École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Computer Aided Decision Support
International Research Master 2
2013 - 2014

Operational research project report

Ant colony optimization algorithm for a combined routing and scheduling problem

Supervisors
Jean-Charles BILLAUT
jean-charles.billaut@univ-tours.fr
Nicolas MONMARCHÉ
nicolas.monmarche@univ-tours.fr

Student
Thomas NOGUER
thomas.noguer@etu.univ-tours.fr

M2RI CADS 2013 - 2014

Université François-Rabelais, Tours

Version of January 18, 2014

Contents

1 Introduction **6**

2 Formalization of the problem **7**

3 Ant colony optimization **9**

3.1 General definitions 9

3.2 Application to the combined routing and scheduling problem 10

3.2.1 Definition of the Ant colony for the mixed optimisation problem 10

3.2.2 Algorithms 13

4 Results **16**

5 Perspectives **19**

6 Assessment **20**

7 Conclusion **21**

List of Figures

2.1 A solution of the problem for a simple instance 8

3.1 The construction of a solution by an ant. 11

List of Tables

4.1 Results for heuristics 17
4.2 Results for the ant colony 17
4.3 Difference of the results between the ant colony and the heuristics 17

Introduction

An unexplored topic in scheduling is the mixed problems. This domain is vast and contains a wide variety of problems. Some mixed problems can be very close to other known problems or the problems they are composed of. However there is another sort of mixed problems that can change the way we approach them completely. Our problem is from this second part, it is a combination of a flowshop, a travelling salesman and a batching problem.

We can picture this problem as follows: a factory with two machine in a flowshop are producing objects that need to be delivered to different locations. The produced objects have a limited lifespan, they must reach their destination before a certain date. The objects once produced are organised in delivery batches. A unique truck making the deliveries of the batches choosing a certain delivery route between the factory and the different destinations of the objects. We want to minimise the tardiness of arrival of the objects to their destination, we do not want the objects to reach their destination after their expiration date. The problem can be resumed in three steps:

- Organise the production of the objects on the two production machines of the flowshop,
- Organise the produced objects into delivery batches,
- Choose a delivery route for each delivery batch between the factory and the destinations.

The travelling salesman problem is known to be NP-hard, our problem contains the travelling salesman problem we can deduce the complexity of our combined routing and scheduling problem as being NP-hard. This is for this reason that we don't try to solve the problem at the optimality, we will look for non-optimal approaches.

In this project, we want to try a first approach of the problem with a metaheuristic: the ant colony. The ant colony can be used in order to take into account several constrains which is the case of our combined problem.

The rest of this report features the formalization of the problem, the general definitions of the ant colony metaheuristic, the ant colony heuristic applied to our problem, the results of this heuristic, the perspectives the project and finally the assessment of this project in regard of my scholarship.

Formalization of the problem

The problem can be defined from three different sub-problems :

- The two machines permutation flowshop. We have two machines M_1 and M_2 organized as a flowshop. A set of n jobs have to be scheduled into a sequence. Each job j has a value a_j and b_j that correspond to the time required to complete the job j on M_1 and M_2 respectively.
- The batching of jobs to be delivered. When the jobs are completed for the two machines permutation flowshop we must form batches of jobs, where a batch must contain at least one job. These groups are then used for the third sub-problem.
- A travelling salesman problem. We consider one truck taking the groups of jobs previously formed where each job has a destination k_j. The idea is to find the sequence of destinations in order to deliver every job from the batch while minimizing the travelling distance. We have a matrix K $(m \times m)$ where m is the number of destinations and K_i^j is the distance between the destination i and j. It is important to know that every delivery has to start from the factory and come back to the factory after delivering every jobs. The factory is always the destination 0 in the matrix K.

The objective function is to minimize the sum of tardiness of the jobs $\sum_{j=1}^{j\leq n} T_j$. The tardiness T_j of a job is defined as follows: $T_j = max(L_j, 0)$ where the lateness L_j of a job j is the difference between the completion time of the job C_j and its due date d_j: $L_j = C_j - d_j$. The completion time of a job is not equal to the completion time of the job for the flowshop problem but for the travelling salesman problem. C_j is equal to the date at which the truck is coming back to the factory after its deliveries. We consider that the truck has no capacity limit, it can carry has many jobs has possible.

The input data of our problem are the following :

- The number n of jobs,
- The values a_j, b_j and d_j of job j $(\forall j \in [1, n])$,
- The number of m destinations,
- The destination k_j of job j $(\forall j \in [1, n])$,
- The matrix K $(m \times m)$ of distances between each destination.

