# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

**4,800**
Open access books available

**122,000**
International authors and editors

**135M**
Downloads

Our authors are among the

**154**
Countries delivered to

**TOP 1%**
most cited scientists

**12.2%**
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

## Interested in publishing with us?
## Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

20

# Job-shop scheduling and visibility studies with a hybrid ACO algorithm

Heinonen, J. and Pettersson, F.
*Åbo Akademi University*
*Finland*

## 1. Introduction

Manufacturing today is primarily cooked down to all-out efforts into profitability. Factories are moved to low-salary countries in order to ensure that profits are maintained and stockholders kept happy. Decisions like these are met with debates about morale, ethics and responsibilities that companies have to society, since losing an entire manufacturing plant can be devastating to a community. An alternative to industrial relocalization is trying to maintain profitability through development of effective production schedules, better utilization of resources and overall better planning in existing manufacturing plants. The significance of effective planning methods has, in other words, increased and will likely continue to do so.

The focus of this chapter is to solve the MT10 job-shop scheduling problem using 4 different variants of the Ant Colony Optimization (ACO) algorithm and to try to rank them. A hybrid model, that uses a postprocessing algorithm to improve the resulting schedule, is also tried for all four ACO versions. The term *visibility* is explained in the context of job-shop scheduling, and incorporated into the test runs.

When we are talking about job-shop scheduling problems (JSP), we mean a set of machines $\boldsymbol{M}$, a set of jobs $\boldsymbol{J}$ and a set of operations $\boldsymbol{O}$. For each operation there is a job to which it belongs, a machine on which it it has to be processed, a predetermined processing time on that machine as well as a predetermined processing order on the machines. The problem is to minimize the makespan while ensuring that no more than one job can be processed at the same time on the same machine, and seeing to that when a job starts, it must be completed (and can't be interrupted).

There have been numerous publications of successful algorithms applied to job-shop problems. Among exact mathematical methods are Mixed integer linear programming and Branch & Bound, among approximation methods there are List Scheduler Algorithms (see Panwalker & Iskander, 1977 for a survey), that assign one operation at a time from a list that is sorted by some priority rule, Shifting Bottleneck by Adams et al. (1988), Simulated Annealing by van Laarhoven et al. (1988), Tabu search was first used in job shop scheduling by Taillard (1989) and a Genetic algorithm approach by Nakano and Yamada (1991).

A newcomer in these approaches to the JSP, ant colony optimization has become an increasingly popular candidate when it comes to algorithms that mimic behaviour of processes that exist in nature. The first ACO algorithm was introduced by Marco Dorigo in



Source: Swarm Intelligence: Focus on Ant and Particle Swarm Optimization, Book edited by: Felix T. S. Chan and Manoj Kumar Tiwari, ISBN 978-3-902613-09-7, pp. 532, December 2007, Itech Education and Publishing, Vienna, Austria

his doctoral thesis (1992) and was called an Ant System (AS). Since then AS has matured into an algorithm that does very well when it comes to problem types that are formulated as a traveling salesman problem (TSP) as well as the quadratic assignment problem (QAP). As a result of research into ACO algorithms, some very successful variants have emerged.

We have the Elitist AS (EAS) proposed by Dorigo et al. (1996), in which the pheromone updating rules are biased towards the best solution found so far, the idea being to exploit the solution components within that solution.

Ant Colony System (ACS) by Dorigo and Gambardella (1997) has several modifications to the original AS. It uses a modified rule when an ant chooses the next travel node, it uses a best-so-far pheromone update rule but applies pheromone evaporation only to the trail that belong to solution components that are in the best-so-far solution. It also uses a local pheromone update rule to decrease the pheromone values on visited solution components, in order to encourage exploration.

Rank-based AS (RAS) by Bullnheimer et al. (1999), is a variant where the elite ant as well as a selection of ants with good solutions during that iteration get to update the pheromone trails.

*MAX-MIN* AS (*MM*AS) by Stützle and Hoos (2000), is an approach that updates the pheromone trails, according to some convergence measure, with either the iteration-best ant or the best-so-far ant. The algorithm uses a lower bound for the pheromones (>0) as well as restricting the maximum amount of pheromone a trail can have. The lower bound encourage ant exploratory behaviour and the upper bound is prohibiting premature convergence due to the elite solution dominating the other solutions.

Hypercube Framework (HCF) by Blum and Dorigo (2004) is more of a framework for implementing ACO algorithms. Among the benefits are automatic scaling of pheromone values to the interval [0,1].

In a paper by Colorni et al. (1993) AS was applied into job-shop scheduling and proved to be a noteworthy candidate when faced with the task of chosing a suitable algorithm for scheduling problems. The conclusions in the aforementioned paper were that AS is one of the most easily adaptable population-based heuristics so far proposed and that its computational paradigm is indeed effective under very different conditions.

As an example of ACO robustness, Jayaraman et al. (2000) used an ACO algorithm in solving a combinatorial optimization problem of multiproduct batch scheduling as well as the continuous function optimization problem for the design of multiproduct plant with single product campaigns and horizon constraints. Further real-world applications with regard to ACO algorithms would be using ACO to solve an established set of vehicle routing problems as done by Bell and McMullen (2004) and a dynamic regional nurse-scheduling problem in Austria by Gutjahr and Rauner (2005). The former paper concluded the results were competetive and in the latter paper ACO was compared to a greedy assignment algorithm and achieved highly significant improvements.

Kuo-Ching Ying et al. (2004) applied the ant colony system to permutation flow-shop sequencing and effectively solved the $n/m/P/C_{max}$ problem, and commented that this suggests that the ant colony system metaheuristic is well worth exploring in the context of solving different scheduling problems.

An example of ACO and flowshops in recent use would be a paper by Gajpal and Rajendran (2006), where they used a new ACO algorithm (NACO) to minimize the completion-

variance of jobs, showing that work with ACO algorithms is an ongoing process to modify and improve the original AS and apply it to a variety of scheduling problems.
For two of the top performing ACO algorithms, ACS and *MM*AS, convergence to the optimal solution has been proved (Dorigo and Stützle, 2004 as well as Stützle and Dorigo, 2002). It is worth to remember that convergence results do not allow prediction of how quickly an optimal solution can be found.

