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

S. Balev, F. Guinand and G. Lesauvage

February 2, 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. From 2000 to 2012, the world container traffic has increased by more than XX% ... 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.

dans cette partie on doit comprendre dans les grandes lignes comment un terminal à conteneurs fonctionne, les principaux termes doivent être introduits dans leur contexte, lanes, quay, pickup, etc.
Such terminal are organized in the following way... lanes, quay, etc. Containers boxes 20 feet, 40 feet, etc. incoming and outgoing containers, ships, trucks and trains, transshipment. Missions, movement, lanes, pickup, delivery, initial location (parking, depot, etc.), straddle carriers capacity, speed, movement, constraints, etc.

Straddle carriers move containers within the terminal. 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.

1.1 The missions

Each move of a container by a straddle carrier can be seen like a mission. A mission contains two phases : the pickup and the delivery of the container. There are four kinds of missions :

- Incoming container missions;
- 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.

2 Modeling and Related Works

Several elements have to be considered separately for modeling the problem:

- the container terminal with its organization (lanes, quay, etc.)
- the ressources: the straddle carriers and,
- the missions.

From what has been explained previously, when a vehicle chains two missions, it has to move from the location of its last delivery to the location of its next pickup. Both locations are precisely defined in the context of a container terminal such that it is possible to compute a shortest path between both locations. In a static context, a matrix of shortest distance between pairs of locations may be computed. Là, c'est différent de ce que tu as écrit. Je pense qu'il est possible de relaxer le problème pour ne pas tenir compte de la machine, mais modéliser l'enchaînement des missions sous la forme d'un graphe comme tu le proposes en section 3.3.1 et 3.3.2.
je poursuis

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, but the first objective has a higher priority. 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.

du coup je pense que la partie vehicle routing problem est inutile, donc je la vire. On pourra en reparler pour la version dynamique du problème
VRP = routage ? voir remarque section 2.4

2.1 Static version of the problem

Dans cette partie, il faut préciser exactement le problème avec les éléments qui sont laissés de côté, on dit que le problème est relaxé.

In the static version of the problem, we consider that no dynamic event can occur in the scheduling after starting its execution. Indeed, we assume that every characteristics if the

problem is known in advance and will not change during all the execution. In this relaxed version of our problem, the travel times will be respected by the machine.

This is the reason why we formulated this problem more as a scheduling problem than a vehicle routing problem. In the next section, we will describe this scheduling problem and formulate it in the $\alpha|\beta|\gamma$ notation given by Graham et al. [12].

2.2 Job Shop Scheduling Problem

The problem consists in finding the schedule S of n jobs $J_i (i = 1, \ldots, n)$ on m machines $M_j (j = 1, \ldots, m)$. This schedule is composed of each machine workload W_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).

j'ai modifié les notations parce que m et n étaient déjà pris

$$\begin{cases} S = \{W_1, W_2 \ldots, W_m\} & \text{and} \\ W_i = \{J_{\alpha_1}, \ldots, J_{\alpha_k}\} & \text{with } k \leq n, \text{ and } W_i \cap W_j = \emptyset, \forall i \neq j \end{cases}$$

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.

ok pour la remarque qui suit, mais est-ce que cette hétérogénéité est prise en compte dans les deux modèles, le modèle statique et le modèle dynamique ?
L'hétérogénéité est prise en compte dans le problème (statique ou dynamique). Maintenant, pour pouvoir comparer nos résultats, il est possible de simplifier les problème pour obtenir des solutions optimales plus facilement

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.

2.2.1 Preemption

There is no preemption in the straddle carrier mission scheduling problem since stopping 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, stopping 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 atomic and cannot be decomposed in sub-missions.

2.2.2 Precedence

The jobs are independent in the problem. It means that there is no precedence constraint between the jobs. C'est intéressant, mais est-ce toujours vrai, y compris dans le cas de transshipment ?Oui car la précédence est induite par les fenetres de temps. Though, there are time windows for the achievement of the two operations of each job. 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. tu soulèves un point important mais à la fin de la lecture on ne sait finalement pas si c'est la méthode qui doit se débrouiller ou si cela relève de la modélisation ou si dans le jeu de données tout est propre. Qu'en est-il ?Oui je pars du principe que les fenêtres de temps fournies ont été

calculées pour représenter la précédence. Donc si les véhicules respectent les fenêtres de temps, il ne devrait pas y avoir de mission de shifting à insérer