During the resolution of the problem we must find the sequence of jobs on the flowshop, the groups of jobs to be delivered and the sequence of delivery for each group.

We proposed a first way of encoding a solution : a table of size n containing the sequence of job for the flowshop sub-problem, a table of size $2n$ containing the groups and sequences for the travelling salesman sub-problem. The first cell of the second table contains the number of jobs for the first group, the following cells contain the id of jobs in the delivery sequence and so on. The figure 2.1 shows a solution for a simple instance of the problem with its encoding.

The sequence here is $\{1, 2, 3\}$ the first batch G_1 of jobs is $G_1 = \{1, 2\}$ and 1 is delivered first, the second batch G_2 only contains one job $G_2 = \{3\}$.

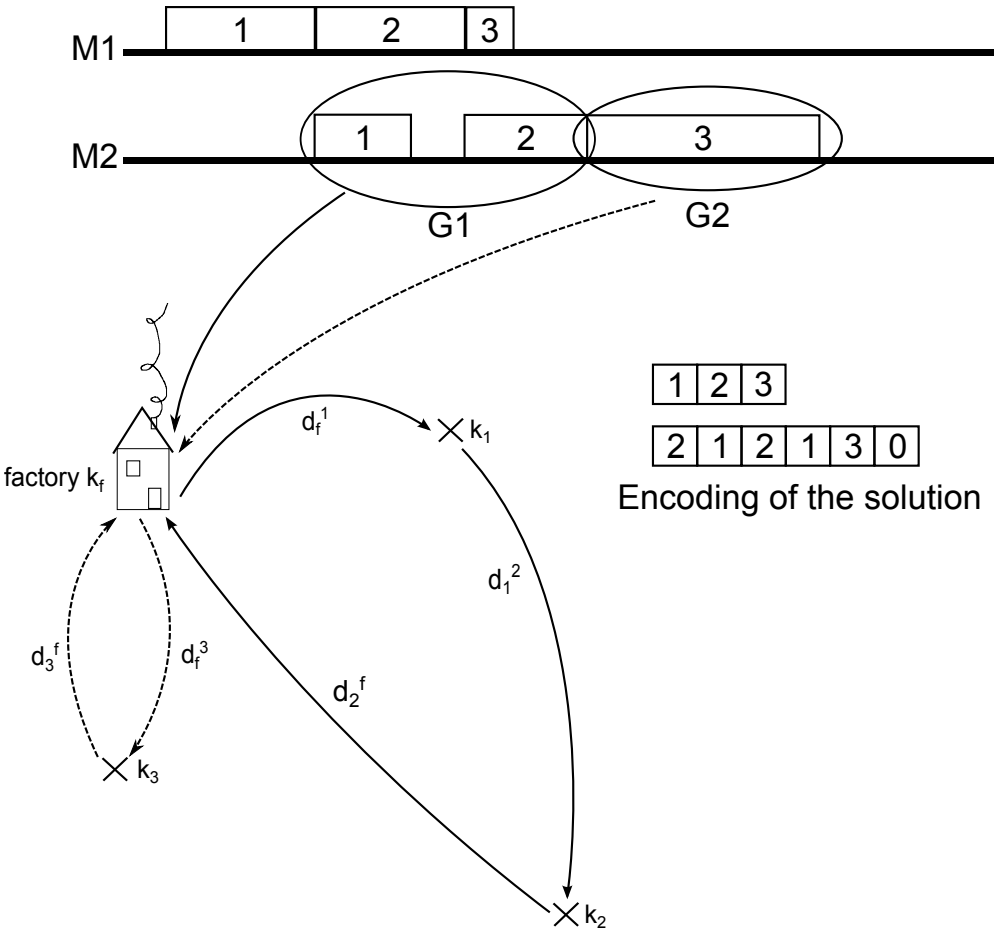

Figure 2.1: A solution of the problem for a simple instance

Ant colony optimization

3.1 General definitions

In this first part we will consider a travelling salesman problem in order to explain the mechanism of the ant colony metaheuristic. The graph representation of cities is an intuitive approach for such a method.

