

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2016-2017

Simulation d'un problème de tournées avec des commandes rapides



POLYTECH[®]
TOURS

Tuteurs académiques

Yannick KERGOSIEN

Jorge MENDOZA

Mustapha HAOUASSI

Étudiant

Lorris PIGEON (DI5)

2 avril 2017



Liste des intervenants

Nom	Email	Qualité
Lorris PIGEON	lorris.pigeon@etu.univ-tours.fr	Étudiant DI5
Yannick KERGOSIEN	yannick.kergosien@univ-tours.fr	Tuteur académique, Département Informatique
Jorge MENDOZA	jorge.mendoza@univ-tours.fr	Tuteur académique, Département Informatique
Mustapha HAOUASSI	mustapha.haouassi@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Loris Pigeon surnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Yannick Kergosien, Jorge Mendoza et Mustapha Haouassi surnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Lorris Pigeon, *Simulation d'un problème de tournées avec des commandes rapides*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2016-2017.

```
@mastersthesis{
  author={Pigeon, Lorris},
  title={Simulation d'un problème de tournées avec des commandes rapides},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2016-2017}
}
```



Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	iv
I Analyse	1
Modélisation du problème de tournées	2
1 Définition du problème	2
2 Les données du problème	2
3 Les variables du problème	3
4 Les contraintes du problème	3
5 Fonction objectif	3
II Etat de l’art	4
1 Les problèmes de routage	5
2 La simulation	7

III	Cahier de spécification système	10
	Introduction	11
3	Contexte de la réalisation	12
1	Contexte.....	12
2	But du projet et objectif.....	13
3	Base méthodologique	13
4	Description générale	14
1	Environnement du projet	14
2	Caractéristiques des utilisateurs	14
3	Fonctionnalités du système	14
3.1	Paramétrage de la simulation et calculs.....	14
3.2	Affichage de la solution.....	15
4	Contraintes de développement	15
5	Description des interfaces externes du logiciel	17
1	Matériel/logiciel	17
2	Homme/machine.....	17
2.1	Représentation de la solution.....	17
3	Logiciel/logiciel.....	17
6	Architecture générale du système	18
7	Fonctionnalités du système	19
1	Le moteur de la simulation	19
2	L'affichage graphique de la simulation	22
IV	Mise en oeuvre	23
8	Plan de développement	24
9	Implémentation	27
1	Librairies.....	27
2	Structure du code	27
3	Fichiers de données et de configurations.....	27
4	Fichiers de logs	28
5	Performance	28

Bilan et conclusion	29
6 Avancement du projet	29
7 Qualité	29
8 Bilan	30
Remerciements	31
Bibliographie	32
Annexes	33
A Diagramme de classe de la simulation	34
B Cahier de tests	38
1 Introduction.....	38
2 Tests Unitaires	38
3 Tests d'intégration	38
3.1 La classe EvaluationCommandeTest	39
3.2 La classe EvaluationTourneeTest.....	39
3.3 La classe SimulationTest.....	39
C Guide d'utilisation	40
1 Exploitation du sujet	40
1.1 Ouverture du projet sous Eclipse	40
1.2 Exécution du projet	40
1.3 Génération de l'exécutable .jar avec Eclipse	40
1.4 Affichage du GANTT d'une exécution.....	41
2 Développement du projet.....	42
2.1 Ajout d'une classe de décisions	42

Table des figures

3	Contexte de la réalisation	
1	Affichage du GANTT du planning	13
4	Description générale	
1	Diagramme de cas d'utilisation du paramétrage de l'instance de la simulation	15
2	Diagramme de cas d'utilisation de l'affichage dynamique de la solution	16
6	Architecture générale du système	
1	Architecture generale du systeme	18
7	Fonctionnalités du système	
1	Représentation du système.....	19
8	Plan de développement	
1	Détail du plan de développement.....	25
2	Diagramme de GANTT du projet	26
A	Diagramme de classe de la simulation	
1	Partie solution de la simulation	34
2	Partie instance de la simulation	35
3	Partie affichage de la simulation	35
4	Partie calcul de la simulation	36
5	Classe Simulation	37

C Guide d'utilisation

1	Exécution de l'application en ligne de commandes.....	41
2	Arborescence du répertoire de ressources du projet	42
3	Modification d'un des paramètres du fichier de configurations.....	42
4	Affichage d'une classe de décision personnalisée	43

Première partie

Analyse

Une large partie du travail de recherche opéré au premier semestre a porté sur l'analyse et la modélisation du problème proposé par la MOA.

Modélisation du problème de tournées

1 Définition du problème

Une entreprise de livraison reçoit des commandes de clients et doit faire son possible pour effectuer les livraisons dans les temps. Pour ce faire, elle a un entrepôt comportant différents produits, et des livreurs, qui peuvent effectuer plusieurs tournées pour répondre à la demande. Cette demande peut prendre deux formes :

1. Le client souhaite être livré rapidement, ainsi, dès réception de la commande, l'entreprise s'engage à la livrer dans l'heure, si elle n'atteint pas cet objectif, elle reçoit une pénalité. (un coût supplémentaire lui sera facturé).
2. Le client est plus flexible : il souhaite être livré sur une fenêtre de temps donnée, fenêtre qui ne peut commencer qu'à partir d'une heure suivant la demande. Par exemple celui-ci commande à 13h un produit et souhaite être livré de 14h30 à 17h30.

Lors d'une tournée, un livreur peut livrer plusieurs commandes successives, chaque tournée a un coût de déplacement pour l'entreprise (frais kilométriques) et un coût fixe. Le retard lors des livraisons de commandes « express » va engendrer un coût supplémentaire. L'objectif du problème est de diminuer les coûts des tournées et les coûts des commandes express non livrées à temps. Pour cela, il faut déterminer quelle commande sera livrée à quel moment par quel livreur, lors de quelle tournée. Il faudra également déterminer la date de départ de chaque tournée. Chaque commande a une date de réception, c'est à ce moment que l'entreprise prend connaissance de la commande.

2 Les données du problème

Les commandes

L'entreprise reçoit les commandes successivement selon leur date de réception. A chaque commande est associé :

- la date de réception,
- la fenêtre de temps pour la livraison durant laquelle la livraison doit être livrée (dans le cas d'une demande « express » il s'agit de la date de réception + une heure),
- la position du client pour la livraison,

- la quantité de produit à livrer,
- le temps de préparation,
- le temps de livraison chez le client,

L'entrepôt

On se limite pour le moment à un seul entrepôt, on connaît sa position.

Les livreurs

L'entreprise a à sa disposition un nombre limité de livreurs. Chaque livreur a une quantité de produit qu'il est possible de transporter dans une même tournée.

Coûts

L'entreprise a connaissance des temps et coûts kilométriques entre chaque lieu par un livreur. Chaque tournée a un coût fixe, en plus des coûts kilométriques. Si l'entreprise ne livre pas une commande « express » dans l'heure, un coût supplémentaire lui est facturé, ce coût est connu.

3 Les variables du problème

Il faut déterminer les tournées : l'ordre des commandes associées à cette tournée, le livreur effectuant la tournée et la date de départ de la tournée.

4 Les contraintes du problème

Livraison unique

Une commande ne peut être livrée qu'une fois. Un seul livreur aura donc pour tâche de la livrer dans son intégralité.

Indisponibilité du livreur

Dès lors que le livreur quitte le dépôt pour effectuer une tournée, il devient indisponible jusqu'à son retour. Il peut effectuer autant de tournées que nécessaires.

Livraison en cours non modifiable

Lorsqu'un livreur effectue une tournée, il n'est pas possible de la modifier. En revanche, il peut recevoir l'ordre de retourner au dépôt sans terminer la tournée qu'il effectue. La partie annulée de la livraison sera à planifier à nouveau.

Attente lors d'arrivée en avance

Si lors d'une tournée la date d'arrivée estimée d'un livreur est antérieure au début de la fenêtre de temps de la commande associée, alors celui-ci devra attendre la durée nécessaire.

Si une commande est livrée après la fenêtre de temps alors un retard est engendré.

5 Fonction objectif

L'objectif est de minimiser les coûts associés à la livraison : coûts kilométrique, coûts fixes par tournée, pénalité de retard des livraisons « express ». Il faudra également tenir compte des retards de livraison.

Deuxième partie

Etat de l'art

L'état de l'art comporte deux parties, la première traite des différents types de problèmes de routage et leur résolution. La seconde aborde la simulation en apportant une définition et une description du fonctionnement.

1

Les problèmes de routage

Résoudre un problème de routage consiste à trouver la meilleure organisation pour répondre à un ensemble de demandes clients. L'optimisation peut se faire selon différentes mesures :

- Le coût minimum,
- La distance minimum,
- Temps minimum de voyage.

La demande peut prendre différentes formes :

- Livrer à partir du dépôt (Amazon),
- Réceptionner une commande à un point donné et retourner au dépôt (Transports scolaires),
- Réceptionner une commande à un point donné et la livrer à un autre point.

L'organisation qui répond au problème est un ensemble de tournées. Une tournée est elle-même un ensemble de demandes. Il faut allouer à chaque tournée une ressource ayant les capacités nécessaires.

TSP : Travelling Salesman Problem

Il s'agit du problème du voyageur de commerce, une tournée est suffisante pour répondre à toutes les demandes. Ce type de problème peut avoir une solution heuristique dans laquelle on crée une solution réalisable qu'on améliore progressivement. Des algorithmes de solutions « exactes » existent, mais ils sont plus lents. La construction de la construction se fait via divers algorithmes :

- Nearest neighbor : créer la tournée en choisissant le point le plus proche.
- Nearest insertion : créer la tournée en choisissant le point le plus proche comparé à tous les points de la tournée.
- Saving method : classer les ralliements entre les points deux à deux et créer la tournée en commençant par les ralliements les plus « économiques ».

De même, l'amélioration de la construction peut se faire via différents algorithmes :

- K-change : changer l'ordre de K arcs afin d'avoir un meilleur coût (la complexité est de l'ordre du nombre de nœuds puissance K).
- K-relocate : on remplace les arcs entre K nœuds par d'autres afin d'avoir un meilleur coût.
- Swap, GENI...

VRP : Vehicle Routing Problem

Problème de routage de véhicules, ici plusieurs tournées sont nécessaires.

VRPTW : Vehicle Routing Problem with Time Windows

En plus de créer différentes tournées, il faut respecter des contraintes de temps : une demande ne peut être répondue que pendant une certaine fenêtre de temps.

Réception ET livraison

Il est également possible de faire de la réception et de la livraison dans le même problème voire dans la même tournée.