## 2. Problem description

The Job-Shop Scheduling Problem (JSP) can be characterized as $n$ jobs to be processed on $m$ machines. In general it is a set of concurrent and conflicting goals to be satisfied using a finite set of resources where resources are called machines and basic tasks are called jobs. Each job is a request for scheduling a set of operations according to a process plan which specifies precedence restrictions. We have

$$M = \{M_1, \ldots, M_m\} \quad \text{a given set of machines}$$

$$J = \{J_1, \ldots, J_n\} \quad \text{a given set of jobs}$$

$$O = \{O_1, \ldots, O_n\} \quad \text{a set of operations}$$

For each operation $u_{ij} \in O$ there is a job $J_i$ to which it belongs, a machine $M_j$ on which it has to be run and a processing time $p_{ij}$ of the operation $u_{ij}$, where $p_{ij}$ is a nonnegative integer. Every job is a chain of operations and every operation has to be processed on a given machine for a given time. The task is to find the starting times of all operations such that the completion time of the very last operation is minimal. The chain order of each job has to be maintained and each machine can only process one job at the same time. No job can be preempted; once an operation starts it must be completed. The solution $s$ to an instance of the $n$ x $m$ JSP specifies a processing order for all of the jobs on each machine and implicitly defines an earliest starttime and earliest completion time for each operation. The maximum of the completion times is called *makespan* and most research address the problem of makespan minimization.
Given an instance of JSP we can associate with it a disjunctive graph $G = (V, A, E)$, where $V$ is the node set, $A$ is the conjunctive arc set and $E$ is the disjunctive arc set. The nodes $V$ correspond to all of the operations and two dummy nodes, a source and a sink. The conjunctive arcs $A$ represent the precedence relationships between the operations of a single job and the disjunctive arcs $E$ represent all pairs of operations to be performed on the same machine. All arcs emanating from a node have the processing time of the operation performed at that node as their length. The source has conjunctive arcs with length zero emanating to all the first operations of the job and the sink has conjunctive arcs coming from all the last operations. A feasible schedule corresponds to a selection of one arc from each disjunctive arc pair such that the resulting directed graph is acyclic, i.e. no loops can be found. The problem of minimizing the makespan reduces to finding a set of disjunctive arcs which minimize the length of the critical path in the directed graph. An in-depth description of disjunctive graphs with regard to job-shop problems can be found in for instance the article about AS and JSP by Colorni et al. (1993).

The MT10 problem is a 10 x 10 instance formulated by Muth and Thompson in 1963. It consists of 10 jobs processed on 10 machines, and every job has 10 tasks to perform. The processing times vary greatly with shortest duration being only 2 time units and longest 99 time units. It has the reputation of being one of the most difficult combinatorial problems ever considered, and was not solved exactly until as late as 1989 by Carlier and Pinson using a branch and bound algorithm. It is a typical job-shop problem.

## 3. ACO

ACO belongs to the class metaheuristics. The term *metaheuristic* is derived from two greek words, *heuristic* which means "to find" and the prefix *meta*, which means "beyond, in the sense of an upper level". It has come to mean a high-level strategy for guiding heuristics in a search for feasible solutions as well as a framework that can be specialized to solve optimization problems. ACO is also a succesful example of swarm intelligence, whose purpose is to design intelligent multi-agent systems by taking inspirations from the collective behaviour of social insects.

ACO is modeled after the foraging behaviour of certain ant species. In the 1940s the French entomologist Pierre-Paul Grassé observed that some species of termites react to what he called "significant stimuli" (Grassé, 1946). He used the term "stigmergy" to describe the communication of ants, and he described this communication as workers being stimulated by the performance they have achieved. Ants alter their environment by means of pheromone trails. A pheromone is any chemical or set of chemicals produced by a living organism that transmits a message to other members of the same species. It is volatile and evaporates quickly, and ants secrete this chemical by walking and follow, in turn, other pheromone trails left by other ants. There are alarm pheromones, food trail pheromones and others that affect behavior or physiology. Strong food trail pheromone concentrations are perceived and stimulate ants to move into that direction. Ants are able to transport food through this mechanism by finding and maintaining the shortest path between the food source and the nest. Occasionally there will be the stray ant taking another route, and this event can be seen as exploration, the ants are constantly trying to find a more effective path. This mechanism was demonstrated by Denebourg et al. (1990), who in an experiment called "the double bridge" connected a nest of Argentine ants with a food source. Figure 1 (a) shows the experimental setup and figure 1 (b) another experimental setup by Goss et al. (1989). If the setup is that of figure 1 (a), initially, each ant randomly chooses one of the two bridges. Due to random fluctuations, after some time one of the two bridges presents a higher concentration of pheromone and attracts more ants. After a while almost the whole colony converges toward the use of the same bridge. With the setup illustrated in figure 1 (b) another mechanism besides random fluctuations was demonstrated: the ants randomly choosing the shorter path travel between the nest and the food source faster and, given time, this means that pheromone will accumulate faster on this path, converging the population towards using this shorter path.

The mechanism can be utilized in order to find the shortest path in, for instance, minimizing makespan for scheduling problems. The underlying problems are formulated as a TSP, that is, a connected, undirected graph $G = (V, E)$ with weights on the edges between the nodes. The nodes V denote the cities, and the edge weight is the distance between two cities. The goal is to find a tour in G that connects all cities once so that the overall length is minimal.

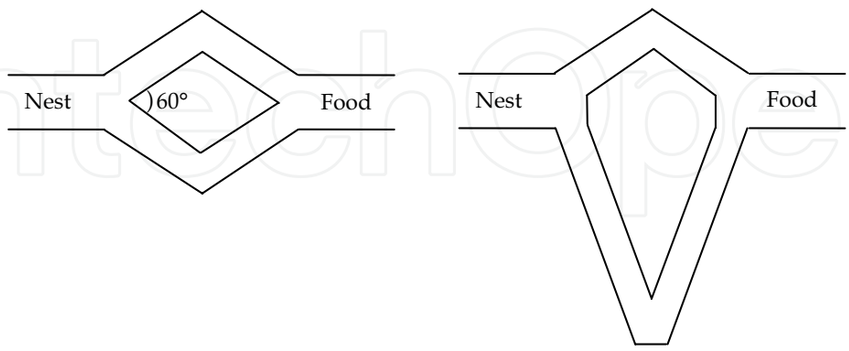



Figure 1. Experimental setup for the double bridge experiments: (a) branches have equal length; (b) branches have different lengths

Having artifical ants search the solution space simulate real ants searching their environment. The artifical ants can be equipped with some oddities that real life ants don't have, for instance a local heuristic function to guide their search through a set of feasible solutions only, or an adaptive memory corresponding to the pheromone trail so that they can remember visited nodes. Also we require the ants to be symmetrical in the sense that they move from the nest to the food and back using the same path. The ACO algorithm also keeps tracks of visited nodes, meaning the ants have a memory which helps them select the next node from a list of possible choices.

