Colored ants for solving dynamic job shop scheduling problem : application to a straddle carriers container terminal

G. Lesauvage, S. Balev and F. Guinand

February 20, 2012

Abstract

1 Introduction

With the development of trade activities which have continually increased, container has become the first mode of packaging for exchanging goods. Container terminals have been created all around the world in order to facilitate the transfer between ships and trucks or trains. The performance of these transfers has to be considered to reduce the waiting cost of the container terminal customers.

Le Havre's harbor is the biggest harbor of France in container traffic. It is located at the North West cost of France, beside the Channel, sea door between the Atlantic and the North Sea. To keep competitive, the harbor has to provide a high quality of service and unceasingly develop new technologies and processes.

Straddle carriers are used within the terminal to move containers. Those vehicles can lift a container from above and are very useful to move containers in the yard by driving over the lanes. They are also used to load or unload trucks or trains. Some of them are able to dynamically adapt the spreader size to any container dimensions while some of them require to be set up in the depot.

Each move of a container by a straddle carrier can be seen like a mission. There are four kinds of missions :

- Incoming container missions;

Figure 1: The Terminal of Normandy, Le Havre's harbor, France.

- Outgoing container missions;
- Transshipment missions;
- Staying container missions.

The first category concerns trucks, trains and ships unloading. Straddle carriers drive to the pick-up locations and unload the vehicles, and then lift the container, drive to the yard to stock it. Concerning ships, they are unloaded by quay cranes which stack the containers on the quay. Then, straddle carriers come to pick-up the containers. The second category concerns trucks, trains and ships loading. In this case, straddle carriers start by picking-up a container from the yard and then drive to the delivery location (trucks areas, trains area or ship areas) to deliver it to their recipient. The third category of missions concerns the move of a container from a ship to another one. Finally, the last kind of missions concerns internal yard optimization process. Indeed, in some cases, it can be useful to reorganize a part of the stock area in order to reduce further delivery times or to free strategic container slots for next unloading missions.

Two time windows are affected to every mission. One concerns the pickup phase, the other one is related to the delivery. These time windows are used to fix an appointment between straddle carriers and hypothetical customer vehicles (trucks, trains or ships) concerned by the missions. Straddle carriers have to reach the pickup or delivery location within the given time window and so does the customers vehicles. If a straddle carrier comes too early, it will have to wait. On the contrary, if it comes too late, the customer vehicle will have to wait.

As a consequence, a time window overrun implies a cost for the terminal because, if a customer has to wait excessively, it may require late fees from the container terminal exploitation company. However, in the case of yard optimization missions, the time windows can be overrun because it has no direct effect on the customers. So, according to the mission kind, time windows can be hard or soft. For incoming missions, the pickup time window is hard and the delivery time window is soft. For the outgoing container missions, the pickup time window is soft but the delivery time window is hard. For transshipment missions, both time windows are hard, and for yard optimization missions both time windows are soft. Those time windows characteristics have to be taken into account in the mission scheduling process[2].

To reduce exploitation costs the mission scheduling must tend to minimize both the time windows overspent time and the distance covered by the straddle carriers. Indeed, time windows must be respected to avoid penalty fees and the distance covered by the vehicles directly impacts the exploitation costs of those vehicles. So, the optimization problem discussed in this paper concerns the allocation and the scheduling of the missions to the straddle carriers.

The main characteristic of this problem in a container terminal is that it is subject to dynamic events occuring during the day and making the environnement unstable. Indeed, the missions can be known at any time of the day. Some of them may be known at the very beginning and others just before the beginning of their pickup time windows. On the other hand, the customers of the terminal may also miss the time windows and by the way make the previous computed schedule unrelevant. The number of straddle carriers can also change during the day if some of broke down or are getting repaired.

All of those dynamic events contribute to the uncertitude of the environemment. Here, the scheduling system must be able to provide a feasible solution at any time and in any condition.

In this paper we will first describe the modeling the straddle carriers scheduling problem in its static and dynamic form. Then we will propose an ant colony based algorithm to solve it. And finally, we will discuss about the performance of this approach.

2 Modeling and Related Works

We will give here the modeling of the problem in its static and dynamic version.

2.1 Static version of the problem

In its static version, we assume that no dynamic event can occur during the day. It implies that the computed schedule will be always relevant. Here the travel time between each location within the container terminal is kown and fixed. The number of straddle carriers is also known and fixed. The time windows are respected by the customers and the missions are known before the begining of the day.

