Dynamical Handling of Straddle Carriers Activities on a Container Terminal in Uncertain Environment - *A Swarm Intelligence approach* -

Stefan Balev, Frédéric Guinand, Gaëtan Lesauvage*, Damien Olivier

Université du Havre, LITIS EA 4108
BP 540, 76058 Le Havre, France

E-mail: {stefan.balev,frederic.guinand,damien.olivier}@univ-lehavre.fr,
gaetanlesauvage@gmail.com

***Abstract*—The CALAS project consists in a laser measure system allowing to localize precisely straddle carriers in a container terminal. The information given by such a tool makes an optimization possible. In fact, a box terminal is an open system subject to dynamics, in which many events can occur. Among others, they concern container arrivals and departures. Within the terminal, straddle carriers are trucks which are able to carry one container at a time in order to move it through the terminal. We aim to optimize the straddle carrier handling in order to improve the terminal management. Moreover, missions come into the system in an unpredictable way and straddle carriers are handled by humans. They can choose to follow the schedule or not. For these reasons, the exact state of the system is unknown. The optimization process that we try to build must be fail-safe and adaptive. In this context, we propose an approach using a meta-heuristic based on Ant Colony to resolve the problem of assigning missions to straddle carriers. We built a simulator which is able to test and to compare different scheduling policies.**

***Index Terms*—swarm intelligence, colored ant colony system, dynamic graph, multiple criteria optimization, vehicle routing problem, container terminal.**

I. System description

The CALAS project aims at localizing precisely handling trucks on a box terminal. It uses a laser localizing system and software which allows to deal with the data sent by laser sensors. This project is the result of a collaboration between *Laser Data Technology Terminal* company and the *Terminaux de Normandie* company. The goal of the CALAS project is to know the state of the terminal in real time, meaning both containers and vehicles location.

A container terminal is divided into three main areas (see Fig. 1). Each part is a set of box rows where containers can be stacked up and these areas are linked by oriented roads. The first area is the quayside. It is beside a channel where ships can tie to the dockside. It is an area bound to prepare the ship (un)loading. The second area, the landside, is used to load or unload trucks and trains. The third part is a storing area linking the two others. Containers are moved into this area when a ship, a truck or a train is unloaded, and containers are moved from this area when a ship, a truck or a train is loaded. Managing a box terminal involves three kinds of tasks:

- Preparing a ship (un)loading;
- Preparing a truck (or a train) (un)loading;
- Optimizing storing area.

In order to accomplish these tasks, containers are moved from one position to another. Such moves are called missions. Each mission is assigned to a straddle carrier.

Fig. 1. Terminal de Normandie, Le Havre, France[1].

The container terminal is an open system subject to dynamics. Though a subset of missions is known before starting the schedule, new missions arise when the schedule has already been established and its execution has started. Moreover, trucks arriving time is not known precisely enough to forecast container delivery. If a truck is late, the straddle carrier which has to load or unload it, could be assigned to another mission instead of staying idle and waiting for the truck. Human behavior also affects the system because straddle carriers are handled by human drivers who can choose to follow the schedule or not.

II. Related Work

In such a system, the turn around time of both vessels and trucks/trains have to be as small as possible. Three different ways have already been used to solve this real problem.

* Corresponding author.

[1] source: http://www.t-n.fr/tn.htm

First, the analytical approach is based on a study of interrelated factors which have to be taken into account to improve the efficiency of the system. In [1], an integrative decision support system is described. It has been created by studying inter-related decisions made daily in a container terminal. The authors evaluated the system at a terminal in Hong Kong and measured a reduction of 30% of the ships turn around time and the costs of container handling have dropped by 35%.

The second approach is the simulation. It consists in building a simulator which is able to test several methods of optimization. In [2], the authors have used both a genetic algorithm and a neural network system for the regulation of container yard operations. With 2 berths, 64 blocks, a planning period of 24h and a forecast period of 3 days, their simulation had shown a reduction of the total ship waiting time from 64h to 46h. In [3], the authors tried to improve the performance of the Rotterdam's Maasvlakte port area in studying its design. Their simulation gives information about quay length, storage capacity and handling and transport equipment of the terminal. Their results are useful for designing the next terminals.