### 3.1 Ant System (AS)

Each edge $e_{ij}$ has a pheromone value $\tau_{ij}$ associated with it, and this pheromone value can be read and modified by the ants. The algorithm starts with the user sprinkling some pheromone on random edges. All ants are initially in their home nest, and move to a node in their feasible list. When located at a node $i$ an ant $k$ uses the pheromone trails $\tau_{ij}$ to compute the probability of choosing node $j$ as the next node:

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha}\eta_{il}^{\beta}}, j \in N_i^k \\ 0, j \notin N_i^k \end{cases} \quad (1)$$

$N_i^k$ is the feasible neighbourhood of ant $k$ when in node $i$, that is, the list of cities that ant $k$ has not yet visited. The parameter $\eta_{ij} = C / d_{ij}$, where $d_{ij}$ is the distance between nodes $i$ and $j$, and $C$ is a positive constant, is a measure of heuristic information, in other words $\eta_{ij}$ is our *visibility*. Parameters $a$ and $\beta$ determine the relative influence of the pheromone trail and the heuristic information. If $a = 0$ then the closest cities are more likely to be selected. If $\beta = 0$ then only pheromone amplification is at work, which generally leads to the rapid emergence

of a stagnation situation, all ants eventually follow the same path and construct the same tour. Dorigo found in his studies (Dorigo et al. 1996) that typically $\beta > a$.
Once all ants have completed their tour the pheromone trails get updated. The pheromone values are modified in order to bias ants in the future iterations to construct solutions similar to the best ones previously constructed. First the pheromone on all arcs is lowered by a constant, and then pheromone is added on the arcs that the ants have passed in their tour. Evaporation is implemented by:

$$\tau_{ij} \leftarrow (1-p)\tau_{ij} \tag{2}$$

where $0 < p \leq 1$ is the evaporation rate. This enables the algorithm to "forget" previous bad decisions and avoids unlimited accumulation on the edges. The deposition of pheromone on the edges is done by means of global trail update

$$\tau_{ij} \leftarrow (1-p)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \forall (i,j) \in L \tag{3}$$

where $\Delta\tau_{ij}^{k}$ is the amount of pheromone ant $k$ deposits on the arcs it has visited, which usually amounts to the value $Q / C^{k}$, where $C^{k}$ is the length of the tour and $Q$ is a positive constant. This means that arcs used by many ants, and therefore part of short tours, receive more pheromone and are therefore more likely to be chosen by ants in future iterations of the algorithm. When using an elitist ant system, the solution presented by the best-solution-so-far adds extra pheromone on its arcs. Equation (3) becomes

$$\tau_{ij} \leftarrow (1-p)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs}, \forall (i,j) \in L \tag{4}$$

where $e$ is a parameter that defines the weight given to the best-so-far tour and

$$\Delta\tau_{ij}^{bs} = 1/C^{bs} \text{ if } e_{ij} \text{ belongs to the ants tour, 0 otherwise} \tag{5}$$

where $C^{bs}$ is the length of the best-so-far tour.
When initializing the system, all ants can be placed in the starting node or sprinkled randomly over all nodes. Dorigo studied the differences and came to the conclusion that it had little effect, though placing them randomly gave sometimes slightly better performance.
Also, the differences between three AS algorithms, *ant-cycle*, *ant-density* and *ant-quantity* were studied in the same paper. In the latter two models each ant lay its trail at each step, without waiting for the end of the tour, whereas in the *ant-cycle* pheromone updates occur at the end of the tour. In the *ant-density* model a quantity $Q$ of trail is left on edge $(i,j)$ every time an ant goes from i to j, whereas in the *ant-quantity* model the amount of pheromone left was $Q/d_{ij}$. The *ant-cycle* model performed best and was chosen, and is the one depicted in the equations above.

### 3.2 Rank-based Ant System (RAS)

This version is an extension to the original AS. After all $m$ ants have generated a tour, the ants are sorted by tour length and the contribution of an ant to the pheromone trail update is weighted according to the rank $\mu$ of the ant. An elitist strategy is used as well.
Only the $\omega$ best ants are considered and $\omega = \sigma - 1$, where σ is the number of elitist ants in the system. This means that equation (4) is modified accordingly

$$\tau_{ij} \leftarrow (1-p)\tau_{ij} + \sum_{\mu=1}^{\sigma-1} \Delta\tau_{ij}^{\mu} + \Delta\tau_{ij}^{*}, \forall (i,j) \in L \tag{6}$$

where $\Delta\tau_{ij}^{\mu} = \begin{cases} (\sigma-\mu)\dfrac{Q}{L_{\mu}} & \text{if the } \mu\text{–th best ant travels on edge } (i,j) \\ 0 & \text{otherwise} \end{cases}$

and $\Delta\tau_{ij}^{*} = \begin{cases} \sigma\dfrac{Q}{L^{*}} & \text{if edge } (i,j) \text{ is part of the best solution found} \\ 0 & \text{otherwise} \end{cases}$

where

- $\mu$ ranking index
- $\Delta\tau_{ij}^{\mu}$ increase of trail level on edge $(i,j)$ caused by the $\mu$-th best ant
- $L_{\mu}$ tour length of the $\mu$-th best ant
- $\Delta\tau_{ij}^{*}$ increase of trail level on edge $(i,j)$ caused by the elitist ants
- $\sigma$ number of elitist ants
- $L^{*}$ tour length of the best solution found

In the paper (Bullnheimer et al. 1999) RAS is put to the test against AS, EAS as well as simulated annealing and a genetic algorithm. The conclusion was that RAS could for all problem instances compete with the classical metaheuristics regarding speed and quality, and that the ranking improved the performance of the ant system algorithm in every respect.

### 3.3 Ant Colony System (ACS)

ACS proposed by Dorigo and Gambardella (1997) introduced a new state transition rule to provide a direct way to balance between exploration of new edges and exploitation of a priori and accumulated knowledge about the problem.

$$j = \begin{cases} \underset{j \in N_k^i}{arg\,max} \left\{ \tau_{ij} \cdot \eta_{ij}^{\beta} \right\} & \text{if } q < q_o \quad \text{(exploitation)} \\ J & \text{otherwise (biased exploration)} \end{cases} \tag{7}$$

where $q$ is a random number uniformly distributed in [0,1] , $q_o$ is a parameter ( $0 \le q_o \le 1$) and $J$ is a random node selected according to the probability distribution given in equation 1.
This means that every time an ant in city $i$ has to choose a city $j$ to move to, it samples a random number $q$. If $q \le q_o$ then the best edge according to equation 3 is chosen, otherwise and edge is chose according to equation 1.
While ants are constructing a solution a *local pheromone updating rule* is applied

$$\tau_{ij} \leftarrow (1-\sigma)\cdot\tau_{ij} + \sigma\cdot\Delta\tau_{ij}^{k}, \forall(i,j)\in L \tag{8}$$

and σ is a parameter $0 < \sigma < 1$ and $\Delta\tau_{ij}^{k}$ is $1/(nL_{nn})$, where $n$ is the number of nodes in the problem and $L_{nn}$ is the tour length produced by the nearest neighbour heuristic (see Rosenkrantz et al. 1977).
The *global pheromone updating rule* is applied only to edges that belong to the best ant tour

$$\tau_{ij} \leftarrow (1-p)\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}, \forall(i,j)\in L \tag{9}$$

where $$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{1}{L_{gb}} & \text{if } (i,j) \text{ is part of the global best tour} \\ 0 & \text{otherwise} \end{cases}$$

and $L_{gb}$ is the length of the globally best tour.
Noticeable in ACS is that the local updating rule is applied in parallel, every time an ant selects a new node to travel to, but the global updating rule after all ants have completed their tour.