This problem can be seen like two different problems : the Vehicle Routing Problem and the Job Shop Scheduling Problem.

2.1.1 Vehicle Routing Problems

Since the problem is about finding shortest routes for a fleet of vehicles, it seems natural to classify it in the class of Vehicle Routing Problems (VRP)[20, 11] and more precisely as a Pickup and Delivery Problem (PDP) [3]. This version of VRP consists in picking up goods before delivering them to the customers. A variant of PDP takes into account the time windows[16]. Here, the pickup and the deliveries must occur into given time intervals. There is also a version of these problems with restricted capacitate vehicles[20].

In those problems the vehicles usually has a fixed capacity greater than one and the problem is to find shortest paths to deliver the goods to the customers. In our problem, the vehicles have a unit capacity. It means that they can not do more than one delivery per pickup operation, and as a consequence they have to go straight to the delivery location since they picked up the container. So, we do not focus on the combination of pickup and delivery operations. We focus on searching for shortest route between the missions (both pickup and delivery operations) and which respects the time windows.

This is the reason why we formulated this problem more as a scheduling problem than a vehicle routing problem.

2.1.2 Job Shop Scheduling Problem

In [13] and [7], the authors give a survey of Job Shop Scheduling Problems and of the means used to solve the different sub-problems.

In [1] the authors gives a survey of algorithms used to solve Job Shop Scheduling Problems with Sequence Dependent Setup Times (JSSP-SDST). Hybrid genetic algorithms, disjunctive graphs, mixed integer linear programming model with local search scheme, fast tabu search, branch and bound, dynamic programming, polynomial insertion algorithm, Lagrangian relaxation or ant colony algorithms are used. However they are used in a static version of the JSSP-SDST problem and without removal time sequence dependent criteria.

Since any scheduling problem can be described in the $\alpha|\beta|\gamma$ notation given by Graham et al. [12], we will give in this section the formulation of the straddle carrier scheduling problem.

The problem consist in finding the schedule S of n jobs $J_i (i = 1, \dots, n)$ on m machines $M_j (j = 1, \dots, m)$. This schedule is composed of each machine workload W_{M_j} containing the ordered list of jobs allocated to the machine M_j. This problem belongs to the class of the Job Shop Scheduling Problems (JSSP).

$$\begin{cases} S = \{W_{M_1}, W_{M_2} \dots, W_{card(M)}\} & \text{and} \\ W_m = \{J_j, \dots, J_k\} & \text{with } j \neq k,\ \forall j, k \text{ and } W_m \cap W_n = \emptyset,\ \forall m \neq n \end{cases}$$

In the case of the container terminal, the machines are the straddle carriers and the jobs are the missions.

Each job contains two operations which must be processed on the same machine in this order:

- O_1 : the pickup of the container;
- O_2 : the delivery of the container.

The machines are heterogeneous and a job j may not be compatible with machine i. For instance, if the straddle carrier has a spreader only adapted to 40 feet containers, then it will not be able to process 20 feet containers missions.

2.1.3 Preemption

There is no preemption in the straddle carrier mission scheduling problem since stoping a mission consists in deliverying the container to another location and then, resuming the mission by picking up the container from that location to the original delivery one. So, stoping and resuming a mission can be seen as two missions. As a consequence, we assume that the missions given as entry for our algorithm are already optimized to prevent from splitting them again.

2.1.4 Precedence

The jobs are independent in the problem. It means that there is no precedence constraint between the jobs. Though, there are time windows on each task. So, if two jobs are scheduled successively without taking into account their time windows, those time windows can be overrun. In the case of two missions concerning the same delivery slot in the container terminal, if the container of the second mission must be stacked onto the container of the first mission, then the first job should be prior to the second one. But if the second job is processed before the first one, then a new mission will be added in the pool of jobs to switch the two containers. So, in this particular case, the time windows of these missions should take into account the precedence between the jobs.

A really important aspect in our problem is that the process times and the release dates of the jobs depend on the machine executing the jobs. Indeed, the process time and the release date of a mission depends on the speed of the straddle carrier on one hand, and on the location of the vehicle at the beginning of the mission on the other hand. Moreover, the location of the straddle carrier relies on its activity. So if the vehicle is idle, then it will be located at the depot, or on its way to the depot. Else the vehicle will be located at the delivery location of its current mission when it will start the next mission.