The last approach is the multi-agent system (MAS). Thurston and Hu [4] aimed at improving the performance of the terminal by a dynamic and cooperative rescheduling of quay cranes and straddle carriers. Here each part of the system is considered as an autonomous agent able to take decisions according to the information of its own environment. Henesey *et al.* [5]–[8] have developed this idea. Their agents try to reach their own goal by searching, coordinating, communicating, and negotiating with other agents. They take their decisions according to a market based mechanism. Like in an auction, they bid for winning a task. Their system allows to test several policies of berthing, stacking or sequencing. They figured out that good decisions about stacking and berth allocation impact positively on the vessel turn around time.

According to these last conclusions, it appears that optimizing the performance of a container terminal means handling the vehicles' moves and their missions allocation. In this context, we deal with a vehicle routing problem.

III. Vehicle Routing Problems

Vehicle routing problems (VRP) are largely studied and represent practical interest since they appear in many industrial processes. In general, VRP can be formulated as follows. One or many vehicles must start from a depot, visit a set of customers, delivering (or picking-up) some goods, and come back to the depot. The aim is to minimize the vehicles' routes. Many different subproblems belong to the VRP class, such as Capacitated Vehicle Routing Problem (CVRP) or Vehicle Routing Problem with Pickup and Delivery (VRPPD) for instance. Every subproblem contains a little variation of the main one, for example, there can be many depots, or vehicles must respect time windows... We distinguish static and dynamic instances of these problems because the methods to solve them are different.

A. Vehicle Routing Problem with Time Windows (VRPTW)

The Vehicle Routing Problem with Time Windows [9] (VRPTW) consists in visiting a set of cities by a set of capacitated vehicles, optimizing overall path length. For instance, an Italian factory produces toys. It has to deliver a set of stores spread all over the country and goods are carried by trucks. Trucks capacity is restricted and they all start from the factory depot. Deliveries can only be done during a defined time interval. If a truck comes too early, it will have to wait. A solution to this problem should minimize the global length of the trucks runs.

The Dynamic VRPTW (DVRPTW) includes dynamics of the new orders. For the above example, if the stores can ask for deliveries when an already scheduled plan is running, then this problem belongs to DVRPTW class.

B. Pickup and Delivery Problem (PDP)

According to [10], PDP contains three subclasses:

1) Many to Many Pickup and Delivery Problems (M-MPDP) : Here, the vehicles have to pickup many objects to many locations. This kind of problem still relatively neglected because it is not frequently present in real situations.

2) One to Many to One Pickup and Delivery Problems (1-M-1PDP) : In this class, there are two different directions for the goods. They are first delivered to the customer. When the customer has done with them, he will ask for bringing back the goods to the depot. These problems may be with single or combined demands. In the first case, each customer asks either for a delivery or a pickup. With combined demands, the same customer can ask for both a delivery and a pickup.

3) One to One Pickup and Delivery Problems (1-1PDP) : This is the main subclass of Pickup and Delivery Problem, meaning the most frequently encountered problem in real life. It deals with picking-up one object at one location and delivering it to one destination. The main problem of this class is the Vehicle Routing Problem with Pickup and Delivery (VRPPD). In this problem, we have to compute the best routes for a fleet of vehicles in order to move objects on a graph. Every route has to start and to end at the depot. The difference to a 1-M-1PDP is that here, each object has its own pickup and delivery location.

When the problem deals with people, it is called Dial-A-Ride Problem (DARP). Some particular cases of VRPPD problems like the Stacker Crane Problem (SCP) are also common in practical life. This is a single vehicle with unit capacity problem. In another subproblem, vehicles are allowed to temporarily drop their loads on specials locations called transshipment points to be able to answer customers demands faster. This is called Vehicle Routing Problem with Pickup, Delivery and Transshipment.

When some requests are not known in advance the above static problems may become dynamic. Those Dynamic Pickup and Delivery Problems [11]–[13] (DPDP or DVRPPD)

consist in optimizing vehicles routes in order to pickup a load somewhere, then to deliver it to its destination, adapting these routes to the new incoming orders without recomputing from scratch. Most of the time, DVRPPD has to handle time windows (DVRPPDTW). Indeed, to start a mission, vehicles have to wait the beginning of its time window. If it is not respected, the vehicle will have to wait for the right time and, meanwhile this vehicle becomes useless.

