International Journal of Computational Intelligence Systems, Vol. 6, No. 1 (January, 2013), 156-173

An Agent-Based Distributed Scheduling For Crisis Management Supply Chain

Ayda Kaddoussi, Nesrine Zoghlami, Slim Hammadi
Ecole Centrale de Lille, Cité Scientifique – BP,
Villeneuve d'Ascq, France
E-mail :{ayda.kaddoussi;nesrine.zoghlami;slim.hammadi}@ec-lille.fr

Hayfa Zgaya
Institut Lillois d'Ingénierie de la Santé
42, Rue Ambroise Paré – LOOS, France
E-mail :hayfa.zgaya@univ-lille2.fr

Received 25 August 2012

Accepted 27 September 2012

Abstract

There has been a significant increase in the improvement of response to disasters in crisis management supply chain. Due to their sudden occurrence, these disasters require a consequent quick and efficient response that depends on the ability of logistics systems to generate plans under a variety of constraints. The supply chain studied in this work is a crisis management supply chain, composed of several elements such as transportation means, loading units, suppliers, equipment, resources, persons... We propose an innovative method for solving a distributed delivery scheduling problem, based on a multi-agent system, for the delivery of goods (food, water, clothes, etc.) to the areas affected by the disaster. The covered areas are geographically distributed and partitioned into multiple sub-regions. Each area is assigned a delivery scheduling sub problem. By employing a distributed cooperative framework, we achieved an incorporation of various evaluation parameters in the process of scheduling in order to maintain a high level of synchronization of all the supply chain, and so to insure a better response to the crisis.

Keywords: Decision support systems, Crisis Management, Supply Chain, Distributed Scheduling, Optimization, Multi-agent systems.

1. Introduction

Military logistics planning involves supplying and transporting resources and military assets[1]. The presented work concerns the definition of a modeling approach and an oriented agent simulation of supply chains in a context with strong disturbances: a Crisis Management Supply Chain (CMSC). However, working in an uncertain environment; incite to be equipped with optimization and cooperation mechanisms assuring all the chain actors satisfaction, while acting in a collective way to reach a common objective: the crisis management. To resolve this kind of problem, we define a distributed delivery scheduling algorithm based on the cooperation between agents. The problem which a delivery schedule attempts to address is that of planning tasks involving multiple transportations' means to deliver goods from a delivery center to numerous locations. The schedule includes meeting deadlines, vehicle allocation, delivery route selection and other parameters, with consideration paid to delivery costs.

Our collaboration with the European Aeronautic Defence and Space Company (EADS) has led to the development of a supply chain engineering tool called OBAC (Optimization Based on Agents Communication) which is being prototyped at the Defence & Security (DS) department. The OBAC is investigating, developing and demonstrating technologies to make a fundamental improvement in logistics planning.

The contribution of this paper is an optimization model for a combined scheduling and synchronization

distributed problem. Our concern is to propose a distributed scheduling of the delivery tasks to be executed all along the CMSC. The purpose is to supply the different zones in need while minimizing tardiness in delivery and maintaining a good synchronization between the different areas. We were faced to several constraints such as delivery dates, limited transportation's means in number and capacity, warehousing costs and useable routes. The delivery schedule implemented decides times for departure of the tasks, determines delivery routes, and allocates vehicles while ensuring:

- Meeting deadlines for delivery to the areas.
- Reducing costs.
- Rapid generation of a delivery schedule.

The rest of the paper is organized as follows. Related works in crisis management and optimization algorithms is presented in section 2. The global architecture of the multi-agent system is proposed in section 3, describing the main function of each composing part. In section 4, a detailed formulation of the problem is presented. The global distributed approach for delivery scheduling is described in section 5. Simulation's results are given in section 6, to validate the algorithms developed previously. Conclusion and possible future works are addressed in last section.

2. Related Works

2.1. *Multi-agent systems*

The concept of Multi-Agent System (MAS) is intimately linked to that of Distributed Artificial Intelligence (DAI). The paradigm of Artificial Intelligence, to concentrate the intelligent capabilities into a single entity, proved to be insufficient to solve certain types of complex problems[31]. To correct this limitation, a sub domain has emerged advocating the passage from an "individual intelligence" to a "collective intelligence". The MAS is so far an important branch of the DAI.

The MAS permit to "model the behavior of a set of expert entities, organized more or less according to social type laws. These entities or agents have some autonomy and are immersed in an environment in which and with whom they interact."[32]

Ferber[33] defines an agent as a physical or virtual entity:

- which is capable of acting in an environment.
- which can communicate directly with other agents.
- which is driven by a set of tendencies (in the form of individual objectives or of a satisfaction/survival function which it tries to optimize).

Ferber[33] listed the applications of MAS as follows:

Problem Solving: As an alternative to centralized problem solving, either because problems are themselves distributed, or because the distribution of problem solving between different agents reveals itself to be more efficient way to organize the problem solving - it can be flexible and allow failures in the system - or because, in some cases, it is the only way to solve the problem.

Multi-Agent Simulation: Simulation is widely used to enhance knowledge in biology or in social science and MAS gives us the possibility to make artificial universes that are small laboratories for the testing of theories about local behaviors.

Construction of Synthetic Worlds: These artificial universes can be used to describe specific interaction mechanisms and analyze their impact at a global level in the system. The entities that are represented are mainly inspired by animal behaviors (hunting, searching or gathering habits). The aim of this research is to have societies of agents that are very flexible and can adapt even in cases of individual failure.

Collective Robotics: Defining the robots as a MAS where each subsystem has a specific goal and deals with that goal only. Once all the small tasks are accomplished the big task is too. MAS approaches can also be used in the co-ordination of different mobile robots in a common space.

2.2. *Logistics for crisis management*

There are research projects in the literature that use agent technologies for logistics, but many are focussed towards specific aspects of logistics, such as routing, manufacturing production supply chain, inventory management, etc.

The literature on disaster relief logistics shows that distributed decision-making agent-based paradigm provides an interesting approach to increase agility in Defense systems. The Advanced Logistics Project $(ALP)^2$ for example, is a Defence Advanced Research Projects Agency (DARPA) research project, which aims to automate collaborative technologies for the logistics communities for the U.S military, also by using multi-agent technology. ALP is developed by using the Cognitive Agent Architecture (Cougaar), based on the

human cognitive model[3]. The Cougar architecture is composed of an Expander, which decomposes a task into a workflow of sub-tasks; an Allocator, which allocate tasks to appropriate resources for final handling of further disposition; an Aggregator, which joins a set of tasks into a single super-task; an Assessor, which monitors the plan and the current state to check if objectives are been achieved; and a Data Manager, which is the interface to the environment.

A second existing project is the agent-based Logistics Planning System (LPS), which is a Defence Science and Technology Organization (DSTO) project aimed at developing a support system for the Australian military logistics planners. LPS architecture comprises *User Interface*, which allows the logistics planner(s) to enter logistics goals into the LPS, and presents the user with the logistics plan and analysis; *Organisational Entity* agents, which are grouped into three types based on their roles: *Supply Agents*, *Transport Agents* and *Manager Agents*; and *Information Gathering agents* which are connected to various information sources, such as airfields, ports or aviation fuel databases, and weather Internet sites, to gather information requested by other agents.

Nguyen et al.[4] introduce the idea of using generating functions for evaluating the effectiveness of an air defense system. In his work, Nguyen studies the quantification of benefit from resource allocation for a naval task group having perfect coordination between its assets.

The INGENIAS Development Kit (IDK) is an application developed to manage the crisis situation of a city, in which, a poisonous material is released, and the central services are not enough to heal all the affected people, using a fully functional multi-agent systems[18].

Another example of applying agent technology in crisis is proposed in Beard et al.[19] works. The system is applied to broadband networks to be widely useful with emergencies or natural disasters based on the assumption that integrated networks must treat some connections with greater importance than others, especially in crisis. The ticket server system provides an architecture that automatically and dynamically determines which connections should be treated with higher importance than others, so services can be provided accordingly.

