MATEC Web of Conferences **52**, 01005 (2016)
DOI: 10.1051/matecconf/20165201005

Ant Foraging Behavior for Job Shop Problem

Diyana Abdul Mahad[1] and Zuhaimy Ismail[2]

[1]*Quantitative Methods Unit, Faculty of Management, Multimedia University, 63100 Cyberjaya, Selangor Darul Ehsan, Malaysia*

[2]*Department of Mathematics, Faculty of Science, Universiti Teknologi Malaysia,81310 Skudai, Johor Darul Ta'zim, Malaysia*

Abstract. Ant Colony Optimization (ACO) is a new algorithm approach, inspired by the foraging behavior of real ants. It has frequently been applied to many optimization problems and one such problem is in solving the job shop problem (JSP). The JSP is a finite set of jobs processed on a finite set of machine where once a job initiates processing on a given machine, it must complete processing and uninterrupted. In solving the Job Shop Scheduling problem, the process is measure by the amount of time required in completing a job known as a makespan and minimizing the makespan is the main objective of this study. In this paper, we developed an ACO algorithm to minimize the makespan. A real set of problems from a metal company in Johor bahru, producing 20 parts with jobs involving the process of clinching, tapping and power press respectively. The result from this study shows that the proposed ACO heuristics managed to produce a god result in a short time.

1 Introduction

Scheduling has been defined as "the art of assigning resources to tasks in order to ensure the completion of these tasks in a reasonable amount of time" [1]. A job shop scheduling problem is a problem of as scheduling n given jobs varying sizes on m given identical machines, while trying to minimize the total length or time of the schedule (that is when all the assigned jobs have finished processing). In a finite job shop setting, it has a finite set of jobs on a finite set of machines. Each job is characterized by a fixed order of operations; each of the jobs is to be processed on a specific machine and specified duration. Each machine can process at most one job at a time [2]. Once a job initiates processing on a given machine it must complete processing and must be uninterrupted. It is common that nowadays, the job command are given online where upon completing a job, a system or an algorithm is required to make a decision about the next job. Basically, the Job Shop Problem (JSP) is divided into three common shop models for scheduling namely Open Shops, Flow Shops, and Job Shops. Unlike JSP and FSP, the Open Shop model is a 'build to order' system where no inventory is required.

JSP is a discrete problem of a combinatorial problem and has been widely studied. It is scheduling problem generalized from the Travelling Salesman Problem (TSP). It has also been illustrated as a problem in computational complexity theory which is hard to solve in practice and well known as an NP-hard problem. The general objective of scheduling is to decide in what rules, methods or order to

perform jobs in the sequence, so that certain desired criteria are met. In this research, near-optimal solution methodologies for job shop scheduling are examined.

In searching for the optimal or near optimal solution to the JSP, we explore the possibility of using Ant Colony Optimization (ACO) approach. It is a new algorithm inspired by the foraging behavior of real ants. Many researchers have used and developed ACO algorithm to solve NP-hard problem such as travelling salesman problem [1], graph colouring [3], vehicle routing problem [4] and so on. The main aim of this study is to minimize the makespan that is the total completion time [5]. The makespan can be described as the time required to complete all the n jobs. Minimizing the makespan is a common objective in multiple-machine sequencing problems. A real scheduling problem from a small and medium enterprise producing components for auto industry is used. The rest of the paper is as follows. The full description of JSP is given in section 2 followed by the mathematical modeling. In Section 3, a brief explanation of the ACO approach, the background of the ACO and the pseudo code in the methodology flow chart. In Section 4, the discussion on the adaptation of ACO to solve JSP and the results and discussion is given in the final section.

2 Problem Dissertation

In JSP, it must have a finite set of n jobs, and finite set of m machines. Each job must have a chain of operations, and each machine can be handling on at most one operation at a time. Each operation needs to be processed during an uninterrupted period length of time on a given machine.

