Coordination de systèmes multi-agents Techniques et expérimentations dans le cadre de la RoboCupRescue

ing. B. SOKAY
ECAM – Bruxelles

Durant les situations de crise, la capacité de coordination entre les services de secours conditionne considérablement les chances de survie des victimes. La simulation de systèmes multi-agents offre désormais une plateforme de tests permettant l'évaluation de l'efficacité de cette coordination, de façon prévisionnelle. Cette étude présente le développement d'une stratégie de coordination d'agents logiciels appliquée à la simulation RoboCupRescue.

Mots-clefs : Systèmes multi-agents, agent logiciel, coordination, stratégie RobocupRescue.

During the crisis situations, the capacity of coordination between the first-aid organizations conditions the chances of survival of the victims considerably. Now, the multi-agents systems offer new platforms of tests allowing the evaluation of the coordination effectiveness, in an estimated way. This study presents the development of a strategy of software agents coordination applied to RoboCupRescue simulation.

Keywords : Multi-agent systems, software agent, coordination, strategy, RoboCupRescue.

1. Introduction

La science des systèmes multi-agents est un domaine de l'intelligence artificielle moderne qui ouvre la porte à la résolution de nombreux problèmes. La prise de décisions en situation d'urgence fait partie de ces derniers. Dans une telle situation, l'être humain est soumis à de fortes pressions, qui peuvent influencer considérablement son raisonnement et sa prise de décisions. L'ordinateur et sa logique ne sont pas soumis à ce genre de contraintes. Leur utilisation dans l'optique de remplacer ou d'assister la prise de décision humaine se justifie donc pleinement.

En situation d'urgence, les conséquences de mauvais choix peuvent très rapidement interagir pour finalement plonger le système dans un état chaotique irrémédiable. De plus, ce problème devient d'autant plus complexe lorsque la situation implique plusieurs intervenants. Le besoin de coordination devient alors indispensable afin de pouvoir optimiser l'efficacité des différentes décisions et actions entreprises tout en contrôlant leurs interactions.

Le présent article débute par une brève présentation des concepts liés aux systèmes multi-agents et à leurs spécificités. Il se poursuit par une définition rigoureuse du concept de coordination et par la présentation d'une élaboration de stratégie de coordination appliquée aux agents logiciels de la simulation exploitée dans le cadre de la compétition RoboCupRescue.

La fédération RoboCup, organisation internationale soutenant le développement des milieux de l'intelligence artificielle et de la robotique, organise en effet chaque année cette compétition qui propose à ses différents participants de produire une solution multi-agents permettant la coordination des forces de secours au sein d'une ville touchée par un séisme. Le simulateur fourni pour cet effet s'avère donc être un outil de validation très efficace de la stratégie de coordination élaborée.

Cette dernière à proprement parler a été établie afin de comparer l'intérêt de solutions favorisant la prévention de situations problématiques au cours de la simulation. Elle repose plus précisément sur des concepts de regroupement (clustering) et des principes d'optimisation de parcours assurant une planification efficace des tâches des agents.

2. Agent et Systèmes multi-agents

2.1 Définition de l'agent logiciel

La position d'un agent au sein d'un système multi-agent peut être comparée de manière métaphorique à la position d'une personne agissant au sein d'une société. Celle-ci est capable d'un certain nombre d'actions et possède des attributs qui lui permettent d'obtenir des informations sensorielles sur l'environnement qui l'entoure.

Un agent peut donc être grossièrement vu comme une entité virtuelle intelligente et indépendante, située dans son propre monde et capable d'interagir avec son environnement et ses semblables.

Une définition plus rigoureuse spécifie l'agent logiciel comme étant « *un système informatique, situé dans un environnement et qui agit d'une façon autonome et flexible pour atteindre les objectifs pour lesquels il a été conçu* ». [1]

L'agent logiciel est donc par définition un programme informatique situé dans un cadre où s'exécutent d'autres programmes pouvant également être des agents. Ce programme a la capacité de réaliser des actions pouvant influencer l'environnement suivant les informations qu'il en reçoit. Il est également capable d'échanger des services avec d'autres programmes. Il est finalement autonome, ce qui lui permet de décider et d'agir sans l'intervention d'autrui.

Un système multi-agents est donc en absolu, un système distribué dont les différents modules sont les agents. La particularité de ce système distribué provient essentiellement du fait que les comportements de ses composantes, les agents, sont très souvent peu prévisibles.

Le comportement d'un agent est en effet soumis à de nombreux paramètres. On peut notamment citer :

- L'état de son environnement
- La position de l'agent au sein de l'environnement
- La position de l'agent au sein d'une structure hiérarchique
- L'historique et l'expérience qu'il peut avoir acquis
- Les priorités de ses objectifs face aux contraintes de l'environnement

- La présence d'un ou plusieurs autres agents et leurs influences sur l'environnement
- Les similarités ou divergences d'objectifs des autres agents qui peuvent mener à des coalitions ou à des conflits d'intérêts.

L'agent est donc caractérisé par une certaine flexibilité qui lui permet de percevoir son environnement et d'élaborer une réponse suivant des contraintes temporelles. Il doit également être capable d'afficher un comportement proactif et de saisir les opportunités au bon moment afin de réaliser ses objectifs. Il doit finalement posséder une certaine sociabilité pour lui permettre d'interagir avec les autres agents, lorsque l'exécution de tâches le demande.

2.2 Système multi-agents

Comme présenté précédemment, un système multi-agents est donc un système distribué composé de plusieurs agents logiciels. Ces systèmes bénéficient donc des avantages propres à la résolution distribuée des problèmes tels la modularité, le parallélisme et la redondance.

Les systèmes multi-agents possèdent généralement les quatre caractéristiques suivantes :

- Chacun de ses agents ne possède qu'un point de vue partiel, conséquence directe de leur connaissance limitée de leur environnement et de leur capacité limitée à résoudre des problèmes seuls.
- La multitude d'agents autonomes qui les composent ne leur permet pas d'être totalement contrôlés.
- Les données de ces systèmes sont décentralisées par le fait que chaque agent ne possède qu'une partie des informations.
- Ils fonctionnent selon un principe asynchrone puisque chaque agent du système a la capacité de travailler à son propre rythme.

Finalement, les systèmes multi-agents offrent surtout l'opportunité d'exploiter des principes d'interaction sophistiqués telle la coopération, la négociation et la coordination.

3. Le principe de coordination

3.1 Définition de la coordination

La coordination est un principe souvent utilisé et donc compréhensible pour la plupart des gens. Malgré tout, la présence de coordination n'est pas nécessairement facile à identifier. Son absence et les conséquences qu'elle implique sont bien plus faciles à reconnaître. On se rend bien souvent compte d'un manque de coordination lorsqu'on doit souffrir des répercussions de celui-ci, par exemple lors d'une collision ou d'un retard.
La coordination peut se définir premièrement comme étant « *Le processus gérant les dépendances entre différentes activités »* [2].

