# Ant Colony Optimisation for Job Shop Scheduling

**Sjoerd van der Zwaan**

ISR – Instituto de Sistemas e Robótica,
Instituto Superior Técnico (IST).
Av. Rovisco Pais 1,
1096 Lisboa codex

sjoerd@isr.ist.utl.pt

**Carlos Marques**

ISR – Instituto de Sistemas e Robótica,
Instituto Superior Técnico (IST).
Av. Rovisco Pais 1,
1096 Lisboa codex

cmarques@isr.ist.utl.pt

**Abstract**. A recent adaptive algorithm, named Ant System, is introduced and used to solve the problem of job shop scheduling. The algorithm was first introduced by *Dorigo*, *Maniezzo* and *Colorni* in 1991 and is derived from the *foraging and recruiting* behaviour observed in an ant colony. It can be applied to combinatorial optimisation problems. This paper outlines the algorithm's implementation and performance when applied to job shop scheduling. The algorithm parameter settings seem to play a crucial role in its efficiency and determine the quality of solutions. In this paper we present some statistic analysis for parameter tuning and we compare the quality of obtained solutions for well-known benchmark problems in job shop scheduling.

## 1. Introduction

The study from biology of an ant colony shows that its behaviour is highly structured. Knowing that a single ant has limited capacities (i.e., a single ant is not capable of communicating directly with other ants about past experiences), it is curious to know how the ants co-operate so as to achieve such a complex and organised behaviour of the whole colony. Maybe one of the most studied co-operation phenomena among ants is the so-called *foraging and recruiting* behaviour ([1, 2, 3 and 4]). This behaviour describes how ants explore the world in search of food sources, then find their way back to the nest and indicate the food source to the other ants of the colony. To do so, ants use an indirect way to communicate through tracks of pheromone, a chemical substance that they can deposit and are attracted to. Each ant, upon finding a food source, deposit's fractions of pheromone on the way back to the nest so as to indicate the source to the others. The accumulated pheromone serves as a distributed memory, shared by all the other ants and marks in terms of probability the most visited paths between the nest and the food source. Ants encountering an obstacle between their nest and a food source initially choose a random direction of travel. Once a path has fractions of pheromone deposited, it will influence the choice of the next ant confronting the same obstacle, as this path weights more in terms of probability, due to the attracting pheromone. This is illustrated in figure 1.

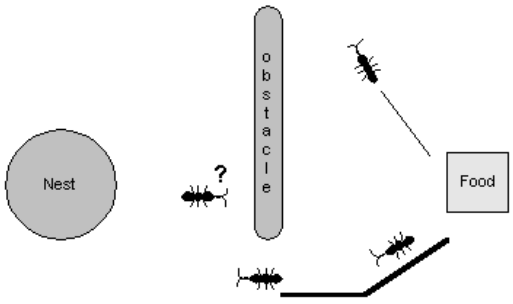


**Figure 1 Positive feedback with accumulated and evaporated pheromone converges to a minimum distance path.**

The effect of a shorter path between the source and nest results in having more ants travelling between

them within the same time-span, thus accumulating more pheromone. Also taking into account the fact that fractions of pheromone evaporate over time, longer paths will be more penalised. This positive feedback with accumulated and evaporated pheromone will automatically eliminate the probability of future ants choosing the longest path.

Inspired by this behaviour, *Dorigo* ([1, 2, 3 and 4]) developed an algorithm named Ant System, which uses a population of co-operating agents, communicating by means of a distributed and shared memory and which can be applied to combinatorial optimisation problems. This paper presents an application of the algorithm applied to the problem of job shop scheduling. The goal of this work is to obtain a properly tuned set of the algorithm parameters such that good quality solutions will be obtained. The outline of this paper is as follows: section 2 introduces the *Ant System* algorithm, section 3 pay's some attention to the problem of job shop scheduling and section 4 deals with the question of how to fit the job shop problem to the *Ant System.* Afterwards, in section 5, convergence properties of the algorithm are investigated so as to obtain a tuned set of parameters. Section 6 presents the obtained simulation results with some benchmark problems. Finally, in section 7, some conclusions are made.

