



Ant System variations with job shop scheduling
by Carla Leslie Peterson

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Science in
Computer Science

Montana State University

© Copyright by Carla Leslie Peterson (2002)

Abstract:

The Ant System is a class of distributed algorithms used for solving NP-hard combinatorial optimization problems. The Ant System has three different instantiations: ANT-quantity, ANT-density, and ANT-cycle. Job shop scheduling is an NP-hard combinatorial optimization problem. ANT-cycle for job shop scheduling has already been used with success in finding good solutions. This thesis outlines the implementation and performance of the three instantiations for the Ant System when applied to job shop scheduling. The three instantiations were successful in finding good solutions to smaller job shop problem instances. The ANT-cycle algorithm was superior to the other two algorithms in all problem instances tested.

ANT SYSTEM VARIATIONS WITH JOB SHOP SCHEDULING

by

Carla Leslie Peterson

A thesis submitted in partial fulfillment
of the requirement for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

July 2002

N378
P4406

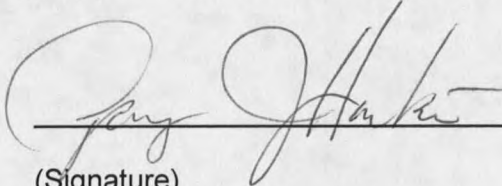
APPROVAL

of a thesis submitted by

Carla Leslie Peterson

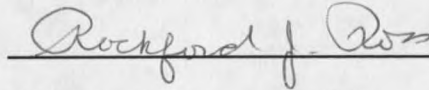
This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Gary Harkin


(Signature)7/18/02
(Date)

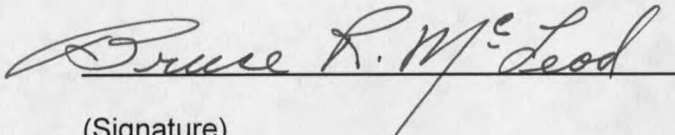
Approved for the Department of Computer Science

Rockford Ross


(Signature)7/18/02
(Date)

Approved for the College of Graduate Studies

Bruce McLeod


(Signature)7-24-02
(Date)

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature



Date

7/23/07

TABLE OF CONTENTS

LIST OF FIGURES	V
LIST OF TABLES.....	VI
ABSTRACT.....	VII
1. INTRODUCTION.....	1
ANT BEHAVIOR	1
ANT SYSTEM ALGORITHM OVERVIEW	3
RELATED WORK	4
EXPERIMENTAL GOALS.....	5
2. ANT SYSTEM ALGORITHM	7
GENERAL AS ALGORITHM	7
GENERAL ALGORITHM CHARACTERISTICS	7
ANT AGENTS	8
PROPERTIES OF ARTIFICIAL ANTS	8
SIMILARITIES WITH REAL ANTS.....	9
DIFFERENCES WITH REAL ANTS.....	9
TABU LIST	10
PROBABILISTIC TRANSITION RULE.....	10
TRAIL INTENSITY	12
ANT-QUANTITY	13
ANT-DENSITY	13
ANT-CYCLE.....	15
3. JOB SHOP SCHEDULING.....	17
JOB SHOP SCHEDULING ALGORITHM.....	17
4. ANT SYSTEM FOR JOB SHOP SCHEDULING (AS-JSP)	21
JSP REPRESENTATION IN THE AS.....	21
5. EXPERIMENTS AND RESULTS.....	23
PROBLEM INSTANCES.....	23
LA01 PROBLEM INSTANCE	23
ABZ5 PROBLEM INSTANCE	28
FT20 PROBLEM INSTANCE	32
6. CONCLUSION	36
CONCLUSION	36
FUTURE RESEARCH	39
REFERENCES CITED.....	40

LIST OF FIGURES

Figure	Page
1. Ants have a direct path between their nest and the food source (Dorigo, 2001).	2
2. An obstacle is placed on the path (Dorigo, 2001).	2
3. Cooperative ants (Dorigo, 2001).	3
4. The longer path from the nest to the food source is left unused (Dorigo, 2001).	3
5. ANT-density and ANT-quantity algorithm.	14
6. ANT-cycle algorithm.	16
7. Representation of a $2/3/G/C_{max}$ job shop problem.	18
8. The AS representation of the JSP in Figure 7.	21
9. <i>la01</i> problem instance with 25 ants	25
10. <i>la01</i> problem instance with 50 ants	27
11. <i>abz5</i> problem instance with 50 ants	29
12. <i>abz5</i> problem instance with 100 ants	31
13. <i>ft20</i> problem instance with 50 ants	33
14. <i>ft20</i> problem instance with 100 ants	34

LIST OF TABLES

Table	Page
1. la01 problem instance results with 25 ants	24
2. la01 problem instance with 50 ants	26
3. abz5 problem instance with 50 ants	28
4. abz5 problem instance for 100 ants	30
5. ft20 problem instance with 50 ants	32
6. ft20 problem instance with 100 ants	34

ABSTRACT

The Ant System is a class of distributed algorithms used for solving NP-hard combinatorial optimization problems. The Ant System has three different instantiations: ANT-quantity, ANT-density, and ANT-cycle. Job shop scheduling is an NP-hard combinatorial optimization problem. ANT-cycle for job shop scheduling has already been used with success in finding good solutions. This thesis outlines the implementation and performance of the three instantiations for the Ant System when applied to job shop scheduling. The three instantiations were successful in finding good solutions to smaller job shop problem instances. The ANT-cycle algorithm was superior to the other two algorithms in all problem instances tested.

CHAPTER 1

INTRODUCTION

Ant Behavior

Computer Science is the study of algorithms to solve problems. One way to classify problems is to place them into classes depending on their time complexities. These classes are: P, NP, NP-complete, and NP-hard. A problem is NP-hard if all problems in NP are polynomial time reducible to the problem (Sipser, 1997). An Ant System (AS) is a class of distributed algorithms used for solving NP-hard combinatorial optimization problems (Coloni, Dorigo, & Maniezzo, 1992). A combinatorial optimization problem has a set of feasible solutions and a cost function over those solutions, and the goal is to find a solution with the optimal cost over all feasible solutions (Ambite, 2001).

AS is based on the foraging behavior observed in a real ant colony (Coloni, Dorigo, & Maniezzo, 1992). Observing a group of ants will reveal that the group is extremely organized in accomplishing their task, and scientists have discovered that the cooperation in an insect colony is self-organized. The coordination of the colony comes from interactions among the ants, which can collectively solve difficult problems.

Jean-Louis Deneubourg and his colleagues showed that ants create pheromone paths from their nest to a food source. Pheromone is a chemical substance deposited by ants as they walk and can be detected by other ants (Bonabeau, & Theraulaz, 2000). If

there is a direct path from the ants' nest to a food source, the ants will simply follow one pheromone path (Figure1).

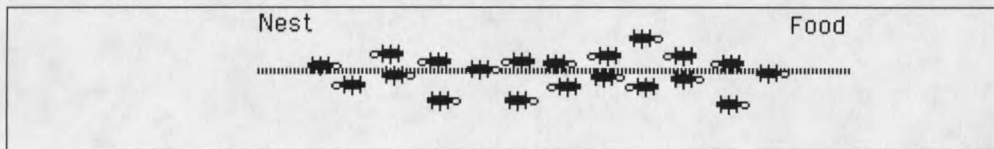


Figure 1: Ants have a direct path between their nest and the food source (Dorigo, 2001).

If an obstacle is placed between the ants and the food source (Figure 2), the ants directly in front of the obstacle must decide which direction to take. It is expected that the ants will choose the left or right path equally. The ants that have chosen the shorter path around the obstacle will be able to deposit pheromone in less time than the ants that have chosen the longer path. In a given amount of time, the ants following the shorter path cover the ground more often making the pheromone denser in concentration on the shorter path. Ants will most likely return to the nest on the same path that they followed to the food source.

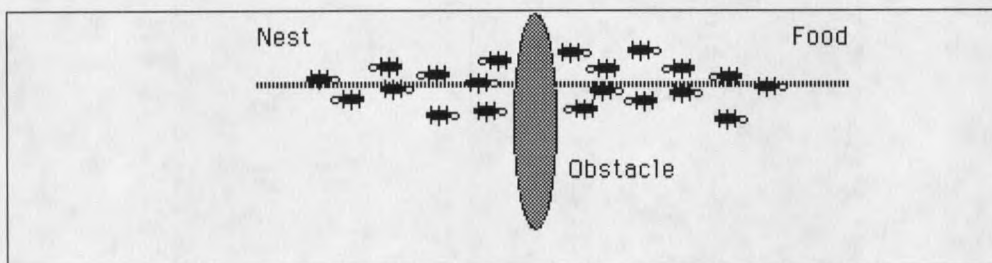


Figure 2: An obstacle is placed on the path (Dorigo, 2001).

Each ant probabilistically prefers to follow a path that has a greater pheromone concentration rather than a path with a lower pheromone concentration. Over time, the pheromone concentration on the shorter path will become greater than on the longer path,

leading future ants to travel on the shorter path. Deneubourg discovered that the path with the highest concentration of pheromone was the shortest path between the nest and the food source. Figure 3 displays the effect that as more ants make their way to the food source (and back) they are more likely to take the shorter route. Over time, all ants will choose the shorter path and the longer path will be left unused as shown in Figure 4 (Colorni, Dorigo, & Maniezzo, 1991).