2.1.5 Setup times and costs

At the beginning of a mission, the straddle carrier has to move to the pickup location. This move can be seen as a setup time or cost. This setup time of a job directly depends on the location of the straddle carrier at the beginning of the mission. If a previous job was executed by the machine then the distance between the two missions corresponds to the distance between the delivery location of the previous job and the pickup location of the current job. If no job was executed before the current job, then the vehicle is located at the depot. Actually, the setup times are sequence dependent (ST_{sd}). The setup cost $setup_{j,i}$ of the allocation of the job j to the machine i is equal to the distance $d(l(last(i), O_2), l(j, O_1))$ between the location $l(last(i), O_2)$ of M_i at the beginning of the mission (at the end of its last job $last(i)$) and the pickup location of J_j.

$$\begin{cases} setup_{j,i} = d(l(last(i), O_2), l(j, O_1)) & \text{if } \exists last(i) \\ setup_{j,i} = d(l(M_i), O_2), l(j, O_1)) & \text{else} \end{cases}$$

2.1.6 Process times and costs

The distance cost $distance_{j,i}$ of the allocation of the job j to the machine i is equal to the setup costs $setup_{j,i}$ plus the distance $d(l(j, O_1), l(j, O_2))$ between the pickup and the delivery location of J_j.

If the job is the last job of the schedule for the vehicle, then it will have to go back to the depot. This move can be seen as a removal time or cost equals to the distance (or travel time) $d(l(j, O_2), l(DEPOT_i))$ from the delivery location to the depot is added to the distance cost of the mission. The removal time is also sequence dependent (R_{sd}).

$$\begin{cases} R_{j,i} = d(l(j,O_2), l(DEPOT_i)) & \text{if } j = card(W_i), \\ R_{j,i} = 0 & \text{else.} \end{cases}$$

$$distance_{j,i} = setup_{j,i} + d(l(j,O_1), l(j,O_2)) + R_{j,i}$$

The process time cost $p_{j,i}$ of the allocation of the job j to the machine i depends directly on the distance $distance_{j,i}$ since the speed of M_i is known.

So, in this problem, both the setup times and process times are sequence dependent.

2.1.7 Release dates

The release date is computed thanks to the beginning of the pickup time window of the corresponding mission and the travel time cost for the machine to reach the pickup location. This travel time relies on the activity of the straddle carrier before starting the mission. So the release date $r_{j,i}$ of the job j for the machine i is the max between the completion time $C_{k,i}$ of the last job k processed by M_i plus the travel time from the delivery location of J_k to the pickup location of J_j, and the beginning of the pickup time window $tw_{min}(O_{j,1})$ of the job j. So the release date are also sequence dependent in this problem.

$$\begin{cases} r_{j,i} = max(C_{last(i),i} + distance(l(last(i),O_2), l(j,O_1)), t_{min}(O_{j,1})) & \text{if } \exists last(i), \\ r_{j,i} = max(distance(l(j,O_1), l(M_i)), t_{min}(O_{j,1})) & \text{else.} \end{cases}$$

2.1.8 Tardiness

If a mission misses one or both its time windows, then there is tardiness. This total tardiness is the sum of the pickup tardiness T_{j,i,O_1} and the delivery tardiness T_{j,i,O_2}. The pickup tardiness is the difference between the arrival time of M_i at the pickup location of J_j and the end of the pickup time window of J_j. The delivery tardiness is the difference between the arrival time of M_i at the delivery location of J_j and the end of the delivery time window of J_j.

$$\begin{cases} T_{j,i,O_1} = max(0, (C_{last(i),i} + t(d(l(last(i),O_2), l(j,O_1)))) - tw_{max}(O_{j,1})) \\ T_{j,i,O_2} = max(0, C_{j,i} - tw_{max}(O_{j,2})) \\ T_{j,i} = T_{j,i,O_1} + T_{j,i,O_2} \end{cases}$$

2.1.9 Optimization criteria

According to the kind of mission, the tardiness may cause penalty costs to the container terminal customers. These potential penalties can be seen like weighted tardiness and the best schedule is the one minimizing this weighted tardiness.

$$F_S = min \sum_{j=1}^{n} (w_j.T_{j,M(j)})$$

Always in order to reduce exploitation costs and once the weighted tardiness has been minimized, the goal is to minimize the distance covered by the vehicles. So, the second criteria to minimize is the process cost of the tasks.

$$D_S = min \left(\sum_{i=1}^{m} . \sum_{k=1}^{card(W_i)} distance(W_i(k), i)) \right)$$

So, according to the classification of Graham et al. in [12], and of Brucker in [7], our problem is $J|ST_{sd}, R_{sd}| \sum w_j.T_j, \sum distance(i)$. As shown in [10], this problem is NP-Complete for $m \geq 2$.

