Journal of Asian Scientific Research
ISSN(e): 2223-1331/ISSN(p): 2226-5724
journal homepage: http://www.aessweb.com/journals/5003

SOLVING JOB SHOP SCHEDULING PROBLEM USING AN ANT COLONY ALGORITHM

Habibeh Nazif[1]

[1]*Department of Mathematics, Payame Noor University, Iran*

ABSTRACT

This paper describes the implementation of an ant colony algorithm (ACA), applied to a combinatorial optimization problem called job shop scheduling problem (JSSP). At first, a rather good solution is generated in negligible computation time and then, the trail intensities areinitiated based on this solution. Moreover, the trail intensities are limited between lower and upper bounds which change dynamically in a new manner. It is noteworthy that in initializing, updating as well as limiting the trail intensities, the goal is to guide the search towards the neighborhood around the best solution found. This paper outlines the algorithm's implementation and performance when applied to job shop scheduling. The computer simulations on a set of benchmark problems are conducted to assess the merit of the proposed algorithm compared to some other heuristics in the literature. The solutions were of good quality and demonstrated the effectiveness of the proposed algorithm.

Keywords: Scheduling- Ant colony algorithm- Job shop- Makespan.

Contribution/ Originality

This study is one of the common studies which have investigated to solve the well-known job shop scheduling problem using meta-heuristic algorithms, specially ant colony algorithms.

1. INTRODUCTION

One of the most difficult problems in the planning and managing of manufacturing processes is the job shop scheduling problem, which has been proved to be a NP-complete problem [1]. The JSSP can be described as follows: There are n different jobs to be processed on m different machines. Each job needs m operations and each operation needs to be processed without preemption for a fixed processing time on a given machine. There are several constraints on jobs and machines:

- A job can visit a machine once and only once.

DOI: 10.18488/journal.2/2015.5.5/2.5.261.268
ISSN(e): 2223-1331/ISSN(p): 2226-5724

- There are no precedence constraints among the operations of different jobs.
- Preemption of operations is not allowed.
- Each machine can process only one job at a time.
- Each job can be processed by only one machine at a time.
- Neither release times nor due dates are specified.

The problem is to find a schedule to minimize the makespan, that is, to minimize the time required to complete all jobs.

A job shop problem instance can be visualized by a directed graph G = (N, A, B), where N represents the set of nodes, A the set of conjunctive arcs and B the set of disjunctive arcs. The set of nodes contains one element for each job operation o_{ij}, a source node S connected to the first operation of each job and a sink node T linked with the last operation of each job. Conjunctive arcs are used to represent the routings of the different operations of the jobs and connect each pair of consecutive operations of the same job. Pairs of disjunctive arcs connect two operations, belonging to different jobs, which are to be processed on the same machine. The disjunctive arcs form a clique for each machine. A feasible solution corresponds to an acyclic subgraph that contains all conjunctive arcs and that contains exactly one disjunctive arc for each pair of disjunctive arcs between two nodes. An optimal solution corresponds to the feasible subgraph with the minimal makespan. An example of a disjunctive graph for a JSSP with three machines and three jobs is given in Fig 1.