# 2. Ant System

The basic principle of the algorithm is to have a population of *l* artificial ants that cyclical construct solutions to a combinatorial optimisation problem. The algorithm imposes a definition of the optimisation-problem into a graph, in which the ants move along every branch from one node to another node and so construct paths representing solutions. Starting in an initial node, every ant chooses the next node in its path according to the *State Transition Rule*:

$$p_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\frac{1}{d_{ij}}\right]^{\beta}}{\sum_{j \in \text{allowed nodes}} \left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\frac{1}{d_{ij}}\right]^{\beta}}$$

$\tau_{ij}$ - quantity of pheromone on the edge between node$_i$ and node$_j$

$d_{ij}$ - heuristic distance between node$_i$ and node$_j$

$p_{ij}$ - probability to branch from node$_i$ to node$_j$

The parameters $\alpha$ and $\beta$ tune the relative importance in probability of the amount of pheromone versus the heuristic distance. Most likely, the definition of the problem will impose some restrictions on the sequence of nodes that generate valid solutions. If, given a partial constructed solution, a next node is not allowed to be chosen, its probability is set to zero. According to this rule, nodes that have a higher amount of pheromone and that are closer to the actual node (in terms of heuristic distance) will have a higher probability to be scheduled in the partial solution. The heuristic distance performs a local greedy search.

When all the ants have constructed a complete solution, that is a sequence of visited nodes resulting in a complete solution to the problem, the cycle is complete and a pheromone update rule is applied. Pheromone *Global Update Rule:*

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

$$\Delta\tau_{ij}(t+n) = \begin{cases} \dfrac{Q}{f_{evaluation}(best\_so\_far)} \\ 0, \textit{otherwise} \end{cases}$$

$\rho$ - evaporation coefficient

Q - quantity of pheromone per unity of distance

This rule consists of two actions. First a fraction of pheromone on all edges in the graph is evaporated. Second an increment of pheromone is given to those edges that are scheduled within the solution of the ant that has the best-so-far solution to the problem. This increase is inversely proportional to some given evaluation function. In the next cycle those edges belonging to the best-so-far solution will have higher probability, thus exploring this information performs a certain kind of reinforcement learning. On the other hand, evaporation prevents searching in the neighbourhood of a local minimum. The evolution of the algorithm is supposed to cyclically minimise the evaluation function.

A more biological approach for updating would be to apply an increment to the visited edges of all the ants in the colony, inversely proportional to their individual evaluation function value. Ants that have better solutions would deposit a higher amount of pheromone on the edges in their path. However simulation has shown that update with an elitist strategy (best so far only) results in a better performance of the algorithm (see section 6). These alternatives are maintained in the developed simulator, which allows one to apply the pheromone update rule with alternatively the best-so-far ant, all the ants of the population or the two

combined. In this case, more importance can be given to the best-so-far ant by incrementing the number of elitist ants (parameter *e*). In the following sections, the best-so-far strategy is taken as default.

# 3. Job Shop Scheduling

This paper refers to a standard model of the *n*-job, *m*-machine job shop problem, denoted by:

$$n/m/G/C_{max},$$

The parameter *G* indicates that jobs are connected with technological production rules, describing their processing order of machines. This order is specified in the technological matrix *T*. An example for *T* could be:

$$T = \begin{bmatrix} M1 & M2 & M3 \\ M2 & M3 & M1 \end{bmatrix}$$

A row of the matrix represents a job, specifying the sequence of machines to be scheduled. Each element of the matrix *T* is referred to as an operation. The processing time of each operation is specified by matrix P:

$$P = \begin{bmatrix} t(O_{11}) & \dots & t(O_{1m}) \\ t(O_{21}) & & t(O_{2m}) \\ . & & . \\ . & & . \\ t(O_{n1}) & \dots & t(O_{nm}) \end{bmatrix}$$

The matrices *T* and *P* together define a job shop problem. The parameter $C_{max}$ stands for the minimum make-span of the job-shop and indicates the performance measure used to minimise (the so-called evaluation function). Given some solution of a job shop problem, the value of $C_{max}$ is then equivalent to the production time that it takes to finish all the jobs, taking into account the imposed restrictions of machine occupation.

# 4. Implementation

Using Ant System for Job Shop Scheduling, it is necessary to define the problem into a graph. To do so, consider the technological matrix *T* given in the previous section. In [1], an idea is proposed for how to define the job shop into a graph. This is illustrated in figure 2, for the example of the 2/3/G/$C_{max}$ job shop defined by T.