The classic algorithm of an ant colony heuristic is the following :

Algorithm 1 Ant colony

```
for k ← 1 to iterationsNumber do
    for each ant do
        Construct solution
        Local pheromone update
    end for
    Local search
    Global pheromone update
end for
return Best found solution
```

In order to construct the solution we need a probability formula based on the following :

$$P_{ij}^{m} = \begin{cases} \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \notin tabu_m} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}} & \text{if } j \notin tabu_m \\ 0 & \text{if } j \in tabu_m \end{cases}$$

Where P_{ij}^{m} is the probability of an ant m in city i to go to the city j, τij the intensity of the pheromone trail from i to j, η_{ij} the visibility of city j from city i (this is typically $1/d_{ij}$ where d_{ij} is the distance between i and j). α and β are parameters used to make the pheromone trail or the visibility more important from one another. And finally, the tabu list is used to prevent ants from going to forbidden cities like the city they come from.

The local pheromone update is used to emulate the evaporation phenomenon of pheromone. This evaporation is calculated from a formula based on the following :

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0$$

Where ρ is the evaporation rate and τ_0 the minimum value of the pheromone trail, which is also usually used as initial value. It is important to consider that the evaporation function is only used for the paths the ants travelled through.

The global pheromone update is used to emulate the deposit of pheromone from the ants. Again this function is only used for the edges the ant travelled while forming its solution. This deposit is calculated from a formula based on the following :

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}$$

Where $\Delta\tau_{ij}$ is the improvement rate between the new best solution and the previous best solution.

3.2 Application to the combined routing and scheduling problem

One main issue to overcome here is the fact that there is no simple graph representation of the complete problem. For the travelling salesman problem, it is easy to see how the pheromone trails and the visibilities come from. For our problem it is quite different. We can try to imagine a graph that an ant could visit and form a complete solution from it. For instance, a node could represent the fact that a job j is placed first in the flowshop, alone in a delivery group and delivered first. But we then have to consider every possibility knowing that the grouping and delivering problems are dependant on the flowshop. If we change the sequence of jobs in the flowshop we obtain a different grouping problem and then a different travelling salesman problem.

Even though we happen to be able to model such graph, its size would be unreasonably huge. It is not rational to chose such solution. We must abandon the idea to apply the ant colony optimization all at once for our problem and think of a different approach.

Instead of trying to use the ant colony metaheuristic for the complete problem, we can consider using it for parts of it. We can chose to use the ant colony on one or several sub-problem and use different method to construct the rest of the solution and then update the pheromone trails based on the results of the whole solution.

Although is not clever to consider using the ant colony for the travelling salesman sub-problem since the problem is different each time you change the groups to be delivered or the sequence of jobs from the flowshop. The pheromone trails would be lost each time you change the solution in any sub-problem above.

We can still consider the other two sub-problems as potential candidate for the ant colony. Indeed, the flowshop problem and the grouping problem can both be considered central in our global problem. We then have three possibilities, use the ant colony on the flowshop, on the grouping problem or both and use simple heuristics coupled with a surrogate mechanism to obtain the value of the objective function for our solution in order to guide the ants towards the best solution. The surrogate mechanism could be very useful in order to save time computation, it would be used in the majority of cases and replaced with heuristics in the others in order to verify if the current solution is still a good one.

3.2.1 Definition of the Ant colony for the mixed optimisation problem

Here we decided to use the ant colony metaheuristic on the batching problem. It is easy to find heuristics to construct the rest of the solution when we have the groups of jobs. We have to specify how to implement the ant colony algorithm to our sub-problem. This typical visualisation we use for an ant building a solution is the complete graph of cities of the travelling salesman problem. The nodes of the graph are cities, when the ant goes from city a to city b is means we have the sequence ab in our solution. The ant has to visit all nodes with a specific path. It chooses a path from the probability formula defined before.

We use a different construction of solution for our algorithm. We take the jobs as the nodes of the graph and two edges (a green and a red) connecting each node together. The graph has then n nodes and $n(n+1)$ edges. During the construction of a solution, and ant can come from a node a to a node b from two different edges: If it takes the green edge, it means node a and b belong to the same batch. If it takes the red edge, it means the two nodes do not belong to the same batch. The figure 3.1 shows how

to construct the batches from the path travelled by an ant on our graph.

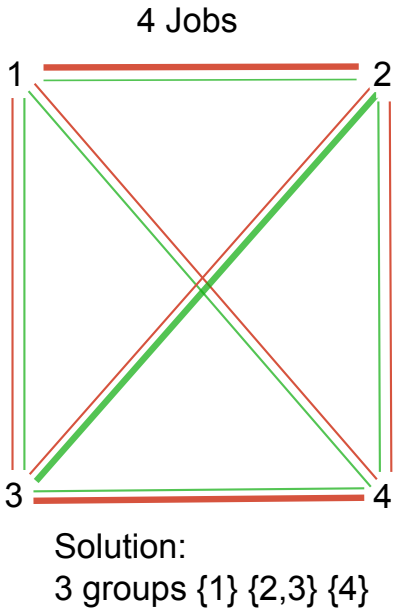

Figure 3.1: The construction of a solution by an ant.