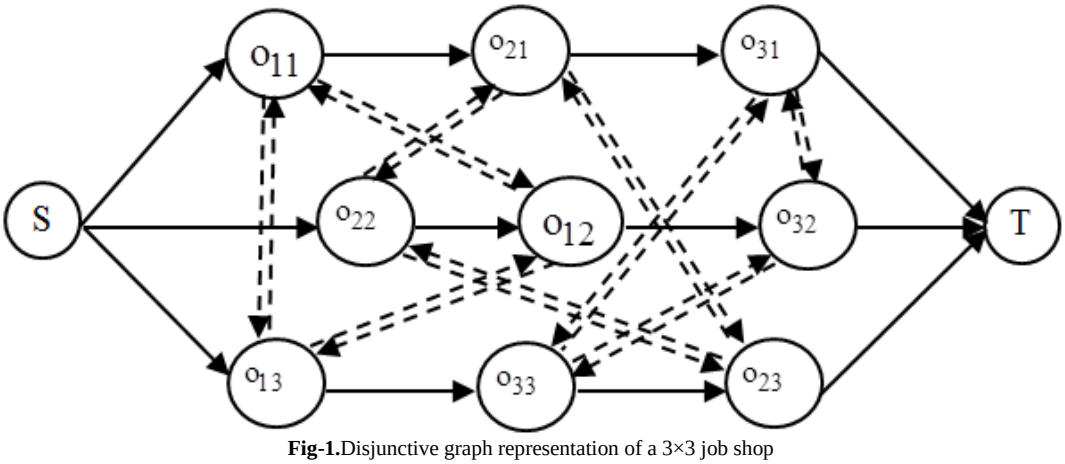

Fig-1. Disjunctive graph representation of a 3×3 job shop

The JSSP with a minimum makespan objective is widely acknowledged as one of the most difficult combinatorial optimization problems. A comprehensive survey of approximation algorithms can be found in, Jain and Meeran [2]. Moreover, real-life application of the JSSP was discussed in Sels, et al. [3].

The most common metaheuristic approaches for makespan minimization include genetic algorithm [4] tabu search [5] simulated annealing [6] ant colony optimization [7] particle swarm optimization [8]. A fair number of authors tried to increase the performance of the genetic algorithm by incorporating other traditional heuristics in the algorithm. This hybridization can happen with local search operator [9] tabu search technique [10] and simulated annealing approach

[11]. Additionally, a genetic algorithm and a scatter search procedure is proposed by Sels, et al. [12] to solve the job shop scheduling problem.

More recent research often focused on extensions of the JSSP. Examples are the inclusions of setup times[13] the adaptation of JSSP to the no-wait job shop [14] the incorporation of alternative objective functions [15] or the extension to the multi-objective JSSP [16].

In this paper, we apply an ant colony algorithm (ACA) to solve the JSSP with the objective of minimizing the maximum completion time, or makespan. The proposed algorithm is based on model designed by Ahmadizar [17] for the permutation flow shop problem. The performance of the proposed approach is evaluated on a set of benchmark problems.

2. ANT COLONY ALGORITHM

The main idea in ant colony optimization algorithms is to mimic the pheromone trails used by real ants searching for feed as a medium for communication and feedback. In the ACA, a rather good solution is firstly generated in negligible computation time, and then the pheromone trails are initialized depending on this solution. In other words, unlike most applications of ant colony optimization algorithms, at the beginning of the ACA an equal initial value is not assigned to all pheromone trails. Each artificial ant starts with an empty sequence and chooses one of the jobs. Then, the ant iteratively appends an unscheduled job to the partial sequence constructed so far until a complete solution is built. At each step, a job is chosen by applying a transition rule based on the pheromone trails. The performance quality of the constructed solution is then improved by means of a local search procedure. Once all ants in the colony have built their solutions, to make the search more directed, the pheromone trails are modified by applying a global updating rule. Moreover, the trail intensities are limited between lower and upper bounds which change dynamically in a new manner. The general structure of the proposed algorithm is represented as follows:

General structure of the ACA:

Step 1. Set parameters; generate a seed solution and initialize the pheromone trails.

Step 2. While the termination condition is not met, do the following:

2.1. For each ant in the colony do:

By repeatedly applying the transition rule, construct a solution;

Improve the solution quality by the local search;

In case of an improved solution, update the best solution generated so far.

2.2. Modify the pheromone trails according to the global updating rule.

2.3. Update the minimum and maximum trail bounds, and limit the pheromone trails.

Step 3. Return the best solution found.

Once a complete sequence of jobs has been generated by an ant, the performance quality of the solution is improved by means of alocal search procedure. Since searching a large neighborhood requires more computational time, a new local search procedure is proposed to achieve a good trade off between the number of solutions constructed by ants and the local search time. Tohandle this issue, a threshold probability T is incorporated for choosing a job to insert into the other

positions of a given sequence. Higher value of T suggests that a larger neighborhood is expected to be searched (see details in Ahmadizar [17]).

The proposed local search procedure is then represented as follows:

Local search procedure:

For each job j (j = 1,…,N), do the following:

1. Generate a random number R uniformly distributed in [0, 1];
2. If $R \leq T$ do:
   2.1. For each position i (i = 1,…,N), do:
      If job j is not in the ith position of the current sequence, insert job j in position i without any change in the other sequence, and then calculate the makespan of the newly obtained sequence.
   2.2. Determine the best sequence among the N-1 newly obtained sequences.
   2.3. If the makespan is improved, replace the current sequence by the best one found.