As we have just seen, the Vehicle Routing Problem class contains a lot of different subproblems. It becomes very important to exactly identify our own problem.

C. Identification of our problem

In our problem [14], several vehicles (straddle carriers) of unit capacity must accomplish missions (by moving containers within the container terminal). They can also use transshipment location to make the tasks more efficient. A very specific aspect of our system is that the straddle carriers can start from anywhere, *i.e.* they do not have to start from the depot. Moreover, every mission has a time window in which the container must be delivered. If a vehicle comes too early for picking up or delivering a container, it will have to wait the beginning of the missions time window. Furthermore, if a straddle carrier is late, meaning its time window is already closed, in some cases, the mission must be aborted and a new one dealing with the same container will appear into the system.

For all these reasons, our problem belongs to the Dynamic Vehicle Routing Problem with Pickup and Delivery and Time Windows (DVRPPD-TW).

Three interconnected problems must be solved:

- Minimize straddle carriers moves: shortest path problem
- Minimize resources: clustering problem
- Minimize customers delays: scheduling problem

In order to construct a good schedule, the system must integrate the shortest path concept. In the same time, scheduling shortest paths tends to reduce straddle carriers moves. Moreover, we have to define a quality of service level to satisfy customers while lowering operation costs. This is a dynamic large scale problem which requires a real time solution. We propose an on-line algorithm based on Ant Colony Optimization [15], [16] and more precisely on a colored version of this swarm algorithm [17].

IV. Ant Colony and Straddle Carrier Handling

Ant Colony [15], [16] is a meta-heuristic which makes a solution appear thanks to the run of artificial ants into the solution space. The system is self-regulated. In fact, ants spread pheromone according to the solution quality (positive feedback) but the pheromone tracks evaporate progressively (negative feedback). The positive feedback makes the algorithm converge to a quality solution, and the negative feedback prevents it to trap into a local extremum.

Ant Colony with one colony provides a sorted list of missions to accomplish [18]–[20]. The problem is to set a mission to a specific straddle carrier.

We propose to employ a solution using colored ants [17]. In our model, every straddle carrier represents a colony with its own color. Convergence is assured by the fact that ants are attracted by the pheromone of their own colony and repulsed by the pheromones of foreign colonies. This approach simulates a mechanism of collaboration and competition between colonies and will provide a sorted list of missions for each straddle carrier.

A. Modelling

1) Graph construction: Our algorithm uses a graph representation of the problem. In this oriented graph, every vertex represents a mission. We first build a precedence graph. We say that a mission is prior to another if its time window starts before the one of the other mission. Once this precedence graph has been built, a colored node is added to the graph for each straddle carrier. Those vertices are linked to every compatible mission by an arc of the same color. Next, the arcs added during the precedence graph construction are colored according to the compatibility between the straddle carrier and the missions. In fact, if two missions, linked by an arc in the precedence graph, match with the straddle carrier of color c, then we color the edge between them with the color c. If there is already a colored arc between these two nodes, then instead of changing the color of this arc, we add a new one colored with the color c. At the end, if uncolored arcs remain, they are removed from the graph. So we obtain a multi-graph allowing to run our colored ant colony algorithm.

- Straddle Carriers:

Name	Color
s0	green
s1	blue

- Missions:

Name	Start	End	Matching vehicles
m0	5:00	6:00	s0 s1
m1	5:30	6:00	s0
m2	7:00	9:00	s0
m3	6:00	7:30	s0, s1

Fig. 2. Example of a simple instance of our problem

2) Example: Consider a simple instance of our problem where two straddle carriers have to execute four missions. The compatibility between these vehicles and the missions are as in Fig. 2. So we first build the precedence graph (see Fig. 3). Then we add the straddle carriers nodes (see Fig. 4). Finally we color the arcs as described above. The Fig. 5 shows the multi-graph obtained using this procedure.