2.2 Dynamic version of the problem

In its dynamic version, the straddle carriers scheduling problem deals with dynamic events. It implies that a computed schedule can become unrelevant during the day of work. The travel times between each location within the container terminal is also known but are time dependent. The number of straddle carriers is also known but can change since the straddle carriers can become unavailable if they broke down and become available again when they got fixed. The time windows can also be overspent by the customers and the missions can be known at any time of the day.

Berbeglia et al described the dynamic version of pickup and delivery problems in [4]. In her PhD Thesis [17], S. Mitrovic-Minic worked on Dynamic Pickup and Delivery Problems with Time Windows (DPDP-TW). In [18], the authors used a multiple Traveler Salesman Problem (m-TSP) formulation of the Vehicle Routing Problem with Time Windows (VRP-TW). They used precedence graphs to model the multiple Traveler Salesman Problem with Time Windows (m-TSPTW). They proposed algorithms to compute bounds on the number of vehicles required to complete the deliveries. They also showed that this problem is NP-Hard.

Regarding the Job Shop Scheduling modeling of this problem, when jobs must be inserted into the computed schedule then the problem is known as the Dynamic Job Shop Scheduling Problem (DSJSSP) [19]. As showed above, the proposed schedule must be failsafe and has to deal with variations of the environment.

We propose in the next section an on-line ant colony based algorithm based able to propose near optimal solutions at any time for the problem in its dynamic version.

3 Colored Ants algorithm

Bio-inspired algorithms have been more and more used to solve optimization problems since the early nineties[9]. Ant Colony System takes advantage of the results of the experiment of Deneubourg in 1983 [8] showing the stigmergia between ants. It is particulary well adapted to dynamic optimization problem because of its intrinsic characteristics such as the decentralized intelligence, the indirect communication between the ants (positive feedback), and the evaporation of the pheromone tracks (negative feedback).

Our model is a multi-colonies version of the ant algorithm. In this version, the ants are attracted by pheromone of their own colony and repulsed by pheromone of foreigns colonies. The result of such an approach is a behavior of collaboration between the ants of a colony and a behavior of competition between the colonies to compete for the missions.

This kind of ant colony algorithm has been used by Bertelle et al. in [5, 6] for determining dynamically the best distribution of a parallel program on a network. They used a collaboration/competition process between colonies of artificial ants to distribute the data and the calculation of a program among heterogeneous processing ressources while minimizing the communication involved. The authors used a threshold to avoid ants from choosing a destination containing too much foreign pheromone and by this way balancing the ants over the solution space. We will see below that we chose to use a linear combination of the repulsion aspect in the pseudo-random-proportional transition rule rather than using a threshold.

In our model, a colony represent a vehicle (machine). The vehicle has to choose the best chain of missions to process, it means the chain which minimizes overrun time of the missions and the covered distance. We modeled the different solutions as pathes on a graph. The ants of the colony will have to colonize the mission graph to find the shortest path from the source node to the sink. When an ant chose a node to colonize, then it spreads pheromone according to the quality of the marked node. This pheromone will be used to lead the other ants of the colony toward the node and to repulse foreign ants towards other nodes. Since each colony use the same behavior, the nodes are split between the different colonies and this distribution tends to minimize the overall covered distance of the vehicles.

Each colony is modeled by a color. In this way, each node of the graph is colored by the color of the highest amount of pheromone on this node. The solution is obtained by constructing the best paths for each color. Each path P_i of color i is built by starting at the source node and by searching among the set $S_{j,i}$ of accessible nodes colored in i the node with the highest level of pheromone. The process is repeated until either the sink node has been reached or the $S_{j,i} = \emptyset$.