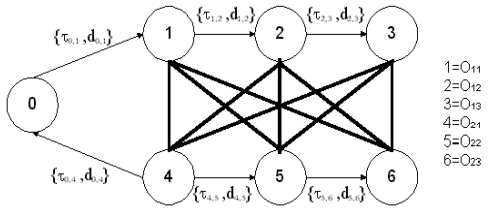


**Figure 2: Definition of a 2/3/G/$C_{max}$ job shop problem into a graph.**

The nodes of the graph represent the operations given by matrix *T* (e.g. $O_{11}$ indicates element $T_{11}$ and equals machine M1). The nodes belonging to the same job are connected by the unidirectional horizontal edges, respecting the technological order of processing a job. The rest of the edges are bi-directional. The maximum number of nodes of a *n*m* job shop is given by:

$$Nodes=(n*m)+1$$

In order to initiate the scheduling, an origin is added from which the first operation of the first chosen job is accessed. If $|O|$ indicates the number of operations in the job shop then the number of edges in the graph is given by:

$$edges = \frac{|O| \cdot (|O| - 1)}{2} + n$$
$$|O| = n * m$$

Each edge has associated a pair of values $\{\tau_{ij}, d_{ij}\}$ representing its amount of pheromone and the heuristic distance between its nodes. The processing time of the operation addressed to by node *j* can be used as a heuristic distance between node *i* and node *j*,. This value can be easily looked up in the *P* matrix. As this heuristic does not take into account the restrictions implied by a possible occupation of machines, the heuristic could turn out to be non-admissible and therefore could easily misdirect local search. Taking into account the restrictions of machine occupation, another heuristic is defined, which is called the *completion time*. For both heuristics, an important note is that in the case of bi-directional edges, their values are not symmetric. The distance for branching from node *i* to node *j* generally is not equal to the distance when branching in opposite direction. Therefore the Ant System has to deal with a non-symmetric graph which implies that the amount of pheromone on bi-directional edges also has to be defined for both directions. The pheromone intensity is defined as a two-dimensional look-up table in memory, with size ([*n*m*]*[*n*m*]) so as to

allow non-symmetric values. The algorithm's spatial complexity (memory) then is given by the size of this table. The spatial complexity of Ant System for Job Shop Scheduling is given by:

$$Spatial\ complexity = O([n*m][n*m])$$

Another important topic is how to guarantee that the Ant System generates a population of feasible solutions, respecting the technological order of the jobs. Each ant simply finds a path trough the graph and will come up with a solution that is a sequence of scheduled operations, represented by a sequence of visited nodes. The algorithm must verify that each ant chooses the nodes in an order that does not violate the technological order of jobs. This is implemented by the idea described by *Bierwith* in [5], proposing a solution for generating feasible solutions with a genetic algorithm applied to job shop scheduling. Accordingly, each ant is equipped with three lists: one list denominated *G* which contains the nodes not yet visited, one list denominated *S* which contains the nodes allowed to be visited in the current iteration (as to verify technological order) and a list denominated *tabu* containing the nodes already visited. Initially all ants will be placed in the origin of the graph and referring to the example of figure 2 the lists for the *k´*th ant will look like:

$$G_k=\{O_{11}, O_{12}, O_{13}, O_{21}, O_{22}., O_{23}\}$$
$$S_k=\{O_{11}, O_{21}\}$$
$$Tabu_k=\{\}$$

After completion of one cycle, each ant should have in its tabu-list a sequence of all the nodes (except for the origin) visited only once and in an order verifying the technological order of the jobs, thus indicating a solution for the job shop problem. Because the algorithm uses a population of artificial ants that every cycle needs to construct a solution, the algorithm temporal complexity (computational time) is given by:

$$Time\ complexity = O(NC*l*[n*m])$$

Where *NC* is the number of cycles, *l* represents the number of ants and *[n*m]* equals the number of operations defined by the job shop problem.

# 5. Parameter Tuning

This section presents an analysis of the performance of Ant System for job shop scheduling in different regions in the parameter space. As in most evolutionary algorithms, the parameter set for which the algorithm performs a desirable convergence is rather problem specific and has to be determined empirical, based on statistic tools, in the absence of a mathematical model.