Due to the limited publicly available information about the military logistics planning systems cited above, it is difficult to make an in depth comparison between them. Even if most of those applications use and apply similar concepts, however, none of them seems to develop a distributed scheduling protocol using agent technology to process the logistics flows involved in the logistics planning. Differences in the domains, i.e. United States, Australian military logistics planning, and the European Aeronautic Defence and Space Company in our case (EADS) may result in having some different functionalities (e.g. level of detail in plans) and research and development issues. Table 1 summarizes the similarities and differences found in the available literature between the three systems.

	Cougaar	LPS	OBAC
Supports highly scalable, distributed systems	Yes	Yes	Yes
Supports modular, parallel software development	Yes	Yes	Yes
Can integrate with other technologies	Yes	No	Yes
Allows zones positioning	No	Not mentioned	Optimize the zones locations
Open source product	Yes	No	No
Classes of problems	Distributed problem, distributed data.	Distributed problem, centralized data.	Distributed problem, distributed data.
Central monitoring agent	Yes	Yes	No

Table 1. Comparison between defense logistics systems

2.3. *Scheduling Algorithms*

In manufacturing systems, scheduling is the process by which materials and production capacities are allocated to meet the clients' demand. Competing priorities and antagonist goals make this process very difficult and complex to do. In distribution systems, the purpose of scheduling is to minimize the total spent time and costs, by telling a distribution facility when to deliver, with which staff, and on which mean of transport. During order entry, each schedule line for an item can contain a requested delivery deadline. The items should arrive at the clients on this date. The objective of the scheduling is to automatically plan when the essential shipping activities such as picking, loading and transporting must be started so that the requested delivery date are met. This aims to maximize the efficiency of the delivery process and reduce costs.

Many scheduling algorithms are developed and different categories are studied in the literature: static and dynamic, online and offline, preemptive and non-preemptive and finally local and global scheduling.

Static scheduling VS dynamic scheduling:

Static scheduling algorithm has complete knowledge of the tasks model and its properties: deadlines, periods, worst case execution time, etc. It calculates schedules for the system by generating a table that includes the tasks execution order[14].

Dynamic scheduling algorithm does not have the complete knowledge of the tasks model and its timing constraints. It calculates the execution order of tasks according to their arrivals and activation instants. Dynamic scheduling has the potential to offer relief from some of the restrictions imposed by strict static approaches. Potential benefits of dynamic scheduling include better tolerance for variations in activities. A dynamic scheduling algorithm may be used with all the kind of tasks (periodic and aperiodic) with fixed or dynamic priorities. (e.g.[15,16])

Offline scheduling VS online scheduling:

A scheduler is called offline if all scheduling decisions are made prior to running the system. A table is generated that contains all the scheduling decisions for use during run-time. This relies completely upon a priori knowledge of tasks behavior.

Online scheduler makes scheduling decisions during the run-time of the system. The scheduling algorithm can be either static or dynamic. The decisions are based on both tasks model properties and the current state of the system.

Preemptive scheduling VS non preemptive scheduling:

A scheduler is called preemptive if it can arbitrarily suspend a task's execution and restart it later without affecting its behavior, except by increasing its elapse time. Preemption typically occurs when a higher priority task becomes run able. Unfortunately in presence of inter-tasks synchronization (e.g., resource sharing), this scheme may lead to an inter-tasks priority inversion, when a lower priority task suspends a higher priority task. This problem may be resolved by methods such as priority inheritance. Example of preemptive scheduling [17] includes multiprocessors tasks on two parallel processors.

A non-preemptive scheduler does not suspend tasks. When the scheduler selects a task to be executed, it runs until its end of execution time. Hybrid systems are also possible. A scheduler may be preemptive but allows a task to continue executing for a short period after the instant it should be suspended on.

Local scheduling VS Global scheduling:

The local scheduler allocates the local processor to the set of tasks present in its node while respecting their timing and resource requirements.

The global scheduler tries to guarantee the tasks constraints by considering and exploiting the processing capacities of all the processors composing the distributed system.

Distributed scheduling:

Most distributed scheduling algorithms have two common features: a global scheduler between nodes and a local scheduler for each individual node. In distributed system, each subsystem may provide its own local scheduling mechanism. The global scheduler takes into account the entire system and provides coordinated scheduling information to the individual subsystems, which will enforce the globally determined scheduling decisions.

From the classification that we presented in this section, we propose to situate our problem in the next Table 2.

Scheduling form	Similarities
Static	*No*
Dynamic	*Tasks arrive dynamically*
Offline Scheduling	*No*
Online scheduling	*During the run-time of the system*
Preemptive scheduling	*No*
Non preemptive scheduling	*A delivery task is executed without preemption*
Local Scheduling	*At a zone level*
Global Scheduling	*At the supply chain level*
Distributed scheduling	*Highly distributed supply chain*

Table 2. Comparison with existent forms of scheduling

2.4. *Models for crisis management delivery logistics*

In the crisis management literature, called also disaster relief, we observe different logistical models that describe different objective functions to optimize. Examples include maximizing the satisfied demand, minimizing completion time and minimizing total delivery delay. Barbarosoglu et al.[20] develop a hierarchical decision support methodology for helicopter logistics planning. The first level of planning concerns the tactical decisions as the selection of helicopter fleet size, and the second level involves the operational decision of vehicle routing and tries to minimize mission completion time. This model can solve very small instances with crisis management supply chain of up to 10 zones and three helicopters.

Mete and Zabinsky[21] describe a two-stage stochastic program that solves the location problem of warehouses in the first stage, and the transportation of resources in the second stage. This model shows some limitations since demand is considered as random variable. The authors solve a small scenario with 21 zone and 14 vehicles.

Some of the works found in the literature is dedicated to evacuation only. For instance, a linear programming approach is described in Chiu and Zheng[22] who deal with multi-priority group evacuation in sudden onset disasters. The authors assume that necessary information such as population number is known with certainty and minimize the objective of total travel time.

In this state of the art, we focus on published work that is conducted in the area of crisis management distribution logistics. For readers who are interested in all aspects of crisis management, we refer to the extensive surveys of Altay and Green and Apte[23].

The proposed system in our work is able to deal with large scale crisis management supply chain. The method developed always produces feasible solutions and offers the flexibility of adjusting the solution quality by imposing different performance indicators restrictions.

3. Multi-agent System Description

The problem addressed in this paper is how to satisfy the demand of the different actors of the CMSC by providing efficient supply chain management and high delivery service level. Delivery accuracy is one of the key objectives of the operations administration in crisis management.

Figure 1 shows a representative model of our distributed delivery scheduling problem. Delivery tasks collected at one zone are divided among multiple means of transportation and are delivered to others dispersed subaltern zones, using existing routes.

- Delivery tasks: a delivery task consists of an amount of good to be delivered to a specific zone under some deadline constraints. The location of the different delivery zones in the logistic chain can vary a lot and depends on the crisis that occurs. The delivery of the products carries a cost, and the size of this cost depends on the distance from the supplier zone to the zone in need. Loading and unloading times are included in the delivery periods.
- Warehouses: We dispose of warehouses in each zone to store the supplies. Stored goods can include any finished goods such as food, water, medicaments, clothes… The major warehousing process includes reception, order preparation, shipping and inventory management.
- Transport means: The transport of goods can be done by trucks, ships or aircrafts, with a preponderance of the use of trucks. The cost and the limited volume of the transport's means imply the need to optimize their use and to carefully plan their exploitation.
- Routes: the road network used in the CMSC is permanent in metropolis zones. In zones at risk, used roads will be determined according to the situation on the field.
- Personnel: it consists of trained personnel belonging to a specific grade and complying with a hierarchical structure.

Our proposition is to consider each actor of the CMSC as an autonomous agent, able to exchange information with other actors[9].