We know how the ants construct the solution and how the graph is made. We need to define the visibility criterion for each edge of our graph. Let's define the visibility for the green edges, what is the formula that will make the ants form groups of jobs? We are handling a complex problem, we need to take into account the two other sub-problems: the flowshop, and the travelling salesman.

For the travelling salesman what we wish to do is group the jobs if their distance is small: $\frac{1}{\delta_{ij}}$ where δ_{ij} is the distance between the destination of job i and the destination of job j.

For the deliveries we know that we have to start from the factory and to end at the factory, the distance between the destinations of the jobs and the factory matters, when the jobs are far from the factory we tend to form groups: $\frac{\delta_{0i} \cdot \delta_{0j}}{\delta_{max}}$ where δ_{0i} is the distance between the factory and the destination of job i and δ_{max} is the maximum distance between the factory and any other destination. We divide by the maximum value in order to keep a value between 0 and 1.

We can then combine those two criteria and add a tuning parameter ω to make one criterion or the other more important:

$$\left(\frac{1}{\delta_{ij}}\right)^{\omega} \cdot \frac{\sqrt{\delta_{0i} \cdot \delta_{0j}}}{\delta_{max}}$$

With this criterion, we will form batches of jobs the further their destinations are from the factory and the closer their destinations are from one another.

For the flowshop sub-problem we want to consider the due dates of the jobs. When the due dates of two jobs are close to each other we will tend to put them into the same group:

$$\frac{1}{|d_i - d_j|}$$

We can combine the criterion from both sub-problem, add tuning parameters and get the formula of the visibility η_{ij}:

$$\eta_{ij}^g = \left(\frac{1}{|d_i - d_j|}\right)^{\psi} \cdot \left[\left(\frac{1}{\delta_{ij}}\right)^{\omega} \cdot \frac{\sqrt{\delta_{0i} \cdot \delta_{0j}}}{\delta_{max}}\right]^{\chi}, \text{ if } |d_i - d_j| = 0 \text{ then } \frac{1}{|d_i - d_j|} = 1 \text{ and if } \delta_{ij} = 0 \text{ then } \frac{1}{\delta_{ij}} = 1$$

We said before that this visibility was to be considered for the green edges of our graph. The visibility η_{ij}^r for the red edges is the complement criteria of the visibility of the green edges:

$$\eta_{ij}^r = 1 - \eta_{ij}^g$$

One last point needs to be defined: How do we choose the edges during the construction of a solution? When forming the batches, an ant has to choose an edge from the node it currently stands to another node. We have our probability formula defined before:

$$P_{ij}^m = \begin{cases} \frac{[\tau_{ij}]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \notin tabu_m} [\tau_{il}]^\alpha \cdot [\eta_{il}]^\beta} & \text{if } j \notin tabu_m \\ 0 & \text{if } j \in tabu_m \end{cases}$$

To make its choice an ant follows the following algorithm:

Algorithm 2 Choosing an edge going from node i

```
a ← Random number ∈ [0; 1]
if a > P_0 then
    return Edge with highest P_ij^m
else
    b ← Random number ∈ [0; Σ_j τ_ij]
    t ← 0
    k ← 1
    while t < b do
        t ← t + τ_ik
        k ← k + 1
    end while
    return Edge k
end if
```

This value P_0 is a diversification probability, it is typically equal to 0.2. It means that 80% of the time the ant will take the edge with the maximum value of P_{ij}^m, and 20% of the time the ant will choose a random edge using the sum of τ_{ij} as probability repartition. The point of such a mechanism is to allow the ants to choose edges disregard their visibility and with a random value in order to diversify the solutions and be able to explore new paths.

3.2.2 Algorithms

Let us call our search graph G, the node set of this graph $\mathscr{N}$, the green edge set $\mathscr{G}$ and the red edge set $\mathscr{R}$.