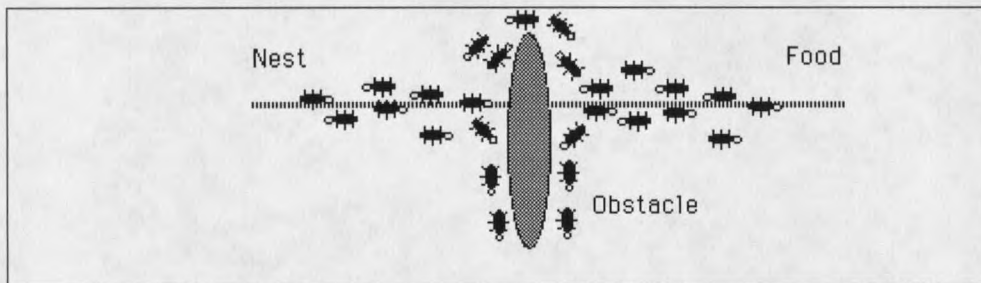


Figure 3: Cooperative ants (Dorigo, 2001).

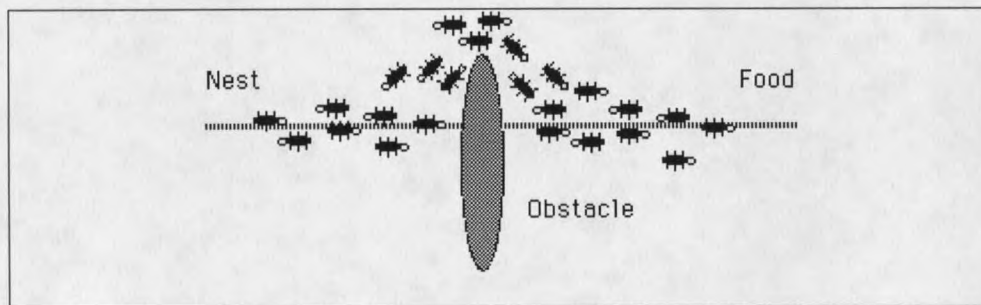


Figure 4: The longer path from the nest to the food source is left unused (Dorigo, 2001).

Ant System Algorithm Overview

Marco Dorigo extended Deneubourg's conclusions and developed the Ant System to solve NP-hard optimization problems in 1992 (Dorigo, 2001). AS uses a set of antlike

agents that cooperate to obtain a good solution to the optimization problem, by using positive feedback to reinforce good solutions. The reinforcement is virtual pheromone, which allows good solutions to be kept in memory so a better solution can be found. Negative feedback through pheromone evaporation is also used to avoid early convergence to a poor solution. AS uses cooperative behavior in which identical ants simultaneously explore different solutions to the problem. The ants that have achieved good solutions influence other ants because their pheromone trail is used as a guide (Bonabeau, Dorigo, & Theraulaz, 1999). Colorni, Dorigo, and Maniezzo proposed three different instantiations of AS: ANT-quantity, ANT-density, and ANT-cycle (Colorni, Dorigo, & Maniezzo, 1991). These instantiations will be described later in detail.

Related Work

Colorni, Dorigo, and Maniezzo were the first to apply AS to job shop scheduling. They applied the AS to problems up to 15 machines and 15 jobs and always found solutions within 10% of the optimal value. They tested for the best values of the AS parameters: α , β , and ρ . α and β are used in the probabilistic transition rule (see Chapter 2) to control the trail intensity and visibility. ρ is the evaporation constant used in calculating the trail intensity. Colorni, Dorigo, and Maniezzo suggested that the best setting for the parameters is $\alpha=1$, $\beta=1$, and $\rho=0.7$ (Colorni, Dorigo, & Maniezzo, 1994). Zwaan and Marques also applied the AS to JSP and tuned the parameters of the AS. They developed an evaporation constant, ϕ , which guides the search towards favored regions of the graph and prevents the searching

in small areas with local optimum values. They also introduced a variation-parameter, ν , which guides the search into more sub-optimal regions of the graph without giving up the algorithm's ability to recover from dead-end solutions. They were able to find an optimum within 8% of the optimal value for the 10/10/G/ $C_{\max}$ problem (van der Zwaan & Marques, 1999).

Dorigo, Colorni, and Maniezzo defined the three possible instantiations of the AS, ANT-density, ANT-quantity, and ANT-cycle, and applied them to the traveling salesman problem. They found that the ANT-cycle instantiation performed better than the others, especially on harder traveling salesman problems (Colorni, Dorigo, & Maniezzo, 1991).

Experimental Goals

This thesis explains how the three instantiations of the Ant System are applied to the job shop scheduling problem. The thesis explores each of the instantiations as they are applied to JSP. The work in this thesis is compared with the previous work completed on the AS for JSP and on the three instantiations of the AS. The solutions found in this thesis are compared with well-known benchmark job shop scheduling problems. This thesis answers the questions:

- 1) Which instantiation of the AS best suits JSP?
- 2) How do the solutions found for the benchmark problems compare with the optimal values for the benchmark problems?

- 3) How do the solutions found compare with other popular algorithms used to solve job shop scheduling?
- 4) What avenues should be further researched?

CHAPTER 2

ANT SYSTEM ALGORITHM

General AS Algorithm

the general Ant System algorithm, there is a population of d artificial ants and an optimization problem defined by a graph. The total number of ants is constant over time. If there are too many ants, sub-optimal paths will occur and lead to an early convergence to poor solutions, and too few ants do not allow cooperation to occur because paths will not be traveled often and pheromone will evaporate. Bonabeau, Dorigo, and Theraulaz proposed in their book, "Swarm Intelligence: from natural to artificial intelligence," that in order to get quality solutions, the number of ants should be equal to the number of nodes in the graph. In the AS, the ants move from node to node building the best solution (Bonabeau, Dorigo, & Theraulaz, 1999).

General Algorithm Characteristics

The general Ant System algorithm has numerous characteristics. One characteristic is that the algorithm must represent a combinatorial problem where ants build and modify solutions by using a probabilistic transition rule and a local heuristic. The local heuristic directs the ant's search. The algorithm must satisfy the problem constraints, which force the ants to find feasible solutions. A pheromone-updating rule that informs the ants how to

modify the amount of pheromone on the trail must exist in the algorithm. In addition to the pheromone-updating rule, the algorithm must have a probabilistic transition rule. The probabilistic transition rule is a function of the heuristic desirability and the pheromone trail (Bonabeau, Dorigo, & Theraulaz, 1999). The time complexity of the general algorithm is $O(n^2d)$ where n is the number of nodes in the graph and d is the number of ants (Colomi, Dorigo, & Maniezzo, 1992).

Ant Agents

The ants in the AS are independent entities that cooperate in solving a combinatorial optimization problem. They cooperate by modifying a shared table of information and the artificial ants also have the ability to react to their environment by making decisions. The artificial ants communicate with each other indirectly through pheromone deposition. Artificial ants have the behavioral traits of real ants as well as extra capabilities to make them more effective and efficient (Dorigo, Caro, & Gambardella, 1999).

Properties of Artificial Ants

Artificial ants are defined by various properties. The goal of artificial ants is to search for minimum cost feasible solutions. Ants have memory that stores path information, which is used to build solutions, evaluate solutions, and retrace the path taken. While at a particular node, ants are allowed to move to any other node in its neighborhood by applying

the probabilistic decision rule. Ants start at an initial node and move around the graph building feasible solutions. Ants can update the pheromone on the trail while finding a feasible solution or by retracing their completed solution at the end. Ants only utilize their private information and information local to the node to make decisions on which path to take. Ants are not adaptive; they modify the problem representation according to their environment making other ants perceive it differently (Dorigo & Caro, 1999).

Similarities with Real Ants

Artificial ants are similar to real ants in many aspects as they both cooperate as a colony to find good solutions to a given task. The common goal is to find the shortest path between a source and a destination. Real ants deposit actual pheromone while artificial ants simulate pheromone deposition through numeric calculations. Pheromone trails are the only communication between ants for both real and artificial ants. Pheromone actually evaporates off of the trails from real ants. In the ant system, the evaporation of pheromone is done by a mathematical calculation. Both real and artificial ants make decisions using a probabilistic decision rule.

Differences with Real Ants

There are also differences between real and artificial ants and one of them is that artificial ants live in a discrete world and have an internal state. Another difference is that

the amount of pheromone an artificial ant deposits depends on the quality of the solution found thus far. Pheromone deposition by an artificial ant depends on the problem and does not reflect a real ant's behavior. Another difference is that extra capabilities can be added to the artificial ants such as local optimization and back tracking (Dorigo, Caro, & Gambardella, 1999).

Tabu List

Each ant keeps a Tabu list, which contains all the nodes the ant has visited. It is used to notify the ant which nodes it can move to from the current node (Bonabeau, Dorigo, & Theraulaz, 1999). It is the ant's memory and is usually represented as a vector where the s^{th} element in the list is the s^{th} node visited by the ant. The ant may also carry other lists in addition to the Tabu list (Coloni, Dorigo, & Maniezzo, 1992).