### 3.4 Max-min Ant System (MMAS)

This version by Stützle and Hoos (2000) differs from the original AS in three ways. Only the iteration best ant is allowed to apply pheronome, the strength of the pheromone trails have lower and upper bounds, and at start, all trails are initialized to their upper bound value to encourage early exploration. Equation 4 is modified

$$\tau_{ij} \leftarrow (1-p)\tau_{ij} + \sum_{k=1}^{m}\Delta\tau_{ij}^{best}, \forall(i,j)\in L \tag{10}$$

where $\Delta\tau_{ij}^{best}$ is the amount of pheromone the iteration best ant deposits on the arcs it has visited.
The pheromone trail upper ($\tau_{max}$ )and lower ($\tau_{min}$ ) bounds for an edge can be calculated, a detailed description can be found in the paper by Stützle and Hoos.

$$\left[x\right]_{\tau_{\min}}^{\tau_{max}} = \begin{cases} \tau_{max} & \text{if } x > \tau_{max} \\ \tau_{min} & \text{if } x < \tau_{min} \\ x & \text{otherwise} \end{cases} \tag{11}$$

At all times should the algorithm see to that the pheromone strength is between the given bounds on any edge.
Studies were conducted in the paper to ascertain if the algorithm should use the iteration best ant or the global best (elite) ant as basis for the pheromone updates, and the results were that the iteration best ant performed better. Also the effects of using $\tau_{min}$ or $\tau_{max}$ as a starting value for the initial pheromone amount on the trails were studied, resulting in $\tau_{max}$ being the better approach.
An additional mechanism called *pheromone trail smoothing* was introduced in the paper for increased performance. Basically when the MMAS has converged, or is very close to convergence, the mechanism increases the pheromone trails proportionally to their difference to the maximum pheromone trail limit. As a conclusion it is stated that MMAS outperformed all other AS variants to date.

## 4. The hybrid-ACO algorithm

The algorithm consists of two parts. We have the ACO part, where ants crawl over the searchspace trying to construct a feasible tour. When all ants have constructed their tour, the timestamps have also been calculated for the individual operations in the schedule defined by a tour, which allows us to calculate the makespan. The postprocessing part springs to life when there is a complete schedule to operate on. The (global) pheromone update of the ACO occurs only after the postprocessing has finished, this is due to the postprocessing affecting the makespan of the schedule formed by the tour of the ant. After the pheromone update ACO continues with the next iteration.

### 4.1 The postprocessing algorithm

After all ants have constructed their tour, a postprocessing algorithm is applied. This algorithm is effectively a local search procedure, based upon the approach of Nowicki and Smutnicki (1996).
The local search begins by identifying the critical path in the constructed schedule. The critical path can be decomposed into a number of blocks where a block is a maximal sequence of adjacent operations that require the same machine. Block length can vary from just one operation to all operations that are scheduled on one machine. Given a block, swapping operations take place. We start from the last block in the critical path which has a size larger than 1 and its last operation in the block. The block size must be larger than 1 since otherwise no swap can be made. The identified operation is swapped with its predecessor in the same block, and the necessary changes are made into the tour of the ant as well as the timestamps of the scheduled operations. If the swap improves the makespan, it is accepted, otherwise the swap is undone and the next pair in the block is up for swapping. If a block contains no more swaps we move to the preceeding block. Note that an accepted swap means that the critical path may change and a new critical path must be identified. If no swap of operations in the critical path improve the makespan, the local search ends.
This means that the tour of an ant may change in the postprocessing part of the algorithm. The tour of the ants after the very first completed postprocessing run may differ radically from the one presented by the first iteration of the ACO, but succeeding postprocessing runs after the first round of calculations are much easier on the ants and are not interrupting the pheromone trails too much.

Figure 1 shows a critical path and possible swaps for an example schedule.

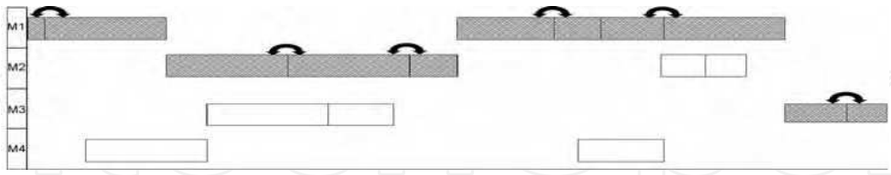



Figure 1. A sample 4-machine schedule with the critical path marked in grey and possible swap pairs with arrows. The path is made of 4 blocks with the largest block consisting of four scheduled operations.

## 5. What is visibility?

An additional problem when working with ant systems is that of visibility. There are similarities between priority rules used in heuristic approaches and the visibility of a single ant, both are trying to evaluate and make a choice of where to go next from a specific node. Usually visibility is referred to as the neighbourhood of the ant, i.e. the nodes that are close to the node the ant is currently staying on. It is a measure of what nodes the ant can see around it when standing on a specified node. In equation 1, the parameter $\eta_{ij}$ is our measure of visibility and in TSP-problems the meaning is clear and all values of $\eta_{ij}$ can be computed a priori, since the node distances are known. No matter which node the ant stands on, the distance to all other nodes can be fetched from a pre-calculated distance table. When it comes to schedules it is not entirely straightforward what visibility is and what effect it has on computations with regard to ACO. The distance in time units from a node in the tour to the next is not known until you have calculated the timestamps for the entire tour so far.

Another thing with ACO and the MT-10 problem is that the tabu list (already visited nodes) alone is not enough. Since the tasks in every job have to be done in correct order, that is, task A3 has to be done before A4 etc., a candidate list is needed. The candidate list has all the legal node choices an ant can make from the node it is currently standing on. This means that only the selection probabilities for the nodes in the candidate list need to be calculated, which speeds up the algorithm. In this case visibility for an ant is restricted to only the nodes in the candidate list. Figure 2 illustrates this phenomena.