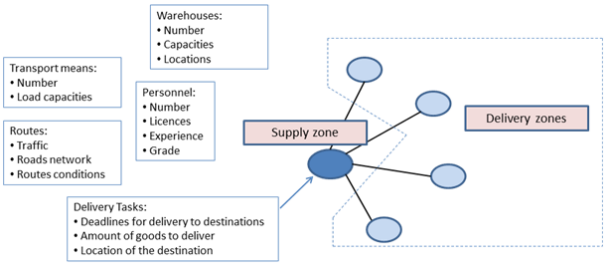

Fig. 1. Typical model of a delivery scheduling system

Our proposition is presented by three-level architecture. The figure 2 illustrates the architecture of the OBAC, containing agents (circles), connected to a network allowing them to communicate: it is the main level which is represented by the middle layer of the figure corresponds to modelling the CMSC by a Multi-Agent System (MAS). In this model, agents working within the multi-agent system continuously receive data from the bottom layer: the Theatre of Operations' level. Based on this information and on the various mathematical models available, these agents adapt their behaviour to respond the best to the different disturbances that occur in the bottom level. It is interesting to see that the behaviour of agents may suggest different actions and decisions, among them the correction and adjustment of the mathematical models if they don't fit to what happens in reality.

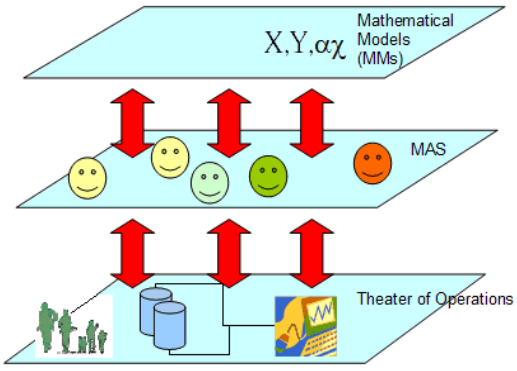

Fig. 2. System architecture

To resolve the problem described previously, we propose a system based on the coordination of different kinds of software agents.

- *Agent-Zone:* It is an organizational agent. Each zone of our CMSC is represented by an agent responsible of providing names and addresses of superior and subordinate areas and providing the future needs of the area and its subordinates, the current stock levels and a forecast upon 7 days. Each *agent-zone* can communicate only with another *agent-zone* that is hierarchically higher/lower to him (an upstream/downstream zone-agent) or with another *agent-zone* from the same hierarchical level. The *Agent-Zone* has to schedule the execution of delivery tasks it receives minimizing total cost and processing time in order to respect due dates.

- *GUI_Agent:* this agent is designed to interact with the users of the system allowing them to dialogue with agents on the platform, pass them information and display results related to each zone. The *GUI_Agent* also feeds the system by data such as stock levels, delivery periods, resources…

- *Weather_Agent*: It is a data agent. This agent gives information related to the environment in a specific area; two types of data were identified: temperature and humidity. The *Weather_Agent* provides an estimate of those data for the actual day of simulation, and the next 6 days.

- *Need Estimating Agent:* The Need Estimating Agent (NEA) is a tool for decision support that is designed to give a future estimation of an *Agent-Zone'*s need in goods (clothes, food, water...). The NEA provides an estimation of these needs, based on data collected from the different agents (*Weather_agent*, *Agent-Zone*...). It mainly works using fuzzy logic calculation[5].

- *Posts_Coordinator_agent and Consumption_agent:* These agents' role is to ensure the smooth functioning of the supply chain. The *Posts_Coordinator_agent* handles haulage and informs areas about sending and reception of packages.
The *Consumption_agent* provides daily each zone with the quantity of supplies consumed by the local population indicated in this area.

- *Integration_and_evaluation_Agent:* this agent has to fusion correctly data in order to compose the global scheduling solution of the whole crisis management supply chain, based on local schedules received from the *Zone-Agents*. The *Integration_and_evaluation_ Agent* uses performance indicators to evaluate the global schedule of the CMSC and selects the agent-zone's schedules to be adjusted.

- *Transport_Management_Agent (TMA):* this agent is responsible of the monitoring and management of the means of transport and the roads network available for the CMSC.

4. Formulation of the Problem

The main concern of the OBAC is to satisfy the regions' needs, respecting delays and minimizing their costs. This is a two-step delivery scheduling problem: first, the local delivery schedules are built by assigning means of transports and routes to the whole supply chain. Then, performance indicators are generated to evaluate the global performance of the supply chain and to identify the assignments' that need to be readjusted, in order to satisfy all connected areas. A zone is satisfied if its request is answered rapidly with a reasonable cost.

We start the description of the mathematical model of the distributed delivery problem by introducing the necessary sets:

- Let R be the set of goods to be delivered, R= $\{r_1, r_2, \ldots, r_u\}$, with u the total number of resources.
- Let S be the set of storehouses, S= $\{s_1, s_2, \ldots, s_h\}$, with h the total number of storehouses.
- Let A be the set of customers or Agent-zones $AgZx_y$, where x is the agent-zone level in the CMSC and y is the rank of the agent-zone in the level x.

The set A is composed of: $A= A_T \cup A_I \cup \{AgZ1\}$:

A_T : the subset of terminal zones. A terminal zone is directly linked to the crisis that erupted. It is the last link of the CMSC.

A_I : the subset of intermediate zones. An intermediate zone is positioned to facilitate the access to terminal zones.

AgZ1: the agent-zone of level one, called also the *Metropolis_Agent*. This agent is responsible for creating all the agents-zones in the CMSC.

$A = \{AgZ1, AgZ2_1, \ldots, AgZ2_{n2}, AgZ3_1, \ldots, AgZ3_{n3}, \ldots, AgZm_1, \ldots, AgZm_n^m\}$, with m is the number of levels of the CMSC, n^e is the total number of agents in the level e.

- Let M be the set of means of transportation (trucks, ships, aircrafts…), M= $\{m_1, m_2, \ldots, m_k\}$, with k the total number of transportation's means.
- Let W_{routes} be the set of available routes, $W_{routes}= \{w_1, w_2, \ldots, w_z\}$, with z the total number of routes.
- Let T be the set of routing tasks to be executed locally, T= $\{t_1, t_2, \ldots, t_L\}$, with L the total number of tasks to be scheduled. A delivery task $t_j \in T$ consists in the delivery of an amount (*delivery_Amount)* of goods (*resourceID)* to a specific zone (the receiver agent-zone: *receiverID)* under some constraints (the delivery date: d_i and the processing time: p_i). A task t_j is formalized as follows:

t_i *: < receiverID;resourceID;delivery_Amount; di; pi >*

5. The Agent-based Parallel and Distributed Scheduling System

5.1. *Main features of the SDS system (Synchronized Distributed Scheduling system)*

Generally, a centralized architecture presents some weaknesses due to its single point of failure and lack of scalability. A centralized system may be easier to implement and manage, but it can rapidly become cause of bottleneck if used for large-scale networks. For these reasons, a distributed architecture is generally considered more suitable for large-sized networks like in our case, while a centralized architecture is preferable for small-scale networks. A distributed architecture delegates the scheduling decision to each node of the distributed supply chain. Thus, it is considered to be a great advantage to decompose or partition the scheduling problem into several simpler interconnected sub-problems which can be more easily solved by the distributed agents. This work focuses on this issue.

Faltings and Yokoo[23] give a survey about the characteristics of distributed methods. We present in this section the features that make our distributed approach appropriate.

- Knowledge management: All data needed for our scheduling problem may be naturally distributed among the set of zone-agents. In fact, each zone-agent is connected to different types of sensors on the field to collect data from the real world. Additional communication is usually needed, but basically, each zone-agent has already the necessary knowledge to

treat the set of tasks that he receives. Also, each zone-agent has different preferences that are hard to articulate through a centralized solver. Gathering all knowledge into one central authority would entail big knowledge transfer costs.
- Robustness: In our distributed approach, a failure in the system is easier to detect. In fact failure of one agent is less critical and other agents can intervene to help finding a solution and maintain the system working without the failed agent.
- Security concerns: In the military logistics, data represent strategic information that should not be revealed to all entities. In fact, each zone-agent belongs to a hierarchical level in the crisis management supply chain and has privacy and security restrictions to respect. Privacy is by far easier to maintain in distributed systems.
- Centralized problem/distributed resolution: Even if our algorithm is designed for distributed problems, many real and centralized problems can be solved using this distributed approach.

5.2. *Flow Processing*

The processing flow of the agent-based algorithm for schedule generation is as follows: Once a crisis occurs somewhere in the world, the metropolis agent receives order to generate dynamically the CMSC zones ((1) in Fig.3). The generation and the positioning of the physical zones is a decision made based on the knowledge of experienced humans (including existing routes and geo-political information, derived from past experience). After creating the physical zones, the *Integration_ and_ evaluation_Agent* is notified ((2) in Fig.3).