Algorithm 3 Ant colony for the mixed optimisation problem

Initialize pheromone matrix with τ_0
Calculate the matrix of visibility
List *path*
for $k \leftarrow 1$ to *iterationsNumber* **do**
    for each *ant* **do**
        / Construction of the solution for the grouping problem*/*
        Select random node u from $\mathscr{N}$
        $i \leftarrow 1$
        repeat
            Select an edge v going from u with algorithm 2
            path.push(v)
            $u \leftarrow$ node connected to v
            $i \leftarrow i + 1$
        until $i = n$
        Each community of nodes linked with edged from *path* form a group
        Each remaining nodes form an individual group containing only this node
        / Construction of the rest of the solution */*
        Use heuristics to construct the rest of the solution
        Computation of $\sum_i T_i$
        if the new solution is better than the best known solution **then**
            Save the new solution as the new best known solution
        end if
        Local pheromone update
    end for
    Local search
    Global pheromone update (see algorithm 4)
end for
return Best found solution

Algorithm 4 Global pheromone update

1: $\%improvement \leftarrow (oldBestSolution - newBestSolution)/newBestSolution$
2: $path \leftarrow newBestSolution.getPath()$
3: **for each** edge e in $path$ **do**
4: **if** $edge.color = green$ **then**
5: // τ_0 is the minimal pheromone value, τ_{ij}^g is the pheromone on the green edge between i and j and ρ is the evaporation rate.
6: $i \leftarrow edge.start$
7: $j \leftarrow edge.end$
8: $\tau_{ij}^g = (1 - \rho) \cdot \tau_{ij}^g + \rho \cdot \%improvement$
9: $\tau_{ji}^g = \tau_{ij}^g$
10: **else**
11: $\tau_{ij}^r = (1 - \rho) \cdot \tau_{ij}^r + \rho \cdot \%improvement$
12: $\tau_{ji}^r = \tau_{ij}^r$
13: **end if**
14: **end for**

When we have formed the batches of jobs, we have to deduce the rest of the solution. In order to do so, we use simple heuristics. To form the sequence of the flowshop problem, we order the batches in increasing order of the minimum due date of the jobs within the batch. The jobs of a batch are then scheduled following Johnson's algorithm.

The final part to form our solution is to find the delivery order of the jobs of a batch. To do so, we use the nearest neighbour criterion. For each jobs of a batch, take the nearest neighbour until all the jobs of the batch are processed. We apply this rule to each batch to obtain the fully formed solution.

The method explained above is the first one ever tried on this problem. In order to be able to evaluate its performance we need another one to serve as comparison. It is in that purpose that we proposed a simple heuristic to solve the problem. The heuristic is actually formed of three methods, one for each sub-problem:

- Johnson's algorithm is used to find the sequence of jobs for the flowshop,
- The batches are formed with a simple method. We follow the sequence of jobs formed by the algorithm and decide to put a job into the current batch or to create a new one if the choice gives gives a lesser $\sum T$ than the other.
- For each batch we choose the delivery order of the jobs following the nearest neighbour criterion.

This simple heuristic is used to see if a naive approach is more efficient that our less naive once.

Results

This part presents the results obtained during this project. The value v correspond to the idle time of the truck, the truck is idle when it is not delivering. The value w correspond to the idle time of the jobs, a job is idle when it is completed on the flowshop and waits for being delivered.

Also each row of n resumes 10 instances. The following simple generation has been used to create the instances:

- 10 instances per value of n,
- $n \in \{10; 20; 50\}$,
- $m = n + 1$,
- $a_i \in [1, 100]$, $\forall i \in [0, n]$
- $b_i \in [1, 100]$, $\forall i \in [0, n]$
- $d_i \in [(\gamma - \alpha) \cdot 100, (\gamma + \alpha) \cdot 100)]$, $\forall i \in [0, n]$
- $k_i \in [1, m]$, $\forall i \in [0, n]$
- $K_{ij} = \text{Manhattan}(x_i, y_i, x_j, y_j)$, $\forall i, j \in [0, m]$
- $x_i \in [0, 50]$, $\forall i \in [0, m]$
- $y_i \in [0, 50]$, $\forall i \in [0, m]$
- $\alpha = 0.5$,
- $\gamma = 0.5$.

The used parameters for the ant colony are the following:

- Number of iterations $= 50$,
- Number of ants $= 10$,
- $P_0 = 0.2$,
- $\tau_0 = 0.1$,
- $\rho = 0.1$,
- $\alpha = \beta = \psi = \omega = \chi = 1$.