We think it is useful to consider the parameter-space as composed of two independent sub-spaces. We therefor classify the Ant System parameters into two groups: those that influence the state transition ($\alpha$ and $\beta$) and those that determine the pheromone update (the evaporation constant $\rho$ and the number of ants *m*). From simulation it appears that the parameters Q (pheromone allocation per unity of distance) and $\tau_0$ (initial pheromone level) are of little importance to the algorithm's performance. As default value of Q, the average of a job's production-time is calculated from all the jobs, without taking into account the imposed restrictions. The default value of $\tau_0$ can be set to any number. As to gain some insight into the influence of the parameters $\alpha$ and $\beta$, a first test criterion is defined, which simply tests for which sets $\{\alpha, \beta\}$ the algorithm converges to the a priori known optimum of a selected problem. The job shop selected is the Muth-Thompson 6/6/G/Cmax benchmark problem taken from [6], with known optimum at $C_{max}=55$. During the simulations the values of the evaporation constant and the number of ants were maintained constant ($\rho$=0.01, l= number of nodes) with values empirically proven to allow the algorithm to converge. The results are given in table 1.

| | **α=0** | **α=1** | **α=2** | **α=3** | **α=4** |
|---|---|---|---|---|---|
| **β=0** | >1000 | >1000 | >1000 | >1000 | >1000 |
| **β=1** | >1000 | >1000 | >1000 | >1000 | >1000 |
| **β=2** | >1000 | >1000 | >1000 | >1000 | >1000 |
| **β=3** | >1000 | >1000 | >1000 | >1000 | >1000 |
| **β=4** | >1000 | 614 | >1000 | 150 | >1000 |
| **β=5** | >1000 | 326 | 442 | 249 | >1000 |
| **β=6** | >1000 | 437 | 314 | 202 | 228 |
| **β=7** | >1000 | 283 | 247 | 148 | 267 |
| **β=8** | >1000 | 310 | 260 | 170 | 121 |
| **β=9** | >1000 | 293 | 187 | 164 | 96 |
| **β=10** | >1000 | 186 | 203 | 132 | 96 |

**Table 1: Number of cycles necessary to reach optimum for different values of α and β. Results for each pair (α,β) are given by the median over five simulation of the Muth-Thompson 6/6/G/Cmax problem (maximum number of cycles $NC_{max}$=1000, ρ=0.01, l=36).**

In the case of $\alpha$=0 and $\beta$=0, every ant simply branches randomly from one node to the other, for which the algorithm is expected not to find the optimum. For all those values $\{\alpha=0, \beta\}$ the ant colony does not use communication through pheromone and every ant simply performs a heuristic greedy search in probability. As could be

expected the algorithm is not able to find the optimum within the maximum number of cycles. For all those values {α, β=0} the ant colony does not make use of local heuristic information and every ant directs its search by the pheromone intensity levels. Also in this case the algorithm was not able to find the optimum which states the necessity of local heuristic information. For the rest of the values {α, β} that did not reach the optimum within the maximum number of cycles, the algorithm directed the search to a local minimum of the evaluation function, not able to recover the optimum. Analysis of the bottom line (β=10) of table 1 shows a declination of the number of cycles for increasing α. One might suspect this trend to continue for α>4.

| | α=5 | α=6 | α=7 | α=8 | α=9 | α=10 |
|---|---|---|---|---|---|---|
| β=10 | 79 | 93 | 73 | 77 | 50 | 49 |

**Table 2: Number of cycles necessary to reach optimum for different values of α with β=10. Results are given by the median over five simulation of the Muth-Thompson 6/6/G/Cmax problem (maximum number of cycles $NC_{max}$=1000, ρ=0.01, l=36).**

Indeed this fact is confirmed and simulation shows that a minimum number of cycles can be obtained with the settings: α=10, β=10.

Once established the values of α and β, the influence of the parameters that define the pheromone update can be investigated. Figure 3 shows the performance of the algorithm for different values of the evaporation constant ρ.