In order to understand more about visibility and its effects, some various approaches to ACO-visibility in schedules are undertaken and studied. Table 1 shortly outlines some different types of visibility.

| Type of visibility | Explanation |
|---|---|
| Distance | Distance-based, the starting time of an operation (counted from $t_0$) |
| SPT | Shortest processing time first |
| LPT | Longest processing time first |
| TLM | Length of unscheduled tasks left on machine |
| TLJ | Length of unscheduled tasks left in job |
| TLJ+TLM(30-70) | Length of unscheduled tasks left in job and on machine, weight 30%-70% |
| TLJ+TLM(50-50) | Length of unscheduled tasks left in job and on machine, weight 50%-50% |
| TLJ+TLM(70-30) | Length of unscheduled tasks left in job and on machine, weight 70%-30% |

Table 1. Various types of visibility for ACO

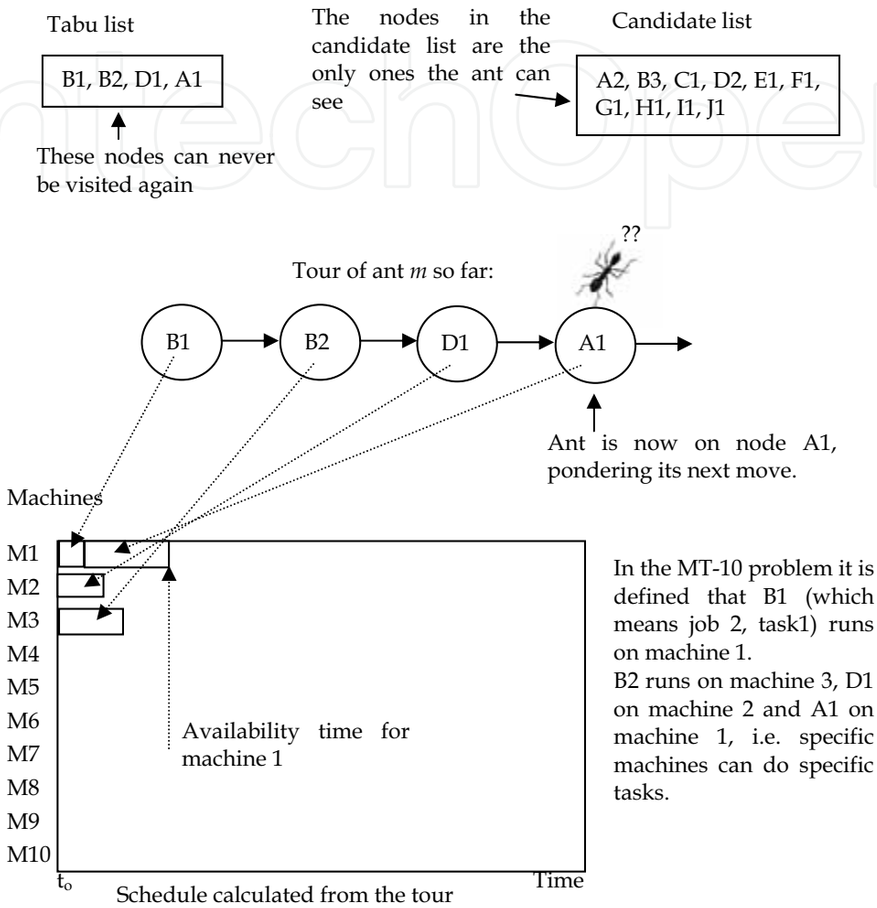



Figure 2. Visibility in scheduling. Since the ant has visited the nodes B1 and B2, the candidate list contains the next in the series, B3. Same for A2 and D2 since both A1 and D1 are in the tour. The rest of the candidates are jobs that have not started yet, the first in their series of tasks. Every time a node is added to the tour, it is placed into the schedule and the timestamps for starting and finishing that task on the specific machine are calculated. When choosing the next node to travel to, visibility can be calculated for all the nodes in the candidate list. The good choices get better visibility rating, according to selected visibility method, and thus a better chance of being selected

When the ant is selecting the next node to travel to, distance-based visibility is the earliest possible start time on the corresponding machine for the possible selections in its allowed list. The task that can start earlier than other candidates gets a higher probability of being chosen than a task that can start later and this can be achieved with a simple formula $Q/t_{start}$

that replaces the definition of $\eta_{ij}$ in equation 1. SPT ranks the candidates according to length of their processing time, shorter processing time means a higher probability of being chosen, whereas LPT is the opposite; longer processing times means higher probability. TLM calculates the total processing time for all unscheduled tasks on the current machine. The longer the total processing time is, the higher the probability of being chosen. TLJ is similar, it calculates the total processing times for all the unscheduled tasks left in the current job. The longer the total processing time, the higher the probability. TLJ + TLM is a combination of TLJ and TLM, where each visibility is weighted differently. To get an outline of the impact of the weighting factors, 30-70, 50-50 and 70-30 proportional weights are used (percentage values).

## 6. Computational experience and results

The experiment setup was to take each ACO method and do 5 runs for each of the different types of visibility. Each run was 2,500 rounds of calculations, then the algorithm was halted. Two sets of runs were made, one without postprocessing, the other with. Average values and mean deviations were calculated. All units in table 3 and 4 are time units. Common parameter settings for ACO can be seen in table 2.

| Parameter | value | meaning |
|---|---|---|
| $m$ | 40 | number of ants |
| $Q$ | 80 | pheromone deposited by an ant |
| $a$ | 1 | bias towards pheromone amplification |
| $\beta$ | 2 | bias towards closest nodes (visibility) |
| $p$ | 0.007 | evaporation rate |

Table 2. ACO parameter settings

These parameters were kept the same for all comparative runs, i.e. for all visibility types during the runs with and without postprocessing.

The column that dictates percentage deviation from optimum solution is calculated for the best found makespan of the runs.

AS and RAS perform about the same, with RAS having the slight edge, smaller standard deviation and better mean values. ACS outperforms both AS and RAS, and MMAS outperforms them all. This is in line with the findings in quoted papers.

The impact of the different visibilities vary for the different ACO methods, and it is quite an interesting read. As can be seen, best solution in table 3 was found by the TLJ visibility with ACS as the ACO method. The results for ACS with different visibilities are a bit jumpy, since ACS also holds the worst solution found. MMAS does good overall with all visibilities.

The best found solution after 2,500 rounds of calculations is really not a very good one, it is still 13.1% from optimum, however, the algorithm has not stagnated and it continues to explore the search space and comes up with new solutions. The meaning of these runs is not to solve to optimality, rather to study the visibility effects and get a feel for the performance of the different ACO methods.