Complexité Supplémentaire

De nombreuses extensions peuvent-être ajoutées au problème :

- Différents types de véhicules, différents types de produits,
- Différentes mesures du coût,
- Priorités associées à des demandes,
- Fenêtres de temps pour la livraison,
- Durée de travail maximal par livreur...

Symétrique ou Asymétrique

Si la distance entre deux points est la même quel que soit le sens du parcours entre ceux-ci, alors le problème est dit symétrique. Si tel n'est pas le cas il est dit asymétrique.

2

La simulation

Un système (ensemble de faits) est étudié puis modélisé pour dégager des aspects intéressants. Si on peut prouver par des moyens mathématiques que le modèle est correct, on parle alors de solution analytique. Mais souvent, pour des systèmes réels à grande échelle il est trop complexe d'effectuer une analyse. Dans ce cas, il convient de faire une simulation. Cette simulation consiste à évaluer le modèle numériquement pendant une période donnée, les données sont assemblées et traitées pour estimer la véracité ou non des caractéristiques dégagées par le modèle.

Système

Un système est une collection d'entités qui interagissent entre elles. Il peut être continu ou discret.

Modèle

Le modèle est la représentation du système développé dans le but de l'étudier.

Le déroulement de la simulation a lieu à l'aide d'un échéancier. L'échéancier contient la liste des événements. A chaque itération, l'échéancier choisit l'évènement futur le plus proche et met à jour l'heure de la simulation à la date de celui-ci. En plus de changer l'heure de la simulation, d'autres opérations sont traitées : il peut s'agir par exemple de changer l'état d'une entité ou de modifier une valeur statistique comme la durée d'attente totale dans le système.

Une simulation se compose de plusieurs éléments :

- L'ensemble des états du système.
- L'horloge, une variable contenant l'heure de la simulation.
- Des compteurs statistiques, des variables mises à jour durant la simulation et qui fourniront a posteriori des données statistiques.
- La routine d'initialisation : une méthode qui prend en charge l'initialisation du modèle.
- L'échéancier gère l'horloge et la liste des événements futurs.
- Un générateur d'observations aléatoires à partir de lois de probabilités.
- Un générateur de rapport qui fournit des informations statistiques générale à partir des données statistiques traitées.
- Une méthode principale, qui utilise les éléments ci-dessus pour dérouler la simulation.

Fin de la simulation

Une simulation peut boucler à l'infini, la fin d'une simulation peut se programmer manuellement de deux manières :

- Après l'occurrence d'un nombre fini d'observations, dans ce cas le temps est une variable aléatoire.
- Au delà d'une certaine durée : dans ce cas la simulation reçoit un événement d'arrêt, le nombre d'observation est une variable aléatoire. Il s'agit de la méthode la plus utilisée

Représentation

La simulation peut se modéliser graphique à l'aide d'un Event-Graph :

Les événements sont des nœuds, reliés entre eux par des arcs orientés (flèches) indiquant l'ordonnement de ceux-ci. Deux types de flèches fournissent des informations supplémentaires sur cette ordonnance :

Flèches épaisses

A la pointe de la flèche, l'événement peut être ordonné à partir de l'événement à la base de la flèche (la quantité de temps est souvent aléatoire et positive, mais elle peut être nulle).

Exemple : Planifier une date de départ à partir de l'arrivée d'un livreur chez un client.

Flèches minces

L'événement à la base la flèche conduit à l'événement en bout de flèche sans temps d'intervention.

Exemple : La fin de livraison d'un livreur chez un client conduit instantanément au départ de celui-ci vers un autre client ou au dépôt.

Règle de simplification du graphe

Si les arcs entrant d'un nœud sont tous des flèches minces, alors le nœud est un événement intermédiaire, il peut être supprimé.

Intérêt de la simulation

Parfois, l'évolution d'un résultat peut être prédite intuitivement en fonction de la variation des paramètres. Cependant, la simulation permet vraiment d'associer une valeur et donc de nuancer l'évolution d'un paramètre sur un autre.

Toutefois, il faut garder un esprit d'analyse et d'abstraction sur les résultats obtenus afin de ne pas tirer de conclusions hâtives. La méthode d'étude suivante fournit une ligne directrice :

1. Formuler le problème et organiser l'étude.
2. Collecter les informations et définir le modèle.
3. Vérifier la cohérence des suppositions établies, si tel n'est pas le cas retourner à l'étape 2.
4. Créer le programme et effectuer les tests.
5. Vérifier que le modèle programmé est valide, dans le cas contraire retourner à l'étape 2.
6. Faire des expérimentations concrètes.
7. Lancer le programme en production.
8. Analyser les données en sortie.
9. Documenter, présenter et utiliser les résultats.

Conclusion

Les gains potentiels de productivité ou d'économies engendrée par une prédiction des besoins sont immenses. Néanmoins, développer une simulation nécessite une équipe de développement conséquente pour répondre à étude de cas personnalisée. Il faut également suivre une démarche analytique sérieuse surtout lors de l'étude des résultats.

Troisième partie

Cahier de spécification système



Introduction

Ce chapitre a pour but de décrire les spécifications du Projet de Recherche et Développement : « Simulation d'un problème de tournées avec des commandes rapides ».

Il définira d'abord le sujet du projet et son contexte. Ensuite, seront décrites les fonctionnalités et leurs utilisations. La dernière partie portera sur la planification du développement de la solution.

Ce projet est proposé par l'équipe ROOT du Département Informatique de Polytech'Tours, les responsables seront Yannick Kergosien, Mustapha Haouassi et Jorge Mendoza qui représentent la MOA. Il est réalisé par Lorris Pigeon, étudiant ingénieur au Département Informatique en 5^{ème} année qui représente la MOE.

Il s'étale sur deux semestres étudiants, de Septembre à Avril, à raison de deux jours par semaine.

Une première partie est orientée recherche et a lieu au premier semestre. C'est durant cette partie que la modélisation, l'organisation et la spécification du projet auront lieu. La recherche prendra la forme d'un état de l'art des problèmes déjà existants, ainsi que d'une analyse du problème étudié.

Une seconde partie consiste à développer le simulateur durant le second semestre. A l'issue de chacune de ces parties, une soutenance aura lieu.

3

Contexte de la réalisation

1 Contexte

Résoudre à un problème d'ordonnancement consiste à choisir une solution qui répond le mieux à un objectif fixé. Cette solution répond à certaines contraintes, et se calcule à partir des données du problème. Le problème de transport étudié dans ce projet est dérivé d'un problème bien connu, le voyageur de commerce :

- Les données sont les villes et leurs positions.
- La contrainte est de visiter toutes les villes.
- L'objectif est de minimiser la distance parcourue.

Ici, le « voyageur » sera en fait des livreurs, et les « villes » des commandes à livrer dans une fenêtre de temps donnée. Un livreur sera à même de faire plusieurs tournées. Les commandes ne sont pas connues à l'avance mais au fur et à mesure du problème.

Définition du problème

Une entreprise de livraison reçoit des commandes de clients et doit faire son possible pour effectuer les livraisons dans les temps. Pour ce faire, elle a un entrepôt comportant différents produits, et des livreurs, qui peuvent effectuer plusieurs tournées pour répondre à la demande. Cette demande peut prendre deux formes :

1. Le client souhaite être livré rapidement, ainsi, dès réception de la commande, l'entreprise s'engage à la livrer dans l'heure, si elle n'atteint pas cet objectif, elle reçoit une pénalité. (un coût supplémentaire lui sera facturé).
2. Le client est plus flexible : il souhaite être livré sur une fenêtre de temps donnée, fenêtre qui ne peut commencer qu'à partir d'une heure suivant la demande. Par exemple celui-ci commande à 13h un produit et souhaite être livré de 14h30 à 17h30.

Lors d'une tournée, un livreur peut livrer plusieurs commandes successives, chaque tournée a un coût de déplacement pour l'entreprise (frais kilométriques) et un coût fixe. Le retard lors des livraisons de commandes « express » va engendrer un coût supplémentaire. L'objectif du problème est de diminuer les coûts des tournées et les coûts des commandes express non livrées à temps. Pour cela, il faut déterminer quelle commande sera livrée à quel moment par quel

livreur, lors de quelle tournée. Il faudra également déterminer la date de départ de chaque tournée.

Chaque commande a une date de réception, c'est à ce moment que l'entreprise prend connaissance de la commande.

2 But du projet et objectif

A partir d'une instance du problème, la simulation va tester les différentes stratégies développées et fournir des statistiques les évaluant. Pour cela, il faudra développer le déroulement de la simulation, définir l'ensemble des indicateurs et des critères d'évaluation des stratégies décisionnelles. Ensuite, à partir des données générées par la simulation, un affichage graphique permettra d'avoir un compte-rendu des différentes étapes :

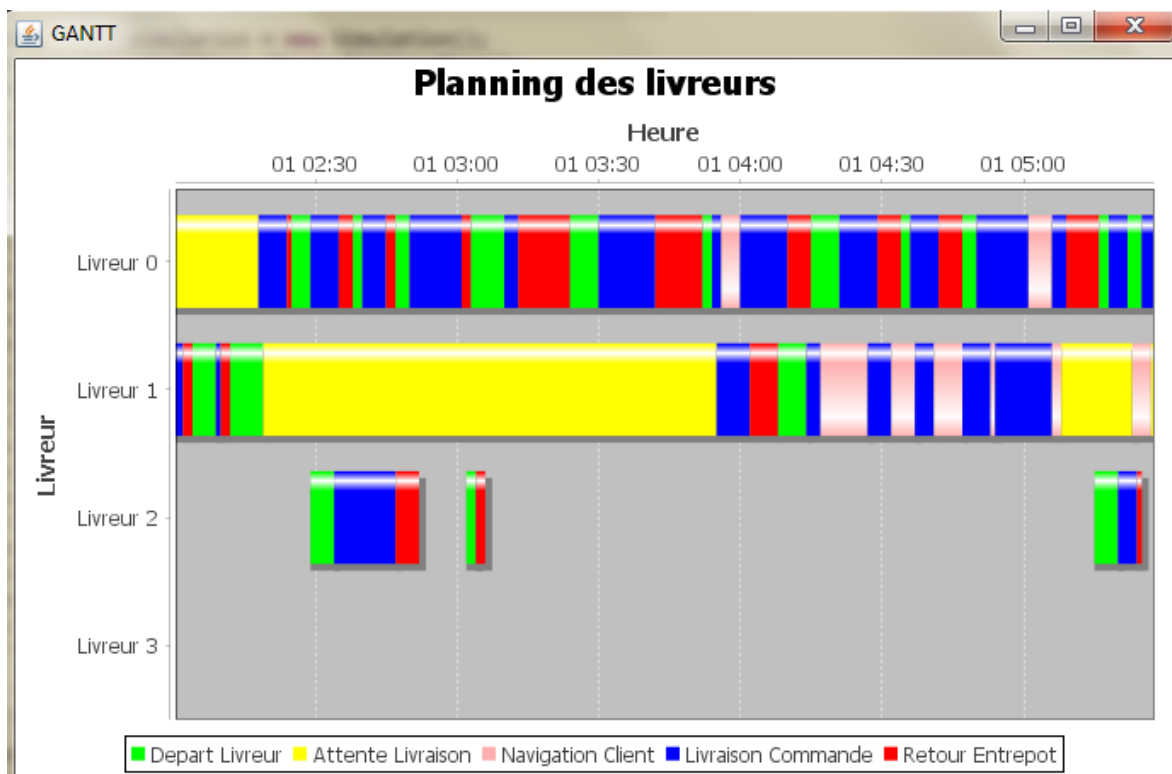


Figure 1 – Affichage du GANTT du planning

3 Base méthodologique

La modélisation du projet se fera en langage UML (Unified Modeling Language), l'outil Astah permettra d'effectuer des diagrammes de cas d'utilisation et de classes normalisées et rapidement compréhensibles. Un outil de versionning type Gitlab sera utilisé pour assurer une meilleure gestion du projet. L'organisation du projet sera faite à l'aide de l'outil de planification MindView. Enfin, les comptes rendus tels que ce document seront rédigés en Latex.