Probabilistic Transition Rule

Every ant starts at an initial node and decides which node to move to next according to the probabilistic transition rule:

$$P_{ij}(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{j \in T} [\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta} \quad (1)$$

t time

p_{ij} probability the ant chooses to move from node i to node j

τ_{ij} trail intensity – the quantity of pheromone on an edge between node i and node j .

η_{ij} visibility

The trail intensity is global information and reflects the experience acquired by the ants. If there is a significant amount of traffic from i to j , then it is desirable (Bonabeau, Dorigo, & Theraulaz, 1999).

The visibility is the greedy part of the equation and states that close nodes should be chosen with higher probability (Coloni, Dorigo, & Maniezzo, 1992). The visibility is based on local information and directs the ants' search. However, using the visibility without trail intensity can lead to very poor solutions. This factor is not modified during the entire algorithm. $\eta_{ij} = \frac{1}{d_{ij}}$, where d_{ij} is the heuristic distance between node i and node j .

α and β control the importance of trail intensity and visibility. If $\alpha = 0$, the closest nodes will most likely be selected. If $\beta = 0$, only pheromone concentration matters and selected paths may not be optimal.

The value of $p_{ij}^k(t)$ can be different for two ants on the same node i since it is based on the partial solution built by that ant so far.

The set T consists of potential nodes an ant can visit from node i excluding nodes the ant has already visited.

Trail Intensity

The trail intensity is a function of both the evaporation rate of pheromone and the quality of solutions built by the ants so far (Bonabeau, Dorigo, & Theraulaz, 1999). The trail intensity from node i to node j at time $t+1$ is:

$$\tau_{ij}(t+1) = \rho \tau_{ij}(t) + \Delta \tau_{ij}(t, t+1) \quad (2)$$

ρ is the evaporation coefficient and $0 \leq \rho < 1$ to avoid unlimited accumulation.

$\Delta \tau_{ij}(t, t+1) = \sum_{k=1}^d \Delta \tau_{ij}^k(t, t+1)$, where $\Delta \tau_{ij}^k(t, t+1)$ is the quantity per unit length of trail

substance (pheromone in real ants) laid on the path from node i to node j by the k^{th} ant between time t and time $t+1$. $\tau_{ij}(0)$ is the initial amount of pheromone and should be set to small arbitrary values. The values can all be the same (Coloni, Dorigo, & Maniezzo, 1992).

There are three different ways to compute $\Delta \tau_{ij}^k(t, t+1)$ and when to update the trail intensity, $\tau_{ij}(t)$. Coloni, Dorigo, and Maniezzo proposed three different instantiations of the AS algorithm in 1991 based on the various methods used to compute $\Delta \tau_{ij}^k(t, t+1)$ and $\tau_{ij}(t)$. The three instantiations of the AS are, ANT-quantity, ANT-density, and ANT-cycle.

ANT-quantity

The ANT-quantity algorithm has the same general characteristics described above for the AS. The requirement for ANT-quantity is that a constant quantity of pheromone, Q_1 , is left on the path every time an ant goes from node i to node j . The value of $\Delta\tau_{ij}^k(t, t+1)$ is calculated as follows:

$$\Delta\tau_{ij}^k(t, t+1) = \begin{cases} \frac{Q_1}{d_{ij}} & \text{if the } k^{\text{th}} \text{ ant goes from } i \text{ to } j \text{ between time } t \text{ \& } t+1 \\ 0 & \text{Otherwise} \end{cases} \quad (3)$$

Equation (3) reinforces the visibility parameter making shorter paths more desirable to ants.

ANT-density

ANT-density is different from ANT-quantity in that a particular ant will leave Q_2 units of pheromone for every unit of length on the path. The value of $\Delta\tau_{ij}^k(t, t+1)$ is calculated as follows:

$$\Delta\tau_{ij}^k(t, t+1) = \begin{cases} Q_2 & \text{If the } k^{\text{th}} \text{ ant goes from } i \text{ to } j \text{ between } t \text{ \& } t+1 \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

The ANT-quantity and ANT-density algorithms can be run for any number of user-defined iterations. The general algorithm for ANT-quantity and ANT-density is given in Figure 5 (Coloni, Dorigo, & Maniezzo, 1991).

```

1 Initialize:
  Set  $t = 0$ 
  Set an initial value  $\tau_{ij}(t)$  for trail intensity on every edge
  Place  $d$  ants on every node  $i$ 
  Set  $\Delta\tau_{ij}(t, t+1) = 0$  for every  $i$  and  $j$ 

2 Repeat until the tabu list is full { this step will be repeated  $n$  times }
  For  $i=1$  to  $n$  do { for every node }
    For  $k=1$  to  $d$  do { for every ant on node  $i$  at time  $t$  }
      Choose the node to move to, with probability  $p_{ij}$  given by equation
      (1), and move  $k^{\text{th}}$  ant to the chosen location
      Insert the chosen node in the tabu list of ant  $k$ 
      Set  $\Delta\tau_{ij}(t, t+1) = \Delta\tau_{ij}(t, t+1) + \Delta\tau_y^k(t, t+1)$  computing
       $\Delta\tau_y^k(t, t+1)$  as defined in (3) or in (4)
      Compute  $\tau_{ij}(t+1)$  and  $p_{ij}(t+1)$  according to equations (2) and (1)

3 Memorize the shortest path found up to now and empty all tabu lists

4 If not(End_Test) then
  Set  $t = t+1$ 
  Set  $\Delta\tau_{ij}(t, t+1) = 0$  for every  $i$  and  $j$ 
  goto step 2
else
  Print shortest path and stop
  {End_Test is currently defined just as a test on the number of
  cycles}

```

Figure 5: ANT-density and ANT-quantity algorithm.

ANT-cycle

ANT-cycle differs from ANT-density and ANT-quantity in that $\Delta\tau_{ij}^k$ is not computed at every step, but after n steps when a solution to the problem is found. The value of $\Delta\tau_{ij}^k(t, t+n)$ is calculated as follows:

$$\Delta\tau_{ij}^k(t, t+n) = \begin{cases} \frac{Q_3}{L^k} & \text{if } k^{\text{th}} \text{ ant goes from node } i \text{ to node } j \text{ in its tour} \\ 0 & \text{Otherwise} \end{cases} \quad (5)$$

where Q_3 is a constant and L^k is the tour length for the k^{th} ant. The trail intensity is updated every n steps according to a calculation similar to (2):

$$\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij}(t, t+n) \quad (6)$$

$$\text{where } \Delta\tau_{ij}(t, t+n) = \sum_{k=1}^d \Delta\tau_{ij}^k(t, t+n).$$

The ANT-cycle algorithm can be run for any number of user-defined iterations. The ANT-cycle algorithm is given in Figure 6 (Colomi, Dorigo, & Maniezzo, 1991).

```

1 Initialize:
  Set  $t = 0$ 
  Set an initial value  $\tau_{ij}(t)$  for trail intensity on every edge
  Place  $d$  ants on every node  $i$ 
  Set  $\Delta\tau_{ij}(t, t+n) = 0$  for every  $i$  and  $j$ 

2 Repeat until the tabu list is full { this step will be repeated  $n$  times }
  For  $i=1$  to  $n$  do { for every node }
    For  $k=1$  to  $d$  do { for every ant on node  $i$  at time  $t$  }
      Choose the node to move to, with probability  $p_{ij}$  given by equation (1),
      and move  $k^{th}$  ant to the chosen location
      Insert the chosen node in the tabu list of ant  $k$ 
      Compute  $\Delta\tau_{ij}^k(t, t+n)$  as defined in (5).

      Compute  $\Delta\tau_{ij}(t, t+n) = \sum_{k=1}^d \Delta\tau_{ij}^k(t, t+n)$ 

      Compute  $\tau_{ij}(t+n)$  and  $p_{ij}(t+n)$  according to equations (6) and (1)

3 Memorize the shortest path found up to now and empty all tabu lists

4 If not(End_Test) then
  Set  $t = t + n$ 
  Set  $\Delta\tau_{ij}(t, t+n) = 0$  for every  $i$  and  $j$ 
  goto step 2
else
  Print shortest path and stop
  { End_Test is currently defined just as a test on the number of cycles }

```

Figure 6: ANT-cycle algorithm.

CHAPTER 3

JOB SHOP SCHEDULING

Job Shop Scheduling Algorithm

Scheduling can be found in manufacturing and production systems as some information-processing environments. Scheduling is a decision-making process and concerns the allocation of limited resources to tasks over time. The goal of scheduling is to optimize one or more objectives (Pinedo, 1995). Job shop scheduling is a deterministic, industrial scheduling problem and can be defined as follows: Given a set of M machines and J jobs where the jobs each have an ordered sequence of operations to be executed on the machines. The order of the operations is fixed and no two jobs can be processed on a machine at the same time. Once an operation starts processing, it cannot be interrupted (van der Zwaan & Marques, 1999). Recirculation occurs when jobs are allowed to visit a machine more than once, but this is not allowed here. The problem is that we want to assign the operations to machines so that the completion time of the last job to leave the system is minimized. This is called the makespan (Pinedo, 1995). Job shop scheduling is a classical NP-hard problem (Lawler, Lenstra, Rinnooy Kan, Shmoys, 1993).