Tweaking the parameter settings for each individual type of visibility may improve the results, but this way all the visibility types are on the same page for easy comparison. Same goes for the ACO methods.

| ACO | Type of visibility | worst | best | mean | σ | % from optimum |
|---|---|---|---|---|---|---|
| AS | Distance | 2174 | 1373 | 1954.0 | 294.4 | 32.3% |
| | SPT | 2273 | 1582 | 2134.8 | 276.4 | 41.2% |
| | LPT | 2314 | 1491 | 2121.8 | 316.8 | 37.6% |
| | TLM | 2406 | 1482 | 2117.4 | 324.7 | 37.2% |
| | TLJ | 2218 | 1502 | 2020.8 | 266.3 | 38.1% |
| | TLJ+TLM(30-70) | 2322 | 1457 | 2072.2 | 311.6 | 36.2% |
| | TLJ+TLM(50-50) | 2357 | 1464 | 2114.4 | 333.0 | 36.5% |
| | TLJ+TLM(70-30) | 2127 | 1459 | 1975.8 | 259.1 | 36.3% |
| RAS | Distance | 2102 | 1488 | 1946. 6 | 230.4 | 37.5% |
| | SPT | 2121 | 1508 | 1981.6 | 237.6 | 38.3% |
| | LPT | 2384 | 1519 | 2151.0 | 318.4 | 38.7% |
| | TLM | 2119 | 1486 | 1852.6 | 205.6 | 37.4% |
| | TLJ | 2230 | 1466 | 2032.8 | 284.8 | 36.6% |
| | TLJ+TLM(30-70) | 2145 | 1364 | 1929.6 | 290.4 | 31.8% |
| | TLJ+TLM(50-50) | 2265 | 1520 | 2090.8 | 286.8 | 38.8% |
| | TLJ+TLM(70-30) | 2008 | 1494 | 1871.0 | 190.3 | 37.8% |
| ACS | Distance | 1251 | 1137 | 1184.6 | 42.0 | 18.2% |
| | SPT | 2072 | 1867 | 2001.0 | 71.2 | 50.2% |
| | LPT | 2213 | 1638 | 2072.6 | 218.8 | 43.2% |
| | TLM | 1473 | 1381 | 1431.4 | 32.1 | 32.6% |
| | TLJ | 1108 | 1070 | 1093.8 | 13.1 | 13.1% |
| | TLJ+TLM(30-70) | 1459 | 1234 | 1373.0 | 81.1 | 24.6% |
| | TLJ+TLM(50-50) | 1404 | 1279 | 1335.0 | 50.6 | 27.3% |
| | TLJ+TLM(70-30) | 1273 | 1168 | 1231.6 | 37.9 | 20.4% |
| MMAS | Distance | 1272 | 1183 | 1243.6 | 31.8 | 21.4% |
| | SPT | 1363 | 1241 | 1332.4 | 46.2 | 25.1% |
| | LPT | 1303 | 1237 | 1276.8 | 25.83 | 24.8% |
| | TLM | 1301 | 1209 | 1273.0 | 33.0 | 23.1% |
| | TLJ | 1286 | 1267 | 1279.8 | 7.0 | 26.6% |
| | TLJ+TLM(30-70) | 1286 | 1211 | 1260.4 | 30.6 | 23.2% |
| | TLJ+TLM(50-50) | 1286 | 1235 | 1260.2 | 19.2 | 24.7% |
| | TLJ+TLM(70-30) | 1295 | 1245 | 1269.0 | 18.1 | 25.3% |

Table 3. Results from computational runs without postprocessing

| ACO | Type of visibility | worst | best | mean | σ | % from optimum |
|---|---|---|---|---|---|---|
| AS | Distance | 1341 | 1083 | 1231.6 | 84.4 | 14.1% |
| | SPT | 1609 | 1055 | 1445.6 | 198.9 | 11.8% |
| | LPT | 1608 | 1079 | 1464.4 | 195.2 | 13.8% |
| | TLM | 1667 | 1048 | 1475.8 | 219.9 | 11.3% |
| | TLJ | 1620 | 1061 | 1458.8 | 207.9 | 12.3% |
| | TLJ+TLM(30-70) | 1599 | 1057 | 1437.8 | 195.1 | 12.0% |
| | TLJ+TLM(50-50) | 1578 | 1059 | 1422.6 | 185.4 | 12.2% |
| | TLJ+TLM(70-30) | 1580 | 1071 | 1420.6 | 183.3 | 13.2% |
| RAS | Distance | 1292 | 1087 | 1207.6 | 71.8 | 14.4% |
| | SPT | 1463 | 1069 | 1346.8 | 142.8 | 13.0% |
| | LPT | 1538 | 1101 | 1351.2 | 148.3 | 15.5% |
| | TLM | 1465 | 1088 | 1330.21 | 129.4 | 14.5% |
| | TLJ | 1358 | 1068 | 1245.6 | 97.4 | 12.9% |
| | TLJ+TLM(30-70) | 1457 | 1093 | 1339.0 | 127.3 | 14.9% |
| | TLJ+TLM(50-50) | 1456 | 1067 | 1281.4 | 125.9 | 12.8% |
| | TLJ+TLM(70-30) | 1502 | 1101 | 1378.6 | 145.9 | 15.5% |
| ACS | Distance | 1053 | 1032 | 1045.0 | 7.4 | 9.9% |
| | SPT | 1340 | 1123 | 1254.0 | 72.5 | 17.2% |
| | LPT | 1178 | 1103 | 1157.2 | 27.5 | 15.7% |
| | TLM | 1137 | 1038 | 1073.0 | 35.5 | 10.4% |
| | TLJ | 988 | 981 | 982.8 | 2.7 | 5.2% |
| | TLJ+TLM(30-70) | 1105 | 1008 | 1052.8 | 31.5 | 7.7% |
| | TLJ+TLM(50-50) | 1060 | 999 | 1033.4 | 20.1 | 6.9% |
| | TLJ+TLM(70-30) | 995 | 977 | 983.0 | 6.6 | 4.8% |
| MMAS | Distance | 1013 | 1001 | 1003.8 | 4.6 | 7.1% |
| | SPT | 1006 | 977 | 989.6 | 9.8 | 4.8% |
| | LPT | 1019 | 991 | 1004.2 | 10.2 | 6.2% |
| | TLM | 1014 | 988 | 1002.2 | 10.6 | 5.9% |
| | TLJ | 1013 | 993 | 1001.2 | 6.7 | 6.3% |
| | TLJ+TLM(30-70) | 994 | 982 | 987.4 | 5.1 | 5.3% |
| | TLJ+TLM(50-50) | 1006 | 989 | 998.6 | 6.8 | 6.0% |
| | TLJ+TLM(70-30) | 1003 | 979 | 990.4 | 8.8 | 5.0% |

Table 4. Results from computational runs with postprocessing

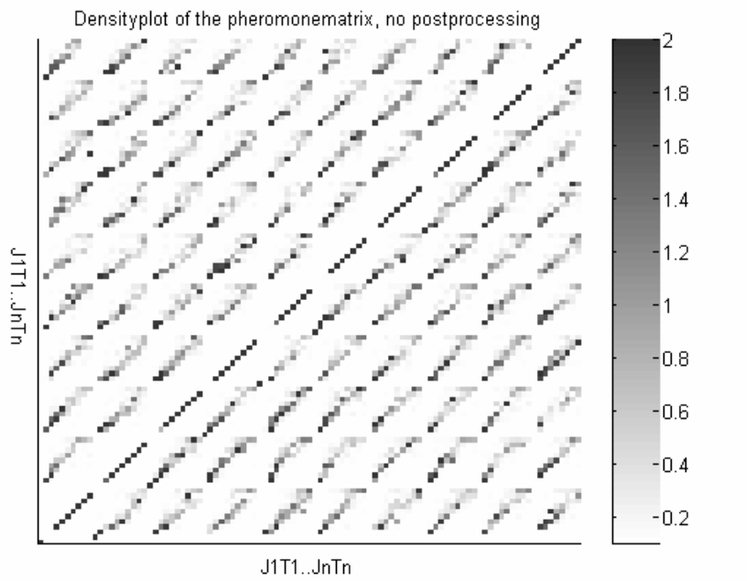



Figure 3. A plot of the pheromonematrix when no postprocessing present. Very clear pheromonetrails are visible