4

Description générale

1 Environnement du projet

Ce projet s'inscrit dans un projet de plus grande ampleur : un thème de recherche sur les problèmes de tournées en moins d'une heure. Dans ce projet de simulation, les commandes « express » et les surcoûts associés à celles-ci modélisent la contrainte de livraison en une heure. Les méthodes de résolution du problème, c'est-à-dire les règles de décision quant à l'insertion d'une nouvelle commande dans une tournée seront fournies par des algorithmes de la MOA.

2 Caractéristiques des utilisateurs

Les utilisateurs de l'application seront des membres de l'équipe ROOT, avec des solides compétences en informatique et en ordonnancement. Ils seront à même d'exploiter le code de la simulation en insérant leurs propres classes de stratégie décisionnelle. Cette simulation ayant pour but d'effectuer des tests de performance sur une méthode de résolution, aucun contenu n'est a priori confidentiel et aucune authentification utilisateur ne sera nécessaire.

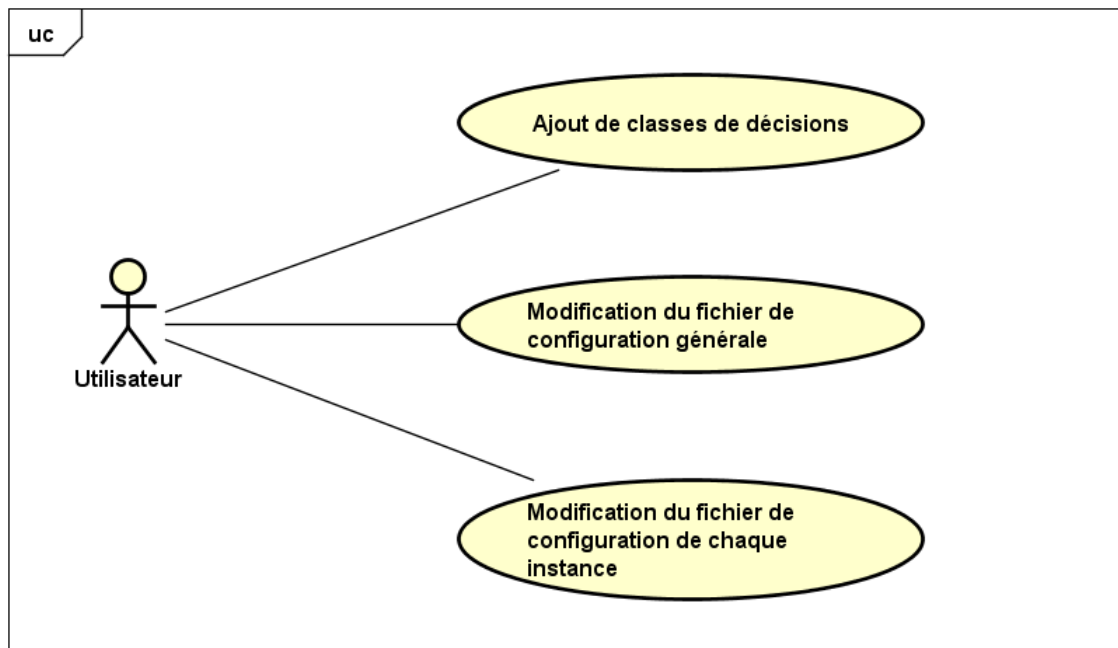
3 Fonctionnalités du système

Les fonctionnalités du système seront organisées en deux modes principaux :

3.1 Paramétrage de la simulation et calculs

Le premier mode consistera à paramétrer les données de la simulation. Des utilisateurs insèrent dans le code de la simulation leurs algorithmes de décision. Ensuite ils peuvent lancer la simulation à partir d'un fichier contenant les données sources du modèle (nombre de livreurs, position de l'entrepôt...). La simulation peut alors se lancer, les décisions seront prises en fonction de la stratégie développée par l'utilisateur. Un historique est stocké sous forme de fichier, il servira plus tard pour l'affichage graphique. En plus de l'historique, un fichier de log contenant les points notables de la simulation sera généré.

Il sera également possible de lancer plusieurs exécutions de simulations avec des données différentes. Un fichier de configuration générale permettra à l'utilisateur de spécifier certains paramètres comme le nombre d'instances à lancer.



powered by Astah

Figure 1 – Diagramme de cas d'utilisation du paramétrage de l'instance de la simulation

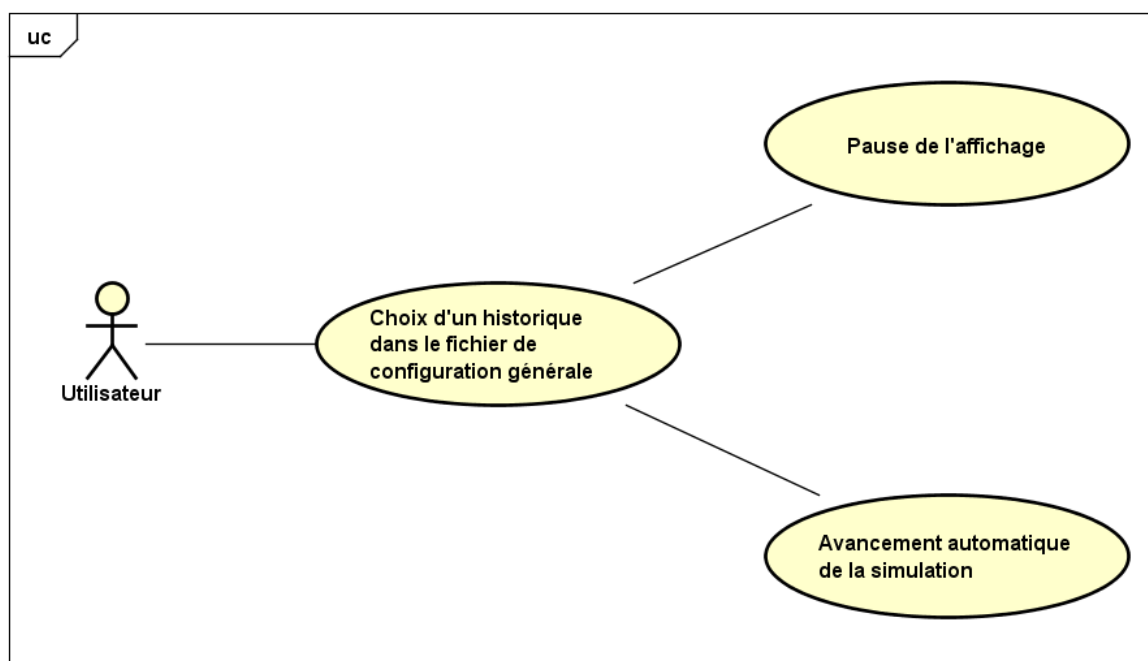
3.2 Affichage de la solution

La solution se représentera sous la forme d'un GANTT récapitulatif des activités des livreurs. Une ligne horizontale sera associée au planning d'un livreur, elle décrira son activité à l'instant t comme dans l'exemple ci-dessous.

Cette affichage permettra d'avoir une vision globale de la solution et de l'activité des livreurs, il sera en outre possible de zoomer et d'avoir des informations plus précises sur une action.

4 Contraintes de développement

Le développement se fera à l'aide du langage Java, la résolution du problème se fera à l'aide de méthodes définies dans des classes de décision Java. La bibliothèque d'affichage graphique sera JFreeChart, très utilisée et documentée. Elle comporte de nombreuses fonctionnalités, notamment l'affichage de GANTT.



powered by Astah

Figure 2 – *Diagramme de cas d'utilisation de l'affichage dynamique de la solution*

5

Description des interfaces externes du logiciel

1 Matériel/logiciel

Aucune interface n'est nécessaire, échange de données entre les entrées et sorties de la simulation.

2 Homme/machine

2.1 Représentation de la solution

A partir de l'historique d'une simulation, il sera possible pour l'utilisateur d'observer le résultat généré par le moteur de la simulation. L'affichage ressemblera à la figure précédente, avec la possibilité de zoomer d'avant en arrière et d'avoir des informations supplémentaires au passage de la souris.

3 Logiciel/logiciel

Le projet sera une application unique qui manipulera éventuellement des fichiers. L'application pourra également utiliser Google Maps, ou OpenStreetMap pour l'affichage, à l'aide d'un projet existant.

6

Architecture générale du système

La classe Simulation comporte une instance d'un problème de tournées. Les données de ce problème saisies en entrée par l'utilisateur sont lues à partir d'un fichier. Les contextes de départ et de fin de la simulation sont définis à partir d'une méthode d'initialisation.

Une fois la préparation de la simulation terminée, son déroulement peut se faire à l'aide de l'échéancier, qui parcourt la liste des événements et traite les actions associées. En plus de l'échéancier, des méthodes définies dans la classe de décision permettront d'appliquer les règles d'optimisation du problème. Lors du déroulement de la simulation une classe d'historique maintiendra en mémoire le détail des ressources et du planning.