A job shop problem can be represented by a disjunctive graph $G = (N, A, B)$. N is equal to the nodes in the graph and corresponds to all of the operations performed on J jobs. A represents the solid (conjunctive) arcs in the graph that display the precedence relationship between the operations of a single job. B represents the dotted (disjunctive)

arcs in the graph that connect two operations, from two different jobs, that are to be processed on the same machine. The disjunctive arcs are bi-directional (Pinedo, 1995).

The standard model for a J -job, M -machine job shop problem is: $J/M/G/C_{max}$, where G represents the precedence rules for the processing order of operations on the machines and C_{max} stands for the minimum makespan. Figure 7 displays the graph representation of a job shop problem $2/3/G/C_{max}$; a 2 job, 3 machine job shop problem. O_{11} represents the operation from job 1 to be performed on machine 1. S is the source node and has conjunctive arcs pointing to the first operations of all jobs. The last operations for all jobs point to the sink node, T . Each operation has its own processing time (van der Zwaan & Marques, 1999).

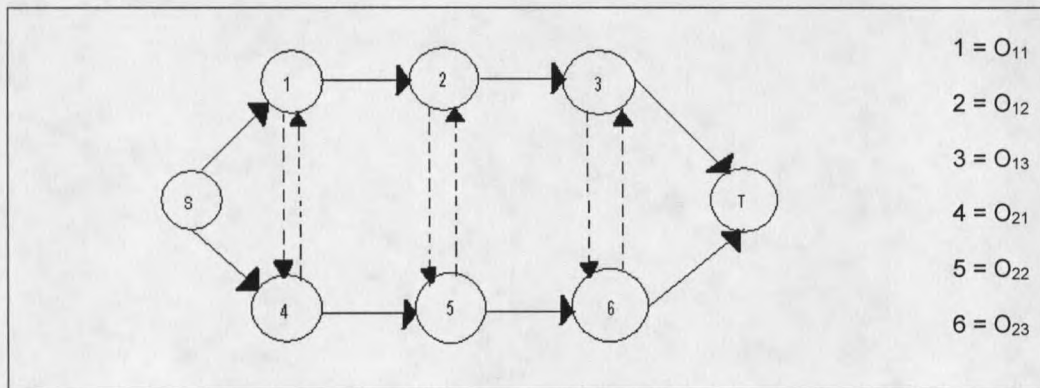


Figure 7: Representation of a $2/3/G/C_{max}$ job shop problem.

JSP can be formally stated as follows:

Given a set $M = \{M_1, \dots, M_m\}$ of machines, a set $J = \{J_1, \dots, J_n\}$ of jobs and a set $O = \{u_{ij}\}$, $(i, j) \in I$, where $I \subseteq [1, n] \times [1, m]$. For each operation $u_{ij} \in O$, there is a job J_i that it belongs to, a machine M_j where it must be processed, and a processing time p_{ij} . O can be broken down into chains that represent the jobs: if the relation

$u_{ip} \rightarrow u_{iq}$ is in a chain, both operations belong to job J_i and there is no u_{ik} with $u_{ip} \rightarrow u_{ik}$ or $u_{ik} \rightarrow u_{iq}$. The objective is to find a starting time $S_{ij} (\forall u_{ij} \in O)$ such that it is minimized. The problem can be formulated into a linear program:

$$\max_{u_{ij} \in O} (S_{ij} + p_{ij}) \quad (7)$$

subject to

$$S_{ij} \geq S_{ik} + p_{ik} \text{ when } u_{ik} \rightarrow u_{ij} \quad (8)$$

$$(S_{ij} \geq S_{ik} + p_{kj}) \vee (S_{kj} \geq S_{ij} + p_{ij}) \quad (9)$$

Equation (8) states the operation precedence constraint where the operations have a fixed order and equation (9) states the constraint that only one job may be processed on a machine at one time. The completion time for the operation $u_{ij} \in O$ is represented as $C_{ij} = S_{ij} + p_{ij}$. The maximum of the all completion times is the makespan (Colomi, Dorigo, & Maniezzo, 1994).

There are many different priority rules for operations selection. A few of the common rules are:

SRT: select the operation with the shortest processing time

LPT: select the operation with the longest processing time

SRT: select the operation belonging to the job with the shortest remaining processing time.

LRT: select the operation belonging to the job with the longest remaining processing time (Colomi, Dorigo, Maniezzo, & Trubian, 1994).

This thesis applied the LRT rule because of its similarities with results of stochastic scheduling. Stochastic scheduling models real life production environments where there are many random events. These random events may include machine breakdowns, unexpected high-priority jobs, and uncertain processing times (Pinedo, 1995).

As stated earlier, job shop scheduling is NP-hard and is one of the most difficult combinatorial problems. Carlier and Pinson who used a branch and bound algorithm, exactly solved a famous instance of the problem, 10 jobs and 10 machines, in 1989. Many algorithms have been developed for job shop scheduling over the years, such as List Scheduler, Shifting Bottleneck, Simulated Annealing, Tabu Search, and the conventional Genetic Algorithm (Colomi, Dorigo, & Maniezzo, 1994).

CHAPTER 4

ANT SYSTEM FOR JOB SHOP SCHEDULING (AS-JSP)

JSP Representation in the AS

JSP can be represented in the AS by a directed, weighted graph $G = (V, E)$ where $V = O \cup \{u_0\}$ and $E = \{(u_{ij}, u_{kl}) : u_{ij}, u_{kl} \in O\} \cup \{(u_0, u_{i1}) : u_{i1} \text{ is the first operation in the chain of job } J_i\}$. Node u_0 is used to specify which job will be scheduled first in case there are multiple jobs with their first operation on the same machine (Colorni, Dorigo, & Maniezzo, 1994). Figure 8 displays how the JSP shown in Figure 7 is represented in the AS. S is the same as u_0 . The solid black edges are bi-directional.

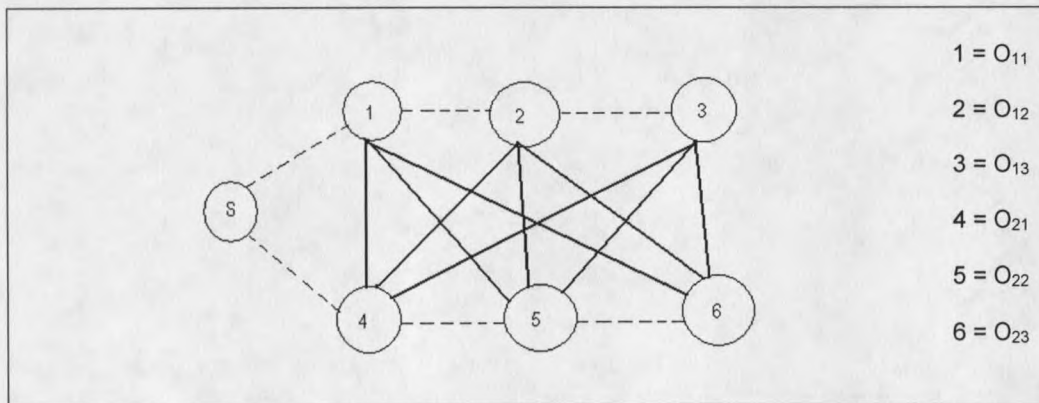


Figure 8: The AS representation of the JSP in Figure 7.

As with the disjunctive graph representation, the nodes (except u_0) represent the operations of the jobs. The maximum number of nodes for an $M \times J$ job shop problem is $(M \times J) + 1$. The extra node added is for u_0 . The number of edges in the graph is

$$\frac{|O| * (|O| - 1)}{2} + J \text{ where } |O| \text{ is the number of operations in the problem and } |O| = n * m.$$

Each edge has a pair of values, $\{\tau_{ij}, \eta_{ij}\}$, that represent the amount of pheromone on the edge and the distance between node i and node j . The amount of pheromone on the bi-directional edges must be defined for both directions, which makes the graph non-symmetric. The trail intensity, τ , is represented by a table with size $([J * M] \times [J * M])$ to allow non-symmetric values (van der Zwaan & Marques, 1999).

Two matrices, T and P , are used to define JSP. The matrix T describes the processing order of the machines for each job. For example, $T = \begin{bmatrix} M1 & M2 & M3 \\ M2 & M3 & M1 \end{bmatrix}$ where a row in the matrix represents a job and each element in T is an operation. The matrix P

describes the processing time of each operation. For example, $P = \begin{bmatrix} t(O_{11}) & \dots & t(O_{1m}) \\ t(O_{21}) & & t(O_{2m}) \\ \vdots & & \vdots \\ t(O_{n1}) & \dots & t(O_{nm}) \end{bmatrix}$.

Each ant maintains three lists during the duration of the algorithm. The ant has a list, W , which keeps track of which nodes it has not yet visited. The second list, X , contains the nodes the ant is allowed to visit in the current iteration. The ant also has a Tabu list that contains the nodes already visited. The solution to the job shop scheduling problem is found in the Tabu list at the completion of the algorithm.