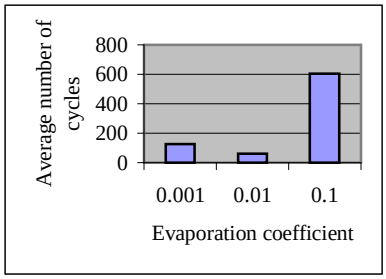


**Figure 3: Influence of different setting of the evaporation coefficient on the average number of cycles to reach optimum. Averages are taken over five runs of the Muth-Thompson 6/6/G/Cmax problem. (Maximum number of cycles $NC_{max}$=1000, α=10, β=10, l=36).**

In the case of ρ=0.1, the algorithm does not always find the optimum. This is due to the fact that when the evaporation rate is too high, only the edges belonging to the path with the early encountered local minimum receive pheromone update every cycle. The rest of the edges have their pheromone rapidly decaying to zero with increasing number of cycles, due to the evaporation. Each new cycle, this effect is amplified and the ants will be more and more attracted to the edges that receive pheromone update, thus searching in the small neighbourhood of the local minimum. Figures 4 and 5 show the pheromone intensity levels after 25 cycles with good and bad parameter settings for ρ.

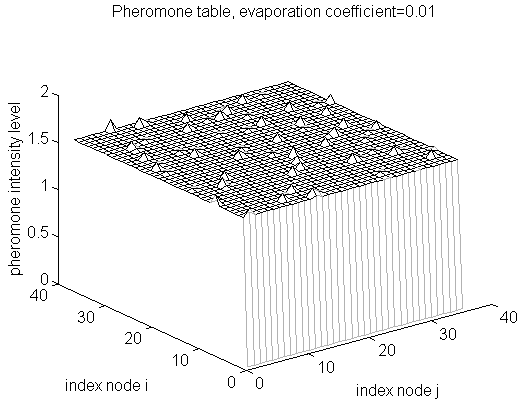


**Figure 4: Pheromone intensity level after 25 cycles for good evaporation coefficient setting (α=10, β=10, ρ=0.01). Search is still exploring the state space with more probability for the local maximums in pheromone intensity level.**

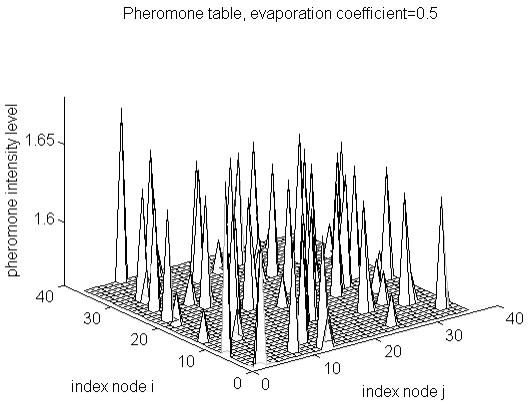


**Figure 5: Pheromone intensity level after 25 cycles for bad evaporation coefficient setting (α=10, β=10, ρ=0.1). Search is in the small neighbourhood of the early encountered peaks of pheromone intensity.**

The small variations of the pheromone intensity obtained with the good evaporation settings will be amplified more in probability by the state transition rule according to parameter α>1. The probability of nodes not belonging to these local maximums in pheromone intensity maintains relevance. For the bad settings, this amplification results in probabilistic elimination of those nodes that do not

belong to the early found peaks in pheromone intensity.

The influence of the number of ants in the colony on the algorithm's performance is shown in figure 6.

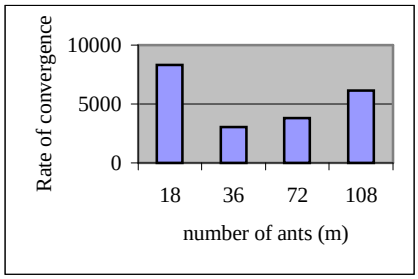


**Figure 6: Influence of the number of ants on the rate of convergence. Average number of cycles to reach optimum is taken over five runs of the 6/6/G/Cmax Muth-Thompson problem. (Maximum number of cycles $NC_{max}$=1000, $\alpha$=10, $\beta$=10, $\rho$=0.01).**

The algorithm's rate of convergence is calculated as the average number of cycles to reach the optimum multiplied by the number of ants. From figure 6 it follows that the optimal number of ants must be somewhere near to the number of nodes in the graph (n*m).

# 6. Results

After optimisation of the parameters in the previous section for the specific Muth-Thompson 6/6/G/Cmax job shop problem, this section shows some simulation results of the Ant System to some more complex benchmark problems, with the set of tuned parameters as obtained in the previous section 5. However, first some typical runs of the Muth-Thompson 6/6/G/Cmax job shop problem are illustrated.

Figure 7 shows a typical run of the Ant System, comparing the evolution of the evaluation function belonging to the best-so-far ant for different parameter settings.