	Heuristics											
n	$\sum T_{min}$	$\sum T_{avg}$	$\sum T_{max}$	Time(s)$_{min}$	Time(s)$_{avg}$	Time(s)$_{max}$	$\%v_{min}$	$\%v_{avg}$	$\%v_{max}$	$\%w_{min}$	$\%w_{avg}$	$\%w_{max}$
10	2735	3483.3	4327	0	0	0	3	22	40	2	15	26
20	11995	13345.1	14885	0	0	0	0	11	24	19	61	95
50	71125	82427.7	89546	0	0	0	3	12	25	50	70	90
AVG	**28618.33**	**33085.37**	**36252.67**	**0**	**0**	**0**	**2**	**15**	**30**	**24**	**49**	**70**

Table 4.1: Results for heuristics

	Ants											
n	$\sum T_{min}$	$\sum T_{avg}$	$\sum T_{max}$	Time(s)$_{min}$	Time(s)$_{avg}$	Time(s)$_{max}$	$\%v_{min}$	$\%v_{avg}$	$\%v_{max}$	$\%w_{min}$	$\%w_{avg}$	$\%w_{max}$
10	2899	32767.90	5177	0.01	0.01	0.01	30	42	57	0	8	18
20	13528	14043.20	14766	0.02	0.02	0.02	27	41	59	0	9	26
50	72856	82069.80	87965	0.07	0.08	0.09	42	52	59	0	9	25
AVG	**29761**	**33293.63**	**35969.33**	**0.03**	**0.03**	**0.04**	**33**	**45**	**58**	**0**	**9**	**23**

Table 4.2: Results for the ant colony

	Ants - Heuristics					
n	$\Delta \sum T_{min}$	$\Delta \sum T_{avg}$	$\Delta \sum T_{max}$	Δ Time(s)$_{min}$	Δ Time(s)$_{avg}$	Δ Time(s)$_{max}$
10	164	284.6	850	0.01	0.01	0.01
20	1533.00	698.10	-119	0.02	0.02	0.02
50	1731	-357.9	-1581	0.07	0.08	0.09
AVG	**1142.667**	**208.27**	**-283.33**	**0.03**	**0.03**	**0.04**

Table 4.3: Difference of the results between the ant colony and the heuristics

A couple of things are to be noticed from those result tables. Firstly, in average the ant colony heuristic gives worst results in exception of the instances of 50 jobs. The execution time of the ant colony is higher which was to be expected but still insignificant. One last interesting datum is the $\%v_{avg}$ and $\%w_{avg}$ for both the ant colony and the heuristics: the ant colony tends to have the truck wait, when for the heuristics it is the jobs. One concern at the beginning of this project was to be able to generate instances that are not too easy on the flowshop part or too easy on the travelling salesman part. This is why we showed the values v and w to be able to evaluate this criterion. It occurs here that both methods have a different tendency, we can think that our instances are good enough.

Perspectives

This project is the first on such problem, there is still much to do and to explore. One first thing to do is to change the values of the parameters of the ant colony and see how they influence the results. It should be possible to enhance the current results in this way. The second thing is to implement a local search for both the ant colony and the heuristic to see how it can affect the results. The study of the instances and the behaviour of the solution couple with the values of v and w on each method could help understand the mechanisms of the problem and how to improve our methods.

Also there is always to possibility to find different resolution methods for this problem and different instance generations.

Assessment

This project was very enriching, the problem is completely unexplored and it was very interesting to see how we can use the ant colony for this problem. It was also the first time I used the ant colony metaheuristic in a project and helped me grasps the machinery behind it.

This project was not easy and I had some difficulties along it. The first difficulty was the application of the ant colony metaheuristic to our problem as explained at the beginning of the part 3.2. Also the project required a lot of implementation in a short amount of time. I did my best to make the code clear and commented so it could be reused for future projects.

Conclusion

I hope that my work will be useful for future work on this problem, and will help to provide good results to show for the ROADEF conference of 2014. I would like to conclude by thanking both supervisors of this project Jean-Charles Billaut and Nicolas Monmarché for their time, help and sympathy.

Ant colony optimization algorithm for a combined routing and scheduling problem

Computer Aided Decision Support
International Research Master 2
2013 - 2014

Operational research project report

Abstract: This document presents an implementation of the ant colony metaheuristic for a combined routing and scheduling problem along with another simple heuristic. The results and comparison between those two methods are part of this report.

Keywords: Combined scheduling routing, ant colony, optimization, heuristic, metaheuristic, flowshop, batching

---

Supervisors
Jean-Charles BILLAUT
jean-charles.billaut@univ-tours.fr
Nicolas MONMARCHÉ
nicolas.monmarche@univ-tours.fr

Student
Thomas NOGUER
thomas.noguer@etu.univ-tours.fr

M2RI CADS 2013 - 2014

Université François-Rabelais, Tours