour de la livraison. Par contre, pour chaque jour complet où une machine est *à l'arrêt*, c'est-à-dire qu'elle est livrée chez le client mais pas encore installée, une pénalité est appliquée. Cette pénalité est spécifique pour chaque type de machine. Ce concept est illustré avec un exemple. Supposons qu'une demande avec trois machines de type A soit livrée à un client le 4ème jour de l'horizon de planification. Les machines sont installées par un technicien le 7ème jour de l'horizon. La pénalité totale liée à cette demande est donc $3 \times 2 = 6$ fois la pénalité associée à la machine de type A, car les trois machines ont été inactives pendant 2 jours (les jours 5 et 6). Si les machines sont livrées le 4ème jour et installées le 5ème jour, alors aucune pénalité n'est appliquée.

Chaque technicien possède un ensemble de compétences qui détermine les types de machines qu'il peut installer. On suppose que l'installation d'une machine n'exige pas de temps d'installation. Le nombre maximum de jours consécutifs qu'un technicien peut travailler est de 5. Si un technicien a travaillé pendant cinq jours consécutifs, il doit avoir deux jours de repos. Si un technicien a travaillé pendant moins de cinq jours consécutifs, un seul jour de repos suffit. Tous les jours en dehors de l'horizon de planification sont considérés comme des jours de repos.

Le parcours quotidien d'un technicien commence et se termine à son domicile. La distance totale qu'un technicien peut parcourir chaque jour est limitée à une valeur maximum donnée. Il en va de même pour le nombre de demandes qu'un technicien peut traiter par jour. Le traitement d'une demande implique l'installation de toutes les machines liées à cette demande.

**Objectif.** Dans une solution, toutes les machines demandées par les clients doivent être livrées et installées pendant l'horizon de planification. L'objectif est de minimiser le coût total. Il existe des coûts :

- par unité de distance parcourue par camion ;
- pour avoir utilisé un camion pendant une journée ;
- pour avoir utilisé un camion pendant la durée de la planification ;
- par unité de distance parcourue par un technicien ;
- pour faire appel à un technicien pour une journée ;

- pour avoir fait appel à un technicien pendant la durée de la planification ;
- pour chaque jour complet où une machine est *à l'arrêt* chez un client.

Afin de déterminer les distances parcourues, les coordonnées sont fournies pour le dépôt, les localisations des clients, et les lieux de résidence des techniciens. La distance entre chaque paire de points $P_1 = (x_1; y_1)$ et $P_2 = (x_2; y_2)$ est la partie entière par excès de la distance Euclidienne entre $P_1$ et $P_2$, c'est-à-dire :

$$d(P_1, P_2) = \left\lceil \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \right\rceil$$

**Récapitulatif des contraintes.** On récapitule ici les contraintes du problème.

- Contraintes liées aux véhicules.
  - Tout véhicule part depuis le dépôt en début de journée.
  - Tout véhicule revient au dépôt en fin de journée.
  - La distance parcourue chaque jour par un véhicule ne doit pas dépasser une distance maximale.
  - Chaque tournée doit respecter la capacité du véhicule.
- Contraintes liées aux demandes.
  - Toute demande doit être livrée.
  - Toute demande doit être installée.
  - La livraison d'une demande ne peut pas être fractionnée.
  - L'installation des machines d'une demande ne peut pas être fractionnée.
  - L'installation d'une machine est possible dès le lendemain de la livraison.
- Contraintes liées aux techniciens.
  - Chaque technicien peut installer seulement les machines pour lesquelles il est habilité.
  - Tout technicien commence la journée de travail à son domicile.
  - Tout technicien revient au domicile en fin de journée.
  - Le planning des installations de chaque technicien doit respecter les jours de repos imposés.
  - La distance parcourue chaque jour par un technicien ne doit pas dépasser une distance maximale.
  - Les demandes traitées chaque jour par un technicien ne doivent pas dépasser un nombre maximal.