Si les actions d'un système multi-agents n'ont aucune dépendance l'une envers l'autre, il n'y aura donc aucune raison d'imposer des contraintes de coordination. La plupart des systèmes multi-agents possèdent malgré tout un certain degré d'interdépendance entre les objectifs et les capacités des différents agents ainsi qu'entre les différentes tâches qu'ils doivent effectuer.

De plus, les agents d'un même système exploitent bien souvent les mêmes ressources, ce qui impose de nouveaux besoins de coordination.
Un principe de coordination efficace doit donc permettre à un agent de déterminer la meilleure manière de concilier la réalisation de ses objectifs propres et ceux de la collectivité.

La coordination peut finalement être définie comme étant « *Le processus qui fait en sorte que les agents agissent ensemble tout en bénéficiant du travail des uns et des autres au travers d'interactions positives (une certaine tâche favorisant ou améliorant une autre tâche), au lieu de se nuire au travers d'interactions négatives (une tâche empêchant ou bloquant une autre). »* [3]

3.2 Les différents principes de coordination

On classe généralement les différents types de coordination suivant trois catégories : la coordination par conventions, celle par communication et celle par apprentissage. [4]

La coordination par conventions est probablement la plus simple et la plus flexible. Par définition, une convention est une loi connue de tous à laquelle chacun adhère. Il existe de nombreux exemples de conventions dans le monde réel. Le code de conduite peut par exemple être vu comme un ensemble de conventions stipulant notamment les obligations de s'arrêter à un carrefour en cas de feu rouge, de respecter les limitations de vitesse, de céder la priorité aux véhicules venant de droite, ou encore de ne dépasser un véhicule que par la gauche.
Les avantages de ce type de coordination sont la simplicité, l'absence ou le peu de communication qu'il impose et le fait qu'il peut être appliqué sans préparation préalable. Ce principe est malgré tout très rigide, demande généralement un travail conséquent de rédaction des conventions, et n'est donc pas très bien adapté aux systèmes de haute complexité.

La coordination par communication est à nouveau une technique humaine très couramment utilisée. Par opposition à la convention, la coordination par communication requiert généralement de la préparation et possède également un certain coût en bande passante.
Ce principe impose également la connaissance d'un langage commun entre les agents et c'est souvent la flexibilité de ce type de langage qui détermine la flexibilité des résultats de coordination.
Dans sa forme la plus simple, la communication peut se résumer à signaler une information. Un système d'échange d'information peut donc s'établir entre les agents. Idéalement on pourrait ainsi permettre à chaque agent d'obtenir le même modèle d'environnement.
Lorsque les agents sont répartis selon une structure organisationnelle, la communication peut alors servir à transmettre des instructions explicites aux agents. La prise de décision est alors centralisée sur un agent décideur ou partagée entre les membres d'un sous-groupe des agents du système.
Suivant le principe de négociation, la coordination par communication permet finalement l'établissement d'une décision commune prise en concertation de tous les agents.

Les méthodes d'apprentissage sont certainement les plus complexes, mais également les principes de coordination les plus flexibles. De plus, elles peuvent facilement être combinées aux principes de convention ou de communication.
Il existe de nombreux intérêts à intégrer un apprentissage au sein d'un système multi-agents. Dans les systèmes complexes, il s'avère souvent

difficile, voire impossible, de pouvoir définir une politique comportementale pour chaque éventualité. De plus si cela reste possible, décrire l'ensemble de ces conventions devient un travail titanesque. L'apprentissage devient dès lors une alternative très intéressante. Plutôt que de tout définir, on initialise l'agent avec une politique simple et on lui laisse apprendre le meilleur comportement à adopter en fonction des situations rencontrées. L'apprentissage s'effectue alors par essais et erreurs ou suivant des algorithmes plus complexes dont la présentation sort du cadre de cet écrit. La coordination par apprentissage est donc une méthode particulièrement adaptée aux systèmes dynamiques où une importante capacité d'adaptation des agents est nécessaire.

4. La simulation de la RoboCupRescue

La compétition RoboCupRescue oriente ses activités autour de simulations et d'expérimentations de conditions post-catastrophiques. L'idée de promouvoir le développement de solutions basées sur l'intelligence artificielle et la robotique provient du grand séisme de Hanshin-Awaji qui frappa Kobe le 17 janvier 1995. On attribue à ce séisme plus de 6 500 morts, au moins 300 000 blessés et la destruction de plus de 80 000 maisons de bois. Suite au manque flagrant de préparation à ce type de catastrophe, la fédération RoboCup en partenariat avec la ville de Kobe décida de créer la compétition RoboCupRescue.

Les participants de cette compétition doivent développer une solution multi-agents de contrôle des forces de génie civil, de pompier et d'ambulancier au sein de conditions post-séisme imposées par un simulateur. Le rôle des participants est donc de définir les stratégies de planification et de coordination des équipes de secours en vue de maximiser le sauvetage de victimes et de minimiser la propagation d'incendies. Le suivi de cette simulation est assuré par une interface graphique livrée avec le simulateur (fig. 1).

En pratique le terme agent ne fait pas nécessairement référence à un individu, mais plutôt à une entité représentant des individus œuvrant de concert pour assumer une fonction. Suivant ce principe, un agent pompier de la simulation représenterait en réalité, une équipe complète de pompiers et leur véhicule. De même, les agents de type centre doivent plutôt être vus comme étant l'entité représentant l'ensemble du personnel habitant ou travaillant au sein du dit bâtiment.

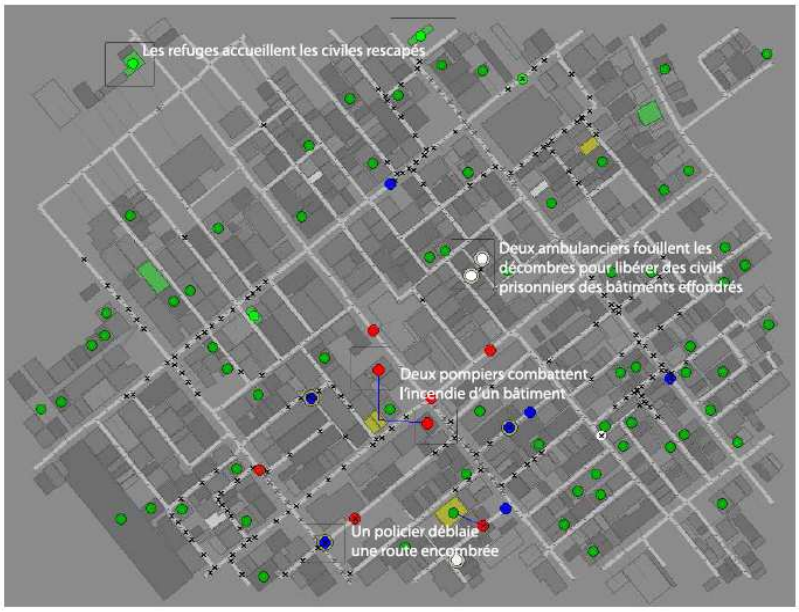

Figure 1 : Exemple de simulation RoboCupRescue

Le rôle des forces de police (●) au sein de la simulation RoboCupRescue s'apparente plus à celui des ouvriers du génie civil qu'à celui des gardiens de la paix. Les différents agents de ce type ont la capacité et les ressources matérielles nécessaires au déblaiement des différentes parcelles de routes encombrées par les débris d'éboulements (+). Ils peuvent assurer la présence d'un minimum de voies praticables que peuvent emprunter les agents des autres types pour lutter contre les incendies (■) et porter secours aux civils (●).

