Proceedings of the 2005 International Conference on Simulation and Modeling
V. Kachitvichyanukul, U. Purintrapiban, P. Utayopas, eds.

HETEROGENOUS ANT ALGORITHM FOR JOB SHOP SCHEDULING

Apinanthana Udomsakdigool and Voratas Kachitvichyanukul

Industrial Engineering & Management
School of Advanced Technologies
Asian Institute of Technology
P.O. Box 4, Klong Luang, Pathumthani 12120, Thailand

ABSTRACT

Ant Colony Optimization (ACO) is a metaheuristic which has been widely used recently to solve many combinatorial optimization problems. In the literature, several successful applications have been reported for Vehicle Routing and Traveling Salesman Problems. In this paper, an ant algorithm is proposed for the Job-shop scheduling Problems (JSP). The objective is to find the schedule that minimizes the makespan. To implement the ACO for the JSP, the original problem is first transformed into a graph based model. Ant algorithm is then developed with several specific features. One important feature is that the ants in the colony are heterogeneous. The performance of the proposed algorithm is tested using the benchmark problems available in OR-Library. The computational results are reported and the possibilities for future studies are suggested.

1 INTRODUCTION

The Job-shop Scheduling Problem (JSP) is one of the shop scheduling problems which is a simplified model of scheduling problems often occurring in the industry. In general, JSP is the NP-hard combinatorial optimization problem where the algorithm would require a number of computational steps that grows exponentially with the problem size. Although optimal solutions of JSP can be obtained via enumeration techniques such as branch and bound method and mathematical programming, these methods may take a prohibitive amount of computation even for moderate size problems. For practical purpose, an efficient methods which can generate good solutions in an acceptable time is needed.

In the past decade, many researchers have studied metaheuristics to solve JSP. Several metaheuristics such as genetic algorithm (Wang and Zheng, 2001; Varela, Vela, Puente, and Gomez, 2003), Simulated Annealing (Kolonko, 1999; Steinhöfel, Albrecht, and Wong, 2002), and Tabu Search (Nowicki and Smutinicki,1996; Pezzella and Merelli, 2000) have been proposed to deal with the JSP and shows that these methods can obtain good results. More recently, Ant Colony Optimization (ACO) has been receiving the extensive attention due to its successful applications to many combinatorial optimization problems (Dorigo, Di Caro, and Gambardella, 1999). However, the application to shop scheduling problem especially JSP has proved to be quite difficult. The earliest ACO algorithm solving the JSP was done by Dorigo et.al (1994). However their algorithm was far from reaching state-of-the-art performance. In this paper an ant algorithm is developed with special feature that the ants in the colony is not homogenous, i.e., each ant possesses different capabilities. The potential of the proposed algorithm is investigated by solving the benchmark instances available in OR-Library (1990).

The remainders of this paper are organized as follows. In the next section, a formal description of the scheduling problem and a graph-based representation to facilitate the development of ACO systems are presented. Section 3 describes the fundamental concept and the features incorporated in the proposed ACO. Computational results on Lawrence's benchmark problems are discussed in section 4. Finally the conclusion and discussion are presented in section 5.

2 THE PROBLEM DEFINITION

The job shop scheduling (Jain, and Meeran, 1999), denoted $n/m/G/C_{\max}$ consists of a set J of n jobs $\{J\}_{i=1}^{n}$ to be processed on a set M of m machines $\{M_j\}_{j=1}^{m}$. O_{ij} is the operation of job J_i which has to be processed on machine M_j for an uninterrupted processing period p_{ij}. Each job J_i consists of a chain of operations $\{O_{ij}\}_{j=1}^{m}$ which represents the predetermined order of job J_i through the machines (precedence constraint). Each machine can process only one job and each job can be processed by only one machine at a time (capacity constraint). In this paper no pre-emption is allowed. The duration in which all operations for all jobs are completed is referred to as the makespan, C_{max}. The ob-

jective considered is to determine the starting time, *Sij* for each operation in order to minimize the makespan while satisfying all precedence and capacity constraints as given in equation 1.

$$\min C_{max} = \min\{\max(S_{ij}+p_{ij}): \forall J_i \in J, \ \forall M_j \in M\} \quad (1)$$
Feasible schedule

3 ACO APPROACH

3.1 Basic of ACO

The ACO algorithm is a kind of algorithm inspired by the natural foraging behavior of real ants. This behavior is the basis for local interaction of a simple agent (ant) which leads to the emergence of shortest path. While walking from food sources to the nest and vice versa, ants deposit a substance called pheromone on the ground. The amount of pheromone in a path is used to guide the colony during the search. The ants select a path via a probabilistic decision biased by the amount of pheromone, the path with strongest the pheromone value will be chosen with the highest the probability. With the continuous action of the colony, the shortest paths are more frequently visited, receiving a higher amount of pheromone and thereby becoming more attractive for the subsequent ants. In contrast, the longest paths are less visited and together with the pheromone evaporation process, these paths are less attractive for subsequent ants. Hence the final path emerges from the global cooperation among all ants in the colony (Dorigo, Maniezzo, and Colorni, 1996; Dorigo, Di Caro, and Gambarella, 1999; and Dorigo, Bonabeau, and Theraulaz, 2000).