Moreover, in the ACA the procedure with T being equal to 10/N is applied five times to improve y the quality of each ant-sequence.

3. COMPUTATIONAL RESULTS

In order to verify the good performance of the proposed algorithm, we use 43 instances from two classes of standard JSSP test problems: Fisher and Thompson [18] instances FT06, FT10, FT20 and Lawrence [19] instances LA01–LA40. Due to the stochastic nature of the ACA, each of the problem instances has been tested for five trials. The best solution has been taken for the five trials.

The proposed algorithm has been coded in Visual C++ and all test runs have been carried out on a 2.0 GHz Intel Core 2 Duo Processor with 2 GB memory. The ACA was compared with some algorithms reported in literature such as: Qing-dao-Er-Ji and Wang [9]; Yang and Sun [20]; Goncalves, et al. [21]; Ombuki and Entresca [22]; Coello, et al. [23] and Binato, et al. [24]. The parameters used in experiments are set similar to Ahmadizar [17]. Moreover, the algorithm terminates when the total number of iterations reaches 150.

Table 1 shows the experimental results. It lists problem name, problem size (number of jobs × number of operations), the best known solution (BKS) found in the literature and the solution obtained by each of the compared algorithms.

As seen from Table 1, the proposed algorithm is able to find the best known solution for 31 instances. For small problems FT06, FT10, FT20 and LA01–LA15, almost all the algorithms can find the optimal solution. For relatively large problems LA16–LA40, the results of the proposed algorithm ACA are better than most of the algorithms.

For each algorithm, we can use formula RD = 100×(MFM-BKS)/BKS for each instance to calculate the relative deviation, where MFM means the minimum makespan found and BKS means the best known solution. We use ARD to denote the average value of relative deviations for all the instances. Table 2 shows the number of instances solved (NIS), and the average relative deviation (ARD). The ARD was calculated for the ACA and the other algorithms. From Table 2, the

proposed algorithm yields a significant improvement in solution quality with respect to other algorithms except the HGA.

Table-2. Average relative deviation to the BKS

Algorithm	NIS	ARD	ARD (ACA)
Qing and Wang (HGA)	43	0.17	0.25
Yang and Sun (MA)	23	0.034	0.033
Goncalves et al. (Param. Active)	43	0.39	0.25
Ombuki and Entresca (LSGA)	25	5.22	0.44
Coello et al. (AIS)	24	1.51	0.21
Binato et al. (GRASP)	43	1.68	0.25

Table-1. Numerical Results

				Qing and Wang	Yang and Sun	Goncalves et al.	Ombuki and Entresca	Coello et al.	Binato et al.
Problem	**Size**	**BKS**	**ACA**	**HGA**	**MA**	**Param. active**	**LSGA**	**AIS**	**GRASP**
FT06	6×6	55	55	55	55	55	-	-	55
FT10	10×10	930	930	930	930	930	-	941	938
FT20	20×5	1165	1165	1165	1165	1165	-	-	1169
LA01	10×5	666	666	666	666	666	-	666	666
LA02	10×5	655	655	655	655	655	-	655	655
LA03	10×5	597	597	597	597	597	-	597	597
LA04	10×5	590	590	590	590	590	-	590	590
LA05	10×5	593	593	593	593	593	-	593	593
LA06	15×5	926	926	926	926	926	-	926	926
LA07	15×5	890	890	890	890	890	-	890	890
LA08	15×5	863	863	863	863	863	-	863	863
LA09	15×5	951	951	951	951	951	-	951	951
LA10	15×5	958	958	958	958	958	-	958	958
LA11	20×5	1222	1222	1222	1222	1222	-	-	1222
LA12	20×5	1039	1039	1039	1039	1039	-	-	1039
LA13	20×5	1150	1150	1150	1150	1150	-	-	1150
LA14	20×5	1292	1292	1292	1292	1292	-	-	1292
LA15	20×5	1207	1207	1207	1207	1207	-	-	1207
LA16	10×10	945	946	945	945	945	959	945	946
LA17	10×10	784	789	784	784	784	792	785	784
LA18	10×10	848	848	848	848	848	857	848	848
LA19	10×10	842	842	844	844	842	860	848	842
LA20	10×10	902	902	907	907	907	907	907	907
LA21	15×10	1046	1050	1046	-	1046	1114	-	1091
LA22	15×10	927	938	935	-	935	989	-	960
LA23	15×10	1032	1032	1032	-	1032	1035	-	1032
LA24	15×10	935	959	953	-	953	1032	-	978
LA25	15×10	977	977	981	-	986	1047	1022	1028
LA26	20×10	1218	1218	1218	-	1218	1307	-	1271
LA27	20×10	1235	1242	1236	-	1256	1350	-	1320
LA28	20×10	1216	1227	1216	-	1232	1312	1277	1293
									Continue

LA29	20×10	1152	1177	1160	-	1196	1311	1248	1293
LA30	20×10	1355	1355	1355	-	1355	1451	-	1368
LA31	30×10	1784	1784	1784	-	1784	1784	-	1784
LA32	30×10	1850	1850	1850	-	1850	1850	-	1850
LA33	30×10	1719	1719	1719	-	1719	1745	-	1719
LA34	30×10	1721	1725	1721	-	1721	1784	-	1753
LA35	30×10	1888	1888	1888	-	1888	1958	1903	1888
LA36	15×15	1268	1275	1287	-	1279	1358	1323	1334
LA37	15×15	1397	1412	1407	-	1408	1517	-	1457
LA38	15×15	1196	1196	1196	-	1219	1362	1274	1267
LA39	15×15	1233	1240	1233	-	1246	1391	1270	1290
LA40	15×15	1222	1222	1229	-	1241	1323	1258	1259

We plot the best solution found by the ACA for all of the instances. As shown in Fig 2, for most of the problems the proposed ACA provides the results that are equal to the best known solution.

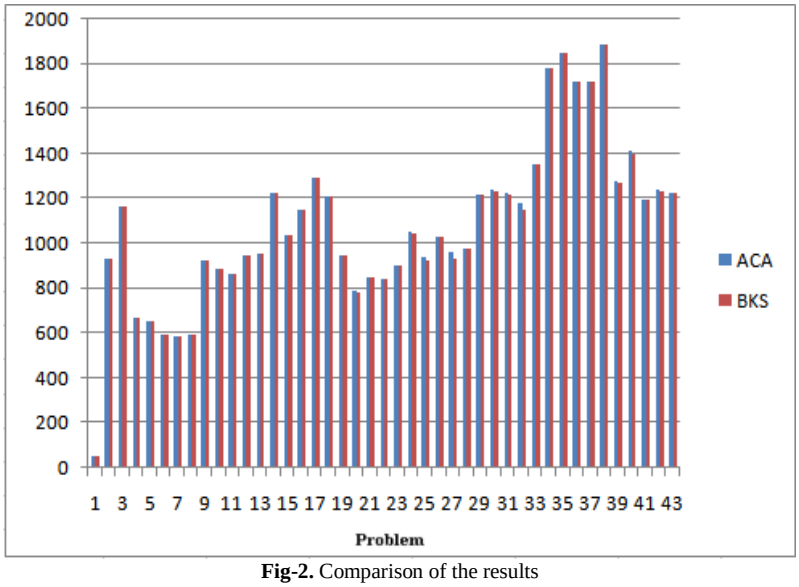

Fig-2. Comparison of the results

4. CONCLUSION

To solve the JSSP more effectively, an ant colony algorithm is developed with the makespan criterion. A novel mechanism is employed in initializing the pheromone trails based on an initial sequence. Moreover, the pheromone trail intensities are limited between lower and upper bounds which change dynamically. The performance quality of a solution constructed by an artificial ant is improved by a job-index-based local search procedure incorporated with a threshold probability for choosing a job to insert into the other positions of the sequence. Once all ants in the colony have generated their solutions, the pheromone trails are modified by applying a global updating rule. The experimental results show that the proposed algorithm is competitive when compared with the best known solutions in the literature.

REFERENCES

[1] J. Lenstra, A. Kan, and P. Brucker, "Complexity of machine scheduling problem," *Annals of Discrete Mathematics*, vol. 1, pp. 343–62, 1977.

[2] A. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *European Journal of Operational Research*, vol. 113, pp. 390–434, 1999.

[3] V. Sels, F. Steen, and M. Vanhoucke, "A hybrid job shop procedure for a Belgian manufacturing company producing industrial wheels and castors in rubber," Technical Report, Ghent University, 2010.

[4] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: Hybrid genetic search strategies," *Computers and Industrial Engineering*, vol. 36, pp. 343–364, 1999.

[5] C. Zhang, P. Li, Y. Rao, and Z. Guan, "A very fast ts/sa algorithm for the job shop scheduling problem," *Computers and Operations Research*, vol. 35, pp. 282–294, 2008.

[6] M. Kolonko, "Some new results on simulated annealing applied to the job shop scheduling problem," *European Journal of Operational Research*, vol. 113, pp. 123–136, 1999.

[7] K. Huang and C. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Computers and Operations Research*, vol. 35, pp. 1030–1046, 2008.

[8] D. Sha and C. Hsu, "A hybrid particle swarm optimization for job shop scheduling problem," *Computers and Industrial Engineering*, vol. 51, pp. 791–808, 2006.

[9] R. Qing-dao-Er-Ji and Y. Wang, "A new hybrid genetic algorithm for job shop scheduling problem," *Computers & Operations Research*, vol. 39, pp. 2291–2299, 2012.

[10] G. Vilcaut and J. Billaut, "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problems," *European Journal of Operational Research*, vol. 190, pp. 398–411, 2008.

[11] H. Zhang and M. Gen, "Multistage-based genetic algorithm for flexible job-shop scheduling problem," *Complexity International*, vol. 11, pp. 223–232, 2005.

[12] V. Sels, K. Craeymeersch, and M. Vanhoucke, "A hybrid single and dual population search procedure for the job shop scheduling problem," *European Journal of Operational Research*, vol. 215, pp. 512–523, 2011.

[13] J. Sun, "A genetic algorithm for a re-entrant job-shop scheduling problem with sequence-dependent setup times," *Engineering Optimization*, vol. 41, pp. 505–520, 2009.

[14] J. Pan and H. Huang, "A hybrid genetic algorithm for no-wait job shop scheduling problems," *Expert Systems with Applications*, vol. 36, pp. 5800–5806, 2009.

[15] I. Essafi, Y. Mati, and S. Dauzère-Pérès, "A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem," *Computers and Operations Research*, vol. 35, pp. 2599–2616, 2008.

[16] B. Qian, L. Wang, D. Huang, and X. Wang, "Scheduling multi-objective job shops using a memetic algorithm based on differential evolution," *International Journal of Advanced Manufacturing Technology*, vol. 35, pp. 1014–1027, 2008.

[17] F. Ahmadizar, "A new ant colony algorithm for makespan minimization in permutation flow shops," *Computers & Industrial Engineering*, vol. 63, pp. 355–361, 2012.

[18] H. Fisher and G. Thompson, *Probabilistic learning combinations of local job-shop scheduling rules*. Englewood Cliffs, NJ: Prentice-Hall, 1963.

[19] S. Lawrence, "Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques," Technical Report, GSIA,Carnegie Mellon University, 1984.

[20] J. Yang and L. Sun, "Clonal selection based memetic algorithm for job shop scheduling problems," *Journal of Bionic Engineering*, vol. 5, pp. 111–9, 2008.

[21] J. Goncalves, J. Mendes, and M. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European Journal of Operational Research*, vol. 167, pp. 77–95, 2005.

[22] B. Ombuki and M. Entresca, "Local search genetic algorithms for the job shop scheduling problem," *Applied Intelligence*, vol. 21, pp. 99–109, 2004.

[23] C. Coello, D. Rivera, and N. Cortez, "Use of an artificial immune system for job shop scheduling," in *Artificial Immune Systems: Proceedings of the ICARIS*, 2003, pp. 1–10.

[24] S. Binato, W. Hery, D. Loewenstern, and M. Resende, "A GRASP for job shop scheduling," *Essays and Surveys in Metaheuristics*, pp. 59–79, 2002.