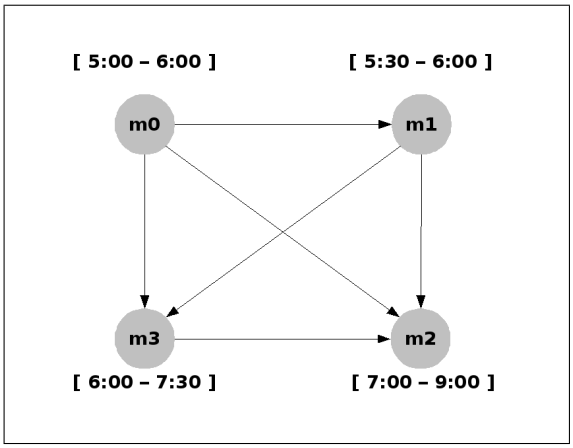

Fig. 3. Precedence graph of the problem described in 2

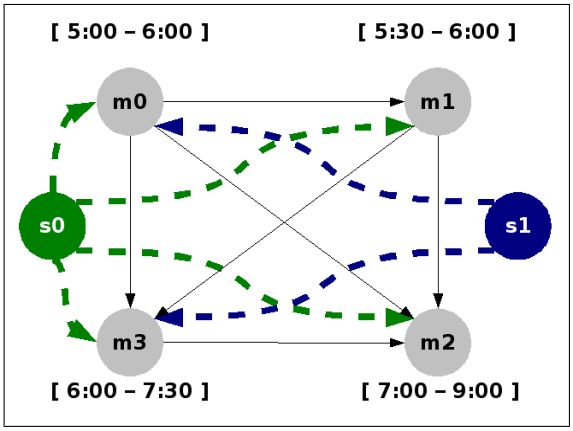

Fig. 4. The vehicles nodes are added to the precedence graph

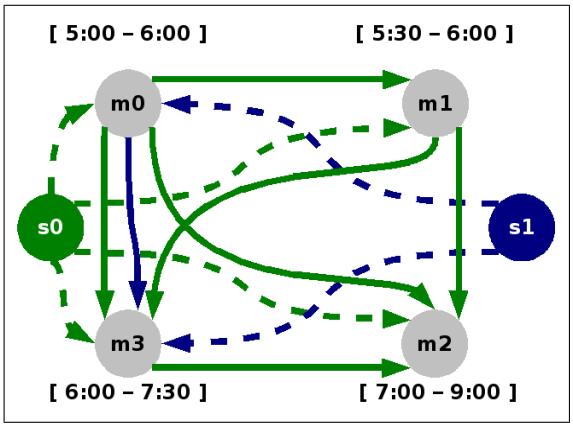

Fig. 5. Mission graph for 2 straddle carriers and 4 missions

3) Arcs weighting: We introduce arc weights which influence the ants when moving in the graph. The weight of an arc measures how efficient it is to assign the two missions connected by the arc to the same carrier. This part of our model provides flexibility and allows to test different weighting policies.

Our policy takes into account both the cost of the mission execution and the time windows proximity. Indeed, if two missions have time windows which are too close, and if they are assigned to the same straddle carrier, then the execution of the first mission will cause the overrunning of the time window of the second mission. It is really important to prevent these phenomena by modelling the linking penalty. We also define a concept of priority. The more the end of the time window is close, the more the priority is high. The weighting function of the arcs takes into account also the distance between the delivery location of the first mission and the pickup location of the second one.

B. Colored Ant Colony Algorithm

In this algorithm, each straddle carrier has a corresponding colony of the same color. Each colony starts from the node representing its straddle carrier. Then, the ants move in the graph using only the arcs of their color. When an ant is in a node, it chooses the next node to visit according to three factors:

- the pheromone rate of its own color
- the pheromone rate of foreign colors
- the weight of the arc

The ant is attracted by the pheromone of its color and repulsed by the pheromone of different colors. Once an ant have reached the chosen node, it spreads pheromone according to the quality of this choice. When a straddle carrier of color c asks for a new mission, it chooses the mission which has the highest rate of pheromone of color c. The overall description of algorithm is shown on Fig. 6.

```
1: for all colony c do
2:     for all ant of colony c do
3:         choose an unvisited destination
4:         move towards it according to the ant speed
5:         spread pheromone
6:     end for
7: end for
8: evaporation
```

Fig. 6. Colored Ant System main algorithm

Our ant colony approach is relevant for solving the considered problem because of its dynamic nature, the large size of the solution space and the real time constraint.