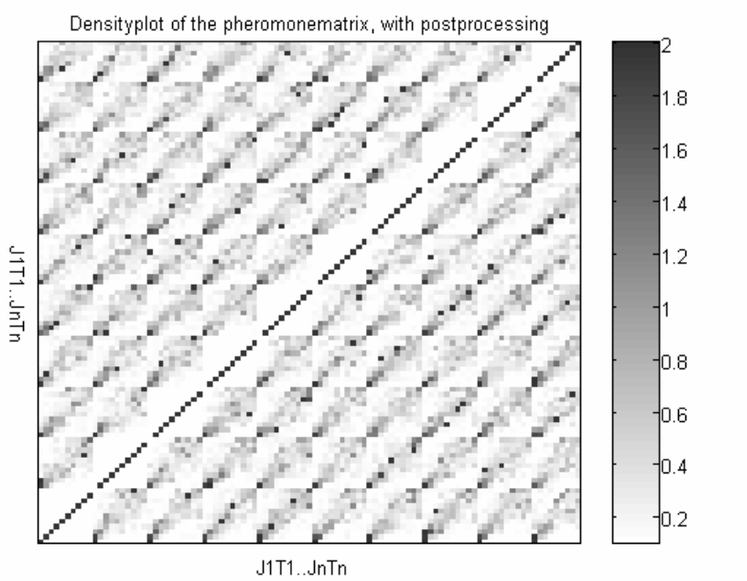



Figure 4. A plot of the pheromonematrix when using postprocessing. Clear pheromonetrails visible but distributed over more edges than in figure 4

As for the postprocessing version of the ACO methods, RAS beats AS in the sense that RAS has less deviation, which means it consistently gives good solutions, though AS did manage to find some better solutions. ACS is still a bit jumpy, it finds very good solutions for some visibility runs, but also performs poorly with for instance SPT and LPT. The various weighted combinations of TLJ + TLM seem to do better, overall, than other visibilities.
TLJ+TLM(70-30) visibility in MMAS seems to work best, after 2,500 rounds of calculations the best found solution is 5.0% from optimum, though TLJ visibility and ACS are very close with a 5.2% solution. MMAS has less deviation, and thus is more likely to continue to produce good solutions every time it runs.
It is clear that the postprocessing closes the performance gap between the different ACO methods, but the same internal ranking still holds true with postprocessing as without. The postprocessing also improves the performance dramatically for all versions of ACO algorithms tested.
The algorithms were stopped after 2,500 rounds of calculations, so the question arises, how good a solution can be found if allowed to run without interruptions for a longer time? An additional run with the best visibility and ACO method from table 3 landed after 30,000 rounds of calculations at a best found makespan of 1012 time units. which is 8.1% from optimum. An additional run with the best visibility and ACO method from table 4 landed after 30,000 rounds of calculations at a best found makespan of 948 time units, which is 1.9 % from optimum.

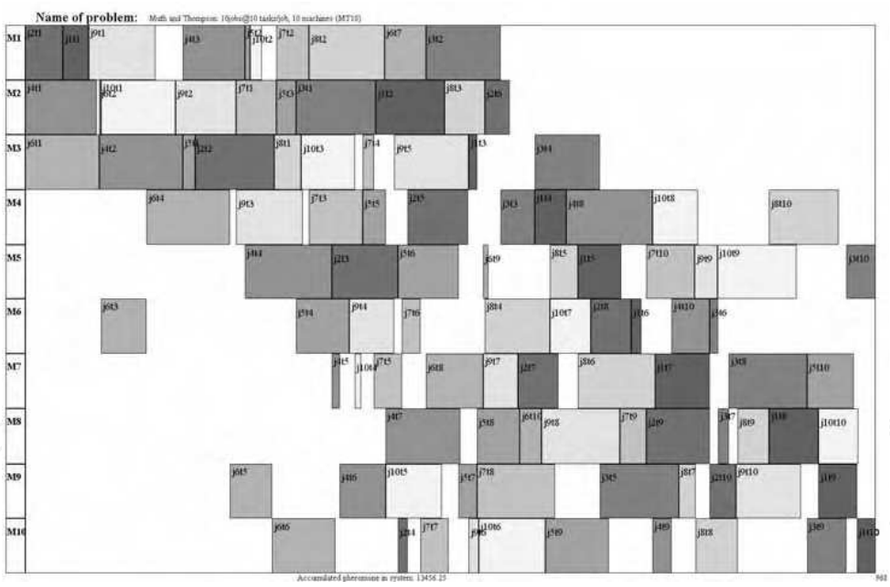



Figure 5. A finished schedule for the MT10 problem, made with the hybrid ACO (MMAS), with a makespan of 968 time units (3.9% from optimum)

Another question that can be asked is does the postprocessing disturb the forming of pheromone trails in the system in any way? Figure 3 is the pheromonematrix of the MMAS with no postprocessing, taken after 2,500 rounds, and figure 4 is a similar one with

postprocessing. The dark dots depict a high concentration of pheromone whereas the presence of a lighter dot means no or very little pheromone is present. As one can imagine, the presence of a postprocessing routine that modifies ant tours messes with the ant pheromone trails, and you can clearly see if you compare figure 3 and figure 4 with each other that figure 4 shows more pheromone distribution in the system. There are still dark dots in figure 4 signifying established pheromone trails so we are not dealing with random search. In light of these figures you could eventually tweak the evaporation setting higher when using postprocessing, or bias the parameters more towards an emphasis on visibility. You could argue that the larger distribution of pheromone over the trails as seen in figure 4 encourages ant exploration more and actually helps in finding better solutions.
A finished schedule produced by a hybrid ACO can be seen in figure 5.

## 7. Conclusion

When paired with the local search the ACO produces noteworthy results very fast (typically 5% from optimum within 200 rounds of calculations). The Max-Min Ant System outperformed all other ACO versions, and it did so for all types of visibility tested, showing that it is indeed a leading candidate for choosing your ant system.
There are various version of ACO available and this chapter served its purpose to both do an attempt at ranking them, showing the impact of various visibility methods as well as proving that pure ACO methods produce good results, but even better when combined with the postprocessing algorithm shown. Naturally, not every combination of ACO and a local search is guaranteed to work better than a pure ACO, but a hybrid version can improve the performance dramatically.
If you are looking for a good, quick solution rather than an all-out effort to find the best solution, ACO perfomance is a noteworthy competitor to existing job-shop scheduling approaches. ACO is an easy algorithm to implement, with roughly the same amount of code and difficulty as that of a genetic algorithm.
ACO is a good example of how harnessing, mimicking and utilizing processes occurring in nature for tough scientific problems can be a successful enterprise.