CHAPTER 5

EXPERIMENTS AND RESULTS

Problem Instances

Three benchmark problems were used to evaluate the ANT-cycle, ANT-density, and ANT-quantity algorithms. The problem instances are: *abz5* from Balas, Adams, and Zawack, *la01* from Lawrence, and *ft20* from Fisher and Thompson (Beasley, 2002). The parameter settings used were: $\alpha = 0.7$, $\beta = 0.7$, $\rho = 0.7$, and the initial pheromone set to 0.5. All problem instances were run on an Intel Pentium IV 1.4Ghz system.

la01 Problem Instance

The problem instance *la01* has 10 jobs and 5 machines with the optimal value equal to 666 (Hestermann, 1996). The graph has a total of 50 nodes with an additional node as the start node (see Chapter 4). Table 1 shows the results obtained when running the three algorithms with 25 ants. The ANT-cycle algorithm was the most successful in finding a good solution, especially after five iterations. The ANT-density and ANT-quantity algorithms produced better solutions when repeated for five iterations. All three algorithms had little variation in the solutions. The three algorithms took an average of 895 CPU seconds (15 minutes) to complete one iteration and 716 CPU seconds (12 minutes) to complete four more iterations. ANT-cycle was the fastest of the three algorithms, taking 30% less time than ANT-Quantity and 25% less time than ANT-density. Figure 9 displays a graph of the

average, maximum, minimum and optimum value for each of the algorithm instances shown in Table 1.

Algorithm Instance	Average	Maximum	Minimum	Average % of the Optimum
Cycle	688.96	749	662	95.3%
Cycle 5 Iterations	615.56	665	586	107.3%
Density	962.44	965	960	69.2%
Density 5 Iterations	782	782	782	85.2%
Quantity	1065.2	1068	1063	62.5%
Quantity 5 Iterations	781	781	781	85.3%

Table 1: *la01* problem instance results with 25 ants

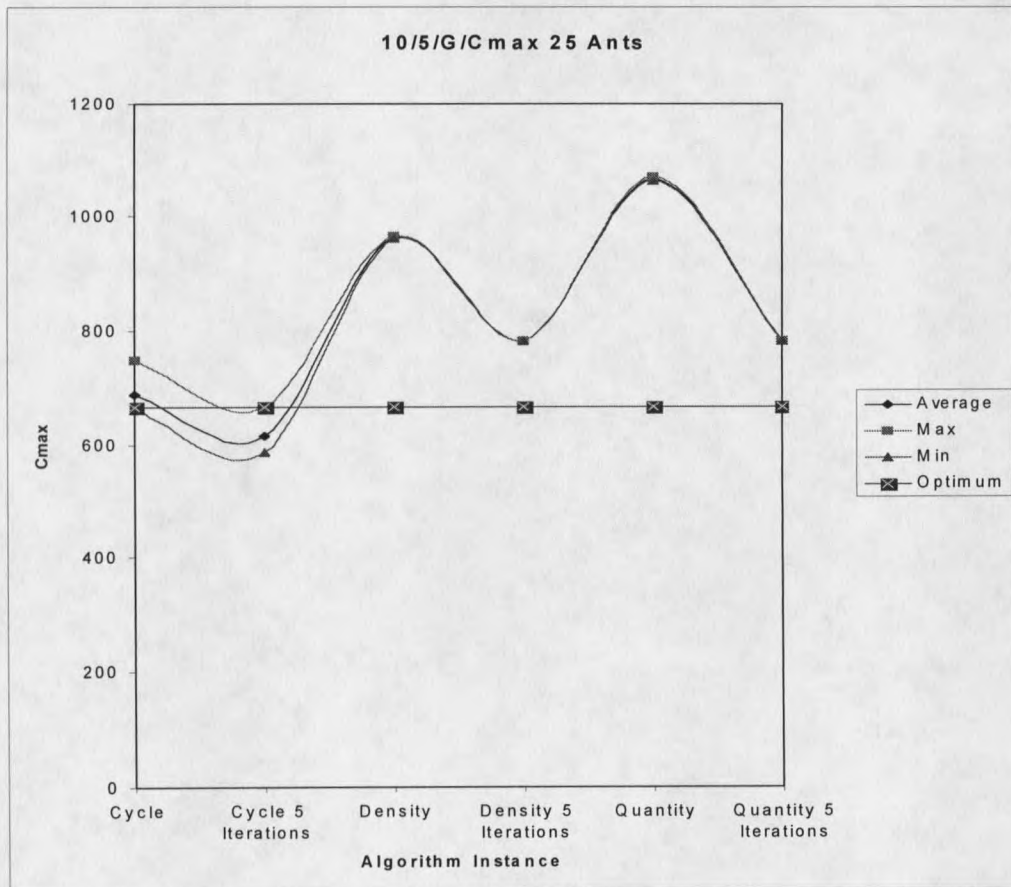


Figure 9: *la01* problem instance with 25 ants

The *la01* problem instance was also run through the three algorithms with 50 ants. The results of this problem instance can be seen in Table 2. It was suggested in the book “Swarm Intelligence: from natural to artificial intelligence,” that in order to get optimal solutions, the number of ants should be equal to the number of nodes (Bonabeau, Dorigo, and Theraulaz, 1999). In the *la01* problem instance, the solutions seemed to get worse when the number of ants equaled the number of nodes. All three algorithms found better solutions when they ran for five iterations. The three algorithms took an average of 1070

CPU seconds (18 minutes) to complete one iteration and 4342 CPU seconds (72 minutes) to complete five iterations. ANT-cycle was once again faster than the other two algorithms. Figure 10 displays a graph of the average, maximum, minimum and optimum value for each of the algorithm instances shown in Table 2.

Algorithm Instance	Average	Maximum	Minimum	Average % of the Optimum
Cycle	5993	9830	990	28.3%
Cycle 5 Iterations	1550.24	2493	783	51.6%
Density	6005	6010	6000	11.1%
Density 5 Iterations	1100	1100	1100	60.5%
Quantity	6047.46	6066	6037	11.0%
Quantity 5 Iterations	1326.56	1327	1326	50.2%

Table 2: *la01* problem instance with 50 ants

The ANT-density and ANT-quantity algorithms had very little variation in the solutions produced by ants. The solutions produced by the ants in the ANT-cycle algorithm were much more variant for both one and five iterations.

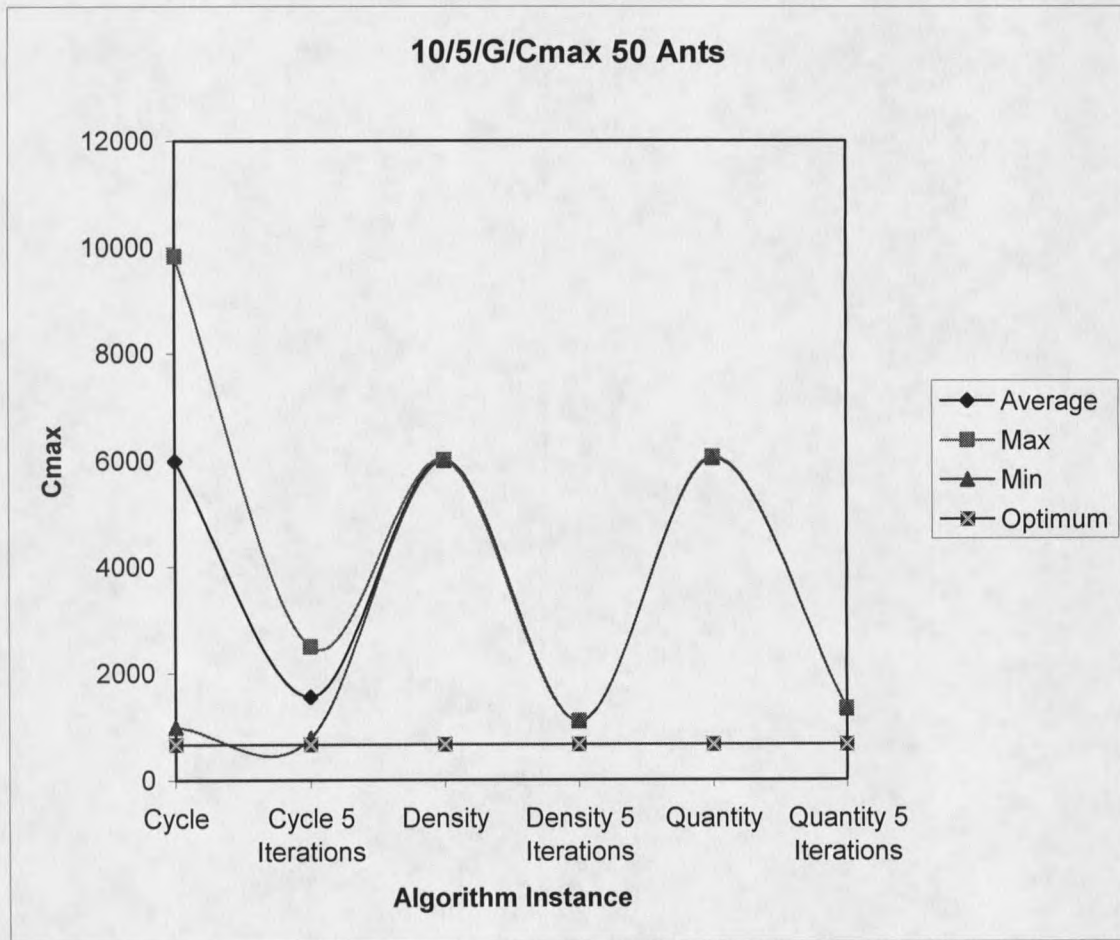


Figure 10: *la01* problem instance with 50 ants

The *la01* problem instance was tested by Esquivel, Ferrero, Gallard, Salto, Alfonso, and Schutz using two evolutionary algorithms (Esquivel, Ferrero, Gallard, Salto, Alfonso, and Schutz, 2001). The conventional evolutionary approach was able to achieve a value within 71% of the optimum. The multistage evolutionary approach was able to achieve the optimum solution.