The main asset of ant colony is to provide an anytime solution. It is an on-line algorithm which adapts easily to the changing environment. Indeed, ants reinforce the pheromone rates to get closer to the best solution. At the same time, evaporation process provides a feedback control of the algorithm by preventing it to get stuck into a local optimum and allowing dynamic events to be handled.

Ant colony deals with many parameters such as evaporation, solution evaluation, ants quantity and speed, dynamic events, etc... Here is the major weakness of this metaheuristic. Solution quality strongly depends on these interdependent settings. We have tried to make these parameters self-adaptive. We use a local method to adapt some of these parameters on-line.

C. Division of labor

As there are several distinct colonies and each ant has only a local vision of its environment, there is no way to use a pheromone spreading process based on a global characteristic. In fact, in this architecture, a colony cannot compare the quality of its own solution to the solutions of the other colonies. So, we must use the same pheromone spreading process for each colony. However, we are able to adapt the quantity of pheromone spread by ants of a colony according to the corresponding vehicle skill for a task. Indeed, we observed that we can reduce the serving time of mission by specializing the vehicles into a kind of missions.

We can increase the quantity of pheromone spread by a given vehicle for tasks concerning a specific area in the container terminal and decrease the quantity spread for the tasks located into the other areas. At the same time, we do the opposite for all other vehicles. In this way, we try to specialize the vehicle in a kind of tasks and we are able to regulate these quantities by taking into account both the preference and the distance criteria.

This regulation keeps the benefit of allowing a vehicle to take a mission for which it is not specialized. It is really important in some cases where the number of missions is high because this regulation prevents the system from having unused vehicles in an area of the terminal and unaffected missions, close of the end of their time window, in an other area.

This original approach has a limit. In fact, the time needed by the adaptive system for affecting a vehicle to a mission which does not belong to its specialization may be considerable. For this reason, the system may become less responsive than with no specialization.

D. Reducing resources

Always in a cost lowering purpose, we try to decrease the number of straddle carriers in the system. Our current solution to the entire problem tends to distribute the missions upon all the vehicles. So, every vehicle has almost the same activity rate. But if this rate is under a defined lower threshold, it is possible to conclude that a vehicle could be removed. Otherwise, if the rate is greater than the upper threshold, it is possible to say that a new vehicle should be added to the fleet.

The thresholds must be computed by taking into account several facts. First, it has to deal with the quality of service. Indeed, the system must answer the requests before the end of their time windows. Furthermore, if a vehicle is ready to serve several missions before the beginning of their time windows, it means that this vehicle is maybe superfluous, and the thresholds must be modified accordingly. On the other hand, the target rate has to deal with other criteria like the covered distance of a vehicle per mission or the ratio between the number of vehicles and the number of missions, and it has to set these criteria against the penalties of transcended time windows. Measuring the time of inactivity of every straddle carrier may also lead the optimization. Concerning this last criterion, we must interrelate the time of inactivity with the penalties of transcended time windows.

So, as for the missions arrivals into the system, the number of vehicles is subject to dynamicity. A vehicle can break down and then must be sent to the maintenance. In function of the failure seriousness, we can estimate the time needed to repair the vehicle and so make it available for routing. We take a rate of fault into account for optimizing the number of vehicles into the system because if this number is as low as possible without transcending some time windows, it will become too low if one vehicle of the fleet breaks down.

V. Simulator

The simulator has two main parts. The first one is the terminal simulation (see Fig. 7), and the second one is the Colored Ant Colony Optimization System (see Fig. 8). The first part contains an implementation of the terminal structure and components. Roads and crossroads provide the network of the terminal on which straddle carriers will be able to go. Some of these roads may contain containers. Quay crane locations are represented by these specials roads, so as the trucks handling locations. This terminal is built at the very beginning of the simulation. A scenario file is read to set the terminal configuration. The second part of the simulator contains the algorithmic view of the simulation, *i.e.* the dynamic mission graph. In this way, it shows how the missions are chosen by the vehicles. This part of the simulator uses GraphStream[2] toolkit which allows to handle dynamic graph easily [21].