3.1 Graph modeling

Lets define the directed graph $G = (V, E)$ where V is the set of vertices and E the set of edges. Each job j to schedule is modeled as a node $v_j \in V$ and the edges $e(v_j, v_k) \in V$ represent the possibility for the machines to chain up the job j with the job k.

Two other vertices are added to the graph: a source node, and a sink node. The source node is connected to each node of the graph which has an in-degree equals to zero. On the other hand, each node with an out-degree equals to zero is connected to the sink node.

3.1.1 Edges

The edges are weighted to represents the cost of processing the target node mission after the source node one. The weight must take into account, on one hand, the travel time between the delivery location of the source mission and the pickup location of the target mission, and on the other hand, the potential lateness involved by chainning the two missions. Each edge contains as many weights as there are machines in the problem. Indeed, since the machines are heterogeneous the travel times may differ from one machine to another.

For the edges linking the source node to a job node, the weight is computed according to the end time of the straddle carriers current activities. If the machine is processing a job then the weight will be based on the travel time between the delivery location of the processed mission and the pickup location of the target node mission and the potential lateness on the pickup time window of the target mission. On the contrary, if the machine is idle then the weight of the edge will be the travel time between the current location of the machine and the pickup location of the target node mission and the potential lateness based on the current time of the day.

For the edges linking the job nodes to the sink node, the weight corresponds to the travel time between the delivery location of the job and the vehicles depot.

Concerning the edges between two job nodes, the weight is the travel time between the delivery location of the origin job node and the pickup location of the target job node.

Since the travel time depends on the location and the activity of the machine, the weights are time dependent even in the static case of the problem.

The graph represent the possibilities of job scheduling for the machines. The allocation problem is next solved using colored ants colonies algorithm on this graph.

The weight $w^{(t)}(n, m, c)$ represents the travel cost of an ant of the colony c to go from the node n to the node m at time t. It corresponds to the weight of the edge linking n and m for the colony c. We chose to compute it as the sum between the travel time $t(n, m, c)$ between the delivery location of mission n and the pickup location of mission m for the straddle carrier c and the potential lateness $lateness^{(t)}(n, m, c)$ at time t on the pickup time window of mission m after executing mission n. F_1 and F_2 are constants used to balance the relative importance of the two criteria : travel time and lateness.

$$w^{(t)}(n, m, c) = t(n, m, c) * F_1 + lateness^{(t)}(n, m, c) * F_2$$

3.2 Dynamic graph

In the dynamic case of the problem, missions can be added, removed or updated. The graph has to allow these modifications.

When a mission is added in the jobs pool, a new node is added in the graph. It is connected by edges as described above. If the new node in the graph causes the creation of an edge to a node connected to the source node, then the edge from the source node to the other node is deleted. As well as this, if the new node insertion causes the creation of an edge to a node connected to the sink node, then the edge between the node and the sink is deleted. This process is required to force the vehicles to process all the missions. Otherwise, the best solution found by the algorithm would be not to process any job because the covered distance would be null.

A node can be removed from the graph for two reasons. On one hand, if the corresponding mission has been canceled, and on the other hand if the corresponding mission has been completed. The node is then deleted from the graph and the edges of adjacent nodes are updated according to the criteria discussed above.

When a mission is updated, the corresponding node is deleted from the graph and then re-added. This process allows to take into account the new characteristics of the mission.

On the other hand, the weights of the edges are updated according to the vehicles activity.

When a vehicle starts a mission, it must complete the mission unless it broke down. In this case, the mission is updated because the pickup location may have changed if the vehicle started to move the container before broking down. Moreover, if the vehicle becomes unavailable, the ants of the corresponding colony are reseted to the source node and must remains at this node until the vehicle becomes available again. In the meantime, the evaporation process makes the previous alocation solution disapeared. In the case where the vehicle does not broke down, it must achieve the mission. To represent this constraint in the algorithm, the ants of the colony of the vehicle starts their path finding from the current mission node. The pheromone of other colonies on this node is also evaporated to make this node totally unaccessible to the ants of other colonies.