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. Là je ne suis pas d'accord. On peut faire l'hypothèse que les cavaliers se déplacent tous à la même vitesse et ont des parcours contraints (ils ne peuvent pas couper à travers le terminal). En fonction de ces hypothèses, on peut calculer l'ensemble des plus courts chemins du graphe des voies de circulation des cavaliers pour aller d'une position à une autre, qu'il s'agisse d'une position de conteneur ou d'une position de dépôt. Donc on doit pouvoir modéliser l'ensemble des missions par un graphe dont les sommets sont les jobs et dont les arêtes sont les déplacements possibles pour l'enchaînement des jobs. Un exemple ci-dessous:

job	**pickup**	**delivery**	**position**	**position**	**$\sim$ distance**
	time window	**time window**	**pickup**	**delivery**	
1	[2-5]	[12-20]	(1,1)	(2,4)	4
2	[3-7]	[6-20]	(8,5)	(3,4)	6
3	[7-12]	[16-25]	(3,8)	(5,0)	6
4	[9-11]	[12-20]	(0,2)	(2,2)	2
5	[12-20]	[18-30]	(6,4)	(2,3)	4
6	[15-25]	[40-50]	(3,1)	(2,4)	4
7	[16-17]	[30-35]	(1,5)	(4,3)	5

je considère une distance et je compte 2 unités de temps pour parcourir une unité de distance. Je ne considère que les jobs qui peuvent être enchaînés, c'est-à-dire ceux dont le delivery au plus tôt+le temps de déplacement est inférieur au pickup au plus tard. Voici la matrice des distances entre jobs (en gros c'est $|x_1 - x_2| + |y_1 - y_2|$). Au plus tôt le delivery de chaque job est (12,6,16,12,18,40,30) et au plus tard la pickup est (5,7,12,11,20,25,17) pour que J_1 puisse précéder J_5 il faut donc que la distance soit inférieure à $(20 - 12)/2 = 4$ puisqu'il faut 2 unités de temps pour se déplacer d'une unité de distance. Indépendamment des distances on sait déjà que les seuls enchaînements possibles pour respecter les délais sont 1 suivi de 5, 6 ou 7, 2 suivi de 3, 4, 5, 6 ou 7, 3 suivi de 5, 6 ou 7, 4 suivi de 5, 6 ou 7, 5 suivi de 6. Voici la matrice des distances pour ces configurations :

	1	2	3	4	5	6	7
1	-	-	-	-	4	4	2
2	-	-	4	5	3	3	3
3	-	-	-	-	5	3	9
4	-	-	-	-	3	2	4
5	-	-	-	-	-	3	-
6	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-

Si on ajoute les distances, on peut encore éliminer des enchaînements possibles.

Je ne peux qu'être d'accord, vu que c'est ce que je fais pour construire mon graphe! Mais les véhicules jouent quand même un role si la flote n'est pas homogène (certains plus rapide que d'autres, incompatibilités entre les missions). Donc dans le cas le plus simple (flotte homogène + pas de problème de compatibilité) les véhicules n'interviennent pas dans le calcul des solutions

.

2.2.3 Graph definition

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.

2.2.4 Edges

The edges are weighted by the travel time between the corresponding nodes. 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 corresponds to the travel time between the machine location at the end of its current activity and the pickup location of the target node job. If the machine is processing a job then the weight will be the travel time between the delivery location of the processed mission and the pickup location of the target node 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.

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.

2.2.5 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.

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

2.2.6 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.

2.2.7 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.

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

2.2.8 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.

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

2.2.9 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 $m \geq 2$. We will explain in the next section the algorithms used to solve the problem and especially our ant colony based meta-heuristic.

2.3 Related works

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. 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. 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.

2.4 Dynamic version of the problem

In the dynamic version, straddle may use other paths than classical ones or predefined ones (c'est ça ? D'où les travaux sur les vehicle routing). Attention, le VRP n'est pas un problème de routage au sens calcul d'un plus court chemin entre 2 points du terminal. Il s'agit bien d'un problème d'ordo, transformé en problème de routage (le point de départ et le point d'arrivé est le dépot et les carrefours sont les missions). Il ne s'agit pas seulement d'optimiser les routes entre 2 missions mais d'optimiser tout l'enchainnement de missions de façon à minimiser le makespan, les retards (dans la version TW) ou la distance totale parcourue (peut être vu comme un m-TSP). Donc il y a plus ici que la dynamique liée aux routes suivies par les conducteurs. Il y a également toute la dynamique liée aux missions (arrivée de missions en cours de journée, annulation de mission, décalage des fenêtres de temps, retards des clients) et aux chariots (panne, conducteur qui se trompe de mission, perte de conteneur...).

2.4.1 Dynamic graph definition

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.