### 2.1 Hypothèses sur les données

On suppose que les données permettent toujours d'obtenir une solution réalisable. Ainsi, la demande de chaque client n'excède pas la capacité des véhicules et, pour chaque type de machine, il existe toujours au moins un technicien habilité à l'installer.

## 3 Solution triviale

Les instances dans les dossiers `ORTEC-early` et `ORTEC-late` ont été proposées par l'entreprise ORTEC (`https://ortec.com/fr-fr`) dans le cadre d'un challenge scientifique. Ces instances ont été modifiées pour créer d'autres instances plus faciles que vous trouvez dans les dossiers `ORTEC-early-easy` et `ORTEC-late-easy`. Pour ces dernières instances, le nombre de techniciens disponibles est suffisamment large pour que la solution triviale suivante soit faisable :

- chaque demande est livrée par un véhicule le premier jour où la livraison est possible ;
- un technicien habilité passe installer les machines chez chaque client le lendemain de la livraison.

Cette solution triviale est bien sûr très coûteuse, mais vous permet de faire les premiers tests et de prendre en main le checker mis à disposition.

Cette solution pourrait ne pas être réalisable pour les instances dans les dossiers `ORTEC-early` et `ORTEC-late`.

Nous vous conseillons d'utiliser d'abord les instances faciles et de passer aux instances difficiles une fois qu'un algorithme d'optimisation plus avancé a été codé.

Pour améliorer la solution triviale, on peut regrouper des livraisons et des installations dans une même tournée ou regrouper des livraisons et des installations de tournées différentes et les affecter à un même camion ou technicien.

Notez que si vous regroupez des livraisons pour les affecter à un même camion, vous avez la possibilité de revenir au dépôt si la capacité du véhicule ne permet pas de transporter toutes les demandes dans une seule tournée.

Pour un technicien, il n'est jamais intéressant de repasser par son domicile pendant la journée de travail.

## 4 Fichiers d'instance et fichiers de solution

### 4.1 Fichiers d'instance

Un ensemble de 100 instances est fourni sur Moodle dans le dossier `instances`. Vous y trouverez 4 dossiers différents, chacun avec 25 instances. Allez à la Section 3 pour plus d'informations sur ces dossiers. 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 espaces. Voici les données que vous allez trouver, dans l'ordre, dans un fichier d'instance.

- `DATASET` : le nom de l'ensemble des instances.
- `NAME` : le nom de l'instance.
- `DAYS` : le nombre de jours dans l'horizon de planification. Les jours sont numérotés 1, 2, et ainsi de suite.
- `TRUCK_CAPACITY` : la capacité d'un camion.
- `TRUCK_MAX_DISTANCE` : la distance maximale qu'un camion peut parcourir chaque jour.
- `TRUCK_DISTANCE_COST` : le coût par unité de distance parcourue par le camion.
- `TRUCK_DAY_COST` : le coût journalier de chaque camion utilisé.
- `TRUCK_COST` : le coût de l'utilisation d'un camion pendant une journée quelconque de l'horizon de planification. Ce coût doit être multiplié par le nombre maximum de camions utilisés pendant la planification. Nous rappelons ici que tous les camions sont identiques.
- `TECHNICIAN_DISTANCE_COST` : le coût par unité de distance parcourue par un technicien.
- `TECHNICIAN_DAY_COST` : le coût journalier de chaque technicien utilisé.
- `TECHNICIAN_COST` : le coût de l'utilisation d'un technicien au cours d'une journée quelconque de l'horizon de planification. Ce coût doit être multiplié par le nombre maximum de techniciens utilisés pendant la planification.

Les quatre sections suivantes commencent par un nom, suivi du nombre d'entrées dans cette section. Par exemple, `MACHINES = 4` indique qu'il existe 4 types de machines. Le détail de chacune de ces machines est décrit dans les 4 lignes suivantes.

- `MACHINES = NbTypeMachines` : les différents types de machines sont répertoriés (un par ligne) dans les `NbTypeMachines` lignes qui suivent. Dans chaque ligne :
  - Le premier chiffre est l'identifiant du type de machine.
  - Le deuxième chiffre donne la taille d'une machine de ce type.
  - Le troisième chiffre indique la pénalité qui est appliquée pour chaque jour complet où une machine de ce type est inutilisée.
- `LOCATIONS = NbLocations` : les différentes coordonnées du dépôt, des clients et des lieux de résidence des techniciens sont indiqués dans les `NbLocations` lignes qui suivent. Dans chaque ligne :
  - Le premier chiffre est l'identifiant de la localisation.
  - Le deuxième chiffre est l'abscisse de la localisation.
  - Le troisième chiffre est l'ordonnée de la localisation.
  - Le dépôt est toujours associé à l'identifiant 1.
- `REQUESTS = NbRequests` : contient les `NbRequests` demandes en machines des clients. Dans chaque ligne :
  - Le premier chiffre est l'identifiant de la demande.
  - Le deuxième chiffre indique l'identifiant de la localisation du client associé à la demande.
  - Le troisième chiffre est le premier jour de la fenêtre de livraison associée à cette demande.
  - Le quatrième chiffre est le dernier jour de la fenêtre de livraison associée à cette demande.
  - Le cinquième chiffre est l'identifiant de la machine demandée.
  - Le sixième chiffre indique le nombre de machines demandées.
  - On suppose que toute demande est associée à un seul type de machine. Si un client demande deux ou plusieurs types de machines, nous aurons plusieurs demandes associées au même client.
- `TECHNICIANS = nbTechniciens` : contient les `nbTechniciens` techniciens disponibles. Dans chaque ligne :
  - Le premier chiffre est l'identifiant du technicien.
  - Le deuxième chiffre indique l'identifiant de la localisation du technicien.
  - Le troisième chiffre indique la distance maximale parcourue par le technicien pendant une journée de travail.
  - Le quatrième chiffre indique le nombre maximal de demandes traitées par le technicien pendant une journée de travail.
  - Les autres chiffres indiquent pour chaque machine si le technicien peut l'installer (1) ou pas (0).
  - Il est possible que la localisation d'un technicien soit la même que celle du dépôt ou que celle d'un client.
- Toute valeur dans le fichier d'instance est un nombre entier.

Vous trouvez sur Moodle, fichier `testInstance.txt` un exemple d'instance.

### 4.2 Fichier de solution

Chaque fichier solution est sous forme d'un fichier texte, avec comme séparateurs des espaces.

Les deux premières lignes (`DATASET` et `NAME`) font référence à l'instance à laquelle est liée la solution. La section suivante, qui énumère certaines caractéristiques de la solution (en commençant par `TRUCK_DISTANCE` et se terminant par `TOTAL_COST`) est facultative. Néanmoins si elle est incluse, elle doit contenir les informations suivantes :

- `TRUCK_DISTANCE` : la distance totale parcourue par les camions.
- `NUMBER_OF_TRUCK_DAYS` : le nombre total de journées camion utilisées.
- `NUMBER_OF_TRUCKS_USED` : le nombre maximal de camion utilisés dans une journée.
- `TECHNICIAN_DISTANCE` : la distance totale parcourue par les techniciens.
- `NUMBER_OF_TECHNICIAN_DAYS` : le nombre total de journées technicien utilisées.
- `NUMBER_OF_TECHNICIANS_USED` : le nombre total de techniciens utilisés.
- `IDLE_MACHINE_COSTS` : le coût des machines *à l'arrêt*.
- `TOTAL_COST` : le coût total de la solution.

Ce paragraphe facultatif permet de vérifier plus facilement le coût total de la solution. Ces résultats sont également retournés par le checker et peuvent donc être vérifiés par l'utilisateur.

Après ces sections, la solution complète est donnée, jour après jour. La partie dédiée à chaque jour contient :

- `DAY = I`, où `I` est l'index du jour.
- Une ligne commençant par l'en-tête `NUMBER_OF_TRUCKS =` suivi par le nombre de camions utilisés ce jour, suivi des itinéraires des camions.
- Une ligne pour chaque camion. Chaque ligne contient :
  - L'identifiant du camion. On rappelle que tous les camions sont identiques, donc on peut les numéroter 1, 2, 3, et ainsi de suite.
  - La séquence (liste ordonnée) de demandes desservies par le camion. Si le camion revient au dépôt entre la livraison de deux demandes, alors il y aura un 0 entre elles.
  - Notez que tous les camions partent du dépôt et doivent y revenir à la fin de la journée, donc cette information n'est pas mentionnée explicitement dans le format de la solution.
- Une ligne commençant par l'en-tête `NUMBER_OF_TECHNICIANS =` suivi par le nombre de techniciens utilisés ce jour.
- Une ligne par technicien. Chaque ligne contient :
  - L'identifiant du technicien.
  - La séquence (liste ordonnée) des identifiants des demandes installées par le technicien.
  - Notez que tous les techniciens partent de leur maison et doivent y revenir à la fin de la journée, donc cette information n'est pas mentionnée explicitement dans le format de la solution.

**Tous les jours de l'horizon de planification doivent apparaître dans la solution, même si aucun camion ou technicien n'est utilisé pendant cette journée**. De plus, pour chaque jour, le `NUMBER_OF_TRUCKS` et le `NUMBER_OF_TECHNICIANS` doivent toujours avoir une valeur, même si elle est de 0 pour ce jour.

Vous trouvez sur Moodle, fichier `testInstance_sol.txt` un exemple de solution, liée à l'instance `testInstance.txt`.

## 5 Validation de la solution

Un *checker* est à votre disposition dans le dossier `checker` qui se trouve sur Moodle. Il est nommé `SolutionVerolog2019.py` et il utilise les fichiers `baseParser.py` et `InstanceVerolog2019.py`.

Le checker est codé en Python. Pour l'utiliser vous pouvez utiliser le site `repl.it`. Dans la section `File` vous devez importer les trois fichiers `SolutionVerolog2019.py`, `baseParser.py` et `InstanceVerolog2019.py`, puis les fichiers instance et solution.

Dans le fichier `main.py` que vous trouvez sur votre session `repl.it`, vous pouvez écrire le code suivant :

```
import os
\# verify the solution
os.system('python SolutionVerolog2019.py
                --solution testInstance_sol.txt
                --instance testInstance.txt'
                )
```

où `testInstance_sol.txt` est le fichier de votre solution et `testInstance.txt` est le fichier de votre instance. Puis cliquez sur le bouton `run`.

## 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 triviale (comme proposé dans la Section 3) ;
- 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 du tout de bonne qualité). Par ailleurs, dans cette première version, il s'agit d'un programme Java sans interface. 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 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).

#### 6.1.3 Version finale

Pour aboutir à la version finale, vous vous focaliserez sur les deux 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 votre programme, ou qui utilise les fichiers d'instance et de solution.

## 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 31 mars 2021, 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 2020/2021** 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 vendredi **18 juin 2021 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 **22 juin 2021 matin, salle IG-201** (si les mesures sanitaires le permettront, sinon à distance). 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 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. Les points sont attribués en tenant en compte des aspects suivants :

- La qualité de votre code. Les aspects pris en compte sont par exemple : :
  - le respect des principes de la programmation orientée objet ;
  - la 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) ;
  - la factorisation du code pour éviter les redondances ;
  - la présence et la qualité des tests.
- L'algorithmique. Les aspects pris en compte sont par exemple : :
  - l'originalité et la variété des algorithmes proposés ;
  - la qualité des solutions obtenues.
- L'interface graphique. Les aspects pris en compte sont par exemple : :
  - la clarté et l'exhaustivité des informations affichées ;
  - le nombre et la pertinence des fonctionnalités ;
  - l'expérience utilisateur.
- La soutenance finale. Les aspects pris en compte sont par exemple : :
  - la clarté de la présentation ;
  - la qualité des supports ;
  - la présentation critique de résultats.
- 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 **18 juin 2021 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.