The problem of scheduling jobs in a machine shop is often modeled as a *classical* job shop problem, which is described as follows. Given is a shop consisting of m machines M_1, M_2,…, M_m. On these machines, a set of n jobs J_1, J_2… $J_n$needs to bescheduled. A schedule defines for each job J_j, a completion time C_i such that the jobs do not overlap in their execution [6]. Each machine is available from time 0 onwards and can process at mostone job at a time. Each job $J_j$consists a chain of operations O_{1j}, O_{2j},…,O_{nj}, wheredenotes the number of operations of job J_j. Operation O_{ij}can only be processedafter the completion of operation O_i–1, j (i=2,…,n_j); operation $O_{1j}$is available fromtime 0 onwards. O_{ij}needs uninterrupted processing on machine m_{ij}during a givennon-negative time p_{ij}. The objective is usually to find a schedule that minimizes themakespan, that is, to find a schedule in which the time to process all jobs is minimal.

This paper refers to the standard model of the n-job, m-machine job shop problem as;

$$J / M / G / C_{max}$$

where J and M stand for the number of jobs and machines, and the G and C_{max} represent the precedence rules and the minimum makespan. Respectively C_{max} is the maximum earliest completion time of the last operation of any job.

An example of $3/3/G/C_{max}$ JSP with three jobs and three machines is shown in Figure 1 with the matrix description given in Table 1.

Table 1: The Matrix represents the description on figure above.

	M_1	M_2	M_3
J_1	12	5	9
J_2	2	9	10

Table 1 show the data of an instance with 3 machines and 2 jobs, where each job consists of 3 operations. Whereas Figure 1 show the representative of the Table 1 into the graph. From Figure 1 we can see that the graph have 8 nodes, 17 edges and 6 operations. This has been calculates from equation below. From Equation 1, 2, and 3 we know that n is stand for jobs, m is for machines, and $|O|$ is operations it takes.

$$\text{Nodes} = n * m + 2 \quad (1)$$

$$\text{Operation } |O| = n * m \quad (2)$$

$$\text{Edges} = \frac{|O| \bullet (|O| - 1)}{2} + n \tag{3}$$

Figure 1 shows the example of the 2 jobs processed with the 3 machines. As the results, the crossing path for the problem is 0- 1 -2 – 6 - 7, and the makespan, as the total completion time is 20.

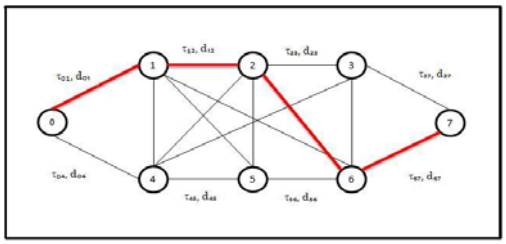

Figure 1. The graphic representation for the result of a 2/ 3/ *G*/ C_{max}job shop problem.

2.1 The Mathematical Model

The mathematical formulation for the job shop with the objective of minimizing makespan are presented below:
Objective:

$$\text{Minimize}\{\text{Max}\ (C_{1j},\ C_{2j},\ C_{3j}\ ,\ \ldots\ ,\ C_{ij})\} \tag{4}$$

Subject to:

$$C_{ij} - S_{ij} - t_{ij} = 0 \qquad \forall(i;\ j) \tag{5}$$

$$C_{i'j'} - C_{ij} + H(1 - \gamma_{ij}) \geq ti'j',\ \forall(i,\ j),\ (i',\ j') : Oij \in N_k\ ,\ Oi'j' \in N_k \tag{6}$$

$$C_{ij} - C_{i'j} + K(1 - \gamma_{ij}) \geq tij,\ \forall(i,\ j),\ (i',\ j') : Oij \in N_k\ ,\ Oi'j' \in N_k \tag{7}$$

$$S_{ij}\ , \geq 0\ \forall i;\ j \tag{8}$$

$$Sij + 1 - Cij \geq 0,\ \forall i,\ j = 1,\ \ldots\ ,\ Ji - 1 \tag{9}$$

$$\gamma = \begin{cases} 1, & \text{if operation} O_{ij}\ \text{precedes} O_{i'j'} \\ 0, & \text{Otherwise} \end{cases} \tag{10}$$

The above model detail the assumptions for the JSP. In Equation 4, let S_{ij}be as the starting point, while $C_{ij}$is the completion time job *i*. While *H* and *K* is assumed as very large positive integer, $N_k$is the set of operations O_{ij}that can be loaded on machine *k* and $\gamma$is a decision variable that generates a sequence betweenthe operations *Oij* and *Oi'j'* . The constraint sets in Equation 3, imposes that thedifference between the completion time and the starting time of an operation is equalto its processing time. This constraint satisfies the assumption that once an operationhas started, it cannot be pre-empted until its completion. Constraint sets in Equation 4and Equation 5 are to ensure that no two operations can be processed simultaneouslyon the same machine. The disjunctive constraint in equation 4, becomes inactivewhen γ= 0 and the disjunctive constraint in Equation 5 becomes inactive whenγ = 1. Constraint set in Equation 6, are to ensures that the start time of an operationis always positive. Constraint set in Equation 7 represents the precedence relationshipamong various operations of a job.