Le rôle des ambulanciers (○) ne se limite pas uniquement à soigner les victimes de la catastrophe et à les rapatrier aux différents refuges (■). Ils sont également chargés des fouilles des différents sites d'éboulement et peuvent entreprendre de libérer les victimes d'enfouissement qu'ils découvrent.

Les escouades de pompiers (●) luttent contre les différents foyers d'incendie afin d'en limiter la propagation. Pour ce faire, chaque escouade est munie d'un camion-citerne de capacité limitée. Une fois vidées, les réserves d'eau peuvent être réapprovisionnées à différentes bouches situées en périphérie des bâtiments refuges.

Chaque service de secours est également pourvu d'un centre possédant des capacités de communication supérieures à celles des simples agents. En effet, afin de simuler les conditions difficiles qu'induit la catastrophe, la communication entre agent a fortement été limitée, d'où le besoin d'exploiter à bon escient ces centres de communication.

5. Besoin de coordination et stratégie développée

Les interactions entre les différents agents sont donc sources de nombreux besoins de coordination. La présente étude s'intéresse essentiellement à la réalisation d'une stratégie pour les forces de police.

Les policiers ont en effet un rôle essentiel au sein de la simulation puisqu'ils facilitent la réalisation des tâches des autres agents. La plupart des stratégies des solutions participant à la compétition RoboCupRescue fonctionne suivant un principe de réactivité. Les policiers n'interviennent donc en général que suite à une demande de déblaiement spécifique introduite par un agent bloqué par des débris.

La stratégie développée explore l'aspect prévisionnel du rôle de déblaiement des policiers. Elle tend à optimiser les déblaiements préalables afin de rendre praticable le parcours de l'ensemble des grandes routes de la cité modélisée et ainsi prévenir toute situation de blocage.

Un bon moyen de diminuer les coordinations négatives entre agents policiers est de leur délimiter une zone de nettoyage privée. Plusieurs procédures de délimitation ont été implémentées afin d'analyser leurs impacts sur la rapidité de déblaiement total et l'importance des répartitions entre agents.

Une première étape a consisté à réaliser un modèle informatique objet des grandes routes à déblayer sur base du modèle objet JAVA défini par l'API livré avec le simulateur. Les différents principes de conventions réalisés consistent à attribuer les différentes grandes routes à explorer et à déblayer entre les différents agents policiers. Un algorithme de planification permet finalement d'optimiser le parcours de ces grandes routes attribuées

5.1 Principes de coordination par conventions

D***ivision en zones uniformes***
Le premier principe de division de la carte est le plus simple. Sur base du nombre d'agents policiers disponibles, on divise la carte en autant de sous-cartes de surfaces équivalentes.
Suivant l'hypothèse que les centres-villes présentent généralement des topologies plus complexes et donc une charge de déblaiement plus lourde, si le nombre d'agents est impair, une dernière zone de même dimension que les autres et localisée sur le centre de la carte est assignée au dernier agent (fig. 2).
En cas d'appartenance d'une grande route à deux zones différentes, ce dernier est défini comme appartenant aux deux sous-cartes.

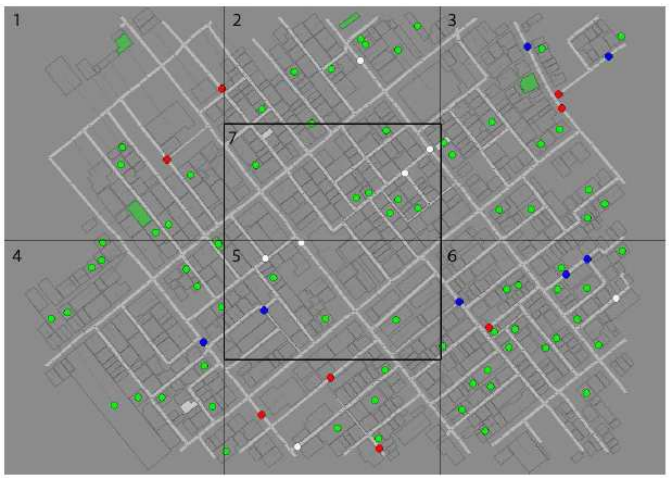

Figure 2 : Division de la carte de Kobe en zones uniformes.

Division en clusters : la méthode des k-moyens
Le clustering est un principe de classification de données en classes présentant des similarités. L'algorithme des k-moyens [5] est une méthode de clustering non supervisée dont l'objectif est de diviser un ensemble de **n** données en **k** classes.
Le caractère non supervisé de la méthode sous-entend que les classes et leurs caractéristiques ne sont pas connues avant application de l'algorithme, mais que leur nombre doit être imposé par l'utilisateur.

Soit une série de **n** données $(x_1, x_2, ..., x_n)$ à classer en **k** clusters $C = \{C_1, C_2, ..., C_k\}$.

L'algorithme des k-moyens définit les clusters finaux C tel que $C = \arg\min_{C} \sum_{i=1}^{k} \sum_{x_j \in C_i} \left\| x_j - c_i \right\|^2$ où c_i est le centroïde du cluster C_i composé des données x_j.

Au final, la méthode des k-moyens peut donc être vue comme une fonction de minimalisation de l'erreur des moindres carrés.

```
1: Function CREATE-KMEANS-CLUSTERS(E,C) returns clusters lists
   Inputs : E : Data elements list, each element defined by its points list P.
            C : Inital Centroids.
2: Create L list of k clusters lists L_i, each initially empty
3: lastCentroid ← null
4: while lastCentroid ≠ C do
5:    lastCentroid ← C
6:    cluster ← null
7:    minDist ← ∞
8:    for each E_i ∈ E do
9:       for each C_j ∈ C do
10:         if ∑ ||P_k − C_j||^2 < minDist for all P_k ∈ E_i then
11:            minDist ← ∑ ||P_k − C_j||^2
12:            cluster ← L_j
13:         end if
14:      end for
15:      Add E_i to L
16:   end for
17:   for each L_i ∈ L do
18:      C_i ← (1/|Li|) ∑_{E_j∈L_i} ∑_{P_k∈E_j} P_k
19:   end for
20: end while
21: return L
```

Figure 3 : Algorithme de création de clusters k-moyen

L'algorithme repose sur un principe itératif en quatre étapes (fig. 3):

1. On définit k points de l'espace des données à classer comme étant les centroïdes initiaux des k clusters.
2. On assigne chaque donnée à classer au cluster dont le centroïde est le plus proche selon un calcul de distance euclidienne.