In ACO algorithm, artificial ants build solution by applying a probabilistic decision policy to move from present state to adjacent state (local move). The decision policy is a function of both prior information represented by the problem specification and the local modifications in pheromone trails induced by the past ants. After they complete their paths, the pheromone values are released depending on the quality of solutions, the shorter the distance the stronger the pheromone value. After the ants repeat this procedure for certain number of iterations, a solution will emerge from the corporation of ants. That is the path that has strongest pheromone value will become the dominant solution. The main components in ACO algorithm are how to construct the solution and how to communicate their experiences to the others.

3.2 Implementation ACO in JSP

The $n/m/G/C_{\max}$ problem can be represented in ACO approach via a disjunction graph (Dorigo, Maniezzo, and Colorni, 1996) as depicted in Figure 1. Given an instance of $n/m/G/C_{\max}$ it can be associated with a disjunctive graph $G = (O,C,D)$, where O is the set of all nodes (processing operations), C is a set of conjunctive directed arcs, and D is a set of disjunctive undirected arcs. C corresponds to the precedence relationship between operations of a single job thus operations belonging to the same job are connected in sequence. D represents the machine constraint of operations belonging to different jobs. The operations of jobs which processed on the same machine are pair-wise connected in both directions. The source and sink operation are the dummy operations of nest and food source respectively. The source operation has conjunctive directed arcs emanating to the first operation of the n jobs and the sink operation has conjunctive directed arcs coming from the last operation of the n jobs. A path P is defined as a sequence of total operations. For simplicity, all arrows of node that are pair wise connected in both direction are omitted in this figure. The arc between nodes is weighted by a pair of numbers (τ, η). The first is the pheromone level on the trail while the second is the visibility which is computed from some problem specific heuristics such as shortest processing time or shortest remaining time. All ants are initially at the sink operation and later on at each step they have to identify a feasible permutation of the remaining operations.

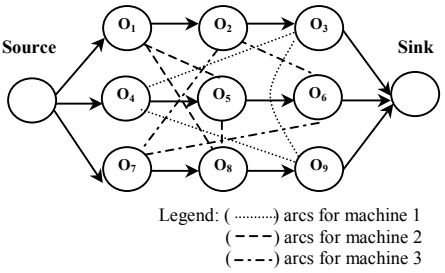

Figure 1: The disjunctive graph of *3/3/G/C_{max} problem*

To cope with this problem, ants construct solutions by choosing the operations according to a probabilistic state transition rule. The selected operation is then added in the visit list and the process is iterated until the last operation is chosen. Once all the ants have completed their solution a global pheromone updating rule is applied. The visit list can determine the sequence of operations. Now, it is possible to calculate the makespan of feasible solution by job processing time subject to precedence and machine constraint.

3.3 Features of ant algorithm

Several specific features is introduced in the proposed algorithm to improve the search performance. This includes the strategy of pheromone initialization and the ants are heterogeneous. These ants have different properties which are characterized by the types of heuristic information. The

detail of all features in ant algorithm will be described below. The algorithmic framework is depicted in Figure 2.

3.3.1 Initial pheromone value

Unlike the strategies use by the previous researches (Colorni et.al, 1994; T'kindt et.al, 2002; and Ying and Liao, 2004), where each path the pheromone value τ_{ij} is given by a constant, in the proposed method the pheromone value on each path is initialized with the value drawn from a random number within the interval(0,1). The reason behind that is to enforce the diversification at the beginning of the resolution. The lower bound of τ_{ij} is set to a small positive constant in order to prevent the algorithm from prematurely converging to a local solution.