Some Properties are :

Property 1: An optimal schedule exists if no idle time between any consecutive jobs for scheduling jobs on a single machine with common due dates.

Property 2: For the optimal schedule, V-shaped property exists around the common due date. This means that jobs completed before or on the common due date *d* are scheduled in non-increasing order of the ratios λ_i/α_i, and jobs starting on or after *d* are scheduled in non-decreasing order of the ratios λ_i/α_i in an optimal schedule.

Property 3: In an optimal schedule, either the first job starts at time zero or the completion time of one job coincides with the common due date *d*.

3 The Methodology

The ant colony optimization was inspired by a colony of real ants in an experiment conducted by Goss *et al*. A laboratory colony is given access to a food source in an arena linked to the colony's nest by a bridge with two branches of different length (see Figure 2).

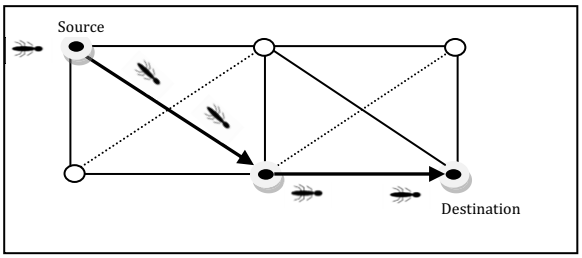

Figure 2. Experiment for the bridge experiment.

Figure 2, also shows how the ants move to on each arc *(i,j)* of the graph, that is associated with a variable τ_{ij} called *artificial pheromone trail*. Pheromone trails are read and written by ants. The amount (intensity) of pheromone trial is proportional to the utility, as estimated by the number of ants using that arc to builds good solutions. Each ant applies a step-by-step constructive decision policy to build problem's solution. At each node local information, maintained on the node itself and/or on its outgoing arcs, is used in a stochastic way to decide the node to move on. The decision rule of an ant k located in node i uses the pheromone trials τ_{ij} to compute the probability with which it should choose node $j \in N_i$ as the next node to move to, where N_i is the set of one-step neighbors of node i:

$$p_{ij}^{k} = \begin{cases} \tau_{ij} & \text{if } j \in N_i \\ 0 & \text{if } j \notin N_i \end{cases} \tag{11}$$

While building a solution ants deposit pheromone information on the arcs they use. In ACO, ants deposit a constant amount $\Delta\tau$ of pheromone. Consider an ant that at time t moves from node i to node j. It will change the pheromone value τ_{ij} as follows:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau \tag{12}$$

Using this rule, which simulates real ants' pheromone depositing on arc (I, j), an ant using the arc-connecting node i to node j increases the probability that ants will use the same arc in the future. As in the case of real ants, autocatalysis and different path length are at work to favor the emergence of short paths.We have to write the data in a more compact form to tackle this job shop problem.

4 ACO Implementation

In this flowchart, we are adding two dummy nodes which classify the start and finish node; as the starting node and the completion of the overall job shop. Thus, now we have the starting point and the destination in the ACO. This flowchart is define as the Ant Colony algorithm. At the beginning of the algorithm, all parameters are initialized. There are *m* ants in the colonies that are placed at the starting points, and one iteration is completed when all ants has arrived at the destination. The local pheromone update when an ant chooses an edge to go on. The global pheromone updating is performed when the whole ant colony reached the destination. Those step influences the next iteration

of ant colonies finding routes. The algorithm stopped when the maximum number of iterations reached, and the minimum makespan of the job shop is found.