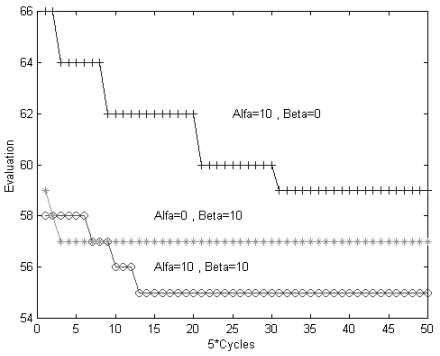


**Figure 7: Typical run for the Muth-Thompson 6/6/G/Cmax job shop problem. The importance of pheromone communication once again is confirmed.**

From figure 8 it follows that the elitist strategy (introduced in section 4) performs better then its biological counterpart in which all the ants apply a pheromone update to the edges in their path, inverse proportional to their evaluation.

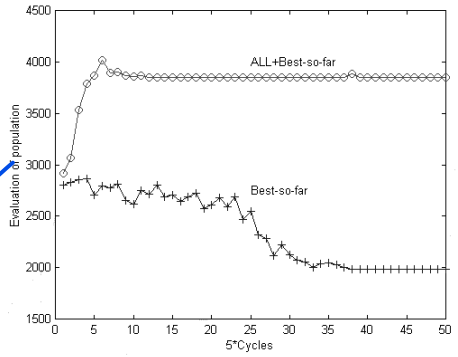


**Figure 8: Comparison between two different pheromone-update strategies.**

Pheromone update with all the ants (the more biological counterpart) results in worse convergence of the evaluation function of the population.

From examining the evaluation of the entire population, it can be concluded that at a given cycle the search gets trapped in the small neighbourhood of a local minimum (figure 9). Variations in the evaluation of the whole colony mean that the ants are exploring in different directions over the search space. To prevent the search for getting trapped, a new operator is introduced called *variation* that substitutes the State Transition rule when triggered.

$$\begin{cases} \text{State Transition Rule}, & n > v \\ p_{ij} = \dfrac{\dfrac{1}{d_{ij}}}{\sum_{j \in \text{nodes allowed}} \dfrac{1}{d_{ij}}}, & n < v \end{cases}$$

$n$ - random number between $[0..1]$

$v$ - percentage of variation $[0..1]$

The effect of different percentages of variation is illustrated in figure 9.

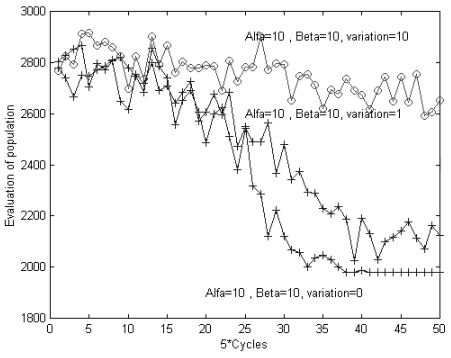


**Figure 9: Applying the *variation* operator helps to open the search. Ants will be forced to explore more in search space.**

Without the variation the population's evaluation function converges to a constant level which means that all the ants are choosing the same path. In the case that this path minimises the evaluation function and finds the optimum, this may be considered a desirable situation. If not so, the search gets trapped in a local minimum. A biological explanation for introducing the variation parameter could be seen as some saturation in pheromone perception for each ant. For a given percentage of variation per cycle, the ants will be insensible to pheromone and thereby guide their search by heuristic information only.

The Ant System was partially tested on three well-known benchmark problems provided by [6]. The first test was the 6/6/G/$C_{max}$ Muth-Thompson problem that was always solved to optimality $C_{max}$=55 (as described in the previous sections). The second test was the 10/10/G/$C_{max}$ Muth-Thompson problem with known optimum for $C_{max}$=930. The third test was the 20/10/G/$C_{max}$ Lawrence problem "la26", with known optimal for $C_{max}$=1218. The obtained results for Ant System applied to these problems are presented in table 3 for different parameter setting for *e* and *v*.