3.3.2 Construct a solution

In the search mechanism, an artificial ant constructs a sequence of operations by performing the construction steps as shown in Figure 2. From the empty visit list, ant k selects an operation from the candidate list that is the subset of unvisited list according to the two step probability transition rule given by the following equation (Dorigo,1997).

$$j^k(t) = \begin{cases} \arg\left\{ \max_{u \in C^k(i)} \left[\tau_{iu}^k(t)\right]^\alpha \left[\eta_{iu}^k\right]^\beta \right. & \text{If } q \le q_o, \\ J & \text{Otherwise,} \end{cases} \quad (2)$$

where τ_{iu} is the pheromone trial between operation i *and* operation u, η_{iu} is the heuristic information between operation i and operation u, $C^k(i)$ is the set of operations that remain to be visited by ant k positioned on operation i and make the solution feasible. Also, α and β are the parameters which determine the relative important of pheromone trial and heuristic information (α>0, and β>0), q is the random number uniformly distributed in [0,1] and $q_0 \in (0,1)$ is the parameter which determines the relative important between exploitation and exploration. In addition J is an operation selected from the random proportional transition rule defined as equation 3.

$$p_{ij}^k(t) = \begin{cases} \dfrac{\left[\tau_{ij}^k(t)\right]^\alpha \left[\eta_{ij}^k\right]^\beta}{\sum_{u \in C^k(i)} \left[\tau_{iu}^k(t)\right]^\alpha \left[\eta_{iu}^k\right]^\beta} & \text{If } j \in C^k(i) \\ 0 & \text{Otherwise,} \end{cases} \quad (3)$$

Input: A problem instance
Set Termination Criteria
Set Max Iteration t_{max} = 1000
Set Algorithm Parameters
  Number of ants N = number of total operations
  Pheromone information weight α = 1
  Heuristic information weight β = 2
  Intensify rate ρ = 0.1
  Exploitation weight q_0 = 0.9
  Restart number = 100
//Initialize pheromone value; τ_{ij}
For each edge of operation *(i,j)* **do**
  if pair of Operation*(i,j)* *O(i,j)* which *Oi* and *Oj* are belong to the same job, *Oi* ≺ *Oj* and *Oi* ≠ *Oj*, set τ_{ij}= *random(0,1)*
  if pair of *O(i,j)* which *Oi* is belong to job *k* and *Oj* is belong to job *l* and job *k* ≠ job *l,* set τ_{ij} = *random(0,1)*
  otherwise set τ_{ij}= 0
End for
Set Global best C_{max}, $GB = \phi$
// Main loop
Set Global C_{max}, $G = \phi$
While terminating criteria is not met **do**
  Set Good ant list, $B = \phi$
  // 1. Construct solution
  Set Unvisited list, $U = \forall O$
  Set Visit list, $V = \phi$
  Set Candidate list, $C = \phi$
  For ant k = 1 to N **do**
    Place ant k randomly on the operations that can be started.
    Keep this information in V_k
    Ant build a tour step by step until $U_k = \phi$ by apply the following steps: (repeat until the U_k is empty)
      Choose the next operation j from C_k according to Eq.(2) and Eq.(3)
      Keep operation j in V_k and delete operation j in U_k
    Compute the makespan C_{max} of the sequence in V_k by applying Eq.(1)
  End for
  For ant k = 1 to m **do**
    Sort N ants of iteration i and keep some good ants in B_i
  End for
  // 2. Update pheromone trial
  For each edge *(i,j)* do
    Ants in B_i update pheromone trials according to the Eq.(3)
  End for
  Empty B_i , U_k , V_k , C_k
  Update G_i
  If the restart criteria is reached
    Update *GB*
    Apply restart process
  End if
End while
Output the global best makespan

Figure 2: An ant algorithm to solve JSP

While building the solution, an ant chooses the operation by combining the pheromone information and heuristic information. Every time an ant in operation i has to select operation j, it samples a random number q. If $q \leq q_0$, the operation according to equation 2 is chosen. Otherwise an operation is selected according to equation 3. When an ant selects the operation, that operation is moved from unvisited list to the visited list. The procedure is repeated until all operations are selected. The sequence of all operations in the visited list is the total order of operations that induces a total order of each job and of each machine. This unambiguously defines a solution to the problem.

Like genetic algorithms, ACO use multiple agents which has its individual decision made based upon the local information or knowledge along with the sharing of information with others in order to achieve the desired kind of collaboration. More precisely, ant colony uses coordination methods to make global information locally available. Ants in colony may be homogenous or heterogeneous and may have common goals. Most previous researches contain homogenous population of ants (Colorni et.al, 1994; T'kindt et.al, 2002; and Ying and Liao, 2004). In this proposed algorithm the ants in the colony are not homogenous. Each ant may have various capability to detect different kinds of heuristic information which are used to guide their search. This is to force the ants to start their search in the different center of search space and hope to increase the chance to find the good solution quickly. The types of heuristic information used in the proposed algorithm are shown in Table 1.