3.3 Algorithm

After each update on the graph, the following algorithm is executed Θ times.

```
for all ant $a$ of each colony $c$ do
  Node $destination \leftarrow choose\_destination(a)$
  Node $n \leftarrow location(a)$
  if $destination = null$ then
    $return\_to\_source\_node()$
  else
    $move(a, destination)$
    $spread\_pheromone(c, n, destination)$
  end if
end for
for all node $n$ of the mission graph do
  $evaporation(n)$
end for
for all colony $c$ do
  $compute\_path(c)$
end for
```

3.4 Pheromone handling

The quantity of pheromone of color c on the node n at time t is noted $\Omega^t(n, c)$. The quantity of pheromone of other colors than c is noted $\hat{\Omega}^{(t)}(n, c)$.

$$\hat{\Omega}^{(t)}(n, c) = \left(\sum_{k \in C} (\Omega^{(t)}(n, k)) \right) - \Omega^{(t)}(n, c)$$

3.4.1 Positive feedback

The algorithm *spread_pheromone*(Colony c, Node n, Node m) computes the quantity of pheromone $\Delta^{(t)}(m, c)$ of color c which will be dropped on the node m between the time t and $t + 1$. This quantity directly depends on the ants previous location n because it takes $w^{(t)}(n, m, c)$ into account.

$$\Delta^{(t)}(m,c) = \lambda * \frac{1}{w^{(t)}(n,m,c)}$$

3.4.2 Negative feedback

The evaporation process computes the new amount of pheromone $\tau^{(t)}(n,c)$ of color c on node n at time t as below:

$$\tau^{(t)}(n,c) = \rho\tau^{(t-1)}(n,c) + \Delta^{(t)}(n,c)$$

3.5 Pseudo-random-proportional rule

The algorithm *choose_destination*(Ant a) (see Algorithm 3.3) returns the destination chosen by the ant a of the colony c according to its current location node n and the probability $p^{(t)}(n,m,c)$ to choose the destination node m at time t computed by the following pseudo-random-proportional rule:

$$p^{(t)}(n,m,c) = \frac{\Omega^{(t)}(m,c)^{\alpha}\left(\frac{1}{w^{(t)}(n,m,c)}\right)^{\beta}\left(\frac{\hat{\Omega}^{(t)}(d,c)}{\sum_{k\in C}\Omega^{(t)}(m,k)}\right)^{\gamma}}{\sum_{d\in n_{out}}\left(\Omega^{(t)}(d,c)^{\alpha}\left(\frac{1}{w^{(t)}(n,d,c)}\right)^{\beta}\left(\frac{\hat{\Omega}^{(t)}(d,c)}{\sum_{k\in C}\Omega^{(t)}(d,k)}\right)^{\gamma}\right)}$$

If the chosen node m has a probability $p^{(t)}(n,m,c) < \Delta$ then the choice is refused and the ant goes back to the source node. This process is used to avoid the continous colonization of the graph by all the colonies. Indeed, if there are more vehicles than missions to allocate, then it is not possible to affect a mission to every vehicle. Moreover, we chose to integrate the competition aspect of the algorithm in the pseudo-random-proportional transition weighted by the γ parameter.

3.6 Solutions

The schedule of each vehicle is obtained at the end of the procedure by building each best path for each color. Since the graph is directed and acyclic, the path is built by starting at the source node and by choosing from each accessible node colored by the same color that the path to build, the one with the highest quantity of pheromone. The built ends when the sink node is reached or when there is no accessible node of the color of the path.

3.7 Path reinforcement

When the first mission of a path has been started by a vehicle, a reinforcement quantity of pheromone of the color of the vehicle is spread on the whole path. This process avoid useless changes in the solution while the previous computed path is being used.

```
for all Node $n$ in the solution $S$ do
    $\tau^{(t)}(n,c) = \tau^{(t-1)}(n,c) + \Lambda$
end for
```

3.8 Parameters

As described above, the algorithm is setted up with the following parameters:

- α : relative importance of the pheromone track in the destination choice;
- β : relative importance of the weight heuristic in the destination choice;
- γ : relative importance of the repulsion process in the destination choice;

- Δ : environment pressure rate. If the chosen destination pheromone rate is less than Δ, then the ant die (start over from the source node);
- η : number of ant per colony;
- λ : fixed quantity of pheromone spread on a destination (with $\lambda > 1$);
- Λ : quantity for reinforcement : quantity of pheromone spread on the whole path when the first mission of the path has been started;
- ρ : rate of pheromone conserved after each evaporation.

We distinguish two classes of parameters: on one hand the ones about the choice of destination for the ants (α, β, γ and Δ), and on the other hand, the ones about the pheromone handling (λ, Λ and ρ).

After testing different values for the parameter η, we decided to fix it to the number of missions in the pool. Indeed, when a mission is added, a new ant is created for each colony compatible with the mission. On the contrary, when a mission is removed from the pool, an ant is removed of each colony.

According to our simulation results, it also appears that the quality of the solution found and also the time required to find this best solution is more strongly connected to the fitness weighting the edges than to the values of the parameters. The parameters of the first class also have more influence on the quality of the solution and the convergence of the algorithm than the parameters of the second class.

4 Experiments and results

We developed an algorithm able to handle dynamics and to provide near-optimal solutions in a reasonable computation time to our problem. Since the algorithm had also to be failsafe, it ensures feasible solutions at anytime. In this section we first perform the algorithm on static versions of the scheduling problem. Then we test it on several dynamic scenarii. All of the simulations are run with D^{2}CTS (Dynamic and Distributed Container Terminal Simulator)[15] with configuration files corresponding to the Terminal of Normandy of Le Havre's harbor.

4.1 Measure of dynamics

In his PhD thesis[14], A. Larsen proposed two measures to determine how dynamic is an instance of a Vehicle Routing Problem. The first one is the degree of dynamism (DOD) and is calculated as the ratio between the number of dynamical requests and the total number of requests. If $dod = 0$ the problem is static, if $dod = 1$ then the problem is fully dynamic. The main weakness of this measure is that it does not take into account the arrival time of those requests. For this reason, Larsen introduced the effective degree of dynamism by the following formula :

$$edod = \frac{\sum_{i=1}^{\eta_{imm}} \left(\frac{t_i}{T}\right)}{\eta_{tot}}$$

Here η_{imm} are the requests arriving during the day which have to be planned immediately. η_{tot} corresponds to the sum of immediate requests and the requests arrived in advance. The planning horizon starts at 0 and ends at T and t_i is the time the i^{th} immediate request is recieved. This measure takes into account the average of the incoming time of the requests and gives a better appreciation of the dynamics of the scenario. Concerning the Vehicle Routing Problem with Time Windows (VRP-TW), Larsen extended the definition of edod by this formula :

$$edod - tw = \frac{1}{\eta_{tot}} \sum_{i=1}^{\eta_{tot}} \left(1 - \frac{r_i}{T}\right)$$

Here, r_i is the reaction time of the task i which is the difference between the latest time the request can starts and t_i which is the incoming time of the i^{th} request.

We will use *dod* and $edod - tw$ to measure the dynamics of our scenarii by setting T to the latest min value of pickup time window of the scenario.

4.2 Measure of performance

As discussed in previous sections, we want to minimize exploitation costs of the container terminal. This is the reason why we measure the performance of our algorithm by two criteria : the overall covered distance of the straddle carriers and the overspent time of the missions time windows. In addition we also count the number of missions having one or both of their time windows overrun.

4.3 Static case

To measure performance of our algorithm, we developed a Branch-and-Bound algorithm to solve the problem in the static case and to obtain optimal solutions. This algorithm is also used to solve the instances of the static problem generated from the dynamic one.

We tested here XX different scenarii of 5 to 10 missions with variable number of straddle carriers between 2 and 5. The small scale of the scenarii in both jobs and machines dimensions is due of course to the impossibility to compute optimal solutions in reasonable time. As we experiment our algorithm in the satic form of the problem $dod = 0$ and there is no dynamic events occuring during the simulations.

$\lvert J\rvert$	$\lvert M\rvert$	**Distance**		**Overspent time**		**Overrun missions**		**Execution time**	
		ACO	B&B	ACO	B&B	ACO	B&B	ACO	B&B
5	2	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00
5	3	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00
5	4	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00
7	2	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00
7	3	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00
7	4	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00
10	2	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00
10	3	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00
10	4	0	0	00:00:00	00:00:00	0	0	00:00:00	00:00:00

4.4 Dynamic case

In the dynamic case, we can not have determinist algorithm to compute optimal solution since the characteristics of the problem change all along the simulation. //Dire que les tests dans le cas statique montrent que l'algo converge vers une near optimal solution

We tested here XX different scenarii of 10 to 100 missions with variable number of straddle carriers between 2 and 20.

//Tester differents edod-tw

//Tester les evenements dynamiques : pannes de vehicules et retards des clients

$\|J\|$	$\|M\|$	*dod*	$edod - tw$	**Distance**	**Overspent time**	**Overrun missions**	**Execution time**
10	2	0.0	0.0	0	00:00:00	0	00:00:00
10	3	0.0	0.0	0	00:00:00	0	00:00:00
20	5	0.0	0.0	0	00:00:00	0	00:00:00
20	7	0.0	0.0	0	00:00:00	0	00:00:00
30	7	0.0	0.0	0	00:00:00	0	00:00:00
30	10	0.0	0.0	0	00:00:00	0	00:00:00
40	7	0.0	0.0	0	00:00:00	0	00:00:00
40	10	0.0	0.0	0	00:00:00	0	00:00:00
50	7	0.0	0.0	0	00:00:00	0	00:00:00
50	10	0.0	0.0	0	00:00:00	0	00:00:00
75	15	0.0	0.0	0	00:00:00	0	00:00:00
75	15	0.0	0.0	0	00:00:00	0	00:00:00
100	20	0.0	0.0	0	00:00:00	0	00:00:00
100	20	0.0	0.0	0	00:00:00	0	00:00:00

5 Conclusions & future work

References

[1] Ali Allahverdi, C.T. Ng, T.C.E. Cheng, and Mikhail Y. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985–1032, June 2008.

[2] Stefan Balev, Frédéric Guinand, Gaëtan Lesauvage, and D. Olivier. Dynamical Handling of Straddle Carriers Activities on a Container Terminal in Uncertain Environment - A Swarm Intelligence approach -. In *ICCSA 2009 The 3rd International Conference on Complex Systems and Applications*, volume 2, page 290, Le Havre, France, June 2009. 7p.

[3] Gerardo Berbeglia, Jean-François Cordeau, Irina Gribkovskaia, and Gilbert Laporte. Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31, 2007.

[4] Gerardo Berbeglia, Jean-François Cordeau, and Gilbert Laporte. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8 – 15, 2010.

[5] Cyrille Bertelle, Antoine Dutot, Frédéric Guinand, and Damien Olivier. Organization detection using emergent computing. *International Transactions on Systems Science and Applications*, 2(1):61–70, 2006.

[6] Cyrille Bertelle, Antoine Dutot, Frédéric Guinand, and Damien Olivier. Organization detection for dynamic load balancing in individual-based simulations. *Multi-Agent and Grid Systems*, 3(1):42, 2007.

[7] Peter Brucker. *Scheduling Algorithms.* Springer Publishing Company, Incorporated, 5th edition, 2010.

[8] Jean-Louis Deneubourg, Jacques M. Pasteels, and J. C. Verhaeghe. Probabilistic behaviour in ants: A strategy of errors? *Journal of Theoretical Biology*, 105:259–271, 1983.

[9] Marco Dorigo, Mauro Birattari, and T. Stützle. Ant Colony Optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

[10] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1:117–129, 1976.

[11] Gilbert and Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345 – 358, 1992.

[12] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.*, 4:287–326, 1979.

[13] Anant S. Jain and Sheik Meeran. A state-of-the-art review of job-shop scheduling techniques. *European Journal of Operations Research*, (113):390–434, 1999.

[14] A. Larsen. *The Vehicle Routing Problem.* PhD thesis, Department of Mathematical Modelling, Technical University of Denmark, 2000.

[15] Gaëtan Lesauvage, Stefan Balev, and Frédéric Guinand. D^{2}CTS : A dynamic and distributed container terminal simulator. In *HMS 2011 : The* 13rd *International Conference on Harbor, Maritime & Multimodal Logistics Modelling and Simulation*, September 2011.

[16] S Mitrovic-Minic. Pickup and delivery problem with time windows: A survey. *SFU CMPT TR*, 12(1):1–43, 1998.

[17] S. Mitrovic-Minic. *The dynamic pickup and delivery problem with time windows.* Simon Fraser University, 2001.

[18] Snežana Mitrović-Minić and Ramesh Krishnamurti. The multiple tsp with time windows: vehicle bounds based on precedence graphs. *Operations Research Letters*, 34(1):111 – 120, 2006.

[19] R. Ramasesh. Dynamic job shop scheduling: A survey of simulation research. *Omega*, 18(1):43 – 57, 1990.

[20] Paolo Toth and Daniele Vigo, editors. *The vehicle routing problem.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.