Integration_and_evaluation_Agent sends a notification to the TMA with the zones' information ((3) in Fig.3). Agent-zones are generated for each created zone ((4) in Fig.3), and each agent-zone is notified of information on goods for delivery ((5) in Fig.3).

Each agent-zone treats the delivery requests received and asks for initial routes and transport means' allocation from the *TMA* to create delivery tasks ((6) and (7) in Fig.3).

The *TMA* uses the delivery zones information received from the *Integration_and_evaluation_Agent* to allocate the initial routes and means of transportation for each agent-zone to start the scheduling ((8) in Fig.3)

Each agent-zone proceeds to the local scheduling of its tasks, using the resources parameters. In this schedule, delivery deadlines are given priority. Each agent-zone sends the local schedule generated for its zone to the *Integration_and_evaluation_Agent* ((9) in Fig.3). *Integration_and_evaluation_Agent* uses performance indicators to evaluate the global schedule of the CMSC and, selects the agent-zone's schedules to be adjusted ((10) in Fig.3). Each agent-zone uses the indicators specified by the *Integration_and_evaluation_Agent* to modify the parameters of the scheduling (used routes, used means of transportation) and notifies the *TMA* ((11) in Fig.3).

The resource management agent uses the data received from each zone-agent to adjust the required resource for re-scheduling and sends notification of the adjusted delivery tasks ((12) in Fig.3). Hereafter, steps (e) through (h) are repeated until schedules free of inconsistencies are obtained for all the zones, or simulation time ends. A schedule is considered as free of inconsistencies when performance indicators, which are generated by the local plans composing the global solution, do not exceed the benchmark values. Otherwise, the system outputs the last best solution obtained. When there is no still adjustments in resources to be made, the final result of the scheduling is sent to the *Integration_and_evaluation_Agent* ((13) in Fig.3).

If the adjustments are successful, *Integration_and_evaluation_Agent* outputs the schedule results ((14) in Fig.3). If there is a user instruction to stop processing, the system is stopped ((15) in Fig.3).

5.3. *The Rooting-tasks construction*

Each agent-zone starts by sending a request to its direct superior to ask for supply. Once the requests are received ((5) in Fig. 3), the responsible agent-zone determines whatever he can satisfy all the demands or not.

If it is the case, requests are considered as tasks to be performed. Otherwise, the responsible agent-zone determines the real amount that he can satisfy in each resource, and creates new tasks with the new parameters of quantity. The responsible agent-zone calculates the new quantities based on requests' due dates and needs' emergency in the concerned area, after a communication process with his subordinates[6].

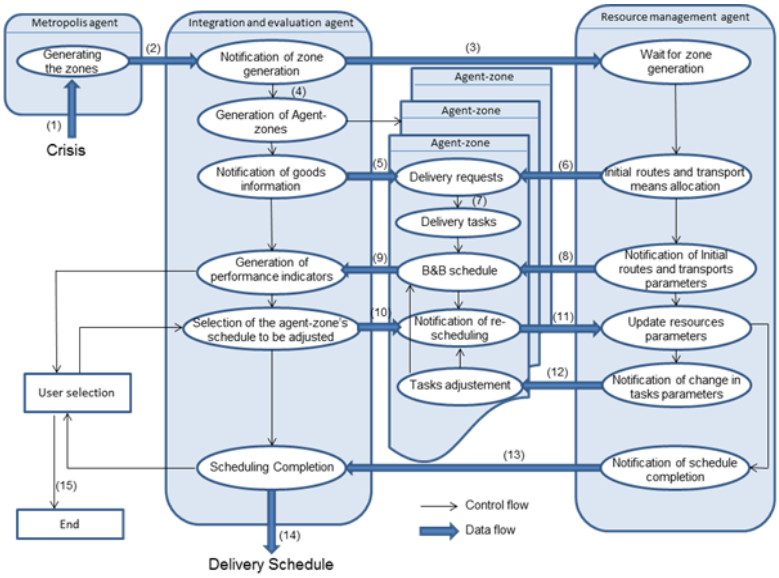

Fig. 3. The synchronized distributed scheduling system

Based on the ideas presented above, the procedure *routingTasksConstruction* gives the list of the tasks ready to be executed.

Let V be the set of requests received by an agent-zone.

Let T be the set of created tasks to be scheduled.

A task $t_i \in T$, and a request $v_i \in V$.

The procedure *createTask*(v_i), $v_i \in V$, determines the amount of resource that can be delivered by the responsible agent-zone for a specific request.

Procedure *routingTasksConstruction*
In: V
Out: T
1: **begin**
2: T= Ø
3: **Do**
4: Select $v_i \in V$
5: t_i := createTask(v_i)
6: V := V- {v_i}
7: T := T+ {t_i}
8: **While** set V of available requests ≠ Ø
9: **end**

When assigning quantities of resources to the tasks is done ((7) in Fig. 3), tasks are waiting to be allocated to transport means and then executed. To do that, the responsible agent-zone proceeds to the scheduling of tasks' execution by minimizing the total tardiness penalty cost. The purpose is to provide an optimal solution for scheduling the routing of supplies to the zones in need. One solution approach was to use the branch and bound technique.

5.4. *Agent-based algorithm for parallel distributed scheduling*

Branch and Bound is by far the most widely used tool for solving large scale NP-hard combinatorial optimization problems. It consists of a systemic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded[7]. When the structure of the scheduling problem such as number of tasks and execution time of tasks are known beforehand, it is interesting to use the B&B algorithm since it implicitly searches the complete space of solutions for the best one. Indeed, explicit enumeration is normally impossible due to the exponentially increasing number of potential solutions, but the use of bounds for the function to be optimized combined with the value of the current best solution enables the algorithm to avoid this inconvenience[8]. The objective is to find a feasible schedule of the execution of the routing tasks for each agent-zone, such that the local total tardiness f is minimized. The scheduling problem can be formulated as follows:

$$\underset{\alpha_k \in Alpha}{Min} \left(f_{\alpha_k} = \sum_{i=1}^{l} \max(c_i - d_i, 0) \right) \qquad (1)$$

With: - c_i = the completion date of the routing task t_i.
- d_i = the theoretical delivery date of the routing task t_i
- l = the total number of tasks of an agent-zone.

- *Alpha* is the set of the nodes α_k in the B&B search tree of the agent-zone.

The search tree Q of B&B technique is a dynamic structure constituted of a set of nodes. Each node α_k is characterized by the following elements:

- a partial assignment *S* of the tasks scheduled. Such an assignment can be noted: $S= t_i \oplus S$; which means adding the task t_i in the sequence S, in the programmed position in the scheduling[9].
- a lower bound *lb* based on the minimum total tardiness time.

The iteration has three main components: selection of the node to process, branching, and bound calculation.

Algorithm1 Local Scheduling algorithm based on B&B with Best-First selection rule.

```
proc GlobOp (G)

begin
    lb= min{f(α0)};
    S :=( );
    G := {α0};
    while G≠ Ø do
        Select the node αk from G to be processed;
        G:= G \ {αk };
        Subdivide (αk, μ(αk));
        v = card(μ(αk));
        lb= min{f(αi)} ∀ αi ∈ μ(αk);
        for i := 1 to v do
            if (f(αi) =< lb)
                then S:= ti ⊕ S;
                     G:= G ∪ {αi};
                     lb:= f(αi);
                     break;
                else nexti
            end if;
        end;
    end;
    UpdateTest(lb,G)
end
```

Branching rule: The solution space of nodes is subdivided into two or more subspaces to be investigated. Then for each of these, the bounding function is calculated and compared to the current best solution keeping the best of these.

Selection rule: The exploration strategy is "Best First, Depth First", in other words, at each step, we branch node with the best lower bound in the search tree, followed by a depth-first search.

Termination rule: The search terminates when there are no unexplored parts of the solution space left, and the optimal solution is then the one recorded as "current best".