| **Problem** | **$C_{max}$** | **e** | **v** | **% within optimum** |
|---|---|---|---|---|
| 10/10/G/$C_{max}$ | 1052 | 1 | 0 | 13% |
| 10/10/G/$C_{max}$ | 1006 | 1 | 1 | 8% |
| 10/10/G/$C_{max}$ | 1063 | 1 | 10 | 14% |
| 10/10/G/$C_{max}$ | 1019 | 2 | 2 | 9.5% |
| 10/10/G/$C_{max}$ | 1054 | 3 | 1 | 13.3% |
| 10/10/G/$C_{max}$ | 1041 | 4 | 3 | 12% |
| 20/10/G/$C_{max}$ | 1604 | 1 | 0 | 31% |
| 20/10/G/$C_{max}$ | 1607 | 1 | 1 | 31.9% |
| 20/10/G/$C_{max}$ | 1535 | 3 | 1 | 26% |

**Table 3: Obtained results for the 10/10/G/$C_{max}$ Muth-Thompson and 20/10/G/$C_{max}$ Lawrence problems. (Maximum number of cycles $NC_{max}$=2000, $\alpha$=10, $\beta$=10, l=n*m).**

# 7. Conclusions

This paper showed how to solve the problem of job shop scheduling with the Ant System. The goal of this work was to gain some insight into the influence of different parameter-settings for Ant System, which seem to play an important role on its performance and determine the quality of solutions. Deriving good statistics helped a lot to gain insight into the system's behaviour and classifying the parameter-space into two independent sub-spaces is a useful way to start experimenting. From this work we conclude that the parameters $\alpha$ and $\beta$ of the State Transition Rule, determine the convergence rate of the algorithm as well as the quality of the obtained solution. To allow the algorithm to converge to a satisfactory solution, the evaporation constant $\varphi$ has to be well tuned so as to guide the search into favoured regions in the search space and at the same time prevent searching in small neighbourhoods of local optima. The introduction of the variation-parameter *v* (similar to the mutation operator in genetic algorithms) allows to guide the search more into sub-optimal regions of the search-space without loosing the algorithm's capability of recovering from dead-ends by imposing to also explore other directions in the search-space. Once the parameters are properly tuned, the algorithm converges satisfactory, thus accomplishing the stated goal of this work. The Ant System was partially tested for more complex job shop problems. In these cases it could always find an optimum within 8% of the best known optimum for the 10/10/G/$C_{max}$ Muth-Thompson problem and within 26% for the 20/10/G/$C_{max}$ Lawrence problem. Reminding that tests were only executed partially ($NC_{max}$=2000) due to the algorithms time-complexity, this can said to be promising.

The main advantage of the Ant System is that it easily deals with combinatorial optimisation problems defined on a non-symmetric graph. The only adaptation to be made for dealing with non-symmetry is the expansion of the pheromone table. This increases its spatial complexity (memory) but does not necessarily require extra computational power. On the other hand it is exactly the Ant System's time-complexity that can be said to be its major disadvantage. The Ant system time complexity in comparison with genetic algorithms increases exponentially with the population size. This is due to the fact that within every cycle, the Ant System needs to construct solutions for all elements in the population, where a genetic algorithm parts from a population with already constructed solutions. Therefore we suggest a genetic algorithm approach to those combinatorial optimisation problems that can be defined by symmetric graphs.

# References


[1] M. Dorigo, V. Maniezzo, A. Colorni, "The Ant System: Optimization by a colony of co-operating agents"" *IEEE Trans. Syst, Man, Cybern.* Vol.26, no.2, pp.29-41, 1996.

[2] M. Dorigo, L. M. Gambardella, "Ant colony System: A Cooperative Learning Approach to the Travelling Salesman Problem", *IEEE Trans. On Evolutionary Computation*, vol.1, no.1, April 1997.

[3] M. Dorigo, V. Maniezzo, A. Colorni, "Distributed Optimization by Ant Colonies", *Proceedings of ECAL91 – European Conference on Artificial Life*, Elsevier Publishing, 134-142, 1991.

[4] M. Dorigo, V. Maniezzo, A. Colorni, "An Investigation of some properties of an Ant Algorithm", *Proceedings of the Parallel Problem Solving From Nature Conference (PPSN92)*, Brussels, Belgium, Elsevier Publishing, 509-520, 1992.

[5] C. Bierwith, "A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms", Department of Economics, University of Bremen, Germany.

[6] Job Shop Scheduling Benchmark, OR-library, http://mscmga.ms.ic.ac.uk/jeb/orlib/jobshopinfo.html.