3. Une fois toutes les données assignées, tous les centroïdes des clusters sont recalculés empiriquement comme étant les barycentres des données de chaque cluster.
4. On répète les étapes 2 et 3 jusqu'à ce que les centroïdes ne changent plus. Les clusters obtenus répondent finalement au critère de minimalisation de la méthode des k-moyens (voir **Figure 3**).

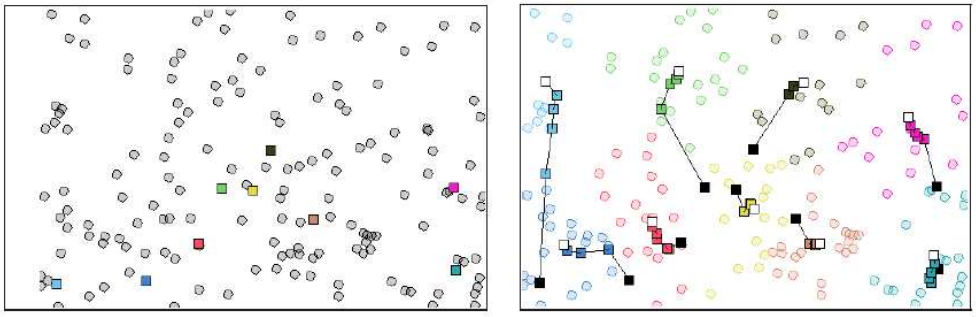

Figure 4 : Exemple d'application de l'algorithme des k-moyens [6]
a) Initialisation des premiers centroïdes.
b) Clusters finaux et déplacements des centroïdes à travers les tours.

Il faut malgré tout remarquer que la méthode des k-moyens ne détermine pas le minimum absolu d'erreur pour tous les classements possibles, mais bien un minimum local conditionné par le choix des centroïdes initiaux.
La méthode des k-moyens a été utilisée dans l'implémentation personnelle pour diviser les grandes routes à nettoyer suivant les hypothèses suivantes :

- Le nombre k de clusters a bien sûr été défini comme étant le nombre d'agents policiers disponibles.
- Les données réellement classées par l'algorithme sont des éléments caractéristiques des grandes routes (extrémités, centre,…).
- Tous les éléments caractéristiques d'une même voie doivent être assignés au même cluster.
- Les valeurs initiales des centroïdes ont été déterminées comme étant les positions des agents policiers en début de simulation.

Un avantage de ce principe est que chaque voie de la carte n'est attribuée qu'une et une seule fois à un cluster. Ce principe diminue sensiblement la charge de déblaiement total généré par la division en zones uniformes, puisqu'une grande route chevauchant deux zones était alors considérée comme appartenant à chacune d'entre elles.

Les résultats de l'application du principe des k-moyens fournissent donc un ensemble de clusters de grandes routes relativement proches des centroïdes initiaux. Les grandes routes d'un même cluster peuvent donc être supposées relativement proches l'une de l'autre. Cet avantage est très intéressant pour l'application recherchée, car elle permet théoriquement de diminuer les déplacements entre tâches. Le principal défaut de ce type de classement est qu'il ne tient pas du tout compte de la densité des clusters générés. Autrement dit, par l'application simple des k-moyens les charges de travail distribuées entre agents ne sont pas nécessairement équilibrées. Une troisième méthode a donc finalement été développée pour tenter de diminuer l'importance de cette inégalité.

Division en clusters : la méthode des k-moyens conditionnée

L'objectif de cette troisième méthode est d'apporter une solution présentant les avantages de division du principe des k-moyens, tout en assurant une distribution plus équilibrée des charges de travail. Il est connu que les résultats de l'application de l'algorithme des k-moyens sont fortement conditionnés par deux critères : le nombre de clusters voulus (k) et les positions des centroïdes initiaux.

Puisque le résultat recherché est d'attribuer un ensemble de grandes routes à chaque agent, il n'est donc pas intéressant de vouloir modifier le nombre de clusters finaux k. La seule façon de pouvoir produire des clusters de densités différentes est donc d'initialiser les centroïdes à des valeurs autres que celles des positions de départ des agents. Le troisième principe de division consiste donc à itérer l'algorithme des k-moyens sur base de centroïdes initiaux choisis aléatoirement, pour finalement choisir le classement le équilibré.

Pour augmenter les chances de générations de clusters différents, la plupart des ouvrages conseillent de choisir aléatoirement la position des centroïdes initiaux. Or il ne faut pas oublier que le système multi-agents est avant tout un programme distribué. En pratique, chaque agent de la simulation est un *thread* distinct qui s'exécute indépendamment des autres. Pour que la même procédure de division en cluster soit exécutée pour chaque agent, il faut que les choix paraissent aléatoires du point de vue d'un agent tout en restant reproductibles dans les mêmes conditions pour les autres agents. Le choix des centroïdes ne peut donc pas s'effectuer par les différentes méthodes de génération pseudo-aléatoires basées sur l'horloge interne du système informatique. Une solution personnelle a donc été développée sur base d'une construction géométrique. (fig.5 et fig. 6)

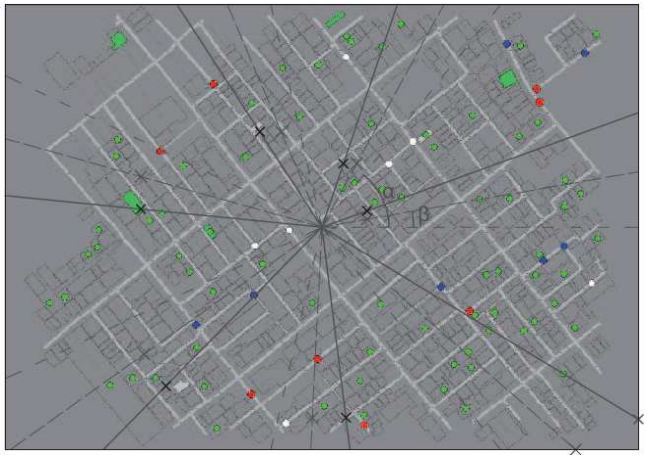

Figure 5 : Principe de génération aléatoire de centroïdes.

Function INIT-CENTROIDS(*center*,*height*,*width*,*k*,*t*) **returns** centroids array

Inputs : *center* : map center of coord (X_C ;Y_C)
height : map height
width : map width
k : clusters number
t : iterations number

Create T **array of t rows and k columns, initially empty**
$\alpha \leftarrow ArcTan(\frac{height}{width})$
$\beta \leftarrow \frac{2\pi}{t}$
for each $i \in [1, ..., t]$ **do**
  $\vartheta \leftarrow \beta i$
  for each $j \in [1, ..., k]$ **do**
    $\varphi \leftarrow \frac{2\pi j}{k}$
    if $\alpha \leq \varphi + \vartheta \leq \pi - \alpha$ **OR** $\pi + \alpha \leq \varphi + \vartheta \leq 2\pi - \alpha$ **then**
      $l = \left| \frac{width}{2Cos(\varphi + \vartheta))} \right|$
      $X_j \leftarrow X_C + \frac{i}{k} lCos(\varphi + \vartheta)$
      $Y_j \leftarrow Y_C + \frac{i}{k} lSin(\varphi + \vartheta)$
      $T_{i,j} \leftarrow$ **Point of coord**(X_j , Y_j)
    else
      $l = \left| \frac{height}{2Sin(\varphi + \vartheta))} \right|$
      $X_j \leftarrow X_C + \frac{i}{k} lCos(\varphi + \vartheta)$
      $Y_j \leftarrow Y_C + \frac{i}{k} lSin(\varphi + \vartheta)$
      $T_{i,j} \leftarrow$ **Point of coord**(X_j , Y_j)
    end if
  end for
end for
return T

Figure 6 : Algorithme de génération pseudo-aléatoire de centroïdes.

Algorithme 2 : Algorithme de génération pseudo-aléatoire de centroïdes.

A. La méthode personnelle des k-moyen conditionnée se résume finalement de la manière suivante. Soit une série de **n** données $(x_1, x_2, ..., x_n)$ à classer en **k** clusters $C = \{C_1, C_2, ..., C_k\}$ après **t** tours de génération suivant un paramètre **P** (nombre de grandes routes, longueur totale de grandes routes ou nombre de parcelles composant les grandes routes).

B. Pour chaque tour :
   1. Appliquer la méthode de génération pseudo-aléatoire de centroïdes pour définir les k centroïdes initiaux du tour $c^t = \{c_1^t, c_2^t, ..., c_k^t\}$.
   2. Exécuter l'algorithme des k-moyens pour générer l'ensemble des clusters du tour $C^t = \{C_1^t, C_2^t, ..., C_k^t\}$.

C. Pour chaque clustering, on compare la valeur moyenne du paramètre relative à chacun des clusters à la valeur moyenne espérée $\overline{P_t}$ si les clusters étaient uniformément répartis suivant ce paramètre. Le clustering finalement choisi est celui qui minimise la moyenne des carrés des écarts à la moyenne théorique. Autrement dit, le classement final est celui dont les clusters présentent une dispersion minimale par rapport à la moyenne représentative de la division la plus équitablement possible suivant le paramètre.

   On détermine donc $C = \arg\min_{C^t} \sum_{i=1}^{k} \sum_{x_j \in C_i^t} \frac{P(x_j) - \overline{P_t}}{k}$ où $P(x_j)$ est la valeur paramétrique de la donnée x_j du cluster C_j^t généré au tour t.

Planification des tâches de déblaiement

La deuxième méthode tend finalement à minimiser les déplacements entre les tâches de déblaiement et repose sur une représentation graphique des différentes grandes routes à déblayer.

En effet, considérons l'ensemble des carrefours d'une zone de la carte comme étant l'ensemble E des nœuds d'un graphe, et l'ensemble des grandes routes les joignant comme étant l'ensemble V des arêtes de ce graphe.

En théorie des graphes, un tel graphe non orienté $G = (E, V)$ est dit connexe si quels que soient les nœuds E_i et E_j de E, il existe un chemin de E_i et E_j.

C'est-à-dire, s'il existe une suite d'arêtes permettant d'atteindre E_j à partir de E_i. Suivant le principe de construction des grandes routes implémenté dans le modèle objet, tous les graphes G sont imposés connexes.

En supposant que le temps nécessaire au déblaiement d'une grande route ne diffère pas durant la simulation et qu'il n'y a pas de nouvelle route bloquée après le début de la simulation
Le problème de détermination du parcours nécessitant le minimum de déplacement peut alors être considéré comme un problème de recherche de chemin eulérien sur le graphe G.

Un chemin est dit eulérien s'il permet le parcours, une et une seule fois, de chaque arête du graphe, ce qui revient à parcourir l'ensemble des grandes routes une seule fois pour les agents policiers.
Malheureusement tous les graphes connexes ne possèdent pas nécessairement de chemin eulérien.

Une condition suffisante pour qu'un graphe connexe possède un chemin eulérien est que ce graphe soit lui-même eulérien.
Par ailleurs, suivant le théorème d'Euler, « *Un graphe connexe est dit eulérien si et seulement si chacun de ses sommets est incident à un nombre pair d'arêtes ».*

Cela revient à dire qu'un graphe connexe est eulérien si et seulement si chacun de ses nœuds est de degré pair (Le degré d'un nœud est égal au nombre d'arêtes qui lui sont adjacentes).

Afin de pouvoir déduire le plus court cheminement permettant le parcours de toutes les grandes routes, il faut donc premièrement transformer les graphes des topologies de chaque zone de la carte pour qu'ils répondent à cette condition

La transformation du modèle graphique en graphe connexe

En théorie des graphes, les techniques de transformation consistent généralement à ignorer certains nœuds afin de modifier la topologie du graphe pour atteindre la configuration voulue.

Dans le problème envisagé, cela signifierait omettre le passage par certaines grandes routes de la zone ce qui n'est pas acceptable pour l'objectif poursuivi.

L'algorithme de transformation développé personnellement modifie les degrés des nœuds du graphe en dupliquant les arêtes déjà présentes. En pratique cela consiste à permettre à l'agent policier de se déplacer plusieurs fois sur une même route. Puisqu'au deuxième passage de l'agent, la grande route est déjà dégagée et en supposant que cette dernière n'est pas trop longue, on peut considérer que cet ajout n'augmente pas trop le nombre de tours nécessaires au déplacement jusqu'à la prochaine grande route à déblayer.

Il est alors intéressant de se demander si tout graphe connexe est transformable en graphe eulérien. Pour répondre à cette question, analysons l'effet d'une duplication d'arête sur les degrés des nœuds d'un graphe joints par cette dernière :

1. Si la nouvelle arête joint des nœuds initialement de degrés pairs, l'ajout les transforme tous les deux en nœuds de degré impair. L'effet se résume donc à créer une paire de nœuds de degré impair.
2. Si la nouvelle arête joint des nœuds initialement de degré impair, l'ajout les transforme tous les deux en nœuds de degré pair. L'effet se résume donc à supprimer une paire de nœuds de degré impair.
3. Si la nouvelle arête joint des nœuds de parités différentes, l'ajout va simplement permuter les parités des deux nœuds. L'effet résultant ne réduit donc pas le nombre de nœuds de degré impair et ne consiste qu'en un déplacement de l'imparité.

Par ailleurs, le théorème dit de *Handshaking* [7] stipule que « *Dans un graphe non orienté, il y a toujours un nombre pair de nœuds de degré impair.* »

Soit G, un graphe non orienté possédant $2n$ nœuds de degré impair, et A_1, A_2, A_3 les trois actions précédemment présentées.
Si on réalise ces actions respectivement n_1, n_2, n_3 fois, le nombre de nœuds du graphe de degré impair vaut alors $2n + 2n_1 - 2n_2$.
Ce nombre devient donc nul si et seulement si $n_2 = n + n_1$.

Par la duplication d'arêtes idéalement choisies, on peut donc déplacer et détruire les imparités. Tout graphe non orienté est donc réductible en graphe eulérien. Les graphes connexes représentant les grandes routes étant

eux-mêmes non orientés, un chemin eulérien peut donc être trouvé pour chaque agent policier.

Algorithme 3 : Algorithme de transformation en graphe eulérien.

En appliquant l'algorithme de transformation en graphe eulérien personnel (fig. 7) au graphe G construit suivant la topologie des grandes routes assignées à l'agent, on s'assure donc que ce dernier ne contient plus de nœuds de degré impair et soit donc eulérien.

L'algorithme de transformation personnel construit le graphe eulérien G^E en minimisant la longueur totale des arêtes qu'il ajoute. L'algorithme assure donc l'obtention d'un graphe possédant le plus court chemin eulérien qu'il est possible de construire suivant la topologie de G.

En pratique l'algorithme analyse les différents nœuds du graphe G et détermine le plus court chemin qui existe entre deux nœuds de degré impair distincts. Cette recherche du plus court chemin est basée sur une recherche en largeur traditionnelle (à partir d'un nœud E, il liste d'abord les nœuds voisins pour ensuite les explorer un par un). Il duplique l'arête adjacente à l'un de ces deux nœuds appartenant au plus court chemin déterminé. Ce processus se réitère alors jusqu'à ce que le graphe G ne possède plus aucun nœud de degré impair et soit donc devenu eulérien (fig. 8).

```
1: Function EULERIAN-GRAPH(G(E,V)) returns the eulerian graph of G
   Inputs : G(E,V) : Graph. E edges list, V vertexs lists.
2: Create O list of all the odd degree vertexs of G(E,V)
3: while O is not empty do
4:    Create P list of paths, Initially empt
5:    for each V_i ∈ O do
6:       path ← LESS-COST-PATH(G(E,V),V_i)
7:       Add path to P
8:    end for
9:    minCost ← ∞
10:   P_* ← null
11:   for each P_j ∈ P do
12:      if P_j cost < minCost then
13:         minCost ← P_j cost
14:         P* ← P_j
15:      end if
16:   end for
17:   Duplicate E_k ∈ P*, adjacent to V_i
18: end while
19: return G^E
```

Figure 7 : Algorithme de transformation en graphe eulérien.

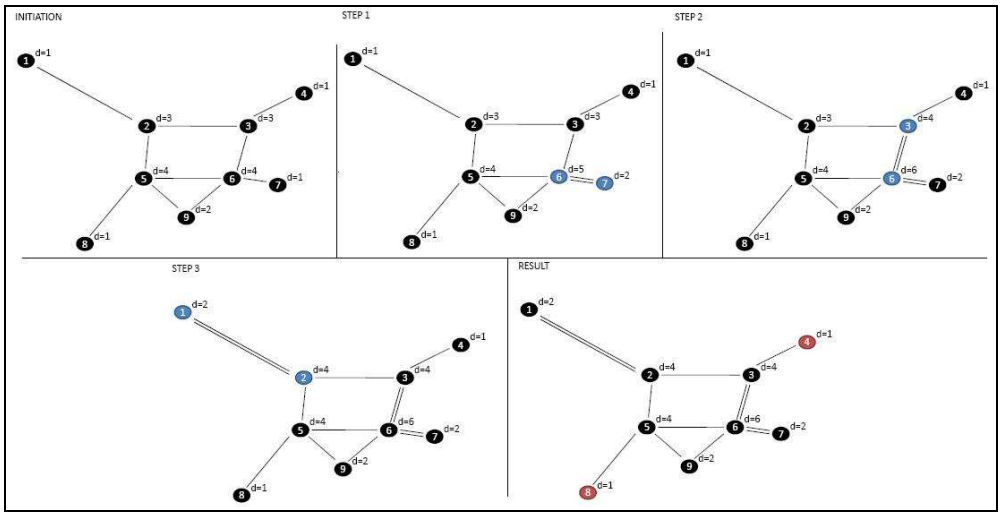

Figure 8 : Exemple de transformation d'un graphe en graphe eulérien.

L'algorithme de *Fleury* nous permet ensuite de déterminer les chemins eulériens du nouveau graphe G^E. [8]
Son principe est assez simple et peut se résumer de la manière suivante :

1. Au départ d'un nœud du graphe eulérien G^E choisir une arête adjacente non marquée qui n'est pas considérée comme un pont. Une arête sera considérée comme un pont si la suppression de cette arête fait perdre la propriété connexe du graphe représentant le parcours restant à effectuer.
2. Si toutes les arêtes non marquées disponibles sont des ponts, choisir l'un d'entre eux.
3. Parcourir l'arête choisie pour atteindre le nouveau nœud.
4. Marquer l'arête parcourue pour qu'on ne puisse plus l'emprunter.
5. Réitérer les points 1 à 4 jusqu'à ce que toutes les arêtes du graphe eulérien G^E aient été parcourues et que l'on soit revenu au nœud initial.(fig. 9)

Il existe donc au minimum autant de chemins eulériens qu'il existe de nœuds $\in G^E$. De plus les topologies des graphes représentant les sections de carte présentent de nombreuses boucles ce qui augmente rapidement le nombre de chemins eulériens disponibles. La stratégie des policiers va donc finalement consister à décider à chaque tour de simulation lequel des chemins eulériens disponibles il va parcourir au tour suivant. Ce choix est

finalement réalisé sur base d'un ensemble de conventions tenant notamment compte de la proximité des incendies et des refuges.

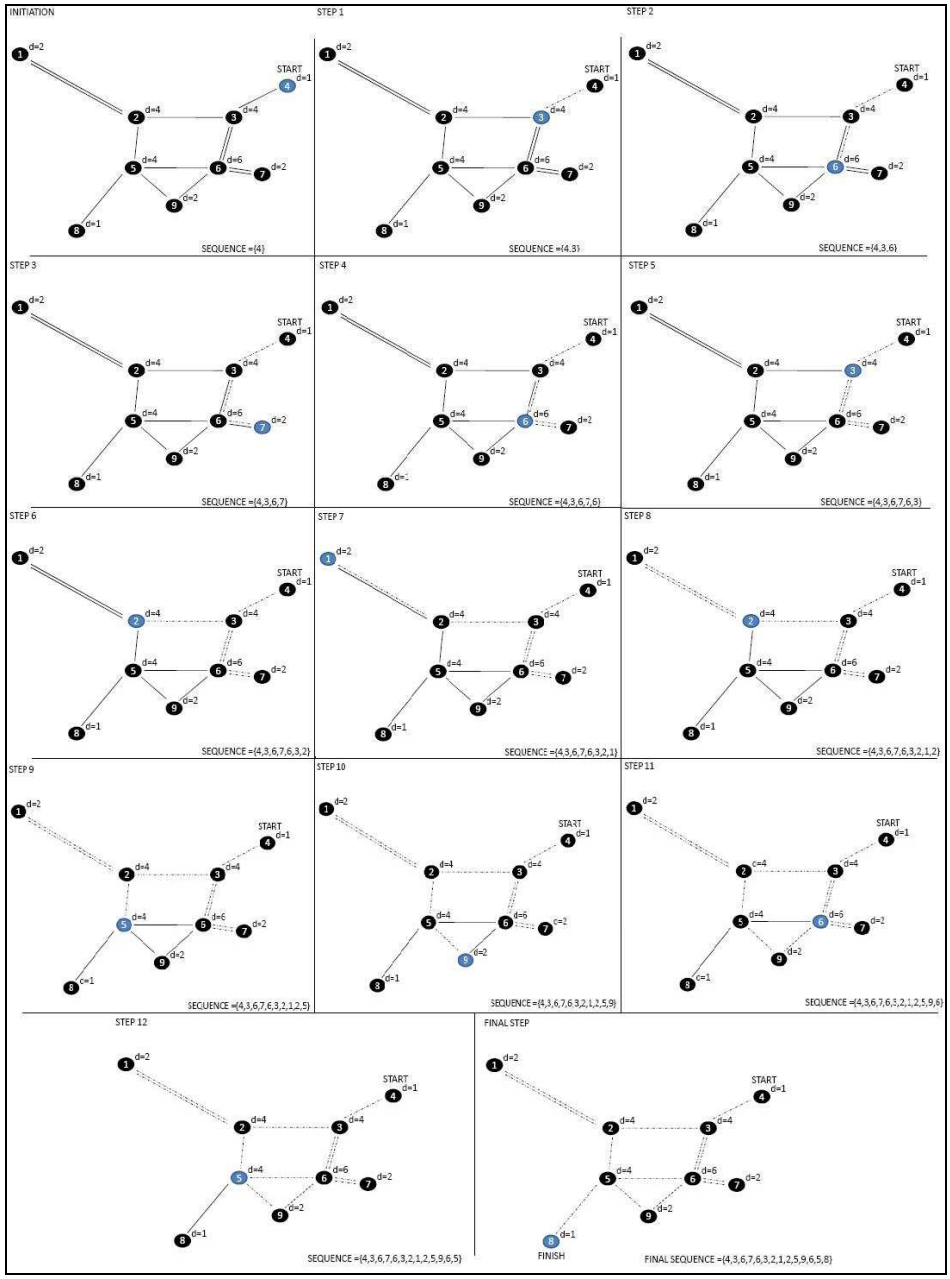

Figure 9 : Exemple d'application de l'algorithme de Fleury

6. Résultats d'expérimentaux

Afin de valider la stratégie réalisée, une série de simulations a été réalisée sur base des cartes de Kobe et de Foligno(fig. 10).

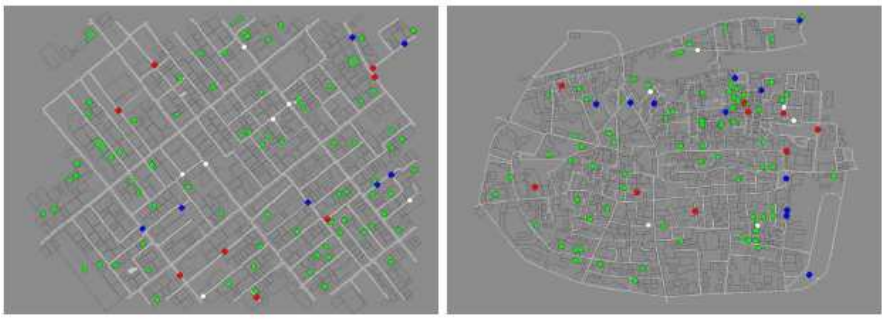

Figure 10 : Topologies simulée des cartes de Kobe et Foligno.

Pour permettre la réalisation de simulations, l'ensemble de la stratégie a été intégré au programme *JAVA* libre de source *SUNTORI* ayant participé à l'édition 2008 de la compétition RoboCupRescue et ayant décroché la seconde place.
L'étude entreprise a pour principal objectif de déterminer l'intérêt d'une optimisation du caractère préventif des agents policiers au sein d'une simulation RoboCupRescue. Le programme *SUNTORI* étant caractérisé par des stratégies essentiellement réactives (les déblaiements ne sont effectués qu'à la demande des autres agents), plusieurs simulations ont donc également été réalisées sur base de son code d'origine afin de disposer d'une base de comparaison pertinente.

Les observations obtenues sont de deux types : les temps de déblaiement qui caractérisent la rapidité des stratégies et les scores attribués en fin de simulation. Ce score est défini par le simulateur suivant une formule empirique complexe tenant notamment comptent du nombre de victimes sauvées et de la propagation des incendies. Puisque la faculté des pompiers et des ambulanciers à remplir leurs tâches dépend directement de l'état des routes à emprunter, le score peut dès lors être perçu comme une fonction de la qualité de déblaiement.
Il faut néanmoins remarquer que seule la stratégie des agents policiers a été modifiée. Les résultats obtenus par le programme modifié sont donc légèrement biaisés par rapport à l'original puisque la coordination entre agents hétérogènes n'a pas été optimisée.

En résumé, les différentes stratégies analysées par le plan d'expérimentation sont :

- La stratégie réactive du programme Suntori original (Suntori)
- La stratégie préventive suivant la division de la carte en zones uniformes (Area Euler)
- La stratégie préventive suivant la méthode des k-moyens (Cluster Euler)
- La stratégie préventive suivant la division par méthode des k-moyens conditionnée par la longueur totale de route à déblayer (Cond L Euler)
- La stratégie préventive suivant la division par méthode des k-moyens conditionnée par le nombre de parcelles de grande route à déblayer (Cond R Euler)
- La stratégie préventive suivant la division par méthode des k-moyens conditionnée par le nombre de grande route à déblayer (Cond GR Euler)

6.1 Comparaison de la rapidité de déblaiement des stratégies

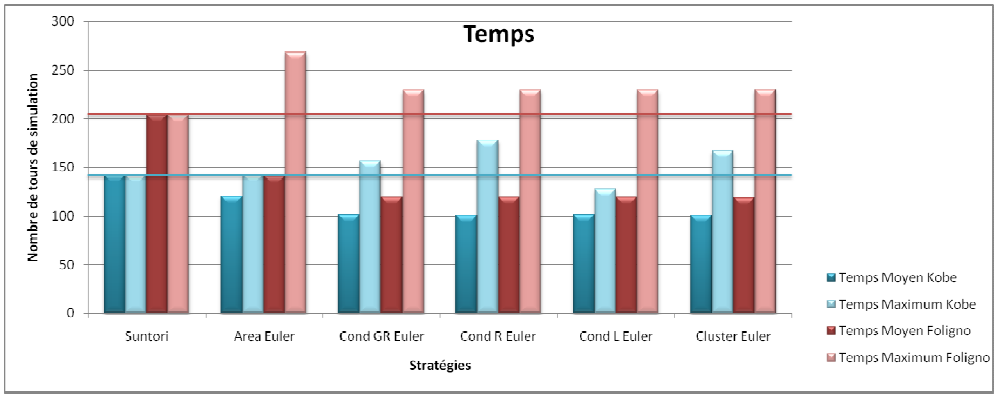

Figure 11 : Temps de déblaiement moyens et maximum obtenus.

On peut premièrement remarquer que chacune des stratégies personnelles présente un temps de déblaiement moyen par agent inférieur à celui de l'équipe *SUNTORI* (fig. 11). Les stratégies personnelles de déblaiement peuvent donc être considérées comme plus rapides que celle de l'équipe *SUNTORI*. L'influence des conventions de divisions en zones uniformes dépend essentiellement de la carte simulée. Pour la carte de Foligno, la division résultante est très déséquilibrée et le temps de déblaiement total en est excessivement prolongé. On peut également noter que sur cette carte, un

agent ne reçoit aucune route à déblayer suivant cette méthode. La capacité générale de déblaiement s'en retrouve donc largement diminuée. Dans la plupart des cas, si elle est totalement déblayée en fin de simulation, la carte ne le sera qu'au cours des cinquante derniers tours.
Les résultats des méthodes des k-moyens conditionnés dépendent eux aussi des topographies des cartes. Les divisions suivant le nombre de grandes routes et le nombre de sections de route semblent néanmoins plus rapides que les autres. On peut donc en déduire que la vitesse des véhicules est suffisante pour négliger la longueur des grandes routes à distribuer.

6.2 Comparaison de l'efficacité des stratégies suivant les scores obtenus

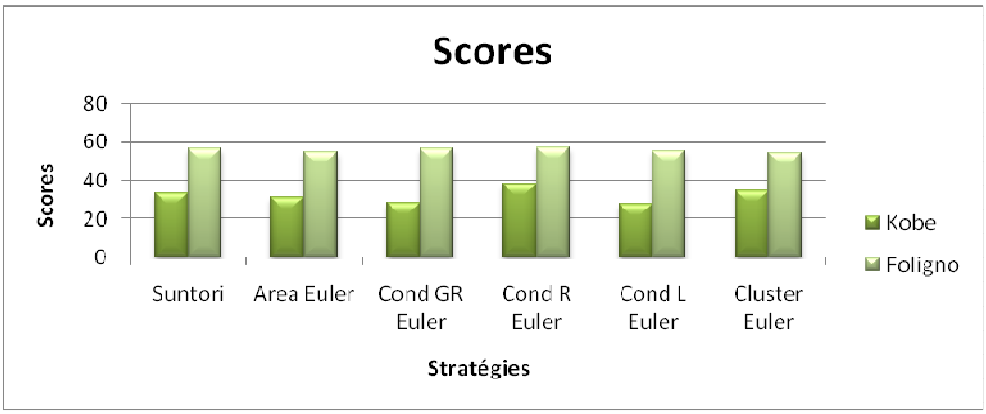

Figure 12 : Scores accordés aux résultats des simulations.

L'efficacité des différentes stratégies semble être très aléatoire et ne semble pas suivre de règle commune à toutes les cartes (fig. 12). On peut malgré tout remarquer les stratégies basées sur une division géographique par clustering conditionné suivant le nombre de parcelles de grande route (R) et le nombre de grandes routes (GR) obtiennent généralement les meilleurs résultats surpassant ceux de l'équipe *SUNTORI* sur les deux cartes. La division en zones uniformes n'arrive par contre jamais à dépasser ces résultats.
L'obtention de ces résultats conforte donc l'idée qu'une stratégie prévisionnelle est intéressante à exploiter au sein de principes de coordination appliqués à la simulation.

7. Conclusion

La conclusion de l'analyse stratégique soulève l'idée qu'il n'existe en fait aucune stratégie optimale pour l'ensemble des cartes pouvant être simulées. L'importance de l'aspect prévisionnel par rapport au réactif est en fait relative à la pression qu'exerce l'environnement sur le système multi-agents. Chaque stratégie possède ses forces et ses faiblesses conditionnées par l'environnement et plus particulièrement la topographie de la carte. Pour pouvoir être efficace sur tout type de carte, la réalisation d'une stratégie devrait pouvoir s'adapter à cette topographie.

Une solution serait par exemple l'exploitation de principe de coordination par rôles, où chaque rôle définirait la stratégie optimale à suivre face aux particularités de la carte. Une autre solution serait d'apprendre ces rôles sur base d'apprentissage. De plus, puisque tous les agents possèdent les informations complètes permettant de d´écrire la topologie de la ville simulée au début de simulation, un traitement *hors ligne* permettant l'exécution d'algorithmes plus lourds est également envisageable.
Malgré tout, la réactivité semble tenir une place importante en début de simulation. Une alternative pourrait donc consister à diviser les policiers suivant deux rôles. Le premier rôle consisterait à n'assurer que les demandes en urgence sur toute la carte alors que le deuxième assurerait le côté préventif de la stratégie. L'attribution des rôles pourrait alors se faire en fonction du déroulement de la simulation et permettre ainsi l'ajustement du côté préventif et réactif tout au long de la simulation.

Le temps moyen de déblaiement est une information qui a son importance. Comme il a déjà été dit, une stratégie nécessitant de nombreux tours pour déblayer la totalité de la carte, mais possédant un nombre de tours moyen par agent relativement bas indique clairement que la répartition des tâches entre agents n'est pas optimale est que c'est principalement le retard d'un ou de quelques agents qui pénalise toute la stratégie. En incluant un simple principe de coordination par communication on pourrait donc encore améliorer l'efficacité des déblaiements des stratégies personnelles. L'agent ayant terminé de déblayer les grandes routes qui lui avaient été assignées pourrait dès lors se renseigner pour déterminer quel agent est le plus en retard et lui proposer son aide. À terme, les temps de déblaiement totaux convergeraient vers des valeurs plus proches des valeurs moyennes pour ainsi améliorer de nouveau l'efficacité des policiers.

8. Références bibliographiques

[1] CHAIB-DRAA, BRAHIM, *Agent et systèmes multiagents*, Canada, Université Laval Québec, Notes du cours IFT 64881A.

[2] MALONE, T.W., *The interdisciplinary study of coordination.*, ACM Comput. Surv. *n*°26, p:87–119.

[3] PAQUET, SEBASTIEN, *Agent Distributed Decision-Making and Task : Coordination in Dynamic, Uncertainand Real-Time Multiagent Environments*., Canada, Faculté de Sciences et Génie, Université Laval, Québec, 1999

[4] BOUTILIER, C., *Planning, Learning and Coordination in Multiagent Decision Processes.,* TARK-96 : Theoretical Aspects of Rationality and Knowledge, p :195–210.

[5] MACQUEEN, J.B., BOUTILIER, C., *Some methods for classification and analysis of multivariate observations.*, 5th Berkeley Symposium on Mathematical Statistics and Probability., p281–297.

[6] MATTEUCCI, M., *K-means clustering demo java applet.*, http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html.

[7] ERICKSON, M.J., *Introduction to Combinatorics.,* Wiley-Interscience, 1996.

[8] FLEURY, M., *Deux problèmes de géométrie de situation*, Journal de mathématiques élémentaires, 1983.