L'affichage de la simulation prendra en source l'historique généré précédemment. Il affichera le travail des livreurs en fonction du temps.

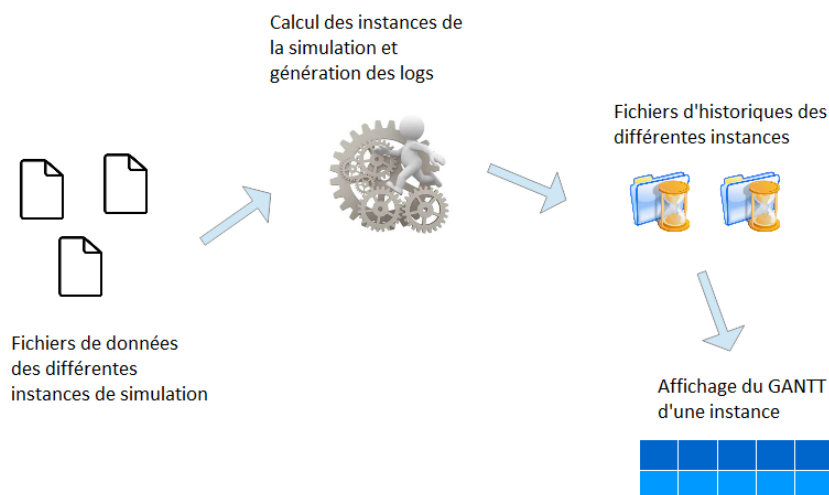


Figure 1 – Architecture generale du systeme

7

Fonctionnalités du système

1 Le moteur de la simulation

Cette partie contient toutes les données nécessaires au déroulement de la simulation.

La figure ci-dessous représente le système et les relations entre les événements :

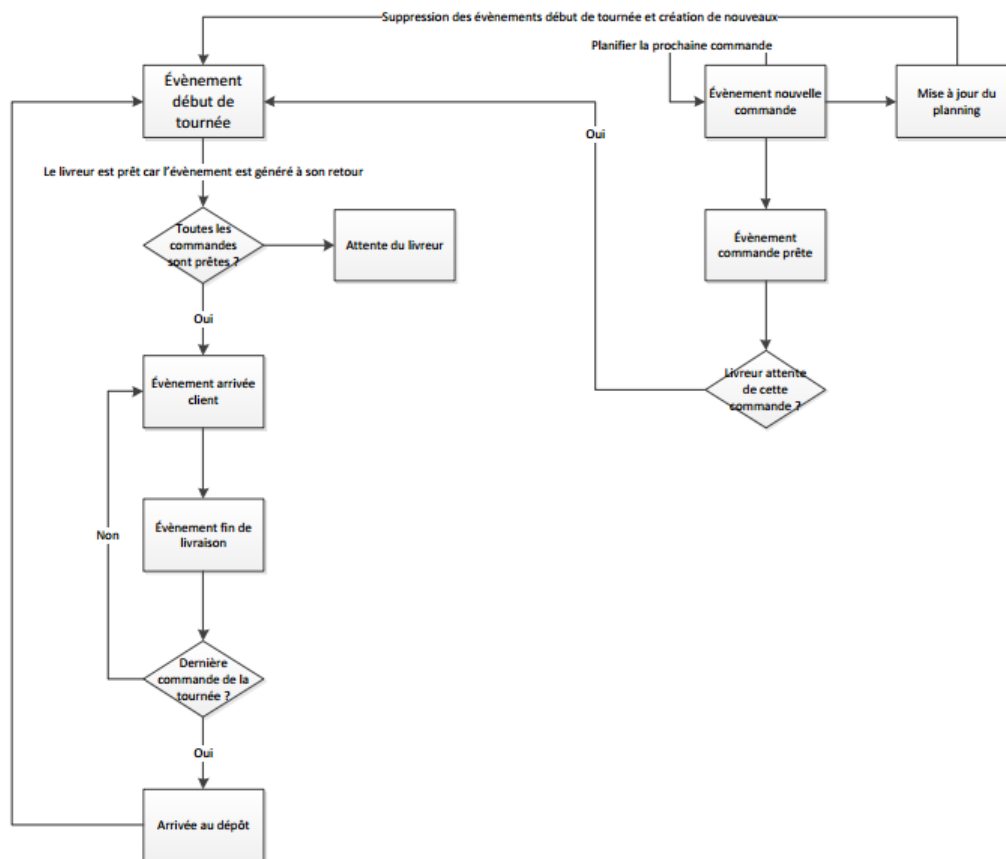


Figure 1 – Représentation du système

La fonction chargerDonnees(File file)

Cette fonction permet de créer l'instance du problème à partir d'un fichier. Le parsing de ce fichier fournira les informations suivantes :

- Les dimensions de la carte.
- La position de l'entrepôt.
- Le nombre de livreur et leur capacité maximale.
- La liste des commandes et leurs attributs.

Cette fonction ne retourne aucune valeur.

Précondition

La structure du fichier et son contenu doivent correspondre.

Postcondition

Les attributs de la classe simulation ont des valeurs correctes. La simulation est prête à être initialisée.

La fonction start()

Cette fonction permet d'effectuer le déroulement de la simulation. Les actions sont les suivantes :

- Initialisation du planning et d'autres variables.
- Exécution des événements selon leur ordre d'arrivée chronologique.
- Finalisation de la simulation et calculs de données statistiques.

Précondition

La fonction chargerDonnees a été lancée afin d'initialiser l'instance de la simulation.

Postcondition

Un fichier de logs et un autre d'historique sont générés.

Les évènements de la simulation

La simulation contiendra une liste d'évènements. Un évènement héritera de la classe abstraite Evenement et redéfinira la méthode run().

Les différents évènements possibles sont les suivants :

- Début de tournée
- Fin de préparation commande
- Nouvelle commande
- Init
- Arrivée client
- Fin attente client
- Fin de livraison client
- Arrivée au dépôt

La suite de cette partie va aborder le détail de la méthode run() pour chaque évènement. Il est à noter que des données statistiques et historiques sont enregistrées à chaque traitement.

Evènement début de tournée

Lors de la réception de commandes, des tournées sont planifiées. L'évènement début de tournée va organiser le départ d'un livreur ou son attente au dépôt si certaines commandes de la tournée ne sont pas préparées.

Précondition

Le livreur était déjà présent au dépôt suite à un évènement arrivée au dépôt.

Evènement fin de préparation commande

Après un certain temps après sa réception, a commande est considérée comme préparée.

Si cette commande avait retardé le départ d'un livreur et que toutes les autres commandes de la tournée sont prêtes, alors un évènement arrivée client est généré.

Précondition

La commande avait été reçue via un évènement nouvelle commande.

Evènement nouvelle commande

Lorsqu'une commande est reçue, il faut planifier la réception d'une prochaine commande en générant un autre évènement nouvelle commande.

Ensuite, il faut insérer la commande dans le planning en faisant appel à une fonction de décision. si une nouvelle tournée est créée lors de l'appel à cette fonction, un évènement début de tournée sera généré.

Enfin, un évènement fin de préparation commande est également généré afin que la tournée puisse débuter.

Evènement init

Cette évènement est généré lors de l'initialisation de la simulation. De nombreuses commandes ont déjà été reçue avant que la simulation ait lieu. Pour les insérer dans le planning, une fonction de décision, en charge de l'optimisation de la livraison de toutes les commandes initiales, sera appelée.

Pour chacune des tournées générées par cette fonction, un évènement début de tournée sera également ajouté dans l'échéancier.

Evènement arrivée client

Cet évènement marque la fin du passage d'un client au suivant au sein d'une tournée. Suivant la commande du client suivant, deux cas sont à distinguer :

L'arrivée du livreur est antérieure à la fenêtre de livraison : Un évènement fin attente client est généré.

L'arrivée du livreur a lieu dans la fenêtre de livraison ou après : Un évènement fin de livraison client.

Evènement fin attente client

La livraison d'un client peut commencer une fois que l'heure de la simulation arrive dans la fenêtre de la commande. Un évènement fin livraison client est généré.

Précondition

L'arrivée du livreur est antérieure à la fenêtre de livraison de la commande.

Evènement fin livraison client

Lorsqu'un la livraison d'une commande chez un client est terminée, un évènement arrivée client est généré pour le client suivant de la tournée. S'il s'agissait du dernier client de la tournée, c'est un évènement arrivée au dépôt qui est généré.

Evènement arrivée au dépôt

Cet évènement marque la fin d'une tournée et la disponibilité du livreur pour une autre tournée. Une fonction de décision est lancée au sein de cet évènement pour attribuer au livreur tournée et créer l'évènement début de tournée.

2 L'affichage graphique de la simulation

L'affichage se fait à partir d'un fichier d'historique. Il est possible de zoomer d'avant en arrière et d'avoir des informations au passage de la souris. Le développement de cet affichage se fera à l'aide d'une librairie libre de droit : JFreeChart. Elle permet d'afficher de nombreux types de graphes et notamment des GANTT.

Choix de l'historique

Le choix de l'historique est entré par l'utilisateur dans le fichier de configuration.

Création du jeu de données

Les données de cet historique sont à adapter à la structure de données de la librairie JFreeChart.

Génération du graphe

Une méthode doit également initialiser le graphe (gestions des étiquettes, des bulles d'informations...) avant de pouvoir lancer l'affichage du GANTT.

Quatrième partie

Mise en oeuvre

8

Plan de développement

La première phase sera d'abord axée sur le moteur de la simulation. Suite à l'effort de modélisation du premier semestre, le développement devrait être assez rapide : huit semaines environ. En revanche, deux autres semaines seront consacrées aux tests, en effet, générer un échantillon de données à tester pour une simulation est assez complexe. De plus, certaines adaptations du code seront sans doute nécessaires pour le bon déroulement de la phase suivante.

La seconde phase portera sur l'affichage graphique, dans la mesure du possible, elle sera préparée par la première phase lors des tests et la génération de l'historique. La durée indicative de cette phase est de trois à quatre semaines. En réalité, comme cette phase est assez dépendante de la précédente, la durée et donc le résultat de l'affichage peuvent varier.

Les complications à envisager et leur résolution sont les suivantes :

Retard sur le développement du moteur de la simulation

De nombreuses tâches sont à effectuer durant cette phase, et il n'est pas impossible que la complexité d'une de ces tâches ne soit pas correctement évaluée. Dans ce cas le retard sera rattrapé lors de la phase de test précédemment évoquée.

Classes de décision non fonctionnelles

Une semaine a été consacrée à l'insertion des classes de décision dans le projet. Dans le cas où celles-ci ne seraient pas disponibles ou utilisables, cette durée permettra le développement d'une classe simple de décision.