abz5 Problem Instance

The problem instance *abz5* has 10 jobs and 10 machines and the optimal solution is 1234 (Hestermann, 1996). The graph has a total of 100 nodes with an additional node as the start node (see Chapter 4). Table 3 shows the results obtained when running the three algorithms with 50 ants. On the average, none of the three algorithms came within 60% of the optimal solution. The ANT-cycle algorithm was the most successful in finding a good solution than the ANT-density or ANT-quantity algorithms. The minimum value for the ANT-cycle algorithm when run once comes within 106.7% of the optimum and is within 110.7% with five iterations. Only the ANT-cycle algorithm was run for five iterations. Better solutions could possibly be found for ANT-density and ANT-quantity by running them for five iterations. The three algorithms on average took 3817 CPU seconds (64 minutes) to complete one iteration. The average time for the ANT-cycle algorithm to complete five iterations was 2597 CPU seconds (43 minutes). Figure 11 displays a graph of the average, maximum, minimum and optimum value for each of the algorithm instances shown in Table 3.

Algorithm Instance	Average	Maximum	Minimum	Average % of the Optimum
Cycle	4198.46	9268	1156	49.8%
Cycle 5 Iterations	4091.54	5585	1114	54.3%
Density	5228.4	5414	5103	23.5%
Quantity	5369.46	5582	5192	23.0%

Table 3: *abz5* problem instance with 50 ants

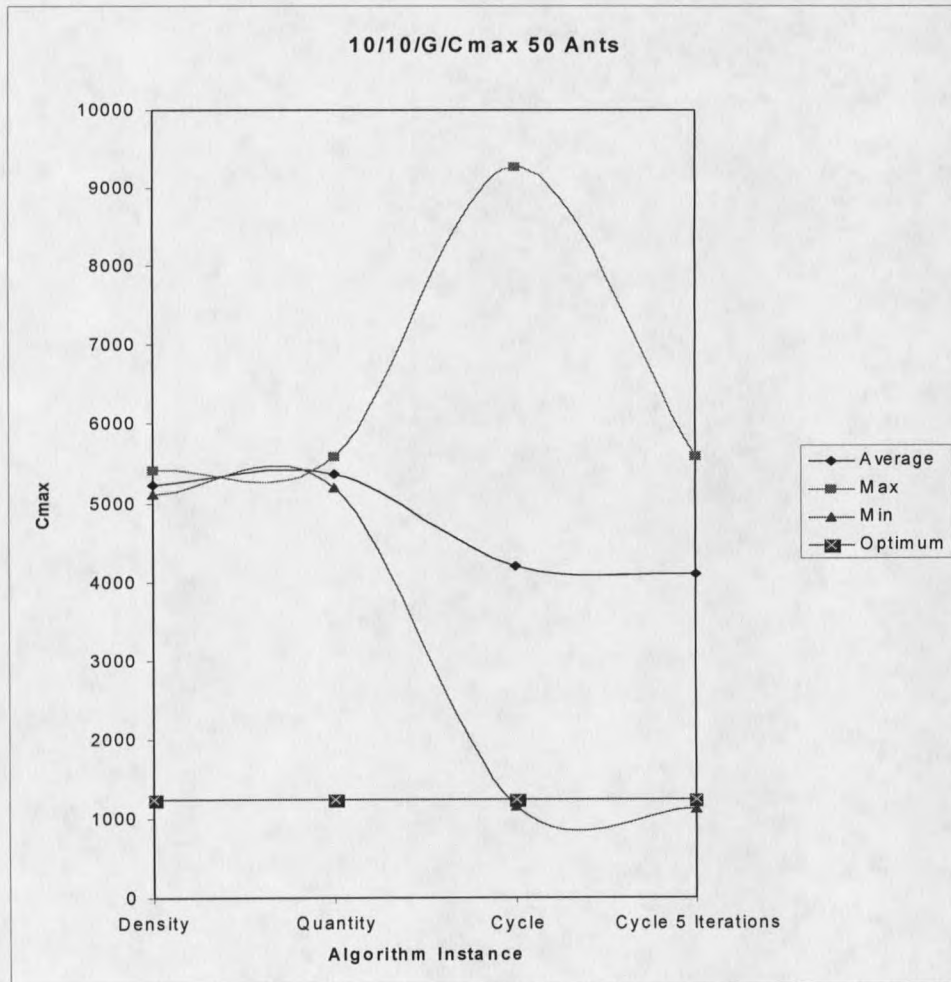


Figure 11: *abz5* problem instance with 50 ants

The *abz5* problem instance was tested with the three algorithms with 100 ants. The results of this problem instance can be seen in Table 4. The *abz5* problem instance was similar to the *la01* problem instance in that the solutions got worse when the number of ants equaled the number of nodes. Four ants in the ANT-cycle algorithm (one iteration) found solutions between 86% and 116.8% of the optimum. The ANT-cycle algorithm did not seem to benefit from running five iterations for this particular instance. The three algorithms on

average took 8174 CPU seconds (136 minutes) to complete to complete one iteration. The ANT-cycle algorithm completed five iterations with an average time of 5511 CPU seconds (92 minutes). Figure 12 displays a graph of the average, maximum, minimum and optimum value for each of the algorithm instances shown in Table 4.

Algorithm Instance	Average	Maximum	Minimum	Average % of the Optimum
Cycle	16978.33	43750	1056	42.3%
Cycle 5 Iterations	5721.36	10149	3663	22.5%
Density	11151.07	11718	10802	11.0%
Quantity	13107.82	13726	12663	9.4%

Table 4: *abz5* problem instance for 100 ants

As for the *la01* problem instance, the ANT-density and ANT-quantity algorithms had very little variation in the solutions produced by ants, regardless of the number of ants. The solutions produced by the ants in the ANT-cycle algorithm were much more variant.

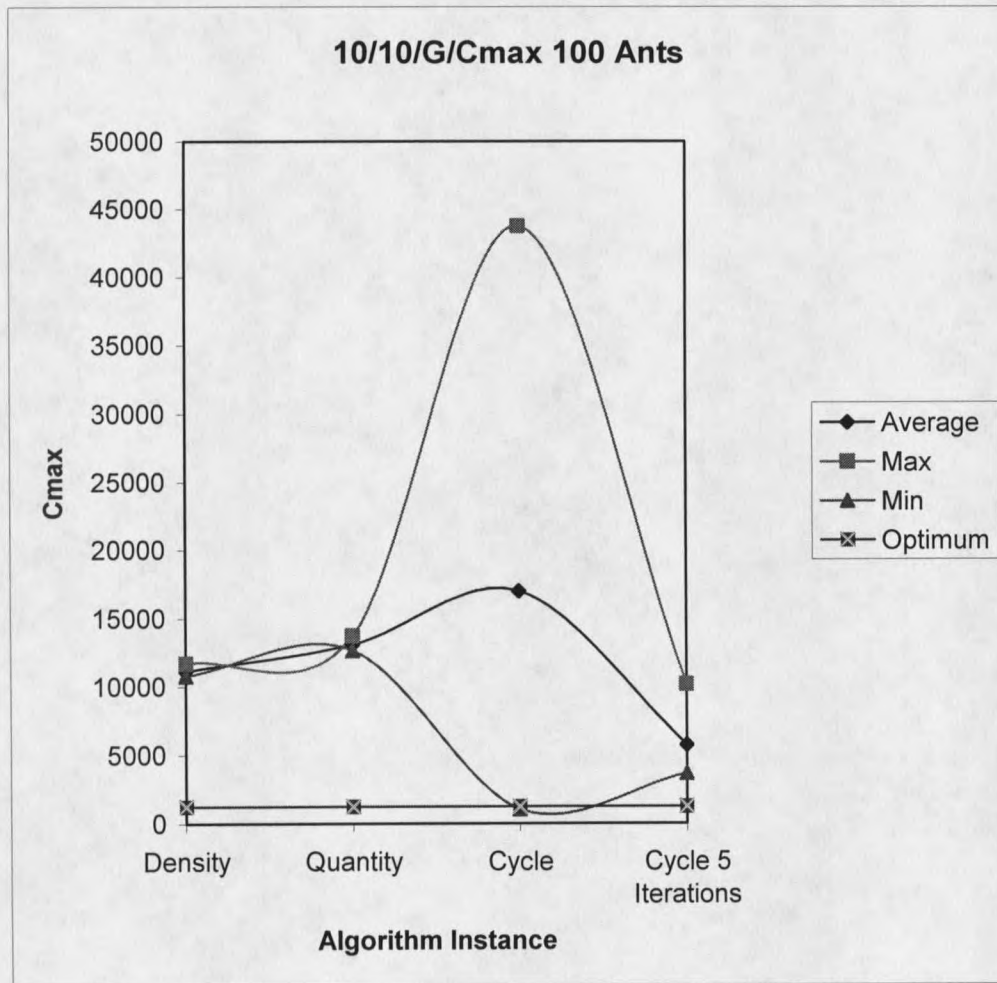


Figure 12: *abz5* problem instance with 100 ants

Chambers and Barnes applied a dynamic tabu search to the *abz5* problem instance in 1996. The dynamic tabu search achieved the optimum value in 55.65 CPU seconds on a DEC 3000/600 (AXP) (Chambers and Barnes, 1996). Adams, Balas, and Zawack used the shifting bottleneck procedure on the *abz5* problem instance. The straight version of the shifting bottleneck produced a value within 94.5% of the optimum in 5.70 CPU seconds.