The simulator uses a discrete time engine which has to iterate every object of the simulation on every time step. Throughout the simulation, the scenario file is read and some dynamic events are sent back to both terminal and Ant Colony views. In this way, the system can simulate the dynamicity of the incoming missions and of the vehicles availability.

In order to have relevant tests and results, we have to define several levels of dynamicity. In [22], Allan Larsen points out two main ways to measure the degree of dynamicity.

First, the degree of dynamism (dod) [23] is the ratio between the number of dynamical requests and the total number of requests. The main weakness of this measure is that it does not take into account the arrival time of these requests into the system. Indeed, with dod if the requests come into the system at the beginning of the day, the system is as dynamic as if they come late in the day. Yet, the later these requests are known, the shorter is the delivery delay. This lateness impacts on the performance of the system.

For this reason, Larsen and al. in [22] defined the effective degree of dynamism ($edod$) by this formula:

[2]http://graphstream-project.org/

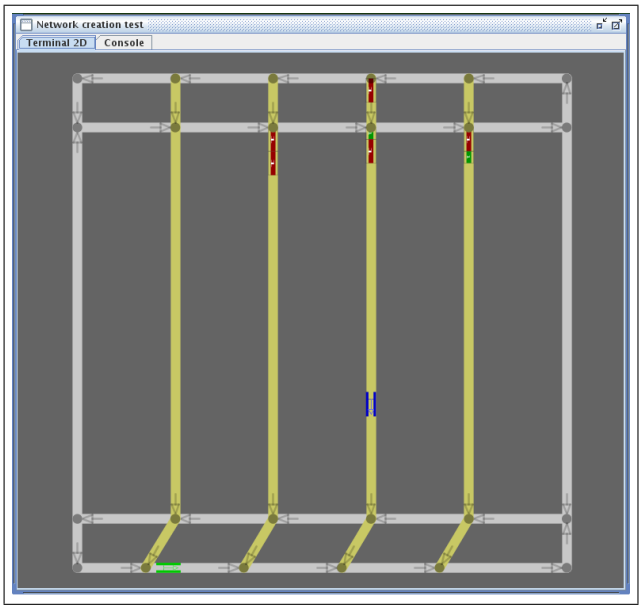

Fig. 7. Terminal view in the simulator

$$edod = \frac{\sum_{i=1}^{\eta_d} \frac{t_i}{T}}{\eta_d + \eta_s} \quad (1)$$

Here, η_s and η_d are respectively the number of static and dynamic requests, and t_i is the arrival time of request i (with $0 < t_i < T$) and T is the time of the simulation end. These measure take into account the average of the incoming time of the requests into the system. The more the dynamical requests come late, the more $edod$ will be high. If $edod = 0$, then the system is totally static. Else, if $edod = 1$, then the system is purely dynamic.

Every straddle carrier on the terminal simulation receives a schedule from the Colored Ant Colony System. Then they act in function of it and move to their pick-up location. Once they have picked-up their container, they move to the delivery location to achieve their mission. At the same time, the mission graph is dynamically updated and the colored ants keep colonizing it.

Simulator gives information about each mission like its length, container, straddle carrier

VI. Preliminary Results

As we still collecting real data from our partners, we are just able to test the relevance of our modelling and of our algorithm in simulated data. For this purpose, we have first run a simulation with a static context, which means that every mission is known at the very beginning of the simulation and that the resources are always available. In a second time, we have added dynamic events such as new incoming missions. For each simulation we have measured the global time needed for achieving all the missions, the number of overrun time windows and the global overrun time. We consider that a time window has been overrun if the non respect of this time window represents a penalty for the container terminal. Indeed, if we overrun the time window of a mission in which we have to move a container from or toward a truck for instance, then the truck will ask for a compensation.

	Static	Half Dynamic	Dynamic
dod	0	0.5	1
$edod$	0	0.25	1
End time	22693	22276	22693
Number of overrun tw	3	5	7
Overrun time penalty	6467	8477	12485

Fig. 9. Results of simulations

Figure 9 shows the results of three instances containing 12 missions and 3 straddle carriers. The only difference between these instances is their degree of dynamicity. Indeed, we have only changed the arrival time of these missions into the system to make them more or less dynamic.