Première phase incomplète

Il est également possible que le résultat de la première phase ne puisse être exploité, dans ce cas un fichier d'historique sera généré "manuellement" à l'issue de la phase de test et en début de seconde phase.

Seconde phase incomplète

Dans le cas où la seconde phase n'aboutirait pas par manque de temps par exemple, les fichiers d'historique et de log de la première phase seront une garantie du succès de la simulation.

Organisation

Le détail du plan de développement se trouve dans le tableau ci-dessous :

	i	Plan...	Nom de tâche	Durée	Début	Fin	Prédécesseur...
1			Initialisation du Projet	3 jours	04/01/2017	11/01/2017	
2			Mise en place du projet (Git...)	1 jour	04/01/2017	04/01/2017	
3			Création des classes de base	2 jours	05/01/2017	11/01/2017	2
4			Création des classes Événement	4 jours	12/01/2017	25/01/2017	3
5			Classe abstraite et création instan...	1 jour	12/01/2017	12/01/2017	
6			Développement des événements	2 jours	18/01/2017	19/01/2017	
7			Tests	1 jour	25/01/2017	25/01/2017	
8			Instanciation du problème	2 jours	26/01/2017	01/02/2017	3
9			Création et lecture du fichier source	2 jours	26/01/2017	01/02/2017	
10			Implémentation de la simulation	12 jours	02/02/2017	15/03/2017	3
11			Fonction principale	2 jours	02/02/2017	08/02/2017	
12			Ajout Classes Décisionnelles	2 jours	09/02/2017	15/02/2017	
13			Gestion de l'historique	1 jour	16/02/2017	16/02/2017	
14			Gestion des logs	1 jour	22/02/2017	22/02/2017	
15			Calculs données statistiques	2 jours	23/02/2017	01/03/2017	
16			Test Corrections Documentation	4 jours	02/03/2017	15/03/2017	
17			Affichage Simulation	7 jours	16/03/2017	06/04/2017	10;8;4
18			Utilisation de l'historique	3 jours	16/03/2017	23/03/2017	
19			Partie Graphique	4 jours	29/03/2017	06/04/2017	18
20			Développement de l'affichage	2 jours	29/03/2017	30/03/2017	
21			Gestion des événements	2 jours	05/04/2017	06/04/2017	

Figure 1 – Détail du plan de développement

Les durées et les tâches du GANTT ont été remplies conjointement avec la MOA afin d'avoir une garantie de pertinence sur l'activité du second semestre. Toujours dans une optique de pertinence, la méthode de gestion agile a été envisagée, mais le premier résultat perceptible aurait eu lieu lors des tests de la simulation planifiés le 2 Mars, des sprints de deux semaines n'auraient pas été pertinents. Il a donc été convenu d'envisager une méthode en cycles incrémentaux. Chaque tâche listé ci-dessous correspond à une itération qui comprend les éléments suivants :

- Réflexion sur l'implémentation technique d'une fonctionnalité
- Développement
- Test de comportement unitaire
- Test d'intégration (relations entre les autres éléments existants)

Cette méthode a permis de développer le projet avec des minis-jalons à l'issue de chaque étape. Elle quantifie donc l'avancement dans chaque grande partie, néanmoins elle ne retire pas certains risques inhérents au projets, développés dans la partie suivante.

Le diagramme de GANTT du projet se trouve ci-dessous :

Suivi de projet

Le suivi de l'avancé du projet a eu lieu toutes les deux semaines environ lors de rendez-vous réguliers avec la MOA (Mr. Kergosien). Ces rendez-vous consistaient à une présentation des développements effectués avec ensuite des conseils et précisions apportés par la MOA. Un

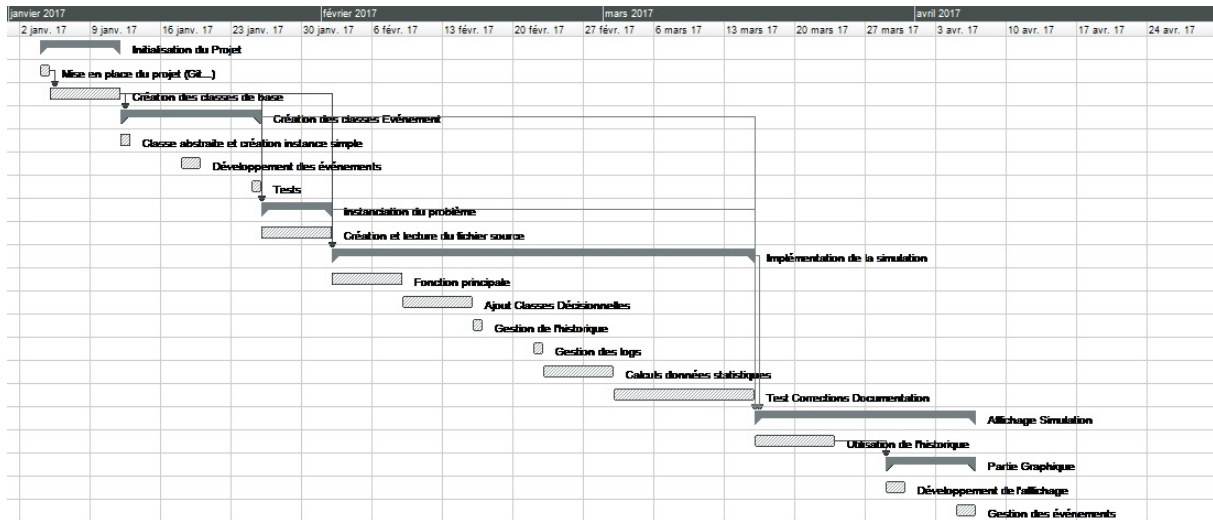


Figure 2 – Diagramme de GANTT du projet

retour de celle-ci a notamment permis un gain de temps : ne pas perdre trop de temps lors du développement de l'affichage en ne réinventant pas la roue mais en utilisant une librairie sous license libre JFreeChart.

9

Implémentation

1 Bibliothèques

Deux bibliothèques majeures ont été utilisées lors de ce projet.

Apache

Une bibliothèque d'Apache a permis de générer des valeurs aléatoires à partir de lois de probabilités (loi de Poisson et loi uniforme) à l'aide de processus de calculs complexes.

JFreeChart

JFreeChart, une bibliothèque Java d'affichage de graphiques a été choisie afin d'afficher le planning d'une simulation. Elle est assez utilisée et correctement documentée avec un exemple d'utilisation de base sur lequel j'ai pu me baser afin d'avoir un affichage pertinent. Même si le rendu offert par cette bibliothèque n'est pas exceptionnel, le gain de temps est conséquent.

2 Structure du code

Les classes respectent le diagramme de classe original, elles sont organisées en package suivant leurs fonctions. Pour plus de détails, se référer au guide d'utilisateur et aux diagrammes de classes présents en annexe.

3 Fichiers de données et de configurations

L'application utilise des fichiers de configurations présent dans le dossier ressources. Cela permet d'avoir une vision d'ensemble des paramètres et une modification rapide de ceux-ci, néanmoins, si un paramètre ne prend pas une valeur cohérente l'application sera arrêtée.

L'application utilise également des fichiers de données en entrée pour recevoir le détail des commandes, le risque est le même que précédemment.

4 Fichiers de logs

De nombreux fichiers de logs renseignent sur le déroulement de la simulation lors de l'exécution de la simulation. L'inconvénient est qu'il faut parfois aller chercher des informations (comme des retards de livraison) qui auraient pu être affichées directement dans l'interface utilisateur de l'application.

5 Performance

La simulation exécute des événements successivement, la performance dépend réellement des classes de décisions implémentées par les utilisateur. Actuellement, la classe `MeilleurInsertion` n'est (volontairement) pas performante puisqu'elle teste toutes les possibilité avant de choisir la meilleur solution. Elle peut donc servir de test pour des méthodes plus performantes.

Bilan et conclusion

6 Avancement du projet

Le premier semestre de recherche a permis un gain de temps conséquent sur les développements du second semestre. En effet l'état de l'art a permis d'opter pour les meilleurs choix technologiques et l'analyse et la modélisation ont apporté des éléments clairs et concrets sur lesquels s'appuyer en seconde phase.

Le moteur de la simulation est fonctionnel et rapidement exploitable avec un exécutable et des fichiers de configurations. Néanmoins, l'affichage graphique est encore améliorable en terme de lisibilité et de d'interaction. Il est actuellement possible de consulter un planning et utilisant le zoom pour accéder aux parties importantes. Il serait pertinent d'utiliser un menu afin de choisir un jour particulier du planning. De plus les retards ne sont pas mis en valeur, il serait également intéressant d'ajouter une coloration particulière afin qu'un utilisateur souhaitant juste consulter un planning puisse les retrouver rapidement.

Globalement, le planning a été respecté, ce qui est très satisfaisant connaissant la difficulté dans l'estimation de la durée de développement. Il est à noter par ailleurs que de nombreuses corrections ont eu lieu lors des phases de tests.

J'ai pris le soin de commencer la phase de développement au plus tôt (début du mois de Janvier). Finalement, il aurait peut-être été judicieux de la commencer en Décembre afin de pouvoir aller plus en avant dans le développement de l'affichage graphique.

7 Qualité

Afin d'avoir un projet de qualité, de nombreuses politiques ont été mises en place :

- Respect des conventions de nommage
- Documentation (Javadoc)
- Utilisation d'une librairie d'affichage graphique (JFreeChart)
- Versionning
- Mise en place d'un guide d'utilisateur
- Création et exécution de tests tout au long de la phase de développement
- Détection de problèmes potentiels dans le code avec Sonarqube

- Structuration du fichier de ressources de l'application
- Création d'un exécutable java (fichier jar)
- Respect des spécifications de base et mises à jour du diagramme de classe fréquentes

Mettre en place ces politiques est une partie intégrante des phases de développement. Elle n'est en aucun cas optionnelle car elle permet une adaptabilité du code, une compréhension de celui-ci par autrui et une exécution optimale de l'application finale.

8 Bilan

Cette expérience m'a beaucoup apporté dans différents domaines. J'ai constaté qu'un projet ne réussit pas ou n'échoue pas par hasard. L'erreur aurait été de penser qu'il ne s'agissait que de l'action de résolution (les développements). En réalité le développement n'a été qu'une partie de l'exercice, et il n'a pu avoir lieu qu'après une réflexion sur la méthodologie (qualité de code), l'organisation (gestion de projet), la modélisation (spécifications)... C'est ce niveau de recul que doit toujours avoir un ingénieur dans un projet.



Remerciements

Je tiens à remercier l'équipe de la MOA et plus particulièrement Yannick Kergosien pour ses conseils et son aide lors de la modélisation.

Je remercie également plus largement l'équipe enseignante de Polytech'Tours pour ses conseils sur différentes parties de mises en place du projet (Gestion de projet, Qualité de code, Analyse et Spécifications...).



Bibliographie

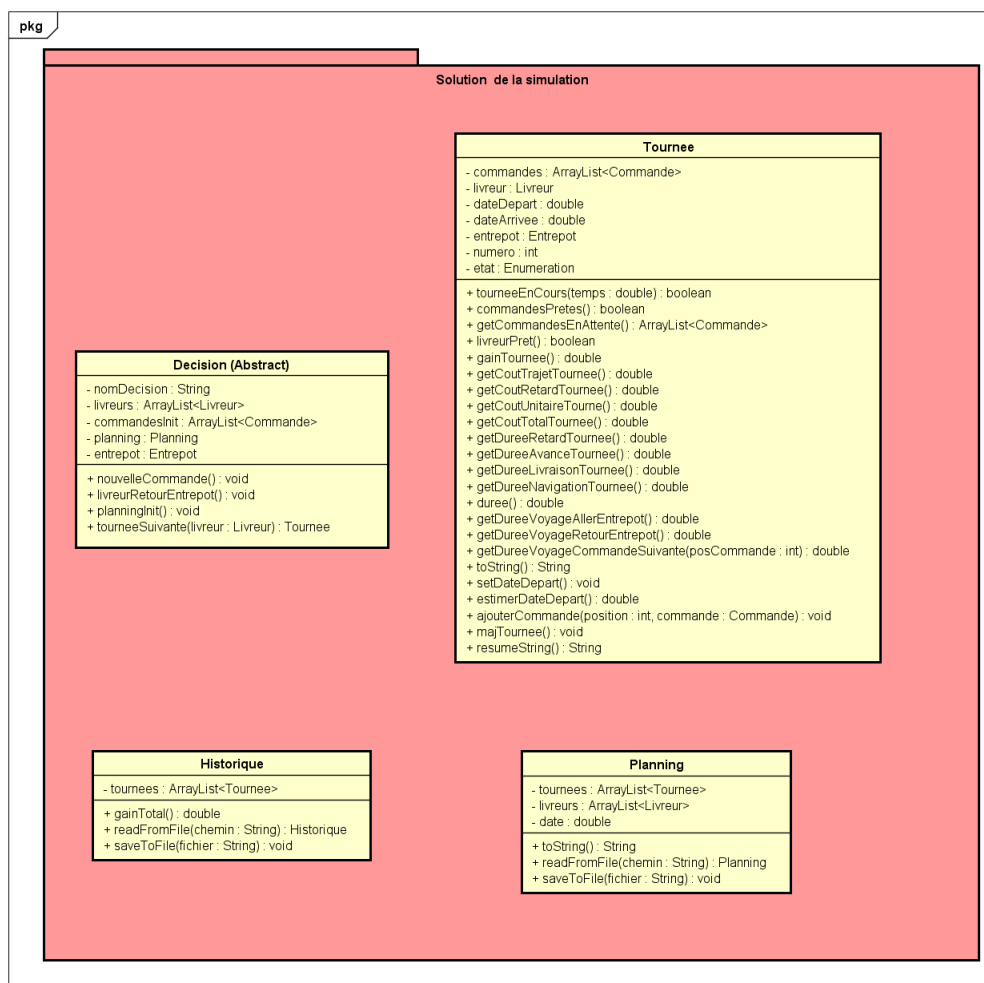
- [1] Simulation Modeling and Analysis, Averill M. Law
- [2] A tabu search algorithm for the open vehicle routing problem Jose Brandao
- [3] A multi-phase constructive heuristic for the vehicle routing problem with multiple trips R.J. Petch, S. Salhi
- [4] An interactive simulation and analysis software for solving TSP using Ant Colony Optimization algorithms Aybars Ugur, Dogan Aydin
- [5] A PARALLEL TABU SEARCH HEURISTIC FOR THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS PHILIPPE BADEAU, FRANCOIS GUERTIN, MICHEL GENDREAU, JEAN-YVES POTVIN, ERIC TAILLARD

Annexes

A

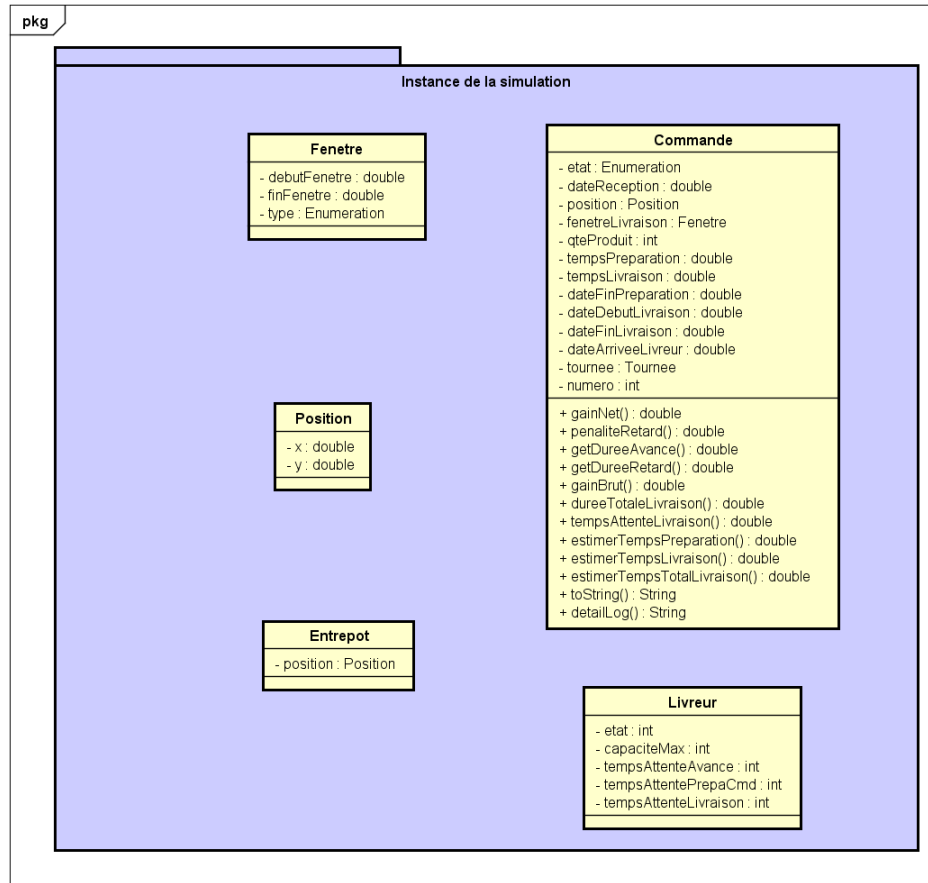
Diagramme de classe de la simulation

Le diagramme de classe initiale a subi de nombreuses modifications, il est donc divisé en parties pour les besoins d'affichage :