2.5 Related works: vehicle routing and dynamic JSSP

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)[18, 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[14]. Here, the pickup and the deliveries must occur into given time intervals. There is also a version of these problems with restricted capacitate vehicles[18]. Berbeglia et al described the dynamic version of pickup and delivery problems in [4]. In her PhD Thesis [15], S. Mitrovic-Minic worked on Dynamic Pickup and Delivery Problems with Time Windows (DPDP-TW). In [16], 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.

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.

2.5.1 Dynamic JSSP

At a container terminal, some missions are known before the beginning of the day, others are known during the day more are less early before the release dates of their corresponding jobs. When all the missions are known in advance, the scheduling problem is considered static. On the other hand, when jobs must be inserted into the computed schedule then the problem is known as the Dynamic Job Shop Scheduling Problem (DSJSSP) [17].

The dynamics also comes from the cancellation of missions. Indeed, a mission can be canceled by the customer and so the schedule needs to be recomputed.

The number of machines can change during the day according to the straddle carriers unavailability (for maintenance, or failure...). Then, the jobs allocated to the unavailable machine must be dispatched to other machines.

Because of all those dynamic events, the mission scheduling algorithm must provide a solution at anytime. Indeed, if a new mission come into the pool at the very begining of its pickup time windows, then the scheduler must insert this mission into the workload of a vehicle as soon as possible to prevent from overrunning the time windows of the mission. In this case, the problem turns to be a nearly real time problem.

Since the Job Shop Scheduling Problem is NP-complete, the Dynamic Job Shop Scheduling Problem is also strongly NP-Hard and it is not possible to get optimal solutions in reasonable time. That is why we must use heuristics or meta-heuristics methods to approximately solve the problem.

3 Static version of the problem

3.1 Branch and bound algorithm

We developed a branch and bound algorithm to find optimal solutions to our problem in small instances (5 machines and 10 jobs). These solutions will allow us to compare the quality of the

solutions found by our ant colony based algorithm (at least in the static case).

It works like a backtracking of every feasible solution but stop the local search each time the upper bound is exceeded. As we wanted to first minimize the weighted tardiness of the solution and then the distance covered by the machines to process the jobs, the algorithm has two steps : first minimizing the weighted tardiness of the solution and then minimizing the distance.

The algorithm starts by generating each allocation possible of each job. Then, for each possibility the weighted tardiness is computed. If it is greater than the bound then the branch is aborted. Else, if it is lower than the bound, the bound is updated and the new bound is propagated. If it is equal to the bound, then the distance criteria is investigated and if the distance of the solution is upper the bound, then the search of this branch is aborted.

The solution construction keep going until finding a leaf in the search tree. If the weighted tardiness is lower than the bound, then it means that a new best solution has been found. In this case, the new bound is propagated.

We can not evaluate the improvement of the distance criteria until we reach a leaf because the removal time cost is sequenced dependent and is taken into account in the distance equation. So, it is only when the solution construction reaches a leaf, the best solution distance is updated if a better solution has been found.

3.2 Heuristics based algorithms

To measure the performance of our algorithm on bigger instances we used heuristics based methods...

4 Dynamic version of the problem

The problem is modeled as a graph to be able to apply the ant colony based algorithm. The goal is to pre-solve the scheduling problem by allowing only missions links which do not lead to overrun the time windows (at least in the static case). Two missions can be chained up if the second mission does not start before the end of the first one and if the travel time between the delivery location of the first mission and the pickup location of the second one is short enough not to overrun the pickup time window of the second mission.

4.0.1 Colored Ants

To solve the allocation problem, an ant colony based algorithm is used. This bio-inspired algorithm has been more and more used to solve optimization problems since the early nineties[9]. It 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 graph.

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 an edge containing too much foreign pheromone and by this way balancing the ants over the graph. 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. 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 $S_{j,i} = \emptyset$.

4.0.2 Algorithm

---

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

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 $\hat{\Omega}^{(t)}(n, c)$.

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

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.

The algorithm *choose_destination*(Ant a) (see Algorithm 4.0.2) 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 go 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.

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.

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

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

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

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.

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

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 strongly connected to the values of the first class parameters.

We will describe in the next section the results of different execution of our algorithm on generated problems about a real context which is the container terminal of Normandy in Le Havre, France.

5 Experiments and results

Thanks to D^{2}CTS we are able to perform several algorithms to solve different optimization problems such as vehicle routing, berth allocation, container positioning or mission scheduling problems. We focused on the mission scheduling problem.

5.1 Static case

5.2 Dynamic case

6 Conclusion

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

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

[16] 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.

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

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