Table 1: Types of heuristic information

Rule	Description
SPT	an operation with shortest processing time
LPT	an operation with longest processing time
SRT	an operation with shortest remaining processing time
LRT	an operation with longest remaining processing time
SMT	an operation with smallest value obtained by multiplying processing time with total processing time
LMT	an operation with largest value obtained by multiplying processing time with total processing time

3.3.3 Pheromone Updating

After all ants complete their solution, global updating is performed. After all N ants have generated their solutions, the ants are sorted by the makespan, and some good ants are permitted to update the pheromone values. For a selected solutions of ants $ant_{r=1}^{m}$, the contribution of those solutions used to update the pheromone values of colony are normalized. The rule of pheromone updating is defined by the following equation.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}^{r}(t), \qquad (4)$$

where

$$\Delta\tau_{ij}^{r}(t) = \begin{cases} \dfrac{1/C_{\max}^{r}}{\sum_{r=1}^{m}(1/C_{\max}^{r})} & \text{If (i,j)} \in \text{tour of } ant \\ 0 & \text{Otherwise} \end{cases}$$

where $\rho \in [0,1)$ is the pheromone evaporating parameter and $C_{\max}^{r}$ is the makespan of ant_r. The evaporating process is the mechanism which allows ant to slowly forget its history and avoids locally convergence and allow more exploration in the search space.

3.3.4 Restart

When the system is stuck in an area of the search space a restart is performed. All pheromone values in every path are deleted and the algorithm is started again. The reason for restart is to diversify to find new, possibly better, solution in other search space.

3.3.5 Parameter setting

The parameters which control the search are set according to the previous studies of Dorigo (1997): α=1, β=2, ρ=0.1, and q_0= 0.9. The number of ants is set to the number of total operations and the number of ants used to update the pheromone values is equal to the number of jobs. When the ants can not improve the global solution of colony after 100 iterations, a restart is initiated. The algorithm terminates when the total number of iterations reaches 1000.

4 EXPERIMENTAL RESULTS

The proposed ant algorithm was tested on Lawrence's benchmark problems which are available from the OR-Library (1990). The dimension of problems is 10 jobs and 5 machines. The algorithm was coded in Borland builder 6.0 and run on an Pentium 4 1.7 GHz PC with 256 MB RAM. To evaluate the algorithm, each of problem instances was executed for ten trials in order to ensure that the random generator did not bear any influences on the quality of results obtained. The final results are listed in Tables 2 and 3. The results in Table 2 are obtained from algorithm A where all ants use the same heuristic information(SPT). The results in Table 3 are obtained from algorithm B where the ants use different types of heuristic information as listed in Table 1. The best solution is obtained from ten trails of this tested algorithm. The average and the standard deviation of solutions across ten trials are reported. The percentage offset from the optimal is calculated as: % offset from the optimal solution = [(best solution-optimal solution)/optimal solution]x100.

Table 2: Results of ten benchmark problems obtained from algorithm A

Problem instance[a]	Optimal solution	Best solution	Average	Standard deviation	%offset
LA01	666	743	769.56	13.92	11.56
LA02	655	844	863.33	17.72	28.85
LA03	597	710	727.11	9.53	18.93
LA04	590	706	719.11	9.30	19.66
LA05	593	599	611.22	9.37	1.01

[a] LA01-LA05 are the instances of 10 jobs and 5 machines.

Table 3: Results of ten benchmark problems obtained from algorithm B

Problem instance[a]	Optimal solution	Best solution	Average	Standard deviation	%offset
LA01	666	712	734.89	10.09	6.91
LA02	655	807	832.11	16.95	23.21
LA03	597	685	707.56	10.76	14.74
LA04	590	678	699.56	11.75	14.92
LA05	593	593	598.56	9.03	0.00

[a] LA01-LA05 are the instances of 10 jobs and 5 machines.

Table 2 and 3 illustrated that the averaged solution obtained from algorithm B is always better than averaged solution obtained from algorithm A. In Table 3 the algorithm finds the best solution of LA01, LA03, LA04, and LA05 within 12% of the optimal value and the standard deviation of solution is nearly the same except for the LA02 in both cases.

5 CONCLUSION AND DISSCUSSION

In this paper, an ACO algorithm is proposed to solve the JSP which is one type of NP-hard . To adapt the ACO in the JSP, the scheduling problem is transformed into the graph based model like a Traveling Salesman Problem (TSP). The strategies included are the randomized initial pheromone to force the ants to diversify their search and starting the search process at the different centers of search space. These strategies help to diversify the search and as a results, with the support of statistical test, yield better solutions. The solutions for the problems which has the same instance are quite consistent; however the reliability of the algorithm should be evaluated by further statistical analysis. Since the CPU time will vary according to hardware, software and the existing research on JSP normally does not provided comparable CPU time information, comparison in terms of CPU time is omitted in this paper. Nevertheless the proposed algorithm provide the solution with a reasonable computing time.