In implementing the model developed, this study explores a real problem of JSP in a Metal Company. This Metal Company involved in the covers product design and development, electrical and electronic enclosure as well as industrial equipment for workshop, warehouse and factory. This company was a premier supplier for the metal components.Data given the general information for 20 parts of Customised E07 Rack which is commonly used to store computers for industrial production.

Initialization: The pheromone trails, the heuristic information and the parameters are initialized
Iterative loop:

2.1 A colony of ants determines starting jobs.
2.2 Construct a complete schedule for each ant:
Repeat
Apply state transition rule to select the next processing job
Apply the local updating rule
Until a complete schedule is constructed
2.3 Apply local search process
2.4 Apply the global updating rule

Cycle: If the maximum number of iterations is realized, then STOP;
Else go to step 2.

5. Results and Analysis

The objective of our study is to determine the minimum makespan, using the ACO. Based on the ACO method, a computerised system was develop and built using the Visual Basic programming language. Several trial run on the system shows that it is very efficient for small size problem but it reduces the average as the problem sizes involves. Table 2 shows the different process, jobs, makespan and the sequence in the JSP.

Table 2. The Results for Makespan Objective for Customized E07 rack.

Process	Total of Jobs	Makespan(seconds)	Machine	Job Sequence
CNC Machining	15	600	1 2 3 4 5 6	1, 7, 13 2, 8, 14 3, 9, 15 4, 10 5, 11 6, 12
Bending	13	104	1 2 3 4 5 6	1, 7, 13 2, 8 3, 9 4, 10 5, 11 6, 12
Clinching	3	72	1 2 6	1 2 3
Tapping	2	40	3 5	1 2
Power Press	5	25	2 3 4 5 6	5 4 3 2 1

For instance, we analyse all the results that receive. We do through each the results on minimizing the makespan. Hence, there are several ways to estimate the running time of a program. To simplify the analysis, we will adopt the convention that there are no particular units of time. Thus, we throw away leading constants. We will also throw away low-order terms, so what we are essentially doing is computing a long running time. Since this is the long running time, we must be careful, never to underestimate the running time of the program. In fact, the answer provided is a guarantee that the program will terminate within a certain time period. The program may stop earlier than this, but never later[7].

Since, we had the heuristic method for the algorithm, there was the parameters that should be aware. We should do some parameter adjustment for the α, β, and ρ that we put in the algorithm. The underlying principle of this heuristic method is that a job with higher total processing time should be given higher priority than a job with less total processing time [8].We should increasing and decreasing the parameters, rather than just to follow the mathematical formulation can also be considered in order to make dynamic so that it is more powerful and can be applied to any range of set of data. Lastly, this implement of the program are very friendly user. However, the best way of balancing these features differed from problem to problem [9]. This program can be use, even there are different case study, which we discuss in this study. Future research for other case study should be done and compiling some improvement of the program.

References

1. M. Dorigo, L.M. Gambardella (1997). *Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Transactions on Evolutionary Computation 1,* page 53–66.
2. Chang, R.-S., Chang, J.-S. and Lin, P.-S. An ant algorithm for balanced job scheduling in grids. *Future Generation Computer Systems*, 2009. URL http://www.elsevier.com/locate/fgcs.
3. E. Salari, K. Eshghi (2005). *An ACO algorithm for graph coloring problem, in: Congress on Computational Intelligence Methods and Applications,* page 15–17.
4. Xiaoxia Zhang, Lixin Tang (2005) *CT-ACO hybridizing ant colony optimization with cycle transfer search for the vehicle routing problem, in: Congress on Computational Intelligence Methods and Applications*, page 15–17.
5. WIKIPEDIA, Website "http://en.wikipedia.org/wiki/Job_shop" Retrieved on 12 September 2008.
6. J.A. Hoogeveen, S.L. Van De Velde, (1991): *Scheduling Around a Small Common Due Date. European Journal and Operational Research 55*, page 237–242.
7. Z. Ismail, D. Abd Mahad (2009). *Ant Colony Optimization for minimizing makespan for job shop problem.* Simposium Kebangsaan Sains ke-17. Melaka.
8. Betul Yagmahan, Mehmet Mutlu Yenisey. (2008). *Ant Colony Optimization forMulti-objective Flow Shop Scheduling Problem.* Turkey.
9. Kathryn A. and Dowslanda. (2008). *An improved ant colony optimisation heuristic for graph coloring.*