The procedure for creating partial solutions is the next: each node α_k in the search tree is defined by a sequence S_{α_k} of already scheduled tasks. $U_{\alpha k}$ is the set of unscheduled tasks at node α_k. The set $\mu(\alpha_k)$ is the set of all the child nodes of α_k, which means the nodes not yet been expended. Branching out from α_k consists in creating new node and copying all the information from the node α_k. Scheduling is done by taking a task t_j from the list $U_{\alpha k}$ and appending it to the programmed position in sequence S.

The procedure *Subdivide* $(\alpha_k, \mu(\alpha_k))$ branches on α_k generating $\mu(\alpha_k)$.

$S: = t_i \oplus S$ means inserting the task $t_i(\alpha_i)$ in the beginning of the sequence S.

$G: = G \cup \{\alpha_i\}$ means inserting in G the next node to be expended.

Since there is no universal bounding algorithm that works for all problems and since there is no such proved lower bound for the specific problem that we are addressing in this paper, a lower bound of every newly created node is calculated in the following function:

$$lb = \underset{\alpha_k \in Alpha}{Min} (f_{\alpha_k})$$

The work tree is then updated by the *updateTest* procedure which at any time removes all the nodes α_k verifying that $f(\alpha_k) > lb$.

Algorithm1 illustrates the generic functioning of the local B&B algorithm for local scheduling, with the following variables: lb: lower bound; S: partial or total scheduling solution (a sequence of scheduled tasks); G: work tree; and α_0: route node (first node to be created).

Once each agent-zone has done the scheduling of his tasks, we obtain a global solution that represents a global scheduling of all the supply chain. The *Integration_ and_ evaluation_Agent* evaluates this solution by using performance indicators to appraise the global cost of the obtained scheduling ((9) in Fig. 3).

5.5. *Performance Indicators Evaluation*

We assign to the CMSC a cost representing the total distribution cost for satisfying the customers. It is composed of fixed performance indicators. To formulate these indicators, we introduce sets of binary variables.

Let p and q be zones belonging to the CMSC, with p the direct hierarchical superior zone of zone q.

The variable concerning the use of means of transportation can be expressed as:

$y_{pq}^{m_j}$ = transportation of flow of product from zone p to its associated zone q, by the mean of transportation m_j.

$$y_{pq}^{m_j} = \begin{cases} 1, & \text{if mean of transportation } m_j \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

The variable representing the transportation flows of products on routes from one zone to another can be defined as:

$x_{pq}^{w_i}$ = transportation of flow of product using route w_i from zone p to its associated zone q.

$$x_{pq}^{w_i} = \begin{cases} 1, & \text{if route } w_i \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

We introduce two variables describing the return flows on routes from zone q back to its direct superior zone p. This flow does not include any products as the transportation means are empty. These variables can be defined as:

$y_{qp}^{m_j}$ = return of mean of transportation m_j from zone q to zone p.

$$y_{qp}^{m_j} = \begin{cases} 1, & \text{if mean of transportation } m_j \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

$x_{qp}^{w_i}$ = return flow from zone q to zone p, on route w_i.

$$x_{qp}^{w_i} = \begin{cases} 1, & \text{if route } w_i \text{ is used} \\ 0, & \text{otherwise} \end{cases}$$

The performance indicators are then expressed as follows:

C^{routes} = transportation cost for the used routes.

$C^{transport}$ = truck, ship and aircraft cost.

C^{delay} = penalty of delay in delivery.

$C^{earliness}$ = cost of warehousing in case of advance in delivery[11].

Let c_{w_i} be the transportation cost for route w_i. Further, let c_{pq} be the transportation cost from zone p to the corresponding zone q. This cost includes the costs of loading and unloading supplies. The total transportation cost for routes can now be expressed as:

$$C^{routes} = \sum_{w_i \in W_{routes}} \sum_{p \in A \setminus A_T} \sum_{q \in A_I \cup A_T} (c_{w_i} + c_{pq}) \left(x_{pq}^{w_i} + x_{qp}^{w_i} \right) \quad (1)$$

Let $c_{pq}^{m_j}$ be the mean of transportation cost, between zone p and zone q. We can express the total truck, ship and aircraft cost as:

$$C^{transport} = \sum_{p \in A \setminus A_T} \sum_{q \in A_I \cup A_T} \sum_{m_j \in M} c_{pq}^{m_j} (y_{pq}^{m_j} + y_{qp}^{m_j}) \quad (2)$$

Let C_p be the cost of delay in delivery per day in zone p. The total penalty of delay in delivery is then expressed as:

$$C^{delay} = \sum_{p \in A \setminus \{AgZ_1\}} C_p D_p \quad (3)$$

with D_p the total days of delay in the delivery of zone p.

Let $c_p^{s_i}$ be the cost of storage in the storehouse s_i of the zone p per day. We can express the total cost of warehousing in case of earliness in delivery as

$$C^{earliness} = \sum_{p \in A \setminus \{AgZ_1\}} \sum_{s_i \in S} c_p^{s_i} E_p \quad (4)$$

with E_p the total days of earliness in delivery of the zone p.

Once each cost is calculated, *Integration_ and_ evaluation_Agent* compares the obtained values to reference costs: C_{Ref}^{routes} , $C_{Ref}^{transport}$, C_{Ref}^{delay} and $C_{Ref}^{earliness}$. Reference costs are fixed based on past experience and the supply chain parameters. Each performance indicator is given a priority degree in the optimization, depending on the actual situation in the zone. A delivery scheduling system must be constructed such that the actual demands all along the supply chain can be satisfied so far as possible.

Based on the values of the different performance indicators, *Integration_and_ evaluation_Agent* detects potential failures in the supply chain. Indeed, if one or more specific indicator exceeds the benchmark value, it implies that there are improvements and adjustments to be made on the proposed solution. The indicators give information about the local schedule to be adjusted and the agent-zone responsible of the overflow. This allows the system to have a sort of traceability of the disturbances in the supply chain. The agent-zone proposes a new routes and means of transports' allocation to generate an adjusted schedule. Then, the steps (e) through (h) of figure 2 are repeated until schedules free of inconstancies are obtained for all the supply chain. If the adjustments are successful, the *Metropolis_agent* validates the global distributed scheduling. If there is a user instruction to stop processing, the system is stopped and the last-best scheduling is chosen. (The generation of the large-scale distributed delivery schedule should be made within a practical time frame).

5.6. *What about convergence?*

Let's discuss the convergence of our solution. The collective convergence of multi-agents has been achieving much attention in many research fields[24-29].

Let's first define the word "strategy": A strategy is the action that agent adopts to behave in the multi-agent society. In our synchronized distributed scheduling system, we assign to each agent an arbitrary initial strategy according to which to behave; but with the time going, there is always a good property that emerges. All agents may finally converge on certain behavior strategy, which is called *collective synchronization*, generating the global schedule of our CMSC. Based on the work of Jiang[30], and for the reason of simplicity, we give a rank to each strategy in the space of strategies S' and we define the strategy as a simple natural number; moreover, we assume that the higher a strategy value is, then the more dominant such strategy is in the collective synchronization.

The collective synchronization of our multi-agents scheduling system always follows several assumptions:

- The synchronization result is the convergence on a common average strategy: all agents' strategies will always converge to a common average strategy of the system which is in our case the global schedule of the CMSC;
- The effects of all agents in the synchronization are identical and all agents have the same synchronization capacity;
- All agent-zones have the same optimization method. The control, input and output data depend on the agent's local perception of its zone in the CMSC and on the strategy of its neighbors.

In order to prove the convergence of our multi-agents scheduling system, we used the definition of Jiuchuan[30] which concern the "Convergence degree of individual strategy":

The convergence of an agent $a_i \in A=\{a_1, a_2, \cdots, a_n\}$, with the strategy ($s_i$) and a predefined tolerance value (ε) :

$Conv(a_i,\varepsilon)=|likeness(s_i,\varepsilon)|/|A|$

Where $|A|$ denotes the total number of agents in the synchronization field, and *likeness*(s_i,ε) : the set of agents that chose any strategy in strategies set S' which satisfies the following situation :

$(likeness(s_i,\varepsilon)\subseteq A)\wedge(\forall a_j \in likeness(s_i,\varepsilon) \Rightarrow W(s_j,s_i)\leq\varepsilon)$

With: $W(s_j,s_i)$: the difference between strategy s_j and s_i : $W(s_j,s_i) = | s_j - s_i |$

Moreover, in real multi-agent societies, there is always an interesting phenomenon: if some agents' social ranks are higher than the ones of other ordinary agents, or their behavior strategies are more dominant than the ones of other ordinary agents (those agents with dominant ranks or strategies are called prominent agents), then the collective synchronization results may converge at such prominent agents' strategies, which is called prominence convergence.