## 8. References

Adams J., Balas E., Zawack D. (1988). The shifting bottleneck procedure for job shop scheduling, *Management Science*, 34, pp. 391-401.

Bell J.E., McMullen P.R. (2004) Ant colony optimization techniques for the vehicle routing problem, *Advanced Engineering Informatics*, 18, pp. 41-48.

Blum C., Dorigo M. (2004). The hyper-cube framework for ant colony optimization, *IEEE Trans Syst Man Cybernet Part B*, 34(2), pp. 1161-1172.

Bullnheimer B., Hartl R., Strauss C. (1999). A new rank-based version of the Ant System: A computational study, *Central European J Operations Res Econom*, 7(1), pp. 25-38.

Colorni A., Dorigo M., Maniezzo V. and Trubian M. (1993). Ant System for Job-shop scheduling, *Belgian Journal of Operations Research, Statistics and Computer Science*, 34: pp. 39--54.

Denebourg, J.-L., Aron, S., Goss, S., Pasteels, J.-M. (1990) The self-organizing exploratory pattern of the Argentine ant, *Journal of insect behaviour*, vol. 3, p. 150.

Dorigo M. (1992). Optimization, Learning and Natural Algorithms. *PhD thesis*, Dipartemento di Elettronica, Politecnico di Milano.

Dorigo M., Gambardella L.M. (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem, *IEEE Trans Evolutionary Comput*,1(1), pp. 53-66.

Dorigo M., Maniezzo V., Colorni A. (1996). Ant system: Optimization by a colony of cooperating agents, *IEEE Trans Syst Man Cybernet Part B*, 26(1), pp. 29-41.

Dorigo, M. and Stützle, T. (2004). Ant colony optimization, MIT press, Cambridge, MA.

Gajpal Y., Rajendran C. (2006). An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops, *International Journal of Production Economics*, 101, pp. 259-272.

Grassé, P.-P. (1946). Les Insects Dans Leur Univers, Paris, France, Ed. Du Palais de la découverte.

Gutjahr W.J., Rauner M.S. (2005). An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria, *Computers & Operations Research* (in print).

Jayaraman V.K., Kulkarni B.D., Karale S., Shelokar P. (2000). Ant colony framework for optimal design and scheduling of batch plants, *Computers and Chemical Engineering*, 24, pp. 190-192.

Kuo-Ching Ying, Ching-Jong Liao (2004). An ant colony system for permutation flow-shop sequencing, *Computers & Operations Research*, 31, pp. 791-801.

Panwalker S.S., Iskander W. (1977). A survey of Scheduling Rules, *Oper.Res.*, 25, 1, pp. 45-61.

Rosenkrantz, D. J., Stearns, R. E., Lewis, P. M. (1977). An analysis of several heuristics for the traveling salesman problem, *SIAM Journal on Computing*, vol. 6, pp. 563-581.

Nakano R., Yamada T. (1991). Conventional Genetic Algorithm for Job Shop Problems, *Proc. of the 4th Int. Conference on Genetic Algorithms*, San Diego, Californa, pp. 474-479.

Nowicki E., Smutnicki C. (1996). A fast taboo search algorithm for the job-shop problem. *Management Science*, 42 (6), pp. 797-813.

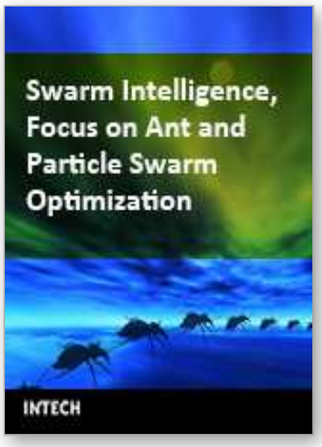
Stützle T., Hoos H.H. (1996). Improving the Ant System: a detailed report on the *MAX-MIN* Ant system, *Technical Report AIDA-96-12*, FG Intellektik, TH Darmstadt.

Stützle T., Hoos H.H. (2000). *MAX-MIN* Ant system, *Future Generat Comput Syst*, 16(8), pp. 889-914.

Stützle, T. and Dorigo, M. (2002). A short convergence proof for a class of ACO algorithms, *IEEE Transactions on evolutionary computation*, vol.6, no. 4, pp. 358-365.

Taillard E. (1989). Parallel Tabu Search Technique for the Jobshop Scheduling Problem, *Internal Report ORWP 89/11*, Departemente de Mathematiques, Ecole Polytechnique Federale de Lausanne, Lausanne.

Van Laarhoven P. J. M., Aarts E.H.L., Lenstra J.K. (1992). Job shop scheduling by simulated annealing, *Operations Research*, 40, pp. 113-125.




**Swarm Intelligence, Focus on Ant and Particle Swarm Optimization**
Edited by FelixT.S.Chan and Manoj KumarTiwari

ISBN 978-3-902613-09-7
Hard cover, 532 pages
**Publisher** I-Tech Education and Publishing
**Published online** 01, December, 2007
**Published in print edition** December, 2007

In the era globalisation the emerging technologies are governing engineering industries to a multifaceted state. The escalating complexity has demanded researchers to find the possible ways of easing the solution of the problems. This has motivated the researchers to grasp ideas from the nature and implant it in the engineering sciences. This way of thinking led to emergence of many biologically inspired algorithms that have proven to be efficient in handling the computationally complex problems with competence such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), etc. Motivated by the capability of the biologically inspired algorithms the present book on "Swarm Intelligence: Focus on Ant and Particle Swarm Optimization" aims to present recent developments and applications concerning optimization with swarm intelligence techniques. The papers selected for this book comprise a cross-section of topics that reflect a variety of perspectives and disciplinary backgrounds. In addition to the introduction of new concepts of swarm intelligence, this book also presented some selected representative case studies covering power plant maintenance scheduling; geotechnical engineering; design and machining tolerances; layout problems; manufacturing process plan; job-shop scheduling; structural design; environmental dispatching problems; wireless communication; water distribution systems; multi-plant supply chain; fault diagnosis of airplane engines; and process scheduling. I believe these 27 chapters presented in this book adequately reflect these topics.

**How to reference**
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Heinonen, J. and Pettersson, F. (2007). Job-shop Scheduling and Visibility Studies with a Hybrid ACO Algorithm, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, FelixT.S.Chan and Manoj KumarTiwari (Ed.), ISBN: 978-3-902613-09-7, InTech, Available from: http://www.intechopen.com/books/swarm_intelligence_focus_on_ant_and_particle_swarm_optimization/job-shop_scheduling_and_visibility_studies_with_a_hybrid_aco_algorithm

INTECH
open science | open minds

**InTech Europe**
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

**InTech China**
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen