A Modified Ant Colony Algorithm for the Job Shop Scheduling Problem to Minimize Makespan

Zhiqiang Zhang, Jing Zhang
Faculty of Computer Science and Engineering
Xi'an University of Technology
Xi'an, China
chinazzq@126.com

Shujuan Li
Faculty of Mechanical and Precision Instrument Engineering
Xi'an University of Technology
Xi'an, China

***Abstract*— This paper presents a modified Ant Colony Optimization (ACO) algorithm for the Job Shop Scheduling Problem (JSSP) with makespan criterion. The traditional ACO algorithms can be simplified with the elimination of pheromone unimportant to the JSSP solution. Also, this paper suggests a new priority rule served as the heuristic information of the proposed algorithm. In order to improve the convergence and solution qualities of the proposed algorithm, the local search procedure based on the neighborhood of the JSSP is introduced. The experimental results indicate that the modified ACO algorithm in this paper is more concise and more effective than ACO algorithms. Furthermore, this paper discloses the existing problems of ACO algorithms for the JSSP.**

Keywords-Job shop scheduling problem; Ant Colony Optimization; Priority rule; Local search.

I. INTRODUCTION

Production Scheduling is the most urgent and realistic problem to be solved in modern manufacturing industry, which is closely concerned with both the production efficiency and economic returns of enterprise. The job shop scheduling problem (JSSP)[1] is an important combinational optimization problem in computer science and operations research, which can be simply described as follows: given n jobs, each composed of several operations that must be processed on m machines. Each operation uses one of the m machines for a fixed duration. Each machine can process at most one operation at a time and once an operation initiates processing on a given machine it must complete processing on that machine without interruption. The operations of a given job must be processed in a given order. The problem consists in finding a sequence of the operations on the machines, taking into account the precedence constraints, which minimizes the makespan (C_{max}), that is, the finish time of the last operation completed in the sequence. Also, the JSSP is NP-complete and has proven to be computationally challenging. The JSSP can be solved by various local search algorithms or metaheuristics in the main including Tabu Search [2, 3], Genetic Algorithms [4, 5] and Simulated Annealing [6, 7], etc. In general, Tabu search [3] is the best scheduling algorithm so far.

In recent years, with the development of research and application of Ant Colony Optimization (ACO) algorithms [8], some researchers have proposed various ACO algorithms for the JSSP, such as Ant System (AS) [9, 10], Ant Colony System (ACS) [11, 12] and Min-Max Ant System (MMAS) [13, 14], etc. Being different from the traditional ACO algorithms, this paper suggests a modified ACO algorithm, whose main thought is to eliminate the pheromone actually having little effect upon the ACO algorithms for the JSSP. This paper presents a new priority rule served as the heuristic information of the proposed algorithm. In order to improve the convergence and solution qualities of the proposed algorithm, the local search procedure based on the neighborhood structure [3] is introduced. The experimental results from many standard JSSP instances indicate that the proposed algorithm is effective and superior to the traditional ACO algorithms.

II. MODIFIED ACO ALGORITHM FOR JSSP

Essentially, local search algorithms are usually based on the iterative exploration of neighborhoods of solutions trying to improve the current solution by local changes. The types of local changes that may be applied to a solution are defined by a neighborhood structure. The choice of an appropriate neighborhood structure is crucial for the performance of a local search algorithm and is problem-specific. Typically, a neighborhood structure is implicitly defined as the possible local changes that can be applied to a solution, and not the set of all possible neighbors.

Being different from local search algorithms, ACO algorithms belong to constructive method and build a solution to a combinatorial optimization problem in an incremental way. Step by step and without backtracking, ACO algorithms add solution components until a complete solution is generated, and usually some kind of heuristic rule is employed [8].

A. Basic Principle

Assuming some JSSP includes 2 jobs and 3 machines, their processing orders and processing times are expressed as: processing orders= $\begin{bmatrix} 1 & 3 & 2 \\ 3 & 1 & 2 \end{bmatrix}$ and processing times= $\begin{bmatrix} 2 & 4 & 1 \\ 3 & 4 & 5 \end{bmatrix}$, respectively. In order to be convenient, the 6 operations can be numbered, whose method is the line (jobs) first and then the column (machines). That is, the operation numbers= $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$. Therefore, ACO algorithms are applied to the JSSP, whose basic procedure can be: the precedence constraint can be realized by the tabu list of ACO algorithms so as to guarantee the generation of feasible solution. It is decided via ACO

978-1-4244-7739-5/10/$26.00 ©2010 IEEE

algorithms that which operation number should be incorporated into the sequencing results one by one. At the same time, in accordance with the following method, the starting time (also the finishing time) of operation *j* can be calculated each time:

1) If the machine needed by operation *j* is occupied, and operation *j* isn't the first one, the starting time of operation *j*= the greater between the machine releasing time and the finishing time of the predecessor operation of operation *j*.

2) If the machine needed by operation *j* is occupied, and operation *j* is the first one, the starting time of operation *j* = the machine releasing time.

3) If the machine needed by operation *j* is idle, and operation *j* isn't the first one, the starting time of operation *j*=the finishing time of predecessor operation of operation *j*.

4) If the machine needed by operation *j* is idle, and operation *j* is the first one, the starting time of operation *j* = 0.

The procedure should be repeated until all operation numbers are incorporated into the sequencing results. Thus, after the starting times and the finishing times of all operation have been determined, ACO algorithms will find some scheduling solutions to the JSSP.

Assuming that the sequencing result found by ACO algorithms is (1, 4, 2, 5, 3, 6), it indicates: the No.1 operation should be firstly arranged, and in closely following, the No.4 operation is arranged until the No.6 operation is arranged at last. Also, it can be determined that: the starting time of the No.1 operation and the No.4 operation=0, the starting time of the No.2 operation and the No.5 operation=3, the starting time of the No.3 operation=7, the starting time of the No.6 operation=8, and the makespan= 8+5=13. It can be known that the critical path is (4, 2, 3, 6), and the length of critical path=13 (i.e. makespan). Since all the adjacent operations on the same machine on critical path can form the blocks, there are (4, 2) and (3, 6) two blocks on the critical path.

B. Modification of ACO Algorithms

Theoretical research on ACO algorithms has indicated that [8]: if the initial value of pheromone is so large that when the pheromone is reduced to small enough, the pheromone released by ants can bring its guiding role into full play. If, vice versa, the search zone will quickly concentrate on the initially produced and limited solution space by ants. For this reason, the initial value of pheromone in ACO algorithms should be very small and combines with a very low evaporation rate in finding the problem solution. However, in fact, we have found that the pheromone has a very small effect upon the function of ACO algorithms for the JSSP. Accordingly, the analysis can be made in the following:

It is well known that ACO algorithms take inspiration from the foraging behavior of some ant species. The foraging procedure is very similar to the TSP, for the both are corresponding to a fully connected graph: the symmetric TSP is completely corresponding to a fully connected undirected graph, and the non-symmetric TSP is completely corresponding to a fully connected directed graph. Essentially speaking, ACO algorithms can build the solutions of combinatorial optimization problems via random travelling on a fully connected graph. The nodes in the graph represent the components of problem (e.g. TSP) solution, and the connection lines or edges between the nodes indicate that there exist some relationships between the nodes (e.g. reachability). Also, the pheromone concentration on the edges with closer relationship is much greater. For this reason, this paper holds that finding solutions to the TSP by ACO algorithms is bound to win success, whereby illustrating that ACO algorithms are very suitable for the combined optimization problem with the fully connected graph model. Nevertheless, the JSSP is much more complicated than the TSP. If graph is used to describe the JSSP, a partly connected directed graph will be obtained, in which there are many nodes (i.e. operations) without any relationships, thus leading to there being no pheromone on the corresponding edges. As a result, when the solution to the JSSP is found, pheromone functions will be weakened, whereby resulting in a poor solution found by ACO algorithms.

Accordingly, based on ACO algorithms, this paper omits the calculation steps and parameters related pheromone in such a way that it is favorable for lowering the difficulty in programming and algorithm parameter setting. Since local search is a particular type of daemon action of ACO algorithms, incorporating the local search procedure [3] can remedy the losses to the convergence of algorithm in this paper, by means of which the quality of solution found by the proposed algorithm can be improved.

C. Heuristic Information of Modified ACO Algorithm

Heuristic information is the basic element of ACO algorithms and embodies the use degree of problem information. Therefore, in the process of building the JSSP scheduling solutions by the modified ACO algorithm in this paper, it is necessary to select the operation whose number is j to incorporate into sequencing results in accordance with random proportional rule described by formula (1):

$$p_{ij}=\frac{(\eta_{ij})^{\beta}}{\sum(\eta_{ij})^{\beta}} \qquad (1)$$

In formula (1), the exact definition of β can be seen in literature [8]. p_{ij} indicates the probability of a arrangement made for the number *j* operation, in the case of a arrangement made for the number *i* operation (i.e. the operation has been incorporated into the sequencing results, and the starting time and the finishing time have been determined). As far as ACO algorithms for the TSP is concerned, heuristic information is the reverse of a number of the distance between two cities, whereas the distance between two cities is, in effect, the increment of objective function (i.e. tour length) of the TSP. Therefore, it can be extended to the JSSP. The definition of heuristic information η_{ij} in this paper is as follows:

$$\eta_{ij}=\begin{cases}\dfrac{1}{\Delta M(i,j)} & \text{if } \Delta M(i,j)>0\\ PT(j) & \text{if } \Delta M(i,j)=0\end{cases} \qquad (2)$$

The definition of function *M(i,j)* in formula (2) is: in the case of making arrangement of the number *i* operation, and then by making the arrangement of the number *j* operation, the current makespan is newly obtained. The current makespan is similar to the tour length via which the ants travel when ACO algorithms are used to find solution to the TSP. In initial stage, *M(i,j)*= the processing time of the number *i* operation =the finishing time of the number *i* operation = the current makespan. After the arrangement of all operations is completed, *M(i,j)*= makespan. Obviously, there is $0 < M(i,j) \leq$ makespan. Therefore, $\Delta M(i,j)$=the increment of the JSSP objective function (i.e. makespan). That is to say, in the case of making arrangement of the number *i* operation, and then by making the arrangement of the number *j* operation, the increment of the current makespan is caused. The function *PT(j)* indicates the processing time of the number j operation.

The heuristic information has actually defined a new priority rule for the JSSP, whose connotation is: if some operation is arranged, the more the makespan increases, the lower the probability that the operation number is incorporated into sequencing result will be. If some operation is arranged, there is no increase in makespan, the longer the processing time of the operation is, and the higher the probability that the operation number is incorporated into sequencing result will be. The former can be called as the priority rule of "*the minimum increment in makespan*" and the latter is, in fact, the priority rule of the longest processing time. The experiments have shown that the priority rule is much better.

D. Local Search Procedure

Neighborhood structure is the basic element of local search procedure in ACO algorithms and embodies the effective use of problem specific knowledge as well as is directly concerned with the quality and efficiency of solution finding. Some related researches indicate that the exchange of the first and second operations in the first block or the exchange of the last and the second reciprocal operation in the last block can obtain a new feasible solution, but the makespan cannot be reduced[3].The operations within block on critical path can be exchanged with the first or last operation so as to obtain a new feasible solution, and the makespan is likely to be reduced[3].If there exist multiple critical paths, one of them can be selected at random, thus having no much effect upon the results[15]. For this reason, randomly selecting a critical path and adopting neighborhood structure suggested in literature [3] can realize the local search procedure of the algorithm in this paper.

III. COMPUTATIONAL RESULTS

The proposed algorithm in this paper is realized with Delphi 7 Personal Edition. The operating system is Windows XP Professional Edition, and hardware configuration is Pentium Ⅳ 2.0 GHz CPU and 512 MB RAM. The experiments have found that the better parameter setting of the proposed algorithm can be: maximum cycle times = number of jobs × number of machines ×20, number of ants = number of jobs × 4 and heuristic factor β=2.0. In order to avoid accidents, the experimental results of the algorithms independent running for 30 times are served as the references. In order to be convenient for the contrast with other algorithms, the same test data of the JSSP are adopted to carry out the algorithm experiments.

A. Pheromone Experiments

In order to investigate the effect of pheromone upon ACO algorithms for the JSSP, and in the prerequisite of adopting heuristic information suggested in this paper and without the local search procedure, the experimental results obtained from ACS algorithm and the modified ACO algorithm in this paper are compared. By advice of literature [8], the parameters setting for ACS algorithm are: pheromone initial value =1.0/ known best solution (i.e. minimum makespan), pheromone factor α=1.0, local pheromone evaporation rate ξ=0.1, global pheromone evaporation rate ρ=0.1, q0=0.9 and the rest is the same as the above mentioned.

FT10 is a notorious instance of the JSSP, whose known best solution (KBS) is 930. Accordingly, the detailed experimental results of FT10 by the algorithm in this paper and ACS algorithm are listed in Table 1.

TABLE I. EXPERIMENTAL RESULTS OF ACS ALGORITHM AND THE PROPOSED ALGORITHM FOR FT10

	ACS Algorithm	**The Proposed Algorithm**
Best	977	971
Worst	1004	1001
Average	992.2	992.1
Minimum cycle times	310	62
Maximum cycle times	1774	1885
Average cycle times	987.6	789.8
Average Time(s)	30.1	28.5

Notes: Cycle times refers to the algorithm in obtaining the best result with the least cycle times. Average time refers to the algorithm in obtaining the best result with the average value of the shortest time, which is closed related to the average cycle times. Literature [14] only gives the experiment results of FT10, but the results given by the algorithm in this paper are much better.

In order to be convenient, the JSSP instances in literatures [11] and [12] are adopted to test ACS algorithm and the algorithm in this paper, with the results given in Table 2.

TABLE II. COMPARISON BETWEEN THE PROPOSED ALGORITHM AND ACS ALGORITHM

Instance	**Size**	**KBS**	**ACS Algorithm**		**The Proposed Algorithm**	
			Best	**Avg.**	**Best**	**Avg.**
ABZ6	10×10	943	990	996.5	978	979.7
LA06	15×5	926	926	928.5	926	926
LA07	15×5	890	910	911.7	894	903.0
LA11	20×5	1222	1222	1222	1222	1222
LA12	20×5	1039	1039	1044.7	1039	1041.4
LA17	10×10	784	834	841.0	804	826.5
LA23	15×10	1032	1061	1065.8	1049	1065.1
LA26	20×10	1218	1335	1362.1	1334	1358.0
LA36	15×15	1268	1387	1402.2	1385	1396.9

It can be found from the above experiment results that as viewed from the qualities of the best solution and the average solution, the algorithm in this paper is superior to ACS algorithm. For this reason, it can be considered that when ACO algorithms are used to find solution to the JSSP, the JSSP corresponding to a partly connected directed graph leads to pheromone having no apparent guiding effects upon the problem solution finding, but the algorithm in this paper

excludes the pheromone part so that it is more concise and more efficient than ACO algorithms.

B. Comparison with Other ACO Algorithms

It is just here that the proposed algorithm without the local search procedure is compared with the two ACS algorithms [11, 12], with experimental results given in Table 3.

TABLE III. COMPARISON BETWEEN THE PROPOSED ALGORITHM AND OTHER ACO ALGORITHMS

Instance	ACS[11]	ACS[12]		The Proposed Algorithm	
	Best	Best	Avg.	Best	Avg.
ABZ6	/	995	1034.6	978	979.7
LA06	926	926	926	926	926
LA07	915	917	931.2	894	903.0
LA11	1222	1222	1224.5	1222	1222
LA12	1047	/	/	1039	1041.4
LA17	825	/	/	804	826.5
LA23	1089	/	/	1049	1065.1
LA26	1397	/	/	1334	1358.0
LA36	1440	1416	1481.8	1385	1396.9

Obviously, the algorithm in this paper is much superior to the ACS algorithms suggested in literatures [11] and [12] whereby illustrating that priority rule suggested in this paper is more effective.

C. Comparison with Tabu Search Algorithm

Finally, the contrast is made between the proposed algorithm with the local search procedure and the TSAB algorithm [3], with the experiment results listed in Table 4.

TABLE IV. COMPARISON OF THE PROPOSED ALGORITHM WITH TSAB ALGORITHM

Instance	Size	KBS	TSAB		The Proposed Algorithm	
			Best	Avg.	Best	Avg.
FT10	10×10	930	941	965.4	930	938.5
LA02	10×5	655	655	672.5	655	667.2
LA19	10×10	842	842	873.4	842	866.3
LA21	15×10	1046	1070	1124.7	1047	1053.3
LA24	15×10	935	951	984.2	944	946.1
LA25	15×10	977	988	1112.9	977	981.3
LA27	20×10	1235	1316	1402.8	1243	1254.4
LA29	20×10	1157	1227	1357.1	1165	1183.7
LA36	15×15	1268	1311	1389.6	1270	1312.3
LA37	15×15	1397	1476	1581.3	1397	1436.6
LA38	15×15	1196	1215	1284.6	1196	1230.4
LA39	15×15	1233	1287	1325.4	1278	1305.1
LA40	15×15	1222	1291	1357.8	1228	1234.5

It can be found via experiment results that as viewed from the solution quality, the algorithm in this paper is superior in the solution qualities to the TSAB algorithm and is a competitive approach for the JSSP.

