Projet Méthodologie de développement objet

Durée totale du projet : 5 séances de 4 heures

Encadrement : S. Duffner et C. Solnon

1 Description du système

Le système vous a été décrit dans sa globalité pour le projet d'IHM. Dans cette deuxième partie, nous allons nous concentrer sur le sous-système *Préparation et supervision des livraisons*, dont nous rappelons la description en 1.1. Vous réaliserez la capture et l'analyse des besoins pour la totalité de ce sous-système. La conception détaillée ne sera faite que pour la partie *Préparation des feuilles de route* de ce sous-système. Enfin, le prototype que vous réaliserez ne concernera que les fonctionalités décrites en 1.2.

1.1 Sous-système *Préparation et supervision des livraisons* (rappel)

Préparation des feuilles de route. Tous les jours, un superviseur des tournées de la société de transport demande au système de calculer les feuilles de route pour les livraisons du lendemain. Pour cela, les livraisons sont réparties en zones géographiques et toutes les livraisons d'une même zone sont effectuées par un même livreur. Nous rappelons que chaque livraison a une plage horaire donnant les heures d'arrivée au plus tôt et au plus tard du livreur. Par ailleurs, le système prévoit 15 minutes d'arrêt à chaque livraison pour décharger les colis et les remettre au client. Pour qu'il soit possible d'effectuer toutes les livraisons dans les plages horaires prévues, le sous-système *Demande de livraison* limite le nombre de livraisons à faire dans une même plage horaire pour une même zone géographique. Si malgré cela il n'est pas possible d'effectuer toutes les livraisons prévues pour une plage horaire, alors la secrétaire contacte les clients concernés pour leur proposer de nouvelles plages horaires. Elle informe (via le système) le superviseur qui modifie la feuille de route en conséquence.

Pour chaque zone géographique, le système visualise sur un plan de l'agglomération la feuille de route du livreur, c'est-à-dire les différentes livraisons à effectuer et l'itinéraire à suivre pour cela (en partant du dépôt et en revenant à celui-ci en respectant les routes en sens unique) avec les horaires de passage prévus aux différents points de livraison. Le superviseur des tournées peut alors modifier interactivement la feuille de route (supprimer une livraison, ajouter une livraison, modifier un itinéraire, ...), et demander au système de mettre-à-jour les horaires de passage en conséquence. Le système signale au superviseur les livraisons pour lesquelles l'horaire de passage ne respecte plus la plage horaire initialement demandée par le client. À tout moment, le superviseur peut demander l'annulation de modifications apportées à la feuille de route.

Lorsqu'une feuille de route a été validée par le superviseur, le superviseur en édite une version papier destinée au livreur. Cette version papier donne la liste des rues à prendre, et les horaires de passage prévus à chaque intersection de rue.

Supervision des livraisons. Un agent de la société de livraison supervise en temps réel le déroulement des livraisons. Pour cela, le système affiche sur un plan les différentes feuilles de route prévues pour la journée en cours. Un code couleur permet de distinguer les livraisons déjà effectuées, les livraisons restant à faire et n'ayant pas pris de retard, et les livraisons restant à faire et ayant pris du retard. Les données concernant les livraisons sont actualisées en fonction des informations transmises par les livreurs lorsqu'ils arrivent aux points de livraison. Ces informations concernent l'heure de la livraison, si la livraison a pu être effectuée, ou la cause de la non-livraison sinon, ainsi que l'heure de départ pour le prochain point de livraison. On notera de plus que chaque camion peut être suivi en temps réel par son positionnement GPS.

À tout moment, le superviseur peut modifier interactivement le parcours d'un livreur, pour les livraisons restant à faire (supprimer une livraison, intervertir l'ordre de deux livraisons, modifier un itinéraire, ...), et demander au système de mettre-à-jour les horaires de passage en conséquence. Lorsque le superviseur a saisi toutes les modifications, le système met à jour la version électronique des feuilles de route des livreurs en conséquence.

1.2 Prototype

Le prototype que vous réaliserez concernera la préparation des feuilles de route. Il devra notamment permettre de :
- charger le plan d'une zone à partir d'un fichier XML ;
- saisir interactivement une demande de livraison (un ensemble de points à livrer avec, pour chaque point, sa plage horaire) ;
- calculer une tournée pour la dernière demande de livraisons saisie ;
- visualiser sur le plan une tournée calculée, avec mise en évidence des livraisons ne pouvant pas être effectuées dans leur plage horaire (le cas échéant), et possibilité d'obtenir des informations sur une livraison (horaire de livraison prévu, plage horaire, ...) en cliquant sur le point correspondant ;
- modifier interactivement une tournée (supprimer ou insérer un point de livraison), avec possibilité d'annuler les dernières modifications effectuées ;
- générer un fichier texte contenant les instructions pour guider le livreur lors de sa tournée (noms des rues à suivre, horaires d'arrivée aux points de livraison, ...).