There are many possibilities for improvement of the proposed algorithm. Firstly, the solution for JSP by the ant algorithm is quite far from the optimal solution. For example, in Colorni(1994) and in the proposed algorithm the best solutions are far from optimal solutions around 10% and 12% respectively. One reason for that is that the probabilistic model may be biased by the heuristic information. To improve the performance of the algorithm, other construction mechanisms should be considered. Secondly, ant algorithm is a population-based method which is better at identifying the promising area than the nonpopulation-based method. However the weakness of population-based method is the high likelihood to miss good solution. Hybridization with other heuristics in order to become a more powerful search tool is another area of interest. Finally, ant algorithms usually have parameters to control the search. The values of such parameters affect the quality of solution. Therefore a variety of parameter settings should be fully examined in order to yield good performance.

REFERENCES

Colorni A., Dorigo M., Maniezzo V. & Trubian M. 1994. Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science, 34(1): 39-54.

Dorigo M., Bonabeau E., & Theraulaz G. 2000. Ant algorithms and stigmergy. Future Generation Computer Systems, 16: 851- 871.

Dorigo M., Di Caro G., & Gambarella L.M. 1999. Ants Algorithm for Discrete Optimization. Artificial Life, 5: 137-172.

Dorigo M, Gambardella LM. 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem.IEEE Transactions on Evolutionary Computation,1: 53–66.

Dorigo M., Maniezzo V., & Colorni A. 1996. Ant system: Optimization by a colony of cooperating agents. IEEE transactions on systems, Man and Cybernetics – Part B, 26(1): 29- 41.

Jain A.S. & Meeran S. 1999. Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research 113: 390-434.

Kolonko M. 1999. Some new results on simulated annealing applied to the job shop scheduling problem. European Journal of Operational Research, 113: 123-136.

Nowicki E. & Smutinicki C. 1996. A fast tabu search algorithm for the job-shop problem, Management Science, 42(6): 797-813.

OR-Library.1990. http://www.brunel.ac.uk/depts/ma/research/jeb/info.html

Pezzella F. & Merelli E. 2000. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. European Journal of Operational Research, 120: 297-310.

Steinhöfel K., Albrecht A., & Wong C.K. 1999. Fast parallel heuristics for the job shop scheduling problem. European Journal of Operational Research, 118: 524-548.

T'kindt V., Monmarché N., Tercinet F., & Laügt D. 2002. An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. European Journal of Operational Research 142: 250-257.

Varela R., Vela C.R., Puente J., & Gomez A. 2003. A knowledge-based evolutionary strategy for scheduling problems with bottlenecks, European Journal of Operational Research, 145: 57-71.

Wang L. & Zheng D.Z. 2001. An effective hybrid optimization strategy for job-shop scheduling problem. Computers and Operations Research, 28(1): 585-596.

Ying K.C. & Liao C.J. 2004. An ant colony system for permutation flow-shop sequencing. Computers and Operations Research, 31(5): 791-801.

AUTHOR BIOGRAPHIES

APINANTHANA UDOMSAKDIGOOL is a lecturer of King Mongkut's Institute of Technology North Bangkok, Thailand. She received her M.Eng. in Industrial Engineering from Chulalongkorn University in 1998. Currently, she is a Doctoral candidate of the School of Advanced Technologies, Asian Institute of Technology, Thailand. Her research interests include Metaheuristics for Combinatorial Optimization Problems, Swarm Intelligence, and Self-organizing Systems. At present, her research work is focused on the Ant Colony Optimization for solving the scheduling problem. Her email address is <Apinanthana.U@ait.ac.th>

VORATAS KACHITVICHYANUKUL is an Associate Professor in the Industrial Engineering & Management field of study, School of Advanced Technologies, Asian Institute of Technology, Thailand. He received a Ph.D. from the School of Industrial Engineering at Purdue University in 1982. He has extensive experiences in simulation modeling of manufacturing systems. He had worked for FORTUNE 500 Companies such as Compaq Computer Corporation and Motorola Incorporated. He had also worked for SEMATECH as technical coordinator of the future factory program. His teaching and research interests include planning and scheduling, high performance computing and applied operations research with special emphasis on industrial systems. His email address is <voratas@ait.ac.th>