IV. SUMMARY

This paper suggests a modified ACO algorithm, which mainly lies in eliminating unimportant pheromone in order to simplify ACO algorithms. Also, being subject to revelation of ACO algorithms for the TSP, this paper presents a new priority rule adaptable to the JSSP and introduces the local search procedure based on the JSSP neighborhood so as to improve the proposed algorithm. The results from the experiments indicate that the proposed algorithm in this paper is effective and also superior to the traditional ACO algorithms.

Although we only analyze and study the JSSP in this paper, the proposed algorithm based on its typicality and complexity can be applied to the other production scheduling problems, and even to other combinatorial optimization problems, whereby being of important reference values and great significance of drawing the experiences.

REFERENCES

[1] Blazewicz J. and W. Domschke, "The job shop scheduling problem: Conventional and new solution techniques," European Journal of Operational Research, vol. 93, pp.1–33, August 1996

[2] Dell'Amico M. and M. Trubian, "Applying tabu search to the job-shop scheduling problem," Annals of Operations Research, vol. 41, pp. 231–252, September 1993

[3] Nowicki E. and C. Smutnicki, "A Fast Taboo Search Algorithm for the Job Shop Problem," MANAGEMENT SCIENCE, vol. 42, pp. 797–13, June 1996

[4] Wang, L. and D. Z. Zheng, "A Modified Genetic Algorithm for Job Shop Scheduling," The International Journal of Advanced Manufacturing Technology, vol. 20, pp. 72–76, July 2002

[5] Kamrul Hasan S. M. and R. Sarker, " Modified genetic algorithm for job-shop scheduling: A gap utilization technique," in IEEE Congress on Evolutionary Computation(CEC 2007), IEEE Press, Piscataway, NJ, pp. 3804–3811, September 2007

[6] Aydin M. E. and T. C. Fogarty, "A simulated annealing algorithm for multi-agent systems: a job-shop scheduling application," Journal of Intelligent Manufacturing, vol. 15, pp. 805–814, December 2004

[7] Yamada T. and B. E. Rosen, "A simulated annealing approach to job shop scheduling using critical block transition operators," in Proc. of IEEE International Conference on Neural Networks (ICNN '94), IEEE Press, Piscataway, NJ, pp.4687–4692, June 1994

[8] Dorigo M. and Stützle T., Ant Colony Optimization. Cambridge, MA: MIT Press, 2004

[9] Zhou P. and X.-p. Li, "An ant colony algorithm for job shop scheduling problem," in Fifth World Congress on Intelligent Control and Automation, IEEE Press, Piscataway, NJ, pp. 2899–2903, June 2004

[10] FiglalI N. and C. Özkale, "Investigation of Ant System parameter interactions by using design of experiments for job-shop scheduling problems," Computers & Industrial Engineering, vol. 56, pp. 538–559, March 2009

[11] URSZULA BORYCZKA, "Ant Colony System for JSP," in ACRI 2004, LNCS,vol. 3305, pp. 296–305, September 2004

[12] Xiao-Ian Z. and Z. Jun, "A new pheromone design in ACS for solving JSP," in IEEE Congress on Evolutionary Computation(CEC 2007), IEEE Press, Piscataway, NJ, pp. 1963–1969, September 2007

[13] Montgomery J., "Alternative Solution Representations for the Job Shop Scheduling Problem in Ant Colony Optimization," In M. Randall, H.A. Abbass, and J. Wiles (Eds.), ACAL 2007, LNAI, vol. 4828, pp. 1–12, Springer, Heidelberg, November 2007

[14] Heinonen J. and F. Pettersson, "Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem," Applied Mathematics and Computation, vol. 187, pp. 989–998, April 2007

[15] Jain A. S. and B. Rangaswamy, "New and "Stronger" Job-Shop Neighbourhoods: A Focus on the Method of Nowicki and Smutnicki (1996)," Journal of Heuristics,vol. 6, pp. 457–480, September 2000