Vous trouverez sur `servif-home` trois exemples de plans de zones (comportant respectivement 25, 100 et 400 intersections de rues) au format XML. Pour chaque intersection i, ces plans donnent les coordonnées x et y de i et la liste des tronçons partant de i avec, pour chaque tronçon : le nom de la rue, la destination (l'intersection suivante), la vitesse moyenne sur ce tronçon (en décimètres par seconde) et sa longueur (en décimètres).

2 Déroulement du projet

Le projet est réalisé par hexanôme. Vous désignerez un chef de projet chargé de la répartition du travail et de la coordination. Un planning prévisionnel indicatif vous est donné ci-dessous :

- Séance 1 : Début de la capture et de l'analyse des besoins
- Séance 2 :
  - Fin de la capture et de l'analyse des besoins
  - Début de la conception détaillée
- Séance 3 :
  - Fin de la conception détaillée
  - Traduction des modèles et début de l'implémentation et des tests unitaires
  - Première prise en main de l'environnement de développement
- Séance 4 : Implémentation et tests unitaires
- Séance 5 :
  - Fin de l'implémentation et des tests unitaires
  - Tests d'intégration
  - Génération de la documentation en ligne
  - Re-génération du diagramme de classes à partir du code
- Semaine suivante : Démonstration du prototype (rendez-vous à fixer) et remise du compte-rendu.

3 Livrables

Capture et analyse des besoins : Les livrables de cette étape concernent la totalité du sous-système *Préparation et supervision des livraisons* décrit en 1.1.
- Planning prévisionnel du projet
- Modèle du domaine
- Diagramme de cas d'utilisation
- Description abrégée des cas d'utilisation

Conception : Les livrables de cette étape ne concernent plus que la partie *Préparation des feuilles de route* du sous-système *Préparation et supervision des livraisons.*
- Description détaillée des cas d'utilisation
- Diagrammes de packages et de classes

– Diagramme de séquences du cas d'utilisation permettant au superviseur de saisir les demandes de livraison et calculer la feuille de route

Implémentation et Test : Les livrables de cette étape ne concernent que le périmètre du prototype décrit en 1.2.
– Code du prototype et des tests unitaires
– Documentation JavaDoc du code
– Diagramme de classes retro-généré à partir du code

Bilan :
– Planning effectif du projet
– Bilan humain et technique

4 Environnement de développement

Vous utiliserez un environnement de développement intégré (IDE) permettant de générer du code à partir de modèles et, inversement, de générer des modèles à partir du code. Il est également conseillé d'utiliser des outils pour automatiser les tests unitaires, pour favoriser le travail collaboratif et la gestion des versions, et pour générer la documentation en ligne du code. Nous vous proposons d'utiliser l'IDE Eclipse et de développer en Java. Si vous souhaitez utiliser un autre IDE ou un autre langage, vous devrez demander notre accord au préalable.

Vous trouverez une documentation complète d'Eclipse sur `http://help.eclipse.org/juno/index.jsp` (voir notamment le Java Development User Guide). Les plug-in d'Eclipse que vous pouvez utiliser sont ObjectAid (`http://www.objectaid.com/`), pour re-générer un diagramme de classes à partir du code Java, et JUnit (`http://www.junit.org/`), pour automatiser la réalisation des tests unitaires. Vous pouvez également utiliser ArgoUML (`http://argouml.tigris.org/`) pour saisir des diagrammes de séquences et de classes, et générer un squelette de code Java à partir d'un diagramme de classes. Cependant, le plug-in ArgoUML installé actuellement sur Eclipse ne marche pas. Si vous souhaitez utiliser ArgoUML, vous devrez l'installer auparavant en tant qu'application indépendante (et non en tant que plug-in d'Eclipse).

Vous génèrerez une documentation en ligne de votre code en utilisant JavaDoc, et vous suivrez le code de style préconisé par Oracle (`http://www.oracle.com/technetwork/java/codeconv-138413.html`).

Pour calculer la tournée, vous pouvez utiliser la librairie Choco et vous inspirer de la classe TSP disponibles sur `servif-home`. Si vous souhaitez utiliser une autre librairie, ou bien programmer votre propre algorithme, vous devrez demander notre accord au préalable.