As we can see in Fig. 9, our algorithm seems to act as expected. It means that the more the missions are known in advance, the better is the performance. The worst case occurs when the mission is known at the very beginning of its time window. These are preliminary results and we have not tested all the parameters of the ant algorithm yet.

VII. Conclusion

The problem considered in this paper belongs to the Dynamic Pickup and Delivery Problem with Time Windows class. However, it does not exactly fit. So it is an original unsolved problem. We propose to solve it using swarm intelligence method. An Ant Colony System is being developed. It uses colored ants and a graph modelling in order to plan a schedule. Moreover, we are trying to minimize the number of vehicles into the fleet in order to both maintain a sufficient quality of service and reduce costs. A simulator able to reproduce the behavior of such a system and to handle dynamic events is being developed. Our preliminary results confirm that our algorithm is able to handle dynamicity and we are actually collecting data in order to compare the performance of our system into a container terminal environment with the current scheduling methods used in a terminal of the seaport of Le Havre in France.

References

[1] K. G. Murty, J. Liu, Y.-w. Wan, and R. Linn, "A decision support system for operations in a container terminal," *Decis. Support Syst.*, vol. 39, no. 3, pp. 309–332, 2005.

[2] C. Jin, X. Liu, and P. Gao, "An intelligent simulation method based on artificial neural network for container yard operation," *Lecture Notes in Computer Science*, vol. 3174, pp. 904–911, 2004.

[3] J. A. Ottjes, H. P. M. Veeke, M. B. Duinkerken, J. C. Rijsenbrij, and G. Lodewijks, "Simulation of a multiterminal system for container handling," *Container Terminals and Cargo Systems*, vol. 2, pp. 15–36, 2006.

[4] T. Thurston and H. Hu, "Distributed agent architecture for port automation," in *COMPSAC '02: Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 81–90.

[5] L. Henesey and J. Törnquist, "Enemy at the gates: Introduction of multi-agents in a terminal information community," *C.A. Brebbia, G. Sciutto (Eds) Maritime Engineering and Ports III*, pp. 23–32, 2002.

[6] L. Henesey, F. Wernstedt, and P. Davidsson, "A market-based approach to container port terminal management," 2002.