The enumerative version of the shifting bottleneck produced a value within 99.6% of the optimum in 1503 CPU seconds (Adams, Balas, and Zawack, 1988).

ft20 Problem Instance

The problem instance *ft20* has 20 jobs and 5 machines with the optimal solution of 1165 (Hestermann, 1996). The graph has a total of 100 nodes with an additional node as the start node (see Chapter 4). Table 5 shows the results obtained when running the three algorithms with 50 ants. The ANT-cycle algorithm was more successful when run for five iterations. The best solution obtained is within 58.3% of the optimum. Solutions generated by all three algorithms were poor for a single iteration and achieved only 51.5% of the optimum for a five iteration cycle. The three algorithms on average took 8313 CPU seconds (139 minutes) to complete one iteration. The average time for the ANT-cycle algorithm to complete five iterations was 2283 CPU seconds (38 minutes). Figure 13 displays a graph of the average, maximum, minimum and optimum value for each of the algorithm instances shown in Table 5.

Algorithm Instance	Average	Maximum	Minimum	Average % of the Optimum
Cycle	11939.26	20821	3738	15.5%
Cycle 5 Iterations	2353.2	2498	1998	51.5%
Density	8776.64	9068	8535	13.3%
Quantity	12913.18	13175	12665	9.0%

Table 5: *ft20* problem instance with 50 ants

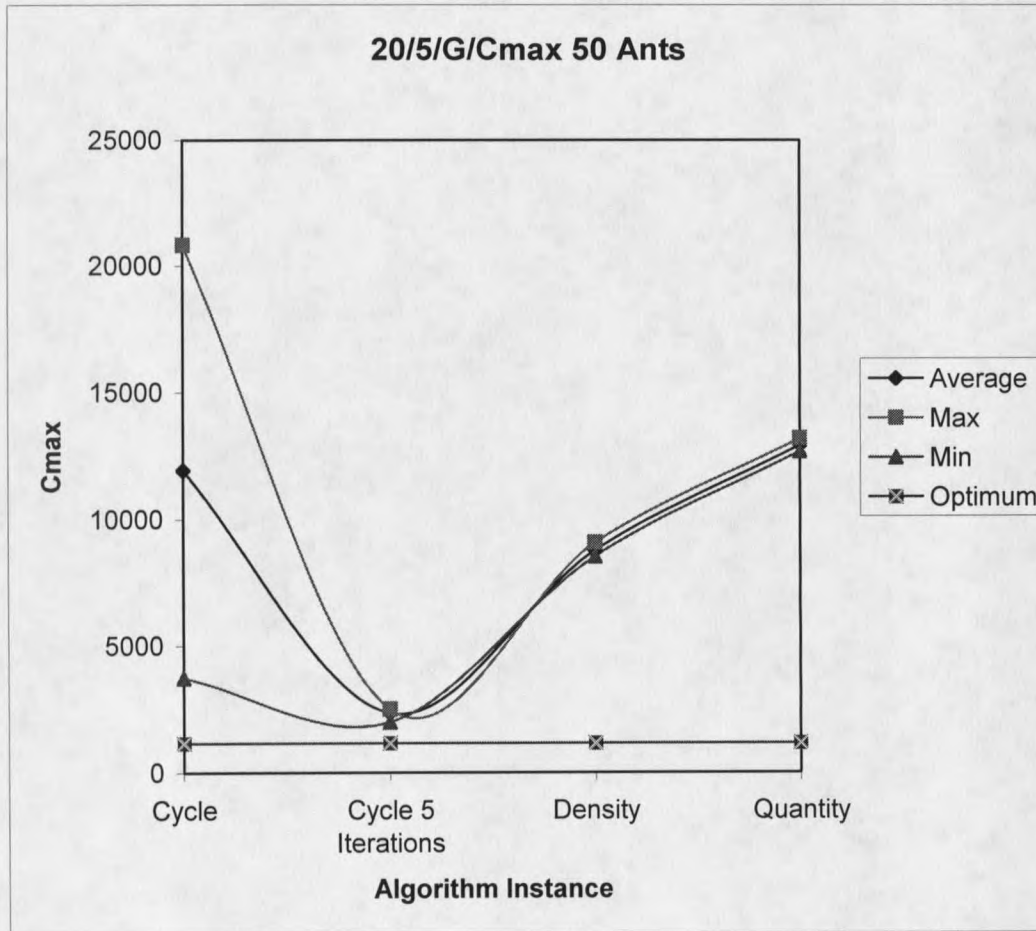


Figure 13: *ft20* problem instance with 50 ants

The *ft20* problem instance was run through the three algorithms using 100 ants, as shown in Table 6. The *ft20* problem instance was similar to the *abz5* and *la01* problem instances in that the solutions got worse when the number of ants equaled the number of nodes. The average CPU time for the *ft20* instance with 100 ants is 18,608 CPU seconds (5 hours). The ANT-cycle algorithm was much faster than the other two algorithms. Figure

14 displays a graph of the average, maximum, minimum and optimum value for each of the algorithm instances shown in Table 6.

Algorithm Instance	Average	Maximum	Minimum	Average % of the Optimum
Cycle	24166.46	43285	3614	13.2%
Density	36146.89	37675	33850	3.3%
Quantity	18837.18	19455	18361	6.2%

Table 6: ft20 problem instance with 100 ants

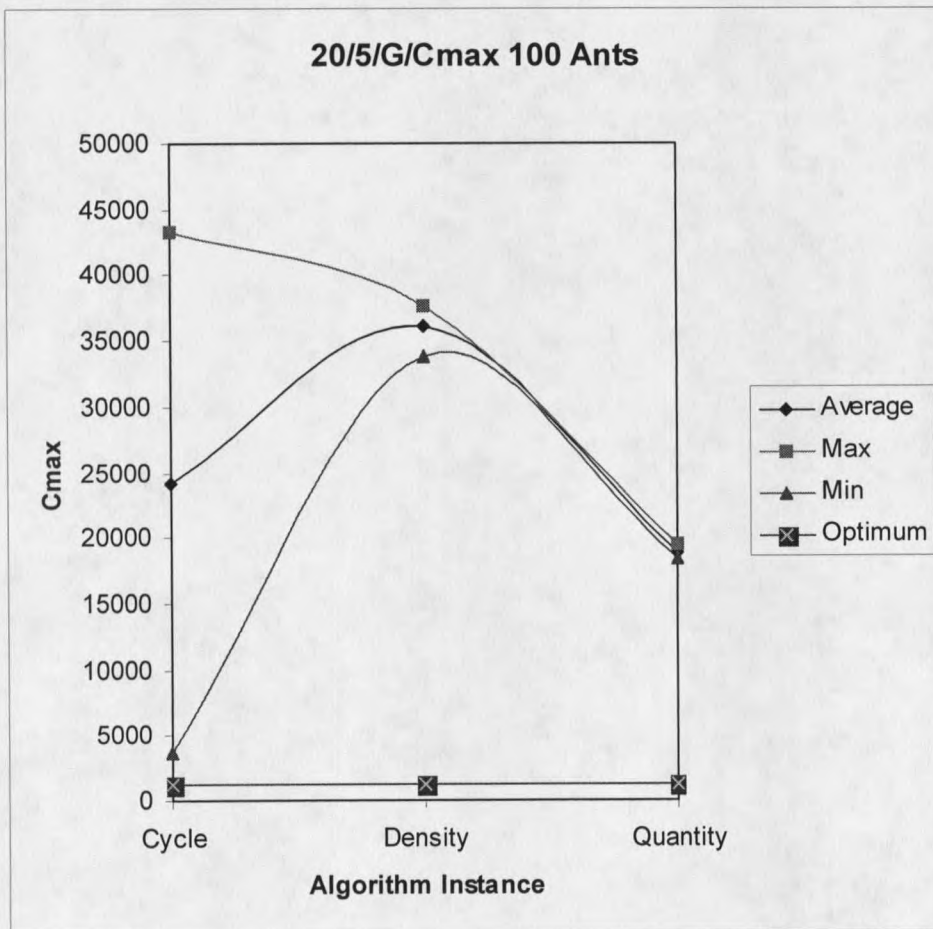


Figure 14: ft20 problem instance with 100 ants

Kaschel, Teich, Kobernik, and Meier used the *ft20* problem instance to test other algorithms in their journal article. The computer used to gather these results was a Pentium II/333 Mhz processor with 128 MB of memory using the Linux operating system. The genetic algorithm gave results within 99.1% of the optimum value and only took an average of 365 CPU seconds. The simulated annealing algorithm gave results equal to the optimal value and took an average of 310 CPU seconds. These results include both the non-homogeneous and homogeneous algorithms (Kaschel, Teich, Kobernik, and Meier, 1999).