5 Modèle mathématique

Le modèle mathématique du problème d'optimisation de tournées de véhicules avec fenêtres de temps vous a été présenté en cours d'aide à la décision, par Maryvonne Miquel, et sa résolution à l'aide de la programmation par contraintes vous a été présentée en cours de méthodologies de développement objet. Nous rappelons ici les grandes lignes de ce modèle mathématique.

5.1 Formalisation du problème initial

Les données en entrée du problème à résoudre sont :
– un graphe $G = (V, A)$ décrivant le plan de la ville, où V est un ensemble de sommets et $A \subseteq V \times V$ un ensemble d'arcs ;
– une fonction $d : A \to \mathbb{R}^+$ associant une durée d_{ij} à chaque arc (i, j) ;
– un entrepôt $e \in V$;
– un ensemble $L \subseteq V$ de points de livraison ;
– un ensemble F de fenêtres de temps numérotées de 1 à $|F|$ et toutes disjointes deux à deux ;
– une fonction $f : L \to F$ associant une fenêtre de temps f_i à chaque point de livraison $i \in L$.

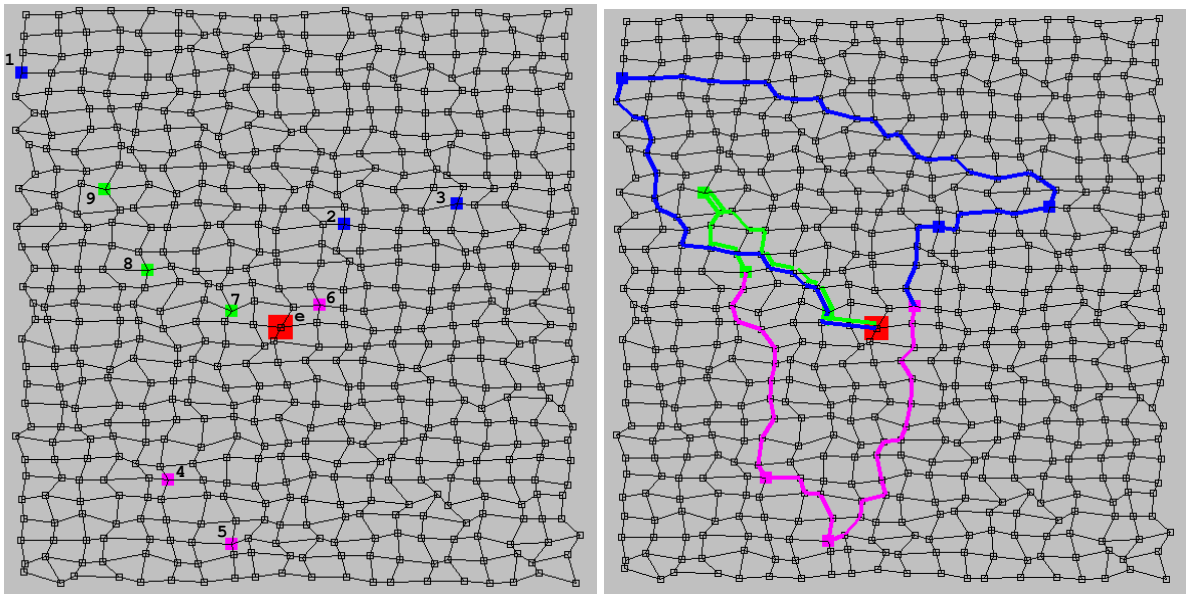


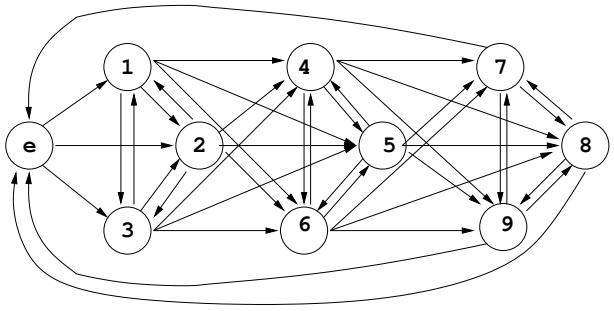
FIGURE 1 – Exemple de problème de tournée (à gauche), et de solution (à droite). Dans cet exemple, la durée d_{ij} associée à chaque arc (i, j) est proportionnelle à la distance euclidienne entre i et j.

A partir de ces données en entrée, il s'agit de trouver un circuit c de G partant de l'entrepôt e et revenant sur e tel que :
- chaque sommet de L apparaît au moins une fois dans c ;
- pour chaque paire de sommets $(i, j) \in L \times L$ telle que $f_i < f_j$, le circuit c passe par i avant de passer par j ;
- la somme des durées des arcs empruntés par c soit minimale.

