University of Bordeaux 1
Science and Technology
Computer Science specialty

Second year of Master - Study and Research Project

Mobility Models for UAV Group Reconnaissance Applications

MEMORY

Customers : AUTEFAGE Vincent and CHAUMETTE Serge

Responsible of Directed Works : AUTEFAGE Vincent and CHAUMETTE Serge

Authors : CASTAGNET Florian, ETCHEVERRY Jérémy, PAZIEWSKY Hayley, TESSIER Alexis & TESTA Mickael

Contents

1 Context **6**

2 Analysis Of Existing **8**
2.1 Summary of the problem 8
2.2 Existing models 8
2.2.1 Random Walk 9
2.2.2 Random Waypoint 9
2.2.3 City section 10
2.2.4 Natural Agent 11
2.2.5 Birds and Fish 12
2.2.6 Wolves 12

3 Scenarios **13**

4 Needs Analysis **14**
4.1 Functional Requirements 14
4.2 Non-Functional Requirements 14

5 Architecture **15**

6 Schedule **16**
6.1 Tasks List 16
6.2 PERT 16
6.3 GANTT 16

7 Works Done **17**
7.1 What we did 17
7.1.1 random model 17
7.1.2 Pheromone model 20
7.2 Difficulties Encountered & Solutions 22

8 Results **23**

9 Tests **24**

10 Improvements **25**

11 Development Environment and Conventions **26**
11.1 Development Environment 26
11.2 Programming Conventions 26

12 Conclusion **28**

List of Figures

Introduction

Dans le cadre de notre formation de Master Informatique à l'Université de Bordeaux 1 et notamment dans la matière : projet d'étude et de recherche, nous devions étudier un article de recherche par groupe de 5 personnes. Chaque groupe devait choisir un sujet de recherche parmi ceux proposés. Notre groupe a choisit un sujet en rapport avec les drones. Le titre de cet article est *Mobility Models for UAV Group Reconnaissance Applications* écrit en collaboration par *Erik Kuiper* et *Simin Nadjm-Tehrani* publié en 2006. Les clients qui nous ont proposé d'étudier cet article sont messieurs Serge Chaumette et Vincent Autefage. Notre chargé de TD est monsieur Pascal Desbarats.

Ce projet, d'une durée de 3 mois, nous a permis de voir le cheminement de l'étude d'un article de recherche, de sa lecture en anglais, à l'implémentation d'un algorithme, tout en passant par une étude de l'existant. Le but de cette matière est donc de nous faire étudier un article de recherche, d'en extraire un algorithme et de l'implémenter. Nous devons donc étudier un modèle qui respecte les propriétés aérodynamiques des drones.

As part of our Master Computer formation at the University of Bordeaux 1 particularly in the matter Study and Research Project, we had to study a research article in groups of 5 people. Every group had to choose a subject of research among those proposed. Our group chose a topic related to the drones. The title of this article is *Mobility Models for UAV Group Reconnaissance Applications* written in collaboration by *Erik Kuiper* and *Simin Nadjm-Tehrani* published in 2006. The customers who suggested us studying this article are misters Serge Chaumette and Vincent Autefage. Our teaching assistant is mister Pascal Desbarats.

This project, for a period of 3 months, we were able to see the progress of the study of a research article, its reading in English, with the implementation of an algorithm, while going through a study of the existing. The aim of this material is to make us study a research article, to extract an algorithm from it and to implement it. We must therefore study a model that meets the aerodynamic properties of drones.

Plusieurs modèles de mobilité existent et se base le plus souvent sur des éléments du monde réel. Cet article précisément étudie deux modèles de mobilité qui sont le Random Mobility Model et le Distibuted Pheromon Model.

Nous avons donc essayé de donner la meilleure image possible de notre projet à travers ce

mémoire. Il représente chaque étapes de celui-ci, de l'étude de l'existant à l'implémentation d'un algorithme. Il montre également notre organisation ainsi que les multiples tests que nous avons établis et réalisés.

This article specifically discusses two mobility models that are Random Mobility Model and distibuted Pheromon Model.

We tried to give the best possible image of our project through this memory. It represents each stage thereof, the study of the existing in the implementation of an algorithm. It also shows our organization as well as the multiple tests which we established and realized.

Chapter 1

Context

Un drone est un aéronef inhabité télécommandé à distance ou autonome. Il est équipé de différents capteurs (accéléromètre, gyroscope, magnétomètre, caméra, etc.) et ceux-ci lui permettent de se mouvoir dans son environnement.

Les drones aujourd'hui sont de plus en plus présents sur le territoire français et partout dans le monde. Il y a différents types de drones avec différentes missions. Par exemple, certains drones font de la surveillance (de bâtiment, de monuments historiques ...), d'autres sont destinés à collecter des renseignements, d'autres ont des usages militaires ou encore de transport. Deux types de drones sont présents actuellement : les drones dit civils, et les drones militaires. Dans le cadre militaire, les drones peuvent être utilisés pour pénétrer des zones qui peuvent être trop dangereuse pour l'homme. Ils peuvent notamment cartographier des zones pour faire du repérage de zone.

De nombreuses recherches sont dédiées à ces appareils depuis de nombreuses années, notamment sur la coopération et la communication de plusieurs drones entre eux. Ces flottes de drone permettent d'accomplir des tâches le plus efficacement possible. Il faut donc savoir comment ils vont bouger ensemble, éviter les collisions, éviter qu'ils fassent deux fois les mêmes tâches etc... Ces différentes contraintes peuvent être résolues grâce à des modèles de mobilités basés sur la communication.
De nombreux sujets de recherche sont dédiés à ces modèles de mobilité dans le cadre des flottes de drones. Ils étudient les différentes façons qu'on les drones pour se déplacer dans leur milieu tous ensemble.

Le but de notre article est de scanner une aire définit auparavant le plus rapidement possible et si possible une fois par heure. La problématique de cet article est donc : **How well scan an area? As much as quickly possible, in a limited time and at least, once every hour**.

A drone is an unmanned aircraft remote-controlled or autonomous. It is equipped

with different sensors (accelerometer, gyroscope, magnetometer, camera, etc..) And they allow it to move in its environment.

Today, UAVs are increasingly present on French territory and around the world. There are different types of drones with different missions. For example, some drones are monitoring (building, historical monuments ...), others are designed to collect information, others have military uses or transport.Two types of UAVs are currently present: the drones said civilian and military drones. In the military context, UAVs can be used to penetrate areas that may be too dangerous for humans. They may include mapping areas for the tracking area.

Much research has been dedicated to these devices for many years, including on cooperation and communication between them several drones. These fleets of drone used to perform tasks as efficiently as possible. We need to know how they will move together, collision avoidance, avoid them to do twice the same work etc ... These constraints can be resolved through mobility models based on communication.
Many research topics are dedicated to these types of mobility within fleets drones. They study the different ways that the drones to move in their environment all together.

The purpose of this article is to scan an area previously defined as soon as possible and if possible once per hour. The problem of this article is: **How well scan an area? As much as Quickly as possible in a limited time and at least, once every hour**.

Chapter 2

Analysis Of Existing

Comme dit auparavant, les flottes de drones sont régis entres elles par des modèles de mobilités. Notre étude de l'existant s'est donc décomposée en plusieurs parties.

As said before, fleets drones are governed by these models mobility. Our study of the current is therefore divided into several parts.

2.1 Summary of the problem

La problématique de notre article est de scanner une zone le plus efficacement possible et le plus rapidement possible, au maximum une fois par heure. Cette zone sera scannée par une flotte de drone qui devront coopérer entre eux et communiquer.

The issue of this paper is to scan an area as efficiently as possible and as quickly as possible, at most once per hour. This area will be scanned by a fleet of drones that will cooperate and communicate.

2.2 Existing models

De nombreux modèles de mobilité existent avec chacun des caractéristiques différentes. Tous ces modèles permettent de définir des mouvements à des flottes de drones. La plupart des modèles sont basés sur des situations de la vie réelle comme la cartographie routière, ou encore sur le comportement de nombreux animaux (fourmis, oiseaux, thermites etc). Nous allons vous en exposer certains que nous considérons les plus importants et les plus intéressants pour notre cas.

Many mobility models exist each with different characteristics. All these models are used to define movements fleets drones. Most models are based on real-life situations

such as road maps, or on the behavior of many animals (ants, birds, termites, wolves etc). We will expose some of which we consider the most important and most interesting for our case.

2.2.1 Random Walk

First described by Einstein in 1926 [29], it was developed to mimic the extremely unpredictable movement of many entities in nature [9]. An MN moves from its position to a new one by randomly choosing a direction and speed choosen by pre-defined ranges, [speedmin, speedmax] and [0, 2PI]. At the end of a constant time interval t or a constant distance traveled d, a new speed and direction are calculated. If a MN during his travel reaches a simulation boundary, it « bounces » off the simulation border with an angle determined by the incoming direction and continues to this new path. It's a memoryless mobility pattern because it retains no knowledge of its old locations and speed values [19]. Therefore, the current position and speed of a MN is independent of its past location and speed. This can generate unrealistic movement because of sudden stops and sharp turns. If the specified time or distance of a MN motion is short, the area of the simulation will be small.

METTRE DES SCHEMA ET DECRIRE PLUS

2.2.2 Random Waypoint

Includes pause times between changes in direction and/or speed.Once this time expires, the MN chooses a random destination and a speed, that is uniformly distributed between [minspeed, maxspeed]. The movement pattern of a MN who uses the RWaypointMM is similar to the RwalkMM if pause time is zero and [minspeed, maxspeed] = [speedmin, speedmax]. During a performance investigation, MNs are initially distributed randomly around the simulation area. Figure 4 shows the average MN neighbor percentage of the MNs. For example, if there are 50 MNs in the network and a node has 10 neighborns, then the node's current neighbor percentage is 20%. Moreover, during the first 600 seconds, due to initially distributed randomly of the MNs around the simulation area, there is an high variability in the average MN neighbor percentage. In the paper, there is presented three possible solutions to avoid this initialization problem. The first is to save the location of the MNs after the initial high variability and use this position as the initial starting point of the MNs in all future simulations. Second, initially distribute the MNs in a specific area to be a distribution more common the model, like fore example, initially placing the MNs in a triangle distribution. Lastly, discard the initial 1000 seconds of simulation time in each simulation trials, to ensure thatthe initialization problem is removed even if the MNs move slowly. But if the MNs move fastly, we can discard fewer seconds of simulation time. This third solution ensures that each simulation has a random initial configuration. Figure 5 : thre is a complex relationship between node speed and pause time. A scenario with fast MNs and long pause times actually produces a more stable networks than a scenario with slower MNs and shorter pause times. Hence, long pause times(over 20 seconds) produce a stable network even at high speeds.

METTRE DES SCHEMA ET a revoir

2.2.3 City section

Le but de ce modèle est de proposer un modèle réaliste. Il se base sur le mouvement de véhicule sur des cartes routières. La spécificité de ce modèle est que la carte dépend de chaque zone géographique. Ici, les véhicules sont des nœuds mobiles.

The aim of this model is to propose a realistic model. It bases itself on the movement of vehicle on road maps. The specificity of this model is that the map depends on every geographical zone. Here, vehicles are mobile nodes.

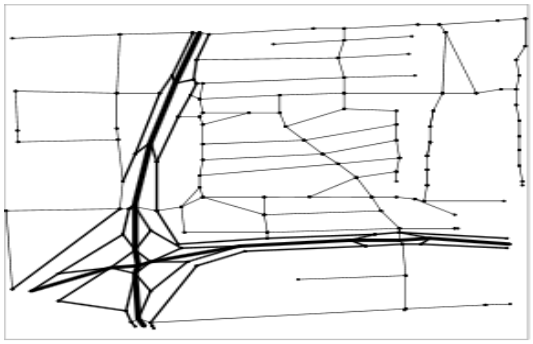
Les données cartographique des rues sont disponibles auprès du bureau américains du recensement. Les informations concernant les cartes sont récupérées dans des fichiers textes. Ces fichiers sont composés de plusieurs éléments : l'identifiant unique pour une route spécifiée. la nature de la route : Cela peut être une autoroute ou une rue. la longitude de départ. la latitude de départ la latitude d'arrivé. la longitude d'arrivé. Toutes les intersections de la carte sont représentées par des nœuds non mobile. Ce modèle prend également en compte l'intensité du trafic routier, plus le trafic est important et plus le trait représentant la route sera gros. Ce modèle est aussi sensible à la vitesse des voitures (en fonction notamment de l'heure). Par exemple, nous pouvons considérer que la vitesse de circulation des véhicules le matin est beaucoup moins importante qu'en début d'après-midi. Voici le déroulement de ce modèle. Chaque nœud commence à un point choisi aléatoirement et sa destination est également choisie aléatoirement. La stratégie de chaque nœud est de rechercher le plus court chemin jusqu'à la destination. Pour accomplir cela, l'algorithme de Dijkstra est utilisé. Plusieurs paramètres rentrent en compte dans cette algorithme comme la vitesse, le trafic. Ce trajet peut être changé dynamiquement. Quand un nœud a atteint sa destination, il recommence tout le processus décrit précédemment. Concernant la vitesse du nœud, il est limité à plus ou moins 5% de la vitesse inscrite dans le fichier.

The mapping data blocks are available from the office Americans of the census. The map information is retrieved from text files. These files are composed of several elements:

- The unique identifier for a specified road.
- The nature of the road: it can be a highway or a street.
- The longitude of the start of position.
- The latitude of the start of departure
- The latitude of arrived.
- The longitude of arrived.

All the intersections of the map are represented by nodes not mobile. **This model also takes into account the volume of road traffic, the more the traffic is important and the more the line(feature) representing the road will be big. ???** This model is also sensitive at the speed of cars (depending in particular on the hour). For example, we can consider that the speed of vehicular traffic in the morning is much smaller than at the beginning of the afternoon.

Here is the progress of this model. Every node begins in a point chosen randomly and its destination is chosen also randomly. The strategy of every node is to look for the shortest way until the destination. To achieve it, the algorithm of Dijkstra is used. Several parameters bring in in account in this algorithm as the speed, the traffic. This route can be dynamically changed. When a node reached its destination, it begins again all the process described previously. Concerning the speed of the node, it is limited to more or less 5% of the speed registered in the file.
For example, here is a representation of the model. We see here that the traffic is little sparse, but with very different speeds.



(b) Region *B*: Street scenario corresponding to a square area of size 1900 m×1900 m

METTRE DES SCHEMA ET a compléter

2.2.4 Natural Agent

Ants

Le modèle de mobilité des fourmis se base sur le fait que les fourmis construisent des réseaux de sentiers qui relient leurs nids avec des sources de nourritures disponible. Chaque fourmi qui se nourrit à le même programme :

- Elle doit éviter les obstacles.
- Aller ou elle veut (déplacement aléatoire) tant qu'il n'y a pas de phéromones.
- Si elle arrive à trouver de la nourriture, elle laisse des phéromones pendant un temps t pour pouvoir indiquer aux autres fourmis l'emplacement de la nourriture.
- Si la fourmi trouve de la nourriture, elle l'a ramène à son nid.

Les fourmis peuvent mourir. Soit elles ne trouvent pas de la nourriture assez rapidement et meurt de faim, soit elles se font tuer par d'autres prédateurs. Tous les chemins de phéromones conduisent à de la nourriture. Mais les phéromones s'évaporent dans le temps. Le taux de phéromone peut être renforcé si plusieurs fourmis passent par le même chemin.

METTRE DES SCHEMA ET a compléter

Termites

Wasps

le modèle basé sur les guêpes est composé de plusieurs caractéristiques :

- un chef qui répartit les différentes guêpes dans différents groupes
-

2.2.5 Birds and Fish

2.2.6 Wolves

Chapter 3

Scenarios

Chapter 4

Needs Analysis

4.1 Functional Requirements

4.2 Non-Functional Requirements

Chapter 5

Architecture

Chapter 6

Schedule

6.1 Tasks List

6.2 PERT

6.3 GANTT

Chapter 7

Works Done

7.1 What we did

Durant les deux mois et demi de ce projet, nous avons implémenté les deux modèles qui étaient décrit dans l'article. Pour rappel, ces deux modèles sont un random simple et un modèle concernant des phéromones. Dans cette rubrique, nous allons vous expliquer le travail de développement que nous avons fourni et les différents choix effectués. Nous n'avons pas trouvé d'utilité à faire un diagramme UML vu la simplicité de l'architecture.

During two and a half months of this project, we implemented both models which were described in the article. As a reminder, these two models are a simple random and a model concerning pheromones. In this section, we are going to explain you the work of development which we supplied and the various made choices. We did not find utility to make a UML diagram seen the simplicity of the architecture.

7.1.1 random model

Pour implémenter ce modèle, nous avons eu besoin de trois classes.
La première classe, qui se nomme Main.java, permet comme son nom l'indique de lancer le programme. C'est dans cette classe que nous définissons la taille de la fenêtre, la matrice qui servira à simuler le scan d'une zone (tous les noeuds auront la même matrice), et l'instanciation des différents objets provenant de la librairies JBotSim. Nous avons par exemple, un objet s'intitulant *topo* qui est une topology. Cet objet est la base pour notre simulateur. C'est à cette topology que l'on va pouvoir ajouter des nœuds mobiles (ici des avions).

To implement this model, we needed three classes.
The first class, which is called *Main.java*, allows as its name indicates it launching the program. It is in this class that we define the size of the window, the matrix which will serve to feign the scan of a zone (all the nodes will have the same matrix), and the instanciation of the various objects coming of JBotSim library. We have for example, an

object being entitled *topo* which is a topology. This object is the base(basis) for our simulator. It is to this topology that we are going to be able to add mobile nodes (here planes).

Nous avons aussi instancié un objet de type Jtopology. Nous expliquerons ce choix dans la suite de la présentation des classes.

La deuxième classe de notre architecture se nomme *MovingNode*. C'est dans cette classe que sera traité le mouvement des différents nœuds. Cette classe hérite donc de la classe *Node* et implémente l'interface *ClockListesner*, toutes les deux provenant de la librairie JBotSim. Elle hérite de la classe *Node* car les avions seront représentés par des nœuds mobiles, il nous faut donc les caractéristiques de ceux-ci. De plus, elle implémente l'interface *ClockListener* car nous avons besoin que nos nœuds bougent tout les certains temps, on a donc besoin d'un clock d'horloge pour réaliser cela. Nous aurons donc un listener pour le temps, ainsi qu'une méthode *onClock* qui sera appelée à chaque top d'horloge.

We also instantiated an object Jtopology's object type. We shall explain this choice during the presentation of classes.

The second class of our architecture is called *MovingNode*. It is in this class that will be handled the movement of the various nodes. This class thus inherits from the class *Node* and implement the interface *ClockListesner*, both resulting(coming) from the JBotSim. It inherits from the class *Node* because planes will be represented by mobile node, we thus need the characteristics of these. Furthermore, It implements the interface *ClockListener* because we need that our nodes move all the seconds, we thus need a top of clock to realize it. We shall thus have a listener for time , as well as a method *onClock* which will be called to every top of horloge.

Cette classe contient un constructeur par défaut, permettant d'instancier et d'initialiser les différents variables, et d'associer des images aux noeuds mobiles (appel de la méthode *setProperty*. Le modèle random a pour caractéristique de n'avoir aucune communication entre les différents noeuds. Pour réaliser cela, nous passons un argument valant 1 à la méthode *setCommunicationRange*.

Comme indiqué dans l'article, les noeuds du modèle random changent d'actions toute les secondes en fonctions de la dernière action entreprise. Pour rappel, voici la table d'action pour le modèle random :

This class contains a constructor by default, allowing to instantiate and to initialize the various variable, and to associate images with the mobile nodes (call of the method *setProperty*. The random model has for characteristic to have no communication between the various nodes. To realize it, we cross a being worth argument 1 in the method *setCommunicationRange*.

As indicated in the article, the nodes of the random model change actions quite the seconds in functions of the last undertaken action. As a reminder, here is the table of action for the random model:

Table 1. UAV random action table.

	Probability of action		
Last action	**Turn left**	**Straight ahead**	**Turn right**
Straight ahead	10%	80%	10%
Turn left	70%	30%	0%
Turn right	0%	30%	70%

Dans notre implémentation, nous avons donc une variable *lastdirection* permettant de savoir la dernière action qui a été faite. Nous avons associé des chiffres aux dernières actions de faite. Si la dernière direction était la gauche, alors la variable *lastdirection* prend la valeur 1, si elle était tout droit, alors on lui associe la valeur 2, et si c'était à droite, on lui associe la valeur 3.

In our implementation, we have a variable *lastdirection* allowing to know the last action which was made. We associated figures with the last actions of made. If the last direction was the left, then the variable *lastdirection* takes the value 1, if it was quite right, then we associate the value 2, and if it was to the right, we associate the value 3.

Pour respecter le plus possible la réalité, nous avons dû modifier la gestion des bords de la fenêtre. En effet, à l'origine dans JBotSim, les noeuds peuvent passer d'un bord à l'autre. Par exemple, s'ils atteignaient le bord gauche, ils se retrouvaient par la suite à droite de la fenêtre, ce qui n'est pas possible dans la vie réelle. Nous avons donc bloqué ce problème en rectifiant les coordonnées du noeud, et en changeant son angle de direction.

To respect as much as possible the reality, we had to modify the management of the edges of the window. Indeed, originally in JBotSim, nodes can pass from an edge to the other one. For example, if they reached the left edge, they found themselves afterward to the right of the window, what is not possible in the real life. We blocked this problem by rectifying the position of the node, and by changing its angle of direction.

A chaque fois qu'un noeud scanne une zone, il regarde dans la matrice si cette position a déjà été scanné (0 si la zone n'a jamais été scanné, 1 sinon). Si c'est le cas, alors il ne fait rien, sinon il met la valeur à 1.

La troisième classe de notre architecture se nomme *Jtopology_Random*. Elle hérite de JTopology et va permettre de dessiner les traces des scans sur la map. Cette classe JTopology hérite de JPanel, et se situe en fait entre une JViewer et une Topology.

Every time a node scans a zone, it looks in the matrix if this position was already scanned (0 if the zone was never scanned, 1 otherwise). If it is the case, then it makes nothing, otherwise it puts the value to 1.

The third class of our architecture is called *Jtopology_Random*. It inherits from JTopology and is going to allow to draw the tracks of scans on the map. This class JTopology inherits from JPanel, and is situated in fact between a JViewer and a Topology.

7.1.2 Pheromone model

Au niveau architecture, l'implémentation du modèle des phéromones est très similaire. On retrouve pratiquement les 3 même classe. Seulement la classe *MovingNode* va réellement être différente car les mouvements des noeuds vont dépendre de beaucoup de paramètre. Tout d'abord, cette classe implémente une interface de plus qui est *MessageListener*. En effet, dans ce modèle, les noeuds vont devoir se transmettre des données (en l'occurrence leur map respective).

La méthode qui va changer le plus par rapport au modèle aléatoire est la méthode *onClock*. En effet, c'est ici que l'on va appliquer les règles comme décrites sur l'image suivante :

Concerning the architecture, the implementation of the model of pheromones is very similar. We find practically 3 even classy. Only the class *MovingNode* is really going to be different because the movements of nodes are going to depend on a lot of parameter. First of all, this class implements an interface furthermore which is *MessageListener*. Indeed, in this model, nodes are going to have to be passed on data (in this particular case their respective map).

The method which is going to change most with compared with the random model is the method *onClock*. Indeed, it is here that we are going to apply rules as described on the following image:

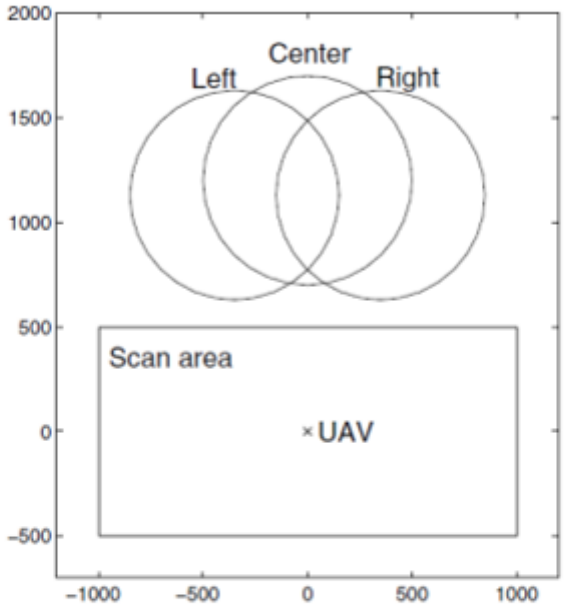


Figure 2. Pheromone search pattern

Table 2. UAV pheromone action table.

Probability of action		
Turn left	**Straight ahead**	**Turn right**
(Total – Left) / (2 * Total)	(Total – Center) / (2 * Total)	(Total – Right) / (2 * Total)

Nous avons traité tous les cas décrits dans l'article, c'est à dire que le noeud regarde les valeurs des phéromones à sa gauche, en diagonale gauche, devant lui, en diagonale droite et à droite.

Une autre caractéristique de notre implémentation est que cette fois-ci, nous ne scannons plus un point de la carte mais les 5 décrits précédemment.

We handled all the cases described in the article, that is the node looks at the values of pheromones at its left, in left diagonal, in front of him, in right diagonal and in right.

Another characteristic of our implementation is that this time, we do not scan any more a point of the map but 5 described previously.

A COMPLETER

7.2 Difficulties Encountered & Solutions

Au niveau de l'implémentation, nous n'avons pas rencontré beaucoup de difficultés. Le seul gros problème est l'affichage des zones scannées. Sur les conseils de Monsieur Casteigts, nous avons dû créer une classe héritant de JTopology et redéfinir la méthode *paint*. En effet, le rafraichissement de l'affichage des zones scannées est un problème car nous perdions les zones scannées précédemment. Nous avons donc créé une ArrayList pour sauvegarder les zones scannées pour les redessiner à chaque fois.

La plus grosse difficultés rencontrées a été la lecture des nombreux articles en correspondance avec notre article d'étude. Ils comportaient de nombreuses pages et la lecture de ceux-ci en Anglais nous a prit énormément de temps. Nous avons dû par la suite mettre à jour les articles lus en recherchant sur internet les mises à jour.

Concerning the implementation, we did not meet many difficulties. The only big problem is the display of the scanned zones. On the advice of Mister Casteigts, we had to create a class inheriting from JTopology and redefine the method *paint*. Indeed, the refreshment of the display of the scanned zones is a problem because we lost zones scanned previously. We created a ArrayList to save area scanned to redraw them in every times.

Chapter 8

Results

Chapter 9

Tests

Chapter 10

Improvements

Chapter 11

Development Environment and Conventions

In this part, we will see the development environment we used, and the programming conventions we've applied.

11.1 Development Environment

- GitHub
- ...

11.2 Programming Conventions

First, we've choosed to used the Java Coding Conventions, which we can see in the following link - http://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Then, in order to create our documentation, we've used the Doxygen Convention, viewable here - `http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html` We've relied heavily on this documentation and have mostly employed these protocoles below.

Doxygen Convention for classes

```
/**
* @class name of the class
* @brief Description of herself
*/
```

Doxygen Convention for methods

```
/**
* @brief Description of the method
* @param the parameters and their descriptions
* @return the description of what return the method (optional)
```

```
*/
```

Doxygen Convention for members

```
int var; /**< Detailed description after the member */
```

We've also resorted to programming conventions that we defined between us.
For example, when a part of a code, was not finished yet, we put the following lines above the concerning part.

Programming convention for unfinished code

```
/**
* @TO_DO
* Description
*/
```

We've also used a convention for the bugs found and wrote these protocoles, depending on whether the bug was resolved or not.
We used it, in line with the Bug Tracking of GitHub.

Programming convention for bugs

```
/**
* @BUG
* @Unfinished/finished
* Description
*/
```

To finish, we've created the five essential files for a project :

- INSTALL.txt : Installation instructions for the project,
- LICENCE.txt : Licence and copyright © of the project,
- README.txt : General description of the project,
- AUTHORS.txt : Authors of the project,
- MANIFEST.txt : Tree structure and files list of the project.

Chapter 12

Conclusion