CHAPTER 6

CONCLUSION

Conclusion

The ANT-cycle algorithm was the most successful in finding good solutions to all the job shop problem instances used in this thesis, and it produced better results when run for five iterations. This is because the trail intensity calculations are calculated after a solution is found instead of after every time an ant moves from one node to the next. When the ANT-cycle algorithm is run once, the ants are working independently finding a solution to the problem. When the ANT-cycle algorithm is run more than once, the solution for a particular ant found in the previous iteration is used to guide the colony of ants towards finding a better solution in the current iteration.

The ANT-density and ANT-quantity algorithms' performances were similar, as expected due to the similarity of the algorithms. The ANT-density and ANT-quantity algorithms had little or no variation in the solutions produced by the ants. When these two algorithms were run once, the ants converged to the same solution leaving no variation in the results. The reasons for this are that the trail intensity is calculated after every move and the number of ants is greater than the number of start nodes. The ants are only allowed to start at a node that is the first operation in each job. This means that the number of nodes the ants can start at is equal to the number of jobs. For example, in the problem instance *la01*, where there were 10 jobs and 25 ants, all of the starting nodes had at least two ants. Since the ANT-density and ANT-quantity algorithms calculated the trail intensity after every

move on the graph, this influenced the ants directly behind other ants to choose a path that might lead to a poor solution. When the two algorithms were run for more than one iteration, the variation decreased more because the ants in the previous iteration converged to a solution within a small range, leaving ants in subsequent iterations to use the same paths to converge to a solution within an even smaller range. The ANT-density and ANT-quantity algorithms produced better results when run more than once. This is due to the trail intensities already on the graph, leading an ant to find a solution by improving on a previous solution. After multiple iterations, the ants converge closer to the optimum value. The ANT-density and ANT-quantity algorithms did not perform well when the problem instance increased in size or complexity. This is due to the increased number of calculations required when the problem instance increases. The ANT-density and ANT-quantity algorithms are making calculations after every move to a node unlike the ANT-cycle algorithm, which makes one calculation after moving through all the nodes in the graph. This would also explain why the ANT-cycle algorithm performs better for the harder problem instances than the other two algorithms.

All algorithms performed worse when the number ants equaled the number of nodes in the graph. The best solutions were produced when the number ants equaled half the number of nodes in the graph. The reason for this is that the number of ants starting on a particular node increases proportionally with the number of nodes. This allows ants to follow other ants whether or not they are finding a poor solution. The statement made by Bonabeau, Dorigo, and Theraulaz in "Swarm Intelligence: from natural to artificial intelligence" was based on the Ant System with the Traveling Salesman Problem (AS-TSP).

In the case of the Traveling Salesman Problem, there are no constraints on which nodes can be visited at a particular time, only that every node been seen in a tour. In the AS-TSP, every ant was placed on a different starting node allowing the ant to chart its own course (during the first iteration) instead of following directly behind another ant.

Coloni, Dorigo, and Maniezzo claimed in a 1991 paper that the settings for α , β , and ρ should be 1, 1, and 0.7 respectively. They used the ANT-cycle algorithm with those parameter settings and good solutions were always found. They found that as β got closer to 1, the number of iterations needed to reach the optimum decreased. This thesis used 0.7 for all three parameters.

As the number of ants increased, the time complexity of the algorithms increased. This fact was more obvious for the ANT-density and the ANT-quantity algorithms. The reason for this was previously stated to be that the number of calculations with the ANT-density and ANT-quantity algorithms is much larger than for the ANT-cycle algorithm. Van der Zwaan and Marques noticed the same result in the 1999 paper on the Ant System with job shop scheduling. However, only the ANT-cycle algorithm was used in their research. A reason for this is the constraints placed on the job shop scheduling problem such as the operations within a job must be done in order and no two operations can be performed on the same machine at the same time.

In 1991, Coloni, Dorigo, and Maniezzo compared the three instantiations of the Ant System. They found that the ANT-cycle algorithm performed significantly better than the other two on the traveling salesman problem, especially on the harder problems. They also showed that the number of cycles needed to reach the optimum, for ANT-density and ANT-

quantity, is much higher when the number of ants is equal to the number of nodes than when the number of ants is half of the number of nodes.

The shifting bottleneck procedure, the genetic algorithm, and the dynamic tabu search all produced near optimal or optimal results in less CPU time than ANT-cycle, ANT-density, and ANT-quantity, finishing in 1503 CPU seconds or fewer on less powerful machines than the three Ant System algorithms. The Ant System algorithms took much longer to complete as the number of nodes and ants increased. ANT-cycle is the most promising instantiation of the Ant System in producing near optimal results in a reasonable amount of time.

Future Research

Further research can be done with the three Ant System algorithms. A change in the parameter values for AS-JSP could result in better solutions, such as trying different values for α , β , and ρ . If the number of iterations the algorithms performed were increased to a higher value, it is probable that better solutions would be processed. If the ANT-density and ANT-quantity algorithms were to run for at least five iterations for the *abz5* and the *ft20* problem instances, it is likely that better solutions would be found. Improved results could also be found if the number of iterations were increased when the number of ants is equal to the number of nodes. More benchmark problems with a higher amount of operations should be tested on the three algorithms.

REFERENCES CITED

Adams, J., Balas, E., and Zawack, D. (1988). The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*. 34(3). 391-401.

Ambite, J. (2001). www-2.cs.cmu.edu/afs/cs/project/jair/pub/volume15/ambite01a.html/node9.html.

Beasley, J.E. (2002). www.ms.ic.ac.uk/jeb/pub/jobshop1.txt.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: from natural to artificial intelligence*. Oxford University Press. New York. 39-71.

Bonabeau, E. and Theraulaz, G. (2000). Swarm Smarts. *Scientific American*. 72-79.

Chambers, J. and Barnes, J. (1996). New Tabu Search Results for the Job Shop Scheduling Problem. *Technical Report Series, ORP 96-06*, Graduate Program in Operations Research and Industrial Engineering, Department of Mechanical Engineering, The University of Texas at Austin.

Coloni, A., Dorigo, M., and Maniezzo V. (1994). Ant System for Job-Shop Scheduling. *Belgian Journal of Operations Research, Statistics, and Computer Science*. 34(1). 39-53.

Coloni A., Dorigo, M., and Maniezzo V. (1991). Distributed Optimization by Ant Colonies. *Proceedings of ECAL91 - European Conference on Artificial Life*. 134-142.

Coloni, A., Dorigo, M., and Maniezzo V. (1992). An investigation of some properties of an "Ant algorithm". *Proceedings of the Parallel Problem Solving From Nature Conference*. 509-520.

Dorigo, M. (2001). About Ant Colony Optimization. iridia.ulb.ac.be/~mdorigo/ACO/about.html

Dorigo, M., Caro, G., and Gambardella, L. (1999). Ant Algorithms for Discrete Optimization. *Artificial Life*. 5(2). 137-172.

Dorigo, M. and Caro, G. (1999). Ant Colony Optimization: A New Meta-Heuristic. *Proceedings of CEC99 - Congress on Evolutionary Computation 1999*.

Esquivel S.C., Ferrero S.W., Gallard R.H., Salto C., Alfonso H., and Schutz M. (2001). Enhanced Evolutionary Algorithms for Single and Multiobjective optimization in the

Job Shop Scheduling Problem. *Special Issue of Journal on Knowledge Based Systems entitled "Evolutionary Computation."*

Hestermann, C. (1996). Using OR-data as benchmarks. ki.informatik.uni-wuerzburg.de/forschung/publikationen/lehrstuhl/Hestermann-ECiMSIO-97/node3.html.

Kaschel, J., Teich, T., Kobernik, G., Meier, B. (1999). Algorithms for the Job Shop Scheduling Problem – a comparison of different methods. *Proceedings of 1999 ERUDIT Conference*.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., and Shmoys, D. B. (1993). Sequencing and scheduling: algorithms and complexity, in *Logistics of Production and Inventory, Handbooks Oper. Res. Management Sci.* Elsevier. New York. 445–522.

Pinedo, M. (1995). Scheduling: theory, algorithms, and systems. Prentice Hall. New Jersey. 1-5, 125-133.

Sipser, M. (1997). Introduction to the theory of computation. PWS Publishing Company. Boston. 273.

van der Zwaan, S. and Marques, C. (1999). Ant Colony Optimisation for Job Shop Scheduling. *Proceedings of the Third Workshop on Genetic Algorithms and Artificial Life (GAAL 99)*.

MONTANA STATE UNIVERSITY - BOZEMAN



3 1762 10359783 5