5.2 Prétraitement

Pour simplifier le problème, on peut commencer par calculer un nouveau graphe $G' = (V', A')$ tel que
- $V' = L \cup \{e\}$
- $A' = \{(e, i) | i \in L, f_i = 1\} \cup \{(i, j) \in L \times L | f_i = f_j\} \cup \{(i, j) \in L \times L | f_j = f_i + 1\} \cup \{(i, e) | i \in L, f_i = F\}$

Considérons par exemple le problème initial de la figure 1, pour lequel il y a 3 fenêtres de livraison telles que $f(1) = f(2) = f(3) = 1$, $f(4) = f(5) = f(6) = 2$ et $f(7) = f(8) = f(9) = 3$. Le graphe G' est :



On calcule ensuite, pour chaque arc $(i, j) \in A'$, le plus court chemin π_{ij} de i vers j dans le graphe initial $G = (V, A)$. On définit enfin la fonction $d' : A' \to \mathbb{R}^+$ associant à chaque arc (i, j) de G' la longueur du plus court chemin π_{ij} dans G. Pour résoudre le problème initial, on peut alors chercher un circuit hamiltonien de G' partant de e et revenant sur e tel que la somme des durées des arcs soit minimale. On reconstruit ensuite la tournée dans G à partir de ce circuit en remplaçant chaque arc du circuit dans G' par le plus court chemin correspondant dans G.

Dans l'exemple précédent, le circuit hamiltonien $< e, 1, 3, 2, 6, 5, 4, 8, 9, 7, e >$ du graphe G' correspond à la solution dessinée dans la partie droite de la figure 1.

5.3 Programmation par contraintes pour le voyageur de commerce

Le problème de la recherche d'un circuit hamiltonien (passant par chaque sommet une et une seule fois) de coût minimal est bien connu sous le nom de problème du voyageur de commerce. Dans notre cas, il s'agit de la version asymétrique du problème car la fonction d' associant une durée à chaque arc n'est pas symétrique.

Ce problème peut être résolu par différentes approches telles que, par exemple, la programmation linéaire en nombres entiers, les méta-heuristiques (optimisation par colonies de fourmis, recherche locale itérée, algorithmes génétiques, ...), ou la programmation par contraintes. Nous vous proposons dans ce projet d'utiliser la bibliothèque de programmation par contraintes Choco. Pour cela, il faut modéliser le problème à résoudre en termes de variables de décision, contraintes à satisfaire, et fonction objectif à optimiser.

Posons $n = |V'|$ et supposons que les sommets de V' sont numérotés de 0 à $n-1$, et que l'entrepôt est le sommet 0. Pour chaque sommet $i \in V'$, nous définissons 3 variables :
- La variable pos_i donne le $i^{\text{ème}}$ sommet visité et prend sa valeur dans l'ensemble $\{0, \ldots, n-1\}$.
- La variable $next_i$ donne le sommet visité après le sommet i et prend sa valeur dans l'ensemble des sommets successeurs de i dans G'.
- La variable $duree_i$ donne la durée de l'arc $(i, next_i)$.

Les contraintes à satisfaire sont :
- On part de l'entrepot, *i.e.*,
$$pos_0 = 0$$
- Le sommet suivant le dernier sommet est l'entrepot, i.e.,
$$next_{pos_{n-1}} = 0$$
- Le sommet suivant le sommet à la position $i-1$ est à la position i, *i.e.*,
$$\forall i \in V \setminus \{0\}, pos_i = next_{pos_{i-1}}$$
- Chaque sommet est à une position différente dans pos, *i.e.*,
$$allDifferent(\{pos_i | i \in V\})$$
- Chaque sommet est le suivant d'un et un seul sommet, *i.e.*,
$$allDifferent(\{next_i | i \in V\})$$
- La durée associée à i est égale à la durée de l'arc entre i et $next_i$, *i.e.*,
$$\forall i \in V, duree_i = d_{i,next_i}$$

La fonction objectif à minimiser est la somme des durées, *i.e.*
$$min \sum_{i \in V'} duree_i$$

Sur l'exemple de la figure 1, la solution correspond à

pos	0	1	3	2	6	5	4	8	9	7
next	1	3	6	2	8	4	5	0	9	7
	0	1	2	3	4	5	6	7	8	9

Le programme Choco correspondant à ce modèle peut être trouvé sur `servif-home`. Vous pouvez bien évidemment utiliser ce programme pour votre projet.