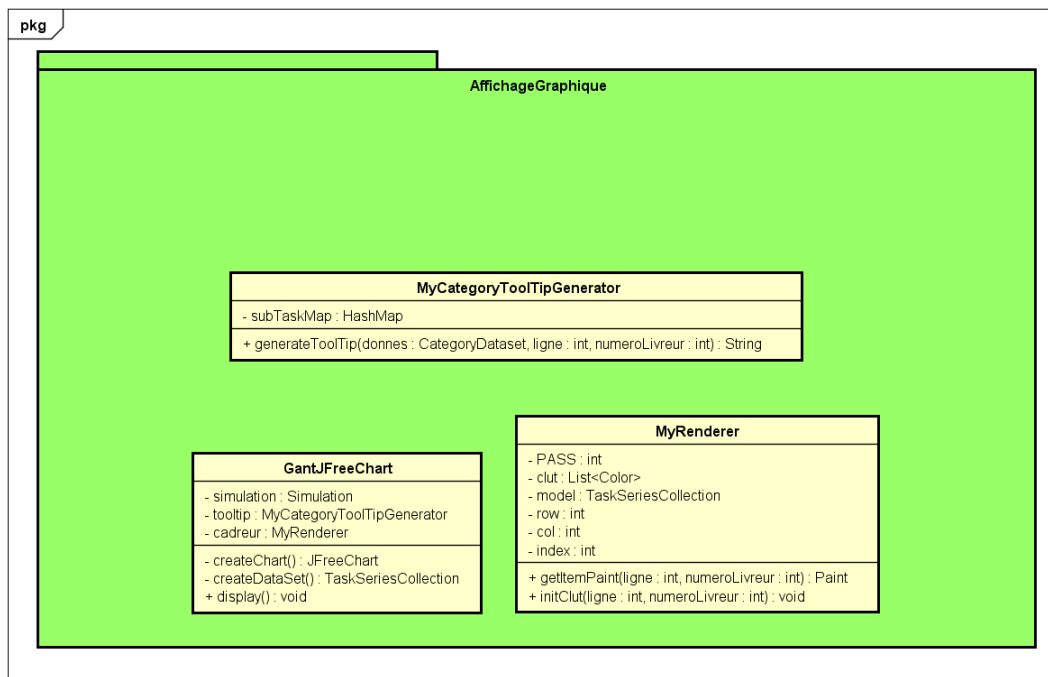
powered by Astah

Figure 1 – Partie solution de la simulation



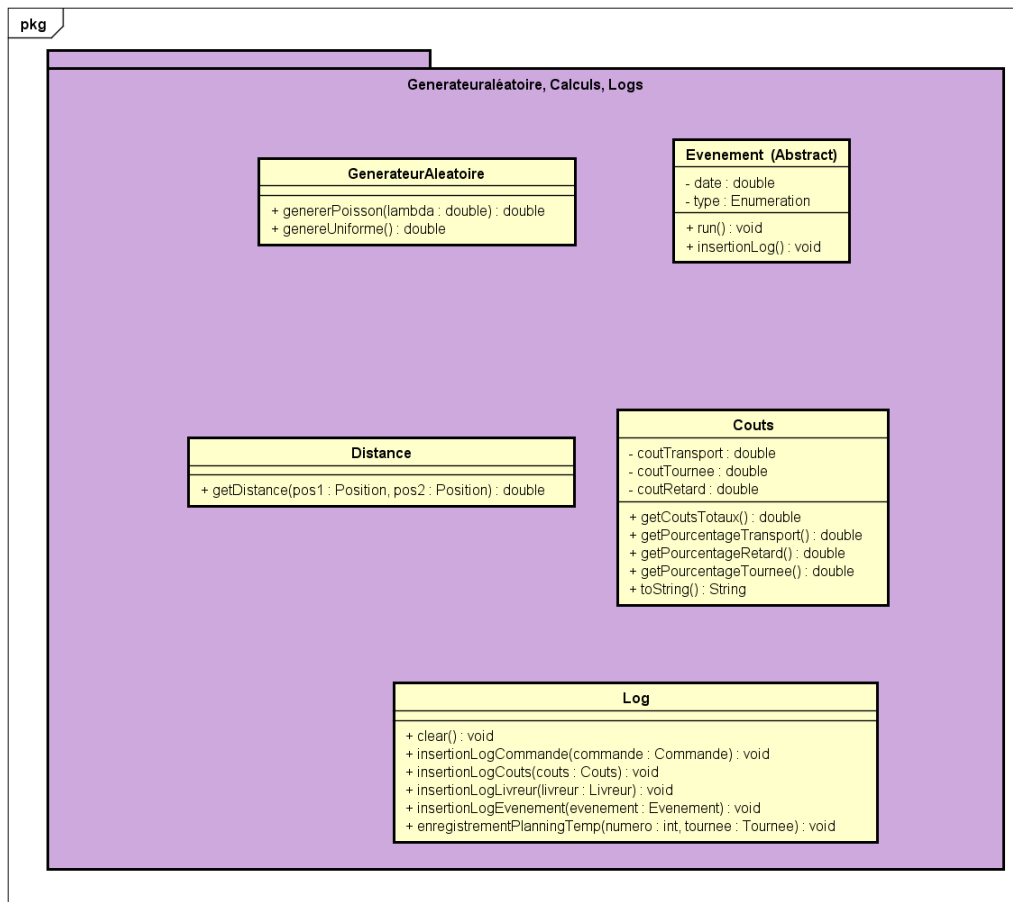
powered by Astah

Figure 2 – Partie instance de la simulation



powered by Astah

Figure 3 – Partie affichage de la simulation



powered by Astah

Figure 4 – Partie calcul de la simulation



Figure 5 – Classe Simulation

B

Cahier de tests

1 Introduction

Cette partie a pour but de rendre compte des différents tests qui ont eu lieu lors du développement du projet de recherche et développement Simulation d'un problème de tournées avec des commandes rapides.

Les classes de tests sont situées dans le package test du projet.

2 Tests Unitaires

Les tests unitaires ont été implémentés à l'aide de JUnit. Les classes testées sont les événements, en effet l'application va consister en une succession d'événements et il convenait de vérifier que la réalisation de ceux-ci avait bien le comportement escompté. Pour ce faire, un contexte est mis en place dans lequel on ajoute artificiellement l'événement testé. Ensuite, l'événement est réalisé et le contexte est réétudié afin de détecter de potentielles anomalies. Les événements testés sont les suivants :

- Arrivée Client dans la classe ArriveeClientTest
- Début Tournée
- Fin Attente Client
- Fin Livraison Client
- Fin Préparation Commande
- Nouvelle Commande

Un objet événement ne comporte qu'une méthode `run()` (dédiée à sa réalisation), c'est donc cette méthode qui sera testée dans les classes de tests.

3 Tests d'intégration

Une fois le moteur de la simulation terminé, il a été nécessaire d'effectuer des tests d'intégration afin d'en vérifier la pertinence qu'elle qu'en soit l'instance en entrée. Différents éléments ont été testés :

- Les fonctions d'évaluation de coûts et de temps durant le calcul de la solution
- Les contraintes de base du problème

3.1 La classe EvaluationCommandeTest

Cette classe va permettre de vérifier les calculs : des coûts d'une commande selon sa date de livraison réelle et sa fenêtre de livraison initiale des durées de retard ou d'avance d'un livreur selon les mêmes paramètres

3.2 La classe EvaluationTournéeTest

Cette classe va permettre de vérifier les calculs des coûts (kilométrique, de retard et de frais unitaire de tournée) d'une tournée en fonction des commandes assignées. Elle va permettre également de vérifier les calculs des durées (livraison, navigation, attente).

3.3 La classe SimulationTest

Cette classe comporte différentes méthodes de tests, deux contraintes du problème sont vérifiées par celles-ci :

- Toutes les commandes sont livrées une seule fois avec la méthode `testCommandeLivreeUneFoisEtUneSeule()`
- Un livreur ne peut effectuer qu'une seule tournée à la fois avec la méthode `testLivreurUneSeuleLivraisonEnCours()`
- Un objet événement ne comporte qu'une méthode `run()` (dédiée à sa réalisation), c'est donc cette méthode qui sera testée dans les classes de tests.



Guide d'utilisation

1 Exploitation du sujet

1.1 Ouverture du projet sous Eclipse

Pour modifier le projet et générer un .jar (exécutable java), il est nécessaire de l'ouvrir sous eclipse. Pour cela, lancer eclipse et naviguer dans le menu File, Export, General, Existing project into workspace, puis choisir le dossier source du projet.

1.2 Exécution du projet

Sous Eclipse

Une fois le projet ouvert avec Eclipse, un clic droit sur la class Main.java permet de choisir le menu run as Java application.

En ligne de commande

Si le fichier exécutable .jar n'existe pas dans l'arborescence du projet, il faut le créer (cf Génération de l'exécutable .jar avec Eclipse). Ouvrir un éditeur de commandes, se déplacer à la racine du projet (commande `cd [chemin]`), qui contient le fichier test.jar. Lancer la commande `java -jar test.jar`. Pour que cette commande fonctionne, java doit être correctement configuré sur la machine cible.

1.3 Génération de l'exécutable .jar avec Eclipse

Ouvrir le projet sous Eclipse puis naviguer dans les menus File, Export, Java, Runnable JAR File. Une fenêtre s'ouvre, choisir la configuration de lancement (cf exécution du projet sous eclipse), et l'emplacement du fichier cible. Enfin valider ces options avec Finish, ne pas tenir compte des avertissements sur la classe Simulation. Exécution d'une instance

Dans le dossier ressources/input/instance/ se trouve le fichier de configuration de l'instance (instance.properties). Les diverses propriétés sont renseignées en commentaires. Il est par

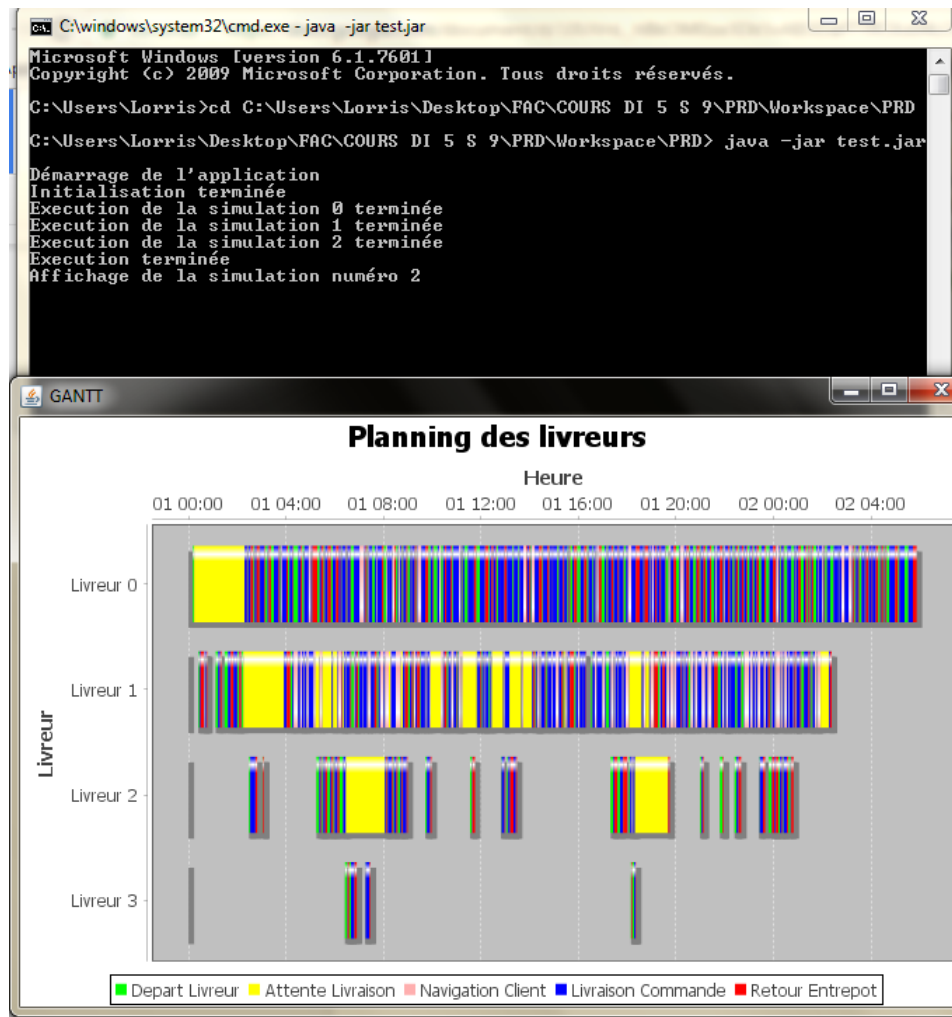


Figure 1 – Exécution de l'application en ligne de commandes

exemple possible de choisir le nombre d'exécution de simulations ou encore le dossier qui comportera les fichiers de configuration de chaque exécution. Ces fichiers se trouvent par défaut dans ressources/input/simulations. Chaque nom de fichier doit suivre le modèle suivant : simulation + numeroExecution + .properties, par exemple simulation0.properties pour la première exécution. De même que pour instance.properties, les paramètres de ces fichiers sont expliqués en commentaires.

Une fois l'exécution d'une instance terminée, des fichiers de logs sont générés dans ressources/output/simulation[numeroExecution].

1.4 Affichage du GANTT d'une exécution

Il est possible d'afficher un GANTT d'une exécution particulière. Pour cela il faut modifier le paramètre `affichageGanttNumero` du fichier d'instance, (ressources/input/instance/instance.properties) en lui fournissant le numéro de l'exécution à afficher (0 pour la première instance).

Ceci fait, il faut lancer l'exécution du projet.

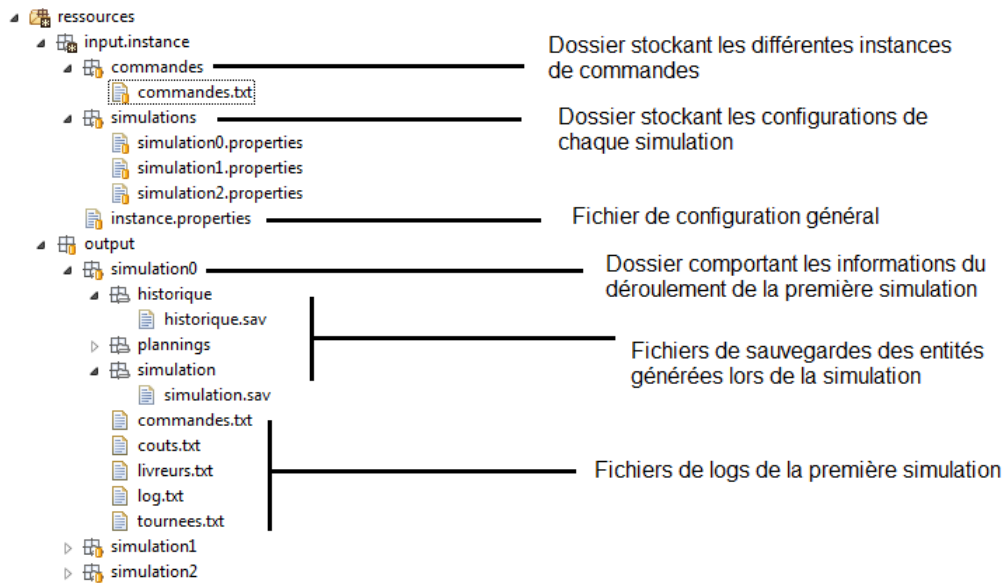


Figure 2 – Arborescence du répertoire de ressources du projet

```
instance.properties
1# Nb de simulations à effectuer
2nbSimulations=3
3# Simulations Identiques ?
4identique=false
5# Répertoire des configurations des simulations
6repertoireSimulationsProperties=ressources/input/instance/simulations/
7# Numéro de simulation à afficher :
8# -1 ---> pas d'affichage
9# 0 ---> première simulation
10# i ---> i + 1 ième simulation ( i >= 0 , i < nbSimulations )
11affichageGanttNumero=2
```

Figure 3 – Modification d'un des paramètres du fichier de configurations

2 Développement du projet

2.1 Ajout d'une classe de décisions

Pour ajouter de nouvelles règles de décisions pour résoudre le problème d'affectation des tournées, il faut ajouter une nouvelle classe héritant de la classe Decision. Cette nouvelle classe doit absolument être ajoutée au sein du package Decision. Des méthodes seront à implémenter :

- `nouvelleCmd(Commande)` : fonction permettant d'ajouter une commande dans le planning des tournées
- `livreurRetourEntrepot()` : fonction appelée à chaque retour d'un livreur à l'entrepôt, permet de modifier le planning des tournées également
- `planningInit()` : fonction en charge de l'ajout des commandes déjà reçues au lancement de l'exécution dans le planning des tournées

Certaines classes de décision déjà existantes (notamment `MeilleurInsertion` ou `DepartDirect`) peuvent être utilisées comme modèle.

ATTENTION

Pour que l'exécution fonctionne correctement, il faut que toutes les commandes soient insérées dans le planning des tournées. Le cas échéant, les commandes non-insérées ne seront jamais

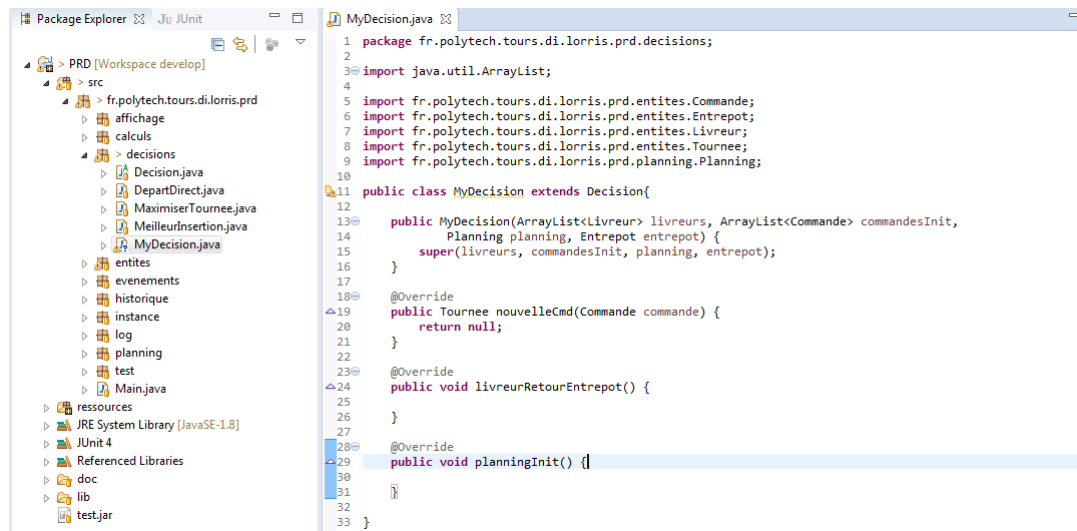


Figure 4 – Affichage d'une classe de décision personnalisée

livrées.

Si une nouvelle classe est ajoutée, il est nécessaire de générer un nouvel exécutable .jar pour l'exécution du projet en ligne de commande.

Simulation d'un problème de tournées avec des commandes rapides

Lorris Pigeon

Objectifs : Yannick Kergosien, Jorge Mendoza et Mustapha Haouassi

Lors d'un achat en ligne, la durée de livraison peut-être un facteur. La livraison en une heure dans les capitales est à l'étude par une société de commerce. Dans le cadre de ce projet, une simulation du système permettrait d'avoir une rapide estimation de la rentabilité d'une telle livraison.

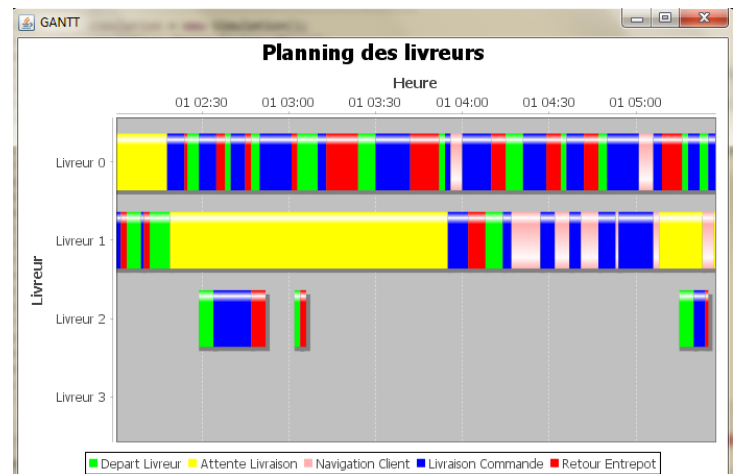
- Modélisation et développement d'une simulation
- Tests de règles de décision
- Affichage graphique du système

Mise en œuvre

Dans un premier temps, il sera nécessaire de développer le moteur de la simulation. Ensuite, des tests permettront d'en vérifier le bon fonctionnement. Enfin, le développement de l'affichage graphique achèvera ce projet.

Résultats attendus

Les données statistiques générées par le moteur de la simulation ainsi que l'affichage graphique du déroulement de la simulation permettront d'évaluer pour la même instance d'un problème deux résolutions différentes.



Laboratoire d'Informatique
EA 6300

Simulation d'un problème de tournées avec des commandes rapides

Lorris Pigeon

Encadrement : Yannick Kergosien, Jorge Mendoza et Mustapha Haouassi

Objectifs

Lors d'un achat en ligne, la durée de livraison peut-être un facteur. La livraison en une heure dans les capitales est à l'étude par une société de commerce. Dans le cadre de ce projet, une simulation du système permettrait d'avoir une rapide estimation de la rentabilité d'une telle livraison.

- Modélisation et développement d'une simulation
- Tests de règles de décision
- Affichage graphique du système

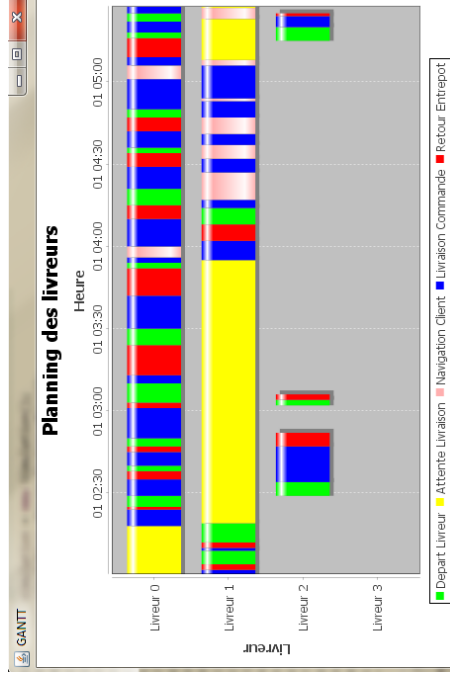


Mise en œuvre

Dans un premier temps, il sera nécessaire de développer le moteur de la simulation. Ensuite, des tests permettront d'en vérifier le bon fonctionnement. Enfin, le développement de l'affichage graphique achèvera ce projet.

Résultats attendus

Les données statistiques générées par le moteur de la simulation ainsi que l'affichage graphique du déroulement de la simulation permettront d'évaluer pour la même instance d'un problème deux résolutions différentes.



Laboratoire d'Informatique
EA 6300



Simulation d'un problème de tournées avec des commandes rapides

Résumé

A partir d'une instance d'un problème de livraisons rapide, une simulation va tester différentes stratégies décisionnelles et fournir des statistiques les évaluant. Ensuite, à partir des données générées par la simulation, un affichage graphique permettra d'avoir un compte-rendu des différentes étapes. Le projet porte sur le développement de la simulation ainsi que de son affichage.

Mots-clés

simulation, recherche opérationnelle, problème de tournées, évaluation de règles de décision, decision rules evaluation

Abstract

Starting from the instance of an express delivery problem, a statistical evaluation will be operated on several decision rules by a simulation. Then, a graphical display from the datas generated by the simulation will provide a general view of the events and their resolution. This project aims to develop the simulation and the display.

Keywords

simulation, operational search, vehicles routing problem

Tuteurs académiques

Yannick KERGOSIEN

Jorge MENDOZA

Mustapha HAOUASSI

Étudiant

Lorris PIGEON (DI5)