Solving Permutation Flow Shop Sequencing using Ant Colony Optimization

Fardin Ahmadizar[1,2], Farnaz Barzinpour[1], Jamal Arkat[1]
[1]Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
[2]Department of Industrial Engineering, University of Kurdistan, Sanandaj, Iran
e-mail: zina@iust.ac.ir (F. Ahmadizar); barzinpour@iust.ac.ir (F. Barzinpour); j_arkat@iust.ac.ir (J. Arkat)

Abstract - **This paper proposes an ant colony algorithm for permutation flow shop scheduling problem. The objective considered is to minimize makespan. Two priority rules are developed as heuristic information based on Johnson's Rule and total processing times. A local search is used for improving the constructed solutions. The proposed ant colony algorithm is tested on the benchmark problem set of Taillard. The obtained results are compared with the previous implementations of ant colony optimization which are available in the literature. Computational results show that the proposed algorithm performs better than other algorithms when the number of machines is less than ten.**

Keywords - **Ant colony optimization, makespan, permutation flow shop, scheduling**

I. INTRODUCTION

The problem of sequencing a set of jobs on one or several machines in a permutation flow shop optimizing has been the subject of extensive research. The permutation flow shop scheduling problem (PFSP) is a NP-hard problem [1], [2]. Some exact algorithms (see [3]-[6]) and many heuristics (e.g., [7]-[24]) have been proposed for solving PFSP. Recently, many researcher have developed metaheuristics such as tabu search (e.g., [25]-[