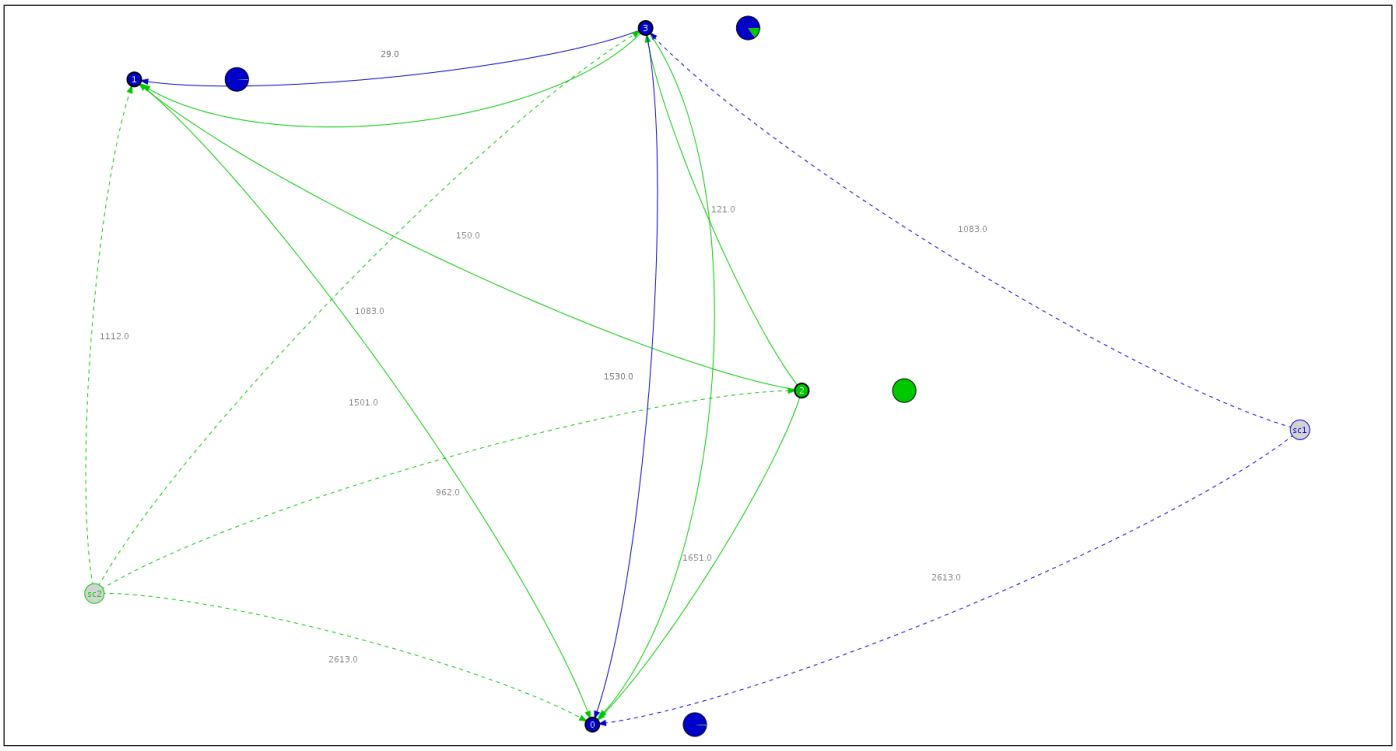

Fig. 8. Mission graph view in the simulator

[7] ——, "Market-driven control in container terminal management," in *Second International EuroConference on computer applications and information technology in the maritime industries*, Hamburg, Germany, 2003.

[8] L. Henesey, "Enhancing container terminal performance: A multi agent systems approach," Ph.D. dissertation, School of Engineering - Dept. of Systems and Software Engineering, Ronneby, 2004.

[9] L. Bianchi, "Notes on dynamic vehicle routing - the state of the art," *Technical Report, IDSIA*, 2000.

[10] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte, "Static pickup and delivery problems: A classification scheme and survey," *TOP*, 2007.

[11] S. Mitrovic-Minic, "The dynamic pickup and delivery problem with time windows," Ph.D. dissertation, Simon Fraser University, 2001.

[12] S. Mitrovic-Minic and G. Laporte, "Waiting strategies for the dynamic pickup and delivery problem with time windows," *Transportation Research Part B*, 2004.

[13] S. Mitrovic-Minic, "Pickup and delivery problem with time windows : a survey," *Technical Report 1998-12, SCS Simon Fraser University*, 1998.

[14] G. Lesauvage, "Gestion dynamique des activités des chariots cavaliers sur un terminal à conteneurs en environnement incertain," Master's thesis, Université du Havre, 2008.

[15] M. Dorigo, "Learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Italie, 1992.

[16] M. Dorigo and L. Gambardella, "Ant colony system : A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, 1997.

[17] C. Bertelle, A. Dutot, F. Guinand, and D. Olivier, "Distribution of agent based simulation with colored ant algorihm," in *14th European Simulation Symposium*.

[18] R. Montemanni, L. Gambardella, A. Rizzoli, and A. Donati, "A new algorithm for a dynamic vehicle routing problem based on ant colony system," *Technical Report, TR-23-02, IDSIA, Galleria 2, Manno, 6928, Switzerland*, 2004.

[19] B. Bullnheimer, R. F. Hartl, and C. Strauss, "Applying the ant system to the vehicle routing problem," in *In Proceedings of the 2nd International Conference on Metaheuristics - MIC97*.

[20] B. Bullnheimer, R. Hartl, and C. Strauss, "An improved ant system algorithm for the vehicle routing problem," *Annals of Operations Research*, 1999.

[21] A. Dutot, F. Guinand, D. Olivier, and Y. Pigné, "Graphstream: A tool for bridging the gap between complex systems and dynamic graphs," in *EPNACS: Emergent Properties in Natural and Artificial Complex Systems*, 2007.

[22] A. Larsen, "The vehicle routing problem," Ph.D. dissertation, Department of Mathematical Modelling, Technical University of Denmark, 2000.

[23] K. Lund, O. B. G. Madsen, and J. M. Rygaard, "Vehicle routing problems with varying degrees of dynamism," *Technical report, IMM, The Department of Mathematical Modelling, Technical University of Denmark*, 1996.