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

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

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 _____ 7/18/02
(Signature) (Date)

Approved for the Department of Computer Science

Rockford Ross _____ 7/18/02
(Signature) (Date)

Approved for the College of Graduate Studies

Bruce McLeod _____ 7-24-02
(Signature) (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 Carla L Peterson

Date 7/23/07

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

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 (Colorni, 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 (Colorni, 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).

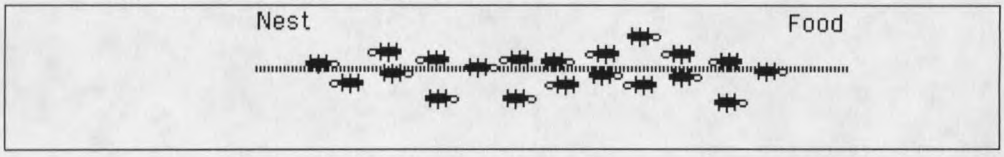

Figure 1: Ants have a direct path between their nest and the food source (Dorigo, 2001).

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.

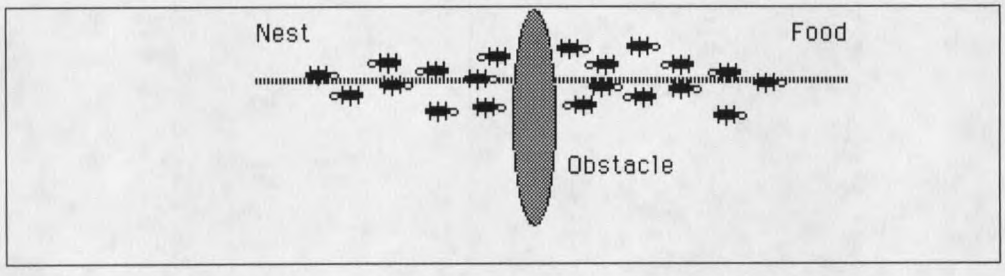

Figure 2: An obstacle is placed on the path (Dorigo, 2001).

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).

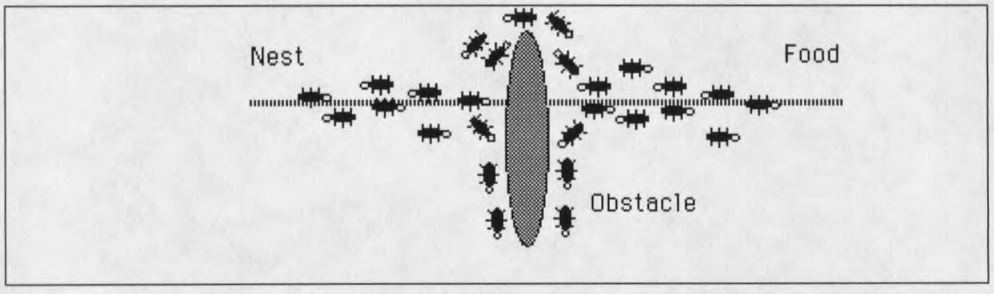

Figure 3: Cooperative ants (Dorigo, 2001).

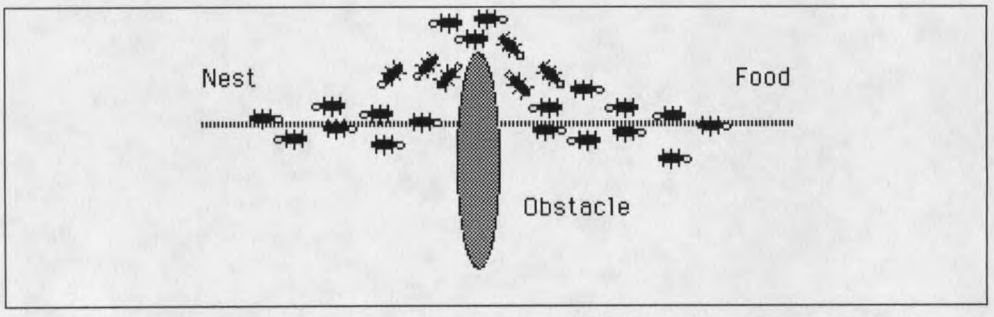

Figure 4: The longer path from the nest to the food source is left unused (Dorigo, 2001).

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 α=1, β=1, and ρ=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 $O(n^2 d)$ where n is the number of nodes in the graph and d is the number of ants (Colorni, 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 (Colorni, 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:

$$p_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum_{j \in T}\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}} \qquad (1)$$

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 (Colorni, 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:

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

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 kth 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 (Colorni, Dorigo, & Maniezzo, 1992).

There are three different ways to compute $\Delta\tau_{ij}^{k}(t,t+1)$ and when to update the trail intensity, $\tau_{ij}(t)$. Colorni, 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:

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

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:

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

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 (Colorni, 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^{th} ant to the chosen location
Insert the chosen node in the tabu list of ant k
Set $\Delta\tau_{ij}(t,t+1) = \Delta\tau_{ij}(t,t+1) + \Delta\tau_{ij}^{k}(t,t+1)$ computing $\Delta\tau_{ij}^{k}(t,t+1)$ 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:

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

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):

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

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 (Colorni, 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).

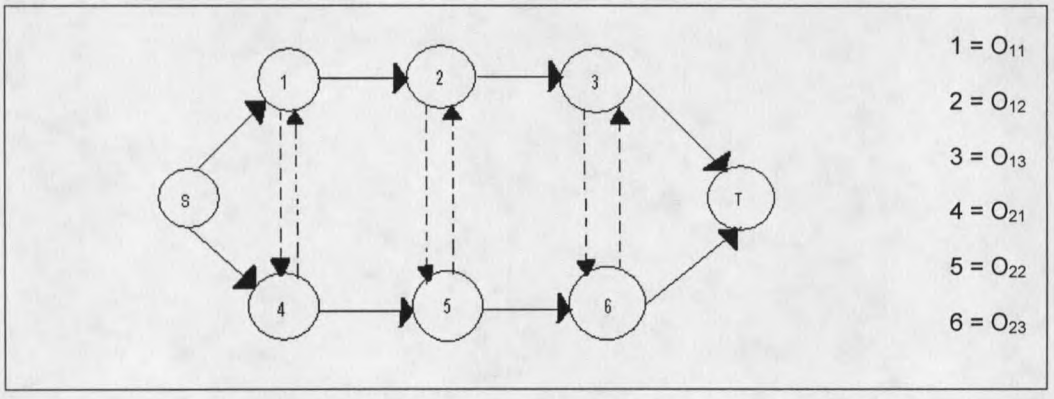

Figure 7: Representation of a 2/3/*G*/C_{max} job shop problem.

JSP can be formally stated as follows:

Given a set $M = \{M_1, \ldots, M_m\}$ of machines, a set $J = \{J_1, \ldots, J_n\}$ 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:

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

subject to

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

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

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 (Colorni, 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 (Colorni, 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 (Colorni, Dorigo, & Maniezzo, 1994).