Programmation Orientée Objet
IG2I, LE4-SI – 2017-2018

Projet - Gestion d'entrepôt : *batching* et *picking*

Diego Cattaruzza, Maxime Ogier

1 Description courte du sujet

L'entreprise *HappyDays* est spécialisée dans la vente de prêt-à-porter. Elle possède quelques centaines de magasins qu'il faut approvisionner tous les jours. Pour ce faire, elle dispose d'un unique entrepôt logistique. Dans cet entrepôt sont stockés tous les vêtements envoyés par les fournisseurs. Tous les jours, il faut préparer quelques colis pour chaque magasin afin de satisfaire leur commande. Il s'agit des opérations de *picking* dans l'entrepôt.

Le *picking* est souvent considéré comme le processus le plus important dans un entrepôt car il compte pour la majorité des coûts opérationnels de l'entrepôt. Les *pickers* (ou préparateurs) se déplacent dans l'entrepôt, en poussant un chariot sur lequel ils collectent des articles afin de préparer les commandes des clients (les magasins). Sur le chariot il y a plusieurs colis, qui sont vides au départ et qui sont remplis au fur et à mesure. Dans un colis, on ne peut mettre que des articles destinés à un même client. Afin de gagner du temps, il est possible (et même souhaitable) sur un même chariot de regrouper des colis pour différents clients. Par ailleurs, comme la commande d'un client requiert souvent plus d'un colis, les articles de la commande sont affectés à différents colis. Tous les colis d'un même client ne sont pas forcément sur un même chariot.

L'entreprise *HappyDays* a développé son propre système d'information (appelé *Warehouse Management System* dans le contexte de la gestion d'un entrepôt). Chaque jour, à partir d'un ensemble de commandes pour les magasins, il faut prendre les décisions suivantes :

- répartir les articles de chaque commande dans un nombre adapté de colis ;
- regrouper des colis sur des chariots ;
- pour chaque chariot, déterminer le parcours des préparateurs pour collecter les articles.

Pour que les opérations de *picking* soient efficaces, ces décisions doivent être prises de telle sorte que la distance totale à parcourir par les préparateurs soit minimale. Notez que la répartition des articles dans les colis et le regroupement des colis ont une influence sur la distance à parcourir.

2 Description de la problématique

Dans cette section, nous présentons de manière générale la problématique.

2.1 Les produits

HappyDays stocke dans son entrepôt un ensemble $\mathcal{P}$ de produits. Chaque produit $p \in \mathcal{P}$ correspond à une référence vendue en magasin, et est défini par un poids unitaire w_p et un volume unitaire v_p.

2.2 L'entrepôt

L'entrepôt de *HappyDays* est constitué de quatre zones (voir Figure 1), qui correspondent aux quatre processus réalisés dans l'entrepôt :

- une zone de réception dans laquelle les produits envoyés par les fournisseurs sont déchargés (et vérifiés) ;
- une zone de stockage dans laquelle les palettes de produits sont stockées sur des étagères ;
- une zone de préparation des commandes (*picking*) dans laquelle les cartons de produits sont stockés sur des étagères, et des opérateurs préparent les colis ;
- et une zone d'expédition depuis laquelle les camions sont chargés pour aller livrer les clients.

Quand un camion d'un fournisseur arrive, il est déchargé, puis toutes les palettes sont stockées dans les étagères de la zone de stockage.

Dans la zone de stockage, on dispose d'une ou plusieurs palettes pour chaque produit $p \in \mathcal{P}$. Chaque palette est elle-même constituée de plusieurs cartons qui contiennent des articles d'un même produit.

Dans la zone de *picking*, les colis en entrée sont posés sur des étagères, qui sont disposées sous forme d'allées réservées aux opérations de *picking*. Entre deux allées, un espace permet d'approvisionner les étagères depuis la zone de stockage (voir Figure 1). On dispose donc de plusieurs allées de *picking*, et chaque allée est composée de 2 rangées d'étagères. Chaque rangée d'étagère contient plusieurs positions pour mettre des cartons. À chaque position, on peut mettre un ou plusieurs cartons contenant des articles d'un même produit $p \in \mathcal{P}$.

L'entrepôt est peu mécanisé. La plupart des mouvements sont faits avec des transpalettes (dans la zone de stockage) ou des chariots (dans la zone de *picking*). Il y a juste un convoyeur qui permet de transporter les colis depuis la zone de *picking* vers la zone d'expédition.

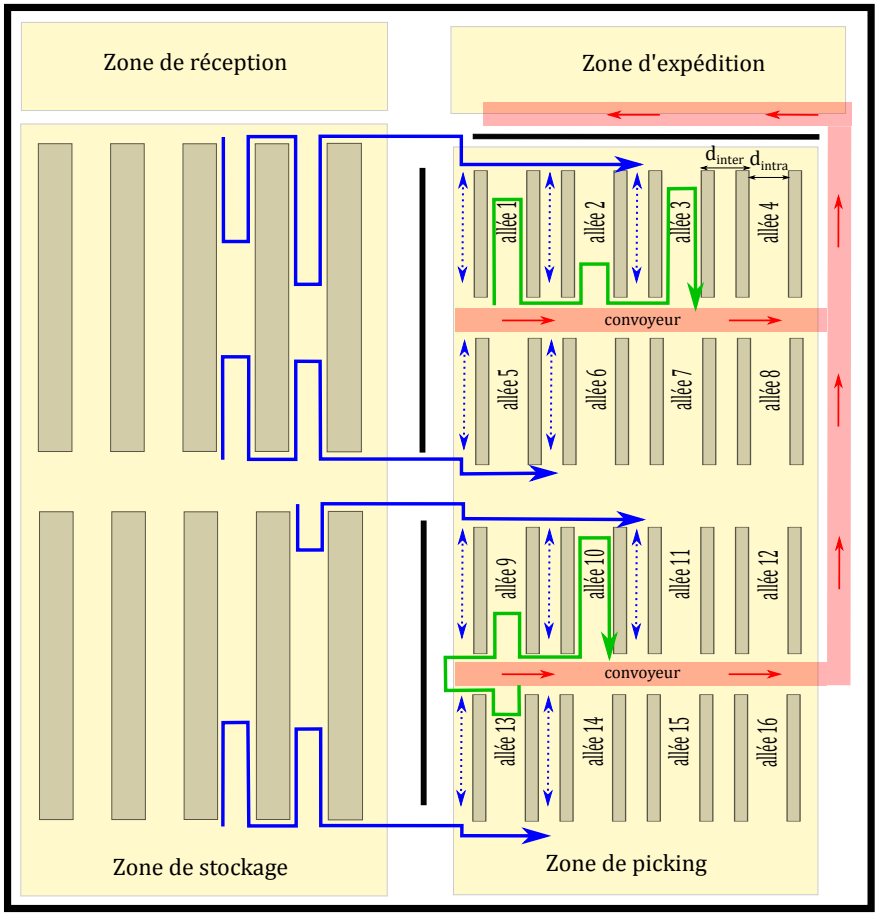

Figure 1: Schéma de l'entrepôt logistique de *HappyDays*.

2.3 Les commandes des clients

HappyDays possède un ensemble de clients. Chaque nuit, un module du système d'information calcule, en fonction des ventes de la journée passée, les commandes à préparer le lendemain pour chaque client. Nous ne nous occupons pas ici du calcul des commandes. Dans la journée, le système d'information est sollicité plusieurs fois pour préparer, dans la zone de *picking*, les commandes d'un ensemble de clients. Donc on ne prépare pas les commandes de tous les clients en même temps, car ce ne serait pas possible de gérer cela. Ainsi, on travaille par *batch* de commandes. Nous nous intéressons ici à la manière de préparer un *batch* de commandes.

Ainsi, à un moment donné, nous disposons d'un ensemble de commandes $\mathcal{C}$ à préparer. Chaque commande $c \in \mathcal{C}$, consiste en un ensemble de lignes de commande, et un nombre maximum de colis M_c à utiliser pour préparer la commande. Étant donné une commande c, une ligne de commande consiste en un produit $p \in \mathcal{P}$, avec une quantité $q_p^c > 0$ associée : il faut prélever q_p^c articles du produit p pour préparer la commande c. Bien évidemment, une commande ne peut pas contenir plusieurs lignes de commandes pour un même produit.

2.4 La constitution des colis

Pour chaque commande $c \in \mathcal{C}$, il faut constituer au maximum M_c colis, de telle sorte que tous les produits de la commande soient contenus dans les colis en une quantité q_p^c pour chaque produit $p \in \mathcal{P}$. Notez que pour un produit p, la quantité q_p^c n'est pas obligatoirement affectée au même colis. Par ailleurs, pour chaque colis, il faut respecter un poids maximal W et un volume maximal V (le poids total des articles dans le colis ne doit pas dépasser W et le volume total des articles dans le colis ne doit pas dépasser V).

2.5 Le processus de *picking*

Pour préparer un ensemble $\mathcal{C}$ de commandes, nous disposons d'un ensemble de préparateurs. Chaque préparateur pousse un chariot qui contient K colis, qui sont vides au départ. Puis le préparateur effectue une tournée dans l'entrepôt, au cours de laquelle il s'arrête à certaines positions pour prélever des articles et les mettre dans les colis. Lorsque les colis sont tous remplis, le préparateur les amène sur le convoyeur, la tournée est terminée. Le préparateur pourra alors repartir avec des colis vides qu'ils remplira à nouveau lors d'une nouvelle tournée. Nous ne nous occupons pas de l'enchainement des tournées.

2.6 La zone de *picking*

Nous nous intéressons ici plus particulièrement à la zone de *picking*. Elle présente une particularité intéressante : les préparateurs doivent faire demi-tour dans chaque allée dans laquelle ils rentrent.

En fait, si vous regardez la Figure 1, vous pouvez voir en bleu les flux réalisés pour approvisionner la zone de *picking* depuis la zone de stockage. Ces flux sont réalisés en majorité avec des engins de type transpalette.

Comme les préparateurs utilisent des chariots qu'ils poussent manuellement, et qu'ils ont peu de visibilité sur ce qui se passe devant, pour des raisons évidentes de sécurité, *HappyDays* a mis en place une politique de circulation qui empêche de croiser les flux des transpalettes avec les flux des chariots.

Ainsi, comme on peut le voir avec les lignes vertes dans la Figure 1, les tournées de *picking* commencent au début d'une allée, puis parcourent quelques allées et se terminent au niveau du convoyeur. Dans chaque allée, on fait demi-tour, on ne peut en ressortir par le fond.

Par ailleurs, il existe un sens de circulation pour éviter au maximum que les chariots se croisent. Ainsi, quand on rentre dans une allée, on parcourt d'abord le côté gauche, puis on fait demi-tour, puis on parcours le côté droit. Et quand on sort d'une allée, il faut repartir vers la gauche. C'est ce qui correspond aux deux tournées de *picking* présentées dans la Figure 1.

Notez que grâce à ce sens de circulation, si on connait l'ensemble des positions à visiter dans une tournée, alors déterminer la tournée optimale (dont la distance est la plus petite) est facile !

3 Problème de *order batching* et *picker routing*

Nous explicitons ici le problème auquel nous nous intéressons : étant donné un ensemble de commandes à préparer, il faut déterminer la constitution des colis, le regroupement des colis sur les chariots, et les tournées de chaque chariot.

3.1 Les données

Nous récapitulons ici toutes les données nécessaires au problème. À un moment donné, si l'on veut préparer un ensemble de commandes dans la zone de *picking*, on suppose que tous les articles nécessaires sont présents dans la zone de *picking*. Cette zone peut être représentée comme présentée dans la Figure 2.

Nous noterons :

- s le dépôt de départ, i.e. l'endroit d'où un préparateur doit partir pour effectuer une tournée de *picking* ;
- t le dépôt d'arrivée, i.e. l'endroit où le préparateur doit arriver à la fin de sa tournée ;
- $\mathcal{V}_L$ est l'ensemble des positions qui contiennent un produit nécessaire dans une commande (ce sont les points à côté d'un carré gris dans la Figure 2) ;
- $\mathcal{V}_I$ est l'ensemble des intersections, i.e. les points qui ne sont pas associés à un produit à ramasser, mais qui permettent au préparateur de se déplacer (de changer de direction) dans l'entrepôt ;

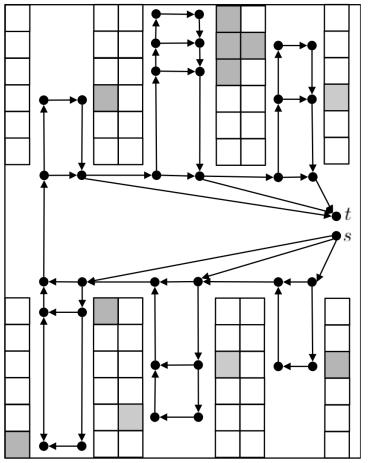

Figure 2: Schéma de la zone de *picking* : les positions en gris contiennent un produit à ramasser.

- $\mathcal{V} = \mathcal{V}_L \cup \mathcal{V}_I \cup \{s\} \cup \{t\}$ est l'ensemble des localisations dans la zone de *picking*.

Par ailleurs, nous notons d_{ij} la distance entre les localisations $i \in \mathcal{V}$ et $j \in \mathcal{V}$, s'il est possible d'aller de i vers j directement sans passer par une autre localisation. Ceci correspond aux arcs en noir sur la Figure 2. Nous notons D_{ij} la plus courte distance pour aller de $i \in \mathcal{V}$ vers $j \in \mathcal{V}$.

Notez qu'à cause du sens de parcours imposé dans la zone de *picking*, il peut être impossible d'aller d'une localisation i vers une localisation j, et dans ce cas D_{ij} est supposée infinie. Par ailleurs, à cause du sens de parcours, l'ensemble $\mathcal{V}_L$ peut être ordonné de telle sorte que $i \prec j$ signifie que la localisation i doit être visitée avant la localisation j dans le sens de parcours.

L'ensemble des produits est noté $\mathcal{P}$, et chaque produit $p \in \mathcal{P}$ est défini par un poids unitaire w_p, un volume unitaire v_p et une localisation dans la zone de *picking* $l_p \in \mathcal{V}_L$.

L'ensemble des commandes à préparer est noté $\mathcal{C}$. Chaque commande $c \in \mathcal{C}$ est définie par un ensemble de lignes de commandes, et un nombre maximum de colis à utiliser pour préparer la commande, noté M_c. Une ligne de commande est une paire contenant un produit $p \in \mathcal{P}$, et un entier q_p^c correspondant au nombre d'articles du produit p à ramasser pour la commande c.

Les préparateurs poussent des chariots qui contiennent K colis, et chaque colis à un poids maximal W et un volume maximal V.

3.2 Les décisions

Le problème auquel nous nous intéressons consiste en plusieurs décisions liées entre elles :

- pour chaque commande $c \in \mathcal{C}$ il faut décider de l'affectation des articles à ramasser dans M_c colis ;
- les colis doivent être regroupés par groupe d'au plus K, chaque groupe sera affecté à un chariot ;
- les tournées des préparateurs pour récupérer tous les articles des colis affectés à un même chariot.

3.3 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 un ensemble de colis $\mathcal{B}_r$, ainsi que la distance d_r à parcourir pour réaliser la tournée. Chaque colis $b \in \mathcal{B}_r$ est associé à une commande $c_b \in \mathcal{C}$. On associe également à chaque colis b l'ensemble des produits $\mathcal{P}_b$ ramassés dans le colis (donc $\mathcal{P}_b \subseteq \mathcal{P}$). Et pour chaque produit $p \in \mathcal{P}_b$, on associe la quantité ramassée q_p^b.

3.4 Contraintes du problème

Pour faciliter les explications, notons $\mathcal{B}_c$ l'ensemble des colis utilisés pour préparer la commande c.

Pour qu'une solution soit réalisable, il faut que :

- chaque tournée $r \in \mathcal{R}$ part de s et arrive à t ;
- chaque tournée r passe par toutes les localisations associées aux produits ramassés dans l'ensemble des colis $\mathcal{B}_r$ de la tournée ;
- chaque tournée r n'a pas plus que K colis, i.e. $|\mathcal{B}_r| \leq K$;
- chaque colis b respecte les capacités en poids et volume : $\sum_{p \in \mathcal{P}_b} q_p^b w_p \leq W$ et $\sum_{p \in \mathcal{P}_b} q_p^b v_p \leq V$;
- une commande c ne peut pas être affectée à plus de M_c colis, i.e. $|\mathcal{B}_c| \leq M_c$;
- pour chaque commande c, chacune de ses lignes de commande doit être ramassée intégralement : pour la ligne de commande associée au produit p, il faut ramasser q_p^c articles de ce produit (peut-être dans différents colis) : $\sum_{b \in \mathcal{B}_c} q_p^b = q_p^c$.

3.5 Fonction objectif

HappyDays souhaite que ses préparateurs parcourent le moins de distance possible.

Ainsi, l'objectif ici est de minimiser la distance totale parcourue, qui étant donnée une solution, s'exprime comme $d = \sum_{r \in \mathcal{R}} d_r$.

Notez que comme l'ensemble $\mathcal{V}_L$ des positions qui contiennent un produit peut être ordonné, alors le calcul de d_r est facile. En effet, notons $\mathcal{V}_r$ l'ensemble des n_r positions contenant un produit à ramasser dans la tournée r. Cet ensemble peut également être ordonné : $\mathcal{V}_r = \{l_0^r; l_1^r; ...; l_{n_r}^r\}$, de telle sorte que $l_i^r \prec l_j^r$ signifie que la localisation l_i^r doit être visitée avant la localisation l_j^r dans le sens de parcours imposé dans la zone de *picking*. Alors $d_r = D_{s,l_0^r} + \sum_{i=0}^{i<n_r} D_{l_i^r, l_{i+1}^r} + D_{l_{n_r}^r, t}$.

3.6 Hypothèses sur les données

On suppose que les données permettent toujours d'obtenir une solution réalisable. Ainsi, pour chaque commande $c \in \mathcal{C}$, il est possible de trouver une affectation des articles dans au plus M_c colis en respectant le poids et le volume maximal de chaque colis.

4 Solution triviale

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

- pour chaque commande c, on affecte les articles à ramasser dans au plus M_c colis en respectant les capacités sur le poids et le volume ;
- on regroupe ensuite les colis par groupe de K ce qui donne une solution.

Attention, cette solution n'est réalisable que si on a bien affecté les articles de chaque commande c dans au plus M_c colis en respectant les capacités. Si vous rencontrez des problèmes de solution réalisable, vous pouvez commencer par affecter les produits les plus gros (en poids ou volume) dans les colis, puis en dernier les plus petits.

5 Fichiers d'instance et fichiers de solution

5.1 Fichiers d'instance

Un ensemble de 10 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.

Les instances sont basées sur des données réelles qui ont été fournies par le groupe français *HappyChic* pour leur entrepôt de la marque *Jules*.

Chaque instance est sous forme d'un fichier texte, avec comme séparateur des espaces, et les lignes de commentaire commencent par '//'. Voici les données que vous allez trouver, dans l'ordre, dans un fichier d'instance.

- `NbLocations` : donne le nombre de localisations (sans compter les dépôts). Donc c'est égal à la cardinalité de $\mathcal{V}_L \cup \mathcal{V}_I$.
- `NbProducts` : donne le nombre de produits, donc la cardinalité de $\mathcal{P}$.
- `K: NbBoxesTrolley` : donne le nombre maximum de colis sur un chariot, donc c'est K.
- `NbDimensionsCapacity` : donne le nombre de dimensions pour les capacités des colis. Dans notre cas, ce sera toujours 2 : poids et volume.
- `B: CapaBox` : donne la capacité des colis dans les deux dimensions : poids (W), puis volume (V).
- `A box can accept mixed orders(0: no, 1: yes)` : dans notre cas ce sera toujours '0' car un colis ne peut contenir que des articles pour une même commande.
- `Products` : donne la liste des produits. Pour chaque produit p, on a sur une ligne (séparés par des espaces) :
  - l'identifiant du produit (cet identifiant est un nombre entre 1 et $|\mathcal{P}|$) ;
  - un identifiant qui réfère la localisation l_p du produit (cet identifiant est un nombre entre 1 et $|\mathcal{V}_L|$) ;
  - le poids unitaire du produit w_p ;
  - le volume unitaire du produit v_p.
- `Orders` : donne la liste des commandes. On trouve dans un premier temps `NbOrders` qui donne le nombre de commandes à préparer, donc la cardinalité de $\mathcal{C}$. Puis, après le commentaire, on a une commande par ligne ; et pour chaque commande c, on a sur une ligne (séparés par des espaces) :
  - l'identifiant de la commande (cet identifiant est un nombre entre 1 et $|\mathcal{C}|$) ;
  - M_c : le nombre maximal de colis pour préparer la commande ;
  - le nombre de lignes de commandes dans la commande ;
  - ensuite chaque ligne de commande correspond à une paire (p, q_p^c) où :
    - p correspond à l'identifiant du produit ;
    - q_p^c correspond à la quantité commandée pour ce produit.
- `Graph` présente le graphe sous-jacent à la zone de *picking*.
- `NbVerticesIntersections` : donne le nombre d'intersections, donc la cardinalité de $\mathcal{V}_I$.
- `DepartingDepot` : donne l'identifiant du dépôt de départ s (ce sera toujours 0).
- `ArrivalDepot` : donne l'identifiant du dépôt d'arrivée t (ce sera toujours $|\mathcal{V}| - 1$).
- `Arcs` : donne la liste des arcs (voir par exemple dans la Figure 2). Pour chaque arc, on a sur une ligne (séparés par des espaces) :
  - l'identifiant de la localisation de départ ;
  - l'identifiant de la localisation d'arrivée ;
  - la distance à parcourir pour aller du départ vers l'arrivée.
- `LocStart LocEnd ShortestPath` : donne les valeurs des plus courtes distances D_{ij}. Plus précisément, on a sur une ligne (séparés par des espaces) :
  - l'identifiant de la localisation de départ ;
  - l'identifiant de la localisation d'arrivée ;
  - la plus courte distance pour aller du départ vers l'arrivée.

  Notez qu'à cause du sens de parcours, certains couples de localisation ne sont pas présents dans la liste des plus courtes distances.
- `Location coordinates LocationName` : donne des informations sur les localisations. Sur chaque ligne on a (séparés par des espaces) :
  - l'identifiant de la localisation ;
  - son abscisse ;
  - son ordonnée ;
  - un champ de texte qui sert à nommer la localisation.

Il est important de noter que dans le fichier d'instances, les identifiants des localisations seront toujours de telle sorte que :

- le dépôt de départ s a l'identifiant 0 ;
- les localisations contenant des produits sont numérotées de 1 à $|\mathcal{V}_L|$;
- les intersections sont numérotées de $|\mathcal{V}_L| + 1$ à $|\mathcal{V}_L| + |\mathcal{V}_I|$;
- le dépôt d'arrivée t a l'identifiant $|\mathcal{V}| - 1$;
- les localisations contenant des produits ($\mathcal{V}_L$) sont numérotées dans un ordre topologique, i.e. que si $i \prec j$, alors la localisation i doit être visitée avant j à cause du sens de parcours (et donc il n'est pas possible d'aller de j vers i).

5.2 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 6).

Chaque fichier solution est sous forme d'un fichier texte, avec comme séparateur des espaces, et les lignes de commentaire commencent par '//'. Voici les données que vous devez présenter, dans l'ordre, dans un fichier de solution.

- Le nombre de tournées dans la solution.
- Puis, pour chaque tournée, on trouve les éléments suivants :
  - un identifiant de la tournée ;
  - le nombre de colis dans la tournée ;
  - puis, pour chaque colis de la tournée, on trouve les éléments suivants :
    - un identifiant du colis ;
    - l'identifiant de la commande pour laquelle ce colis est préparé ;
    - le nombre de produits différents dans le colis (ce n'est pas le nombre d'articles, car pour un même produit on peut prendre plusieurs articles) ;
    - puis, pour chaque produit dans le colis, on trouve les éléments suivants :
      - l'identifiant du produit ;
      - la quantité ramassée pour ce produit.

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.

6 Validation de la solution

Un *checker* est à votre disposition dans le dossier `checker` qui se trouve sur Moodle. Il est nommé `CheckerBatchingPicking.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 CheckerBatchingPicking.jar nomInstance`.

7 Directives informatiques

7.1 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.

7.1.1 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.

7.1.2 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.

7.1.3 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 algorithmes 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).

7.1.4 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.

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 sanctionné comme telle, 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 par groupes de 4, que vous pouvez constituer comme vous le souhaitez. Notez qu'il y aura un groupe avec seulement 3 personnes. Lors de la séance du 9 avril 2018, les personnes non encore affectées à un groupe seront affectées d'office par les enseignants.

9 Rendu

Vous déposez sur Moodle, dans la zone de dépôt **Dépôt du Projet 2017/2018** 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 **14 juin 2018 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 de 30 minutes aura lieu le **15 juin 2018 après-midi**. Cette soutenance comportera une présentation de votre projet d'une durée de 20 minutes maximum 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. Les 10 minutes restantes seront consacrées à des questions.

10 Notation

Une note globale sera donnée sur la base du travail fournit 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 7.1.1–7.1.3) 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.

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.

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).

10.2 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 **14 juin 2018 à 23h59**.

10.3 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.