Let's analyse the convergence of the strategy of each agent of our scheduling system:

- The strategy of *the Integration_and_evaluation_Agent* is to allocate the initial routes and means of transportation for each agent-zone. Sufficient conditions are provided which guarantee the convergence of this **ordinary agent**.
- The strategy of the *Integration_and_evaluation_Agent* is to use performance indicators to evaluate the global schedule of the CMSC and, selects the agent-zone's schedules to be adjusted. Sufficient conditions are provided which guarantee the convergence of this **ordinary agent.**
- The strategy of each agent-zone, which is, in our case, considered as **prominent agent,** is to provide an optimized local scheduling by using the branch and bound method. Thanks to the distributed aspect of our problem, the input and output data are in limited quantities and the different tasks to schedule are in limited number so that each agent zone is faced to a small-sized scheduling problem. In addition the algorithm computes the lower bounds during the branch and bound search by solving an ordinary linear programming problem. The bound calculation, the subdivision scheme and the initialization become both simple and efficient. This should economize considerably the computation time and guarantee **the convergence of each of our prominent agent-zones.**

By referring to the related definition in Jing's work[30], we identified the dominant and ordinary agents and made the collective synchronization for the whole system. We conclude that since the prominent agents have the same optimization method they have the same strategy. Therefore the prominent agents generally have the relatively higher convergence degrees than other ordinary agents after the synchronization. In addition the behavior strategies of our prominent agents are more dominant than the ones of other ordinary agents. Thus

the ordinary agents always support the strategies of prominent agents and the prominence convergence phenomenon emerges finally in our system.

6. Simulation Results

Our application is the result of a significant and sustained work by our research team in the French High School Ecole Centrale (LAGIS – EC-Lille) and the logistics department of EADS to implement a logistic flows demonstrator for crisis management. This demonstrator simulates the running of the CMSC with all its actors. The demonstrator has been tested on real and concrete examples, but respecting the confidentiality clause that binds the LAGIS/EC-Lille and EADS, we cannot present the extracted results. So in order to evaluate the solution proposed in this paper and validates the approach for problem-solving distributed scheduling, we applied the methodology proposed on an academic example of a crisis management supply chain.

We are developing our system, with JADE platform (Java Agent Development platform)[12]. JADE is a middleware which permits a flexible implementation of multi-agents systems; it offers an efficient transport of ACL (Agent Communication Language) messages for agents communication which complies with FIPA specifications[13].

6.1. *Example Description*

We consider 3-levels CMSC scale model composed of the following hierarchical zones: 1 AgZ1 (level 1); 3 intermediate zones (level2) AgZ2: {AgZ21, AgZ22, AgZ23}; and for each intermediate zone, 3 terminal zones (level3) AgZ3: {AgZ31, AgZ32, AgZ33}; {AgZ34, AgZ35, AgZ36}; {AgZ37, AgZ38, AgZ39}.

Each zone corresponds to a crisis management logistic base represented by an agent-zone. We remind that the goal is for each agent-zone to satisfy not only its resources consumptions but also those of all his subordinates.

(in days)

	Intermediate zone		***Terminal zone***	
	Regular mode	*Urgent mode*	*Regular mode*	*Urgent mode*
Metropolis	3	2	-	-
Intermediate zone	-	-	5	3

Table 3. Delivery durations between the delivery zones

Figure 4 shows the interfaces for the definition of the number of intermediate and terminal zones and the number of resources (goods) to be delivered all along the CMSC. The daily consumption of resources per soldier and per civilian is also fixed.

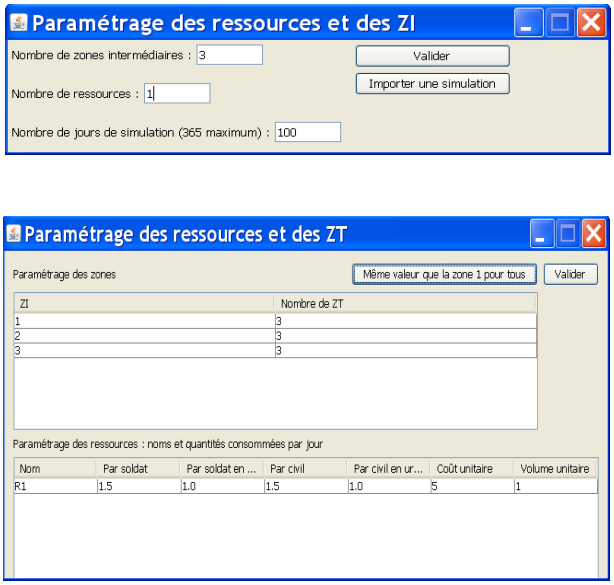

Fig. 4. Interfaces for parameters' initialization

Figure 5 shows the geographical arrangement of the

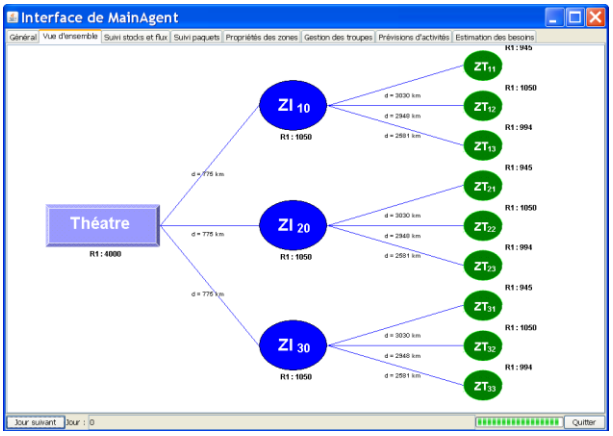

Fig. 5. Geographical arrangement of the CMSC

zones. The *Metropolis_agent* (Théatre) is the delivery center of the whole CMSC. The level-2 zones (ZI_{10}, ZI_{20} and ZI_{30} which corresponds respectively to $AgZ2_1$, $AgZ2_2$ and $AgZ2_3$) are created dynamically and positioned to facilitate the access to the affected terminal zones.

For each zone, we define the initial stocks, the safety stocks, the replenishment stocks, the orders costs and the different supplying times.

Figure 6 shows a schematic representation of the agents' activities with various flexible features for scheduling the delivery tasks: parallel activity, staggered activity, interruption and different duration.

- Parallel activity: the three agents work simultaneously, in a non-sequential way. Each agent generates its local schedule to execute its delivery tasks.

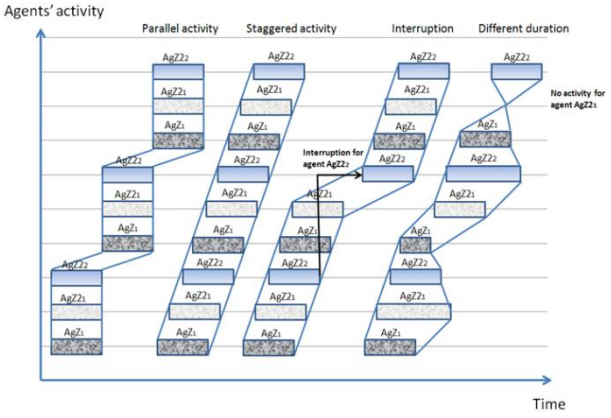

Fig. 6. Agents' activity

An agent waits to finish one scheduling activity to start another one.

- Staggered activity: the three agents have shifted activities with an offset. An agent may start a new scheduling activity while it is still generating a precedent schedule.
- Interruption: in this mode, an agent-zone may interrupt a scheduling activity already started, if it receives an update of the set of tasks to be scheduled.
- Different duration: agents' activities are with different durations. These durations depends on the number of delivery tasks to be scheduled.

6.2. *Results of Experiments*

Each agent-zone starts by sending a request to its direct superior to ask for supply, as it is shown in figure7.

In figure 8, we show a display of the tasks tracking all along the whole CMSC in the day 5 of the simulation. For each delivery task, the sender, the receiver, the resource to be delivered, and the start and end dates are displayed.

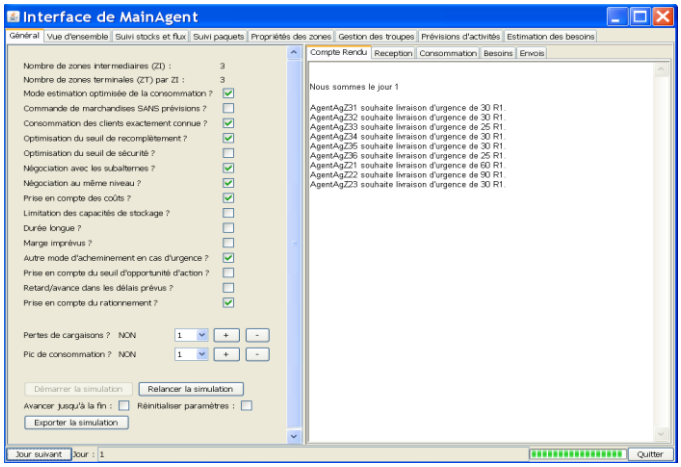

Fig. 7. Example of request generation

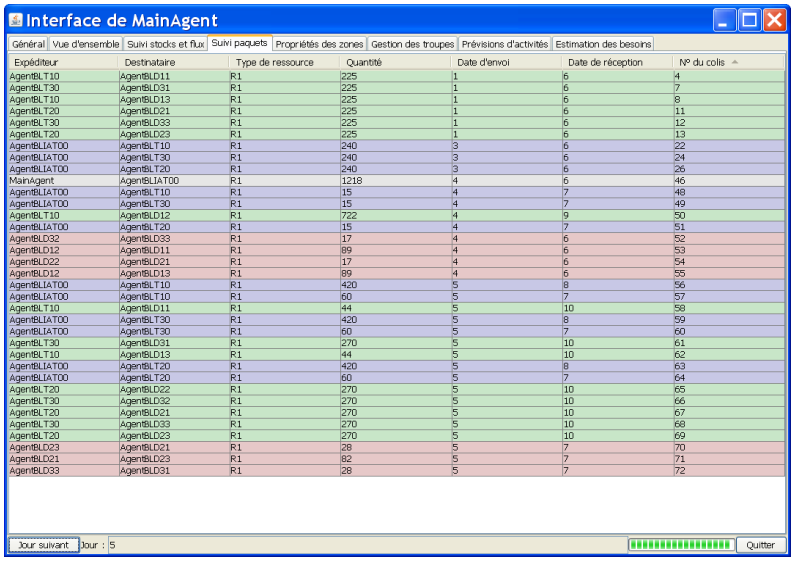

Fig. 8. Interface of tasks tracking

Table 4 reports the performance indicators values for the local schedules obtained with the initial resource allocation. The references values used are obtained in consultation with experts, and depends to a large degree on the geographic positioning of the zones in the CMSC. These values allow to make a tracing and to detect the schedules that need to be adjusted, every time we exceed the reference with a certain margin.

Local Schedule	**Indicator1**		**Indicator2**		**Indicator3**		**Indicator4**		**dicator5**	
	C^{routes}	C^{routes}_{Ref}	$C^{transport}$	$C^{transport}_{Ref}$	C^{return}	C^{return}_{Ref}	C^{delay}	C^{delay}_{Ref}	$C^{earliness}$	$C^{earliness}_{Ref}$
AgZ_1	110	103	513	525	27	20	30	15	0	25
AgZ_{21}	223	214	624	550	42	20	30	15	0	25
AgZ_{22}	**227**	120	660	600	16	20	45	15	0	25

Table 4. Performance indicators values

The preliminary local schedules are then rectified in real time depending to the flow of goods needed and the availability of resources. Adjustments to the scheduling parameters such routes, personnel and means of transport in each delivery zone are made, based on the values of the performance indicators, and an adjusted global schedule is generated.

In table 3, solution with the initial local schedule of agent AgZ_{22} generates a high routes' cost. The SDS system changes the option for route selection and updates the scheduling parameters of the agent AgZ_{22}. A new local scheduling of this zone is then generated based on the tasks adjustments.

Local Schedule	**Indicator1**		**Indicator2**		**Indicator3**		**Indicator4**		**dicator5**	
	C^{routes}	C^{routes}_{Ref}	$C^{transport}$	$C^{transport}_{Ref}$	C^{return}	C^{return}_{Ref}	C^{delay}	C^{delay}_{Ref}	$C^{earliness}$	$C^{earliness}_{Ref}$
AgZ_1	110	103	513	525	27	20	30	15	0	25
AgZ_{21}	223	214	624	550	42	20	30	15	0	25
AgZ_{22}	**164**	120	**690**	**600**	16	20	45	15	0	25

Table 5. Performance indicators values after re-schedulin

We observe amelioration in the value of indicator1in table 5 after the scheduling adjustments. The solution obtained was not necessarily optimal with respect to the five evaluation parameters adopted, but it was confirmed to lie within the range of tolerance.

The Jade platform offers a tool called *sniffer agent* (figure 9) that allows tracking messages exchanged between groups of agents.

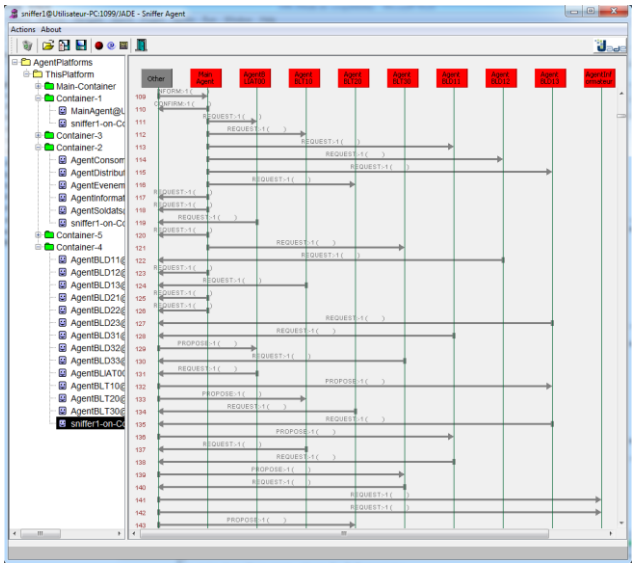

Fig. 9. Agents' communication

The user was provided with a means for selecting to launch the scheduling system for the rooting of the tasks to the zones or not. So that, figures 10 and figure 11 show the results obtained with and without the scheduling solution proposed.

The graphs of figure 10 and 11 represent the inventories evolutions (the y-axis) according to the time, expressed by the number of days (the x-axis). In each graph, the parallel line to the x-axis represents the security inventory level for the given resource.

Furthermore, we distinguish 3 different phases corresponding to the graph evolution:

-Phase 1: represents the past inventory evolution until today: day n°14;

-Phase 2: represents the future previsions taking into account the cargos sent by the providers; between day n°14 and day n°17 for the example agent-zone;

-Phase 3: represents the future previsions of the inventory evolution, according to the future demands not yet sent by the providers; from day n°17 for the selected agent-zone.

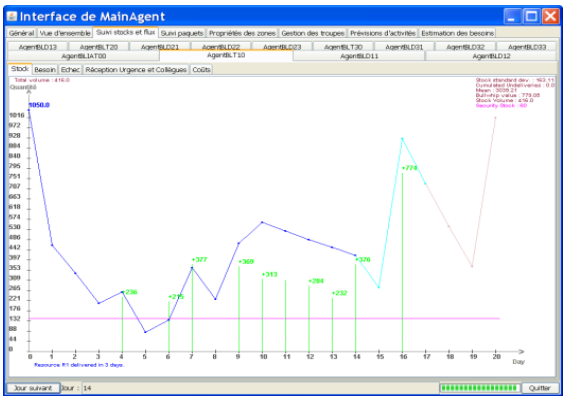

Fig. 10. Stocks evolution for AgZ21 before scheduling

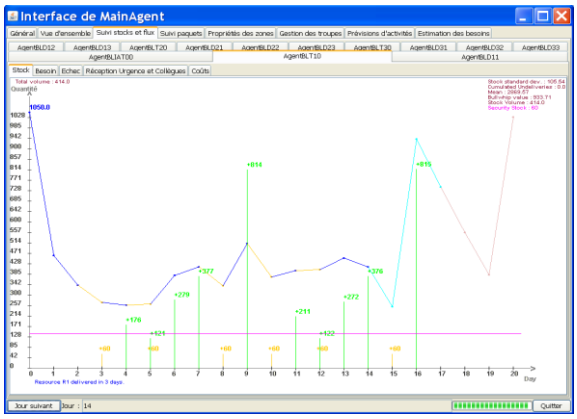

Fig. 11. Stocks evolution for AgZ21 after scheduling

The delivery scheduling method using distributed cooperation between agents has shown its reliability.

We observe that in the figure 11, the delivery scheduling has improved the stock out risk for the selected agent. Indeed, we observe that the safety threshold is initiated before the reception of the delivery (day 5).

7. Conclusion

In this paper, we proposed an intelligent system for delivery schedule based on framework for distributed cooperation. We have confirmed its usefulness and pointed out that some of our objectives had been attained. This work introduces the possibility to satisfy the need in resources all along the CMSC, while

minimizing penalty costs and maintaining a smooth circulation of flows. For the future work, we believe that there is a need of further improvement to find out a new lower bound, and also improvement of coding that may save memory and CPU time. We intend also to study anticipation models on a broader version of the CMSC. We also consider studying the usage of Mobile Agents paradigm in the transport information system [10].

References

1. D. Perugini, D. Lambert, L. Sterling and A. Pearce, "Agents for Military Logistic Planning", In Proceeding of the 15th European Conference on Artificial Intelligence (ECAI-2002), July 21-26, 2002, Lyon, France.
2. *Advanced Logistics Project Homepage*, Defense Advanced Research Projects Agency. http://www.darpa.mil/iso2/alp
3. T. Carrico and M. Greaves, "Agent Applications in Defense Logistics", Defense Industry Applications of Autonomous Agents and Multi-Agents Systems, Whitestein Series in Software Agent Technologies and Autonomic Computing, 51-72, 2008.
4. BU. Nyugen, DF. Reding and D. Nyugen, "An engagement model to optimize defense against multiple attack assuming perfect kill assessment", *Naval Research Logistics*, 1997, 44(7), 687-697.
5. N. Zoghlami and S. Hammadi, "Estimator Agent approach for distributed logistic chain optimization", *In Proceedings of the 50th IEEE International Conference ANIPLA'2006, Rome (Italy), 13-15 November 2006.*
6. A. Kaddouci, H. Zgaya, S. Hammadi, F. Bretaudeau, "PAAN: Partial Agreement Negotiation Network based on Intelligent Agents in Crisis Situation", *International Journal of Mathematics and Computers in Simulation*, Issue 4, Volume 3, December 2009.
7. M. M. Rahman, and M. F. I. Chowdhury, "Examining Branch and Bound Strategy on Multiprocessor Task Scheduling", *In Proceedings of 2009 12th International Conference on Computer and Information Technology (ICCIT 2009)*, 21-23 December, 2009, Dhaka, Bangladesh.
8. J. Clausen, "Branch and bound algorithms principles and examples", *Department of comp sc., university of Copenhagen,Universitetsparken 1, DK-2100 Copenhagen*, 12 March, 1999, Denmark.
9. H. Zgaya, D. Tang, S. Hammadi, F. Bretaudeau. "Negotiation Protocol According To The Perturbation Impact In a Multi-Agent Supply Chain System For The Crisis Management", *In proceedings of the IEEE International Conference on Intelligent Agent Technology (IAT-08)*. Decembre 9-12 Sydney, Australia.
10. H. Zgaya, S. Hammadi. "Distributed Optimisation using the Mobile Agent Paradigm through an adaptable Ontology: multi-operator services research and composition", chapitre en anglais publié en 2009 dans un livre intitué "Multiagent Systems" I-Tech Publications, Artificial Intelligence Series, ISBN 978-3-902613-51-6, pp397-426.
11. H. Gunnarsson, M. Rönnqvist and D. Carlsson, A combined terminal location and ship routing problem. *Journal of the Operational Research Society (2006) 57*, 928-938.
12. Java Agent DEvelopment framework. http://jade.titlab.com/doc
13. Foundations of Intelligent Physical Agents. www.fipa.org
14. C L Liu and J W Layland (1973). Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment. *Journal of the Association for Computing Machinery*, Vol. 20, No. 1, pp. 46-61.
15. A . Daknou, H. Zgaya, S. Hammadi and H. Hubert (2008). "Towards a tool for scheduling tasks of multiple skills in emergency services". *International Francophone Conference of Automatique CIFA*, Bucharest, Romania.
16. K. Ghosh (1994). A survey of real-time operating systems. *Technical Report GIT-CC-93/18, Georgia Institute of Technology, Atlanta, Georgia.*
17. J. Blazewicz, P. Dell'Olmo and M. Drozdowski (2002). "Scheduling multiprocessor tasks on two parallel processors". *RAIRO Operations Research 36*, 37-51.
18. C. Garcìa- Magarino Gutiérrez and R Fuentes-Fernàndez. The INGENIAS Development Kit: a practical application for crisis-management. The 10th International Work conference on Artificial Neuronal Networks (IANN2009).
19. C. Beardm and V.S. Frost, (1999) "Dynamic Agent-Based Prioritized Connection Admission for Stressed Networks", IEEE International Conference on Communications.
20. G. Barbarosoglu, L. Ozdamar and A. Cevik (2008). An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. European Journal of Operational Research 140, 118-133.
21. H.O. Mete and Z.B. Zabinsky (2010). Stochastic optimization of medical supply location and distribution in disaster management. International Journal of Production Economics 126 (1), 76-84.
22. Y.C. Chiu and H. Zheng (2007). "Real time mobilization decision for multi-priority emergency response resources and evacuation groups: model formulation and solution". Transportation Research: Part E43, 710-736.

23. B. Faltings and M. Yokoo (2005). Introduction: speccial issue on distributed constraint satisfaction. Artificial Intelligence 161, 1-5.
24. Reynolds. C. W. Flocks, herds, and schools: a distributed behavioral model. Computer Graphics, 4(21): 25-34, 1987.
25. J. Toner and Y. H. Tu. Long-range order in a twodimensional dynamical XY model: how birds fly together.Physical Review Letters, 23(75): 4326-4329, 1995.
26. T. Vicsek, A. Czirok, E.B. Jacob, and Schochet. Novel type of phase transitions in a system of self-driven particles. Physics Review Letter, 6(75): 1226-1229, 1995.
27. A. S. Mikhailov and D.H. Zanette. Noise-induced breakdown of coherent collective motion in swarms. Physical Review E, 4(60): 4571-4575, 1999.
28. A. Jadbabaie, J. Lin, and A.S. Morese. Coordination of groups of mo-bile autonomous agents using nearest neighbor rules. IEEE Transactions on Automatic Control, 6(48): 988-1001, 2003.
29. Z. Lin, M. Broucke and B. Francis. Local control strategies for groups of mobile autonomous agents. IEEE Transactions on Automatic Control, 4(49): 622-629, 2004.
30. J. Jiang and Y. Jiang. Convergence at Prominent Agents: A Non-Flat Synchronization Model of Situated Multi-Agents (Short Paper). Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), Padgham, Parkes, Mueller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp.1493-1496
31. A. Kadoussi, H. Zgaya, S. Hammadi, E. Duflos and P. Vanheeghe. " Generic need estimating agents for resources forecasting"**.** The 16th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2012. July 17th - 20th, 2012 – Orlando, Florida, USA.
32. J. Erceau and J. Ferber, " L'Intelligence Artificielle Distribuée", La recherche, vol. 22, pp. 750-758, 1991.
33. J.Ferber. "Les Systèmes Multi-Agents vers une intelligence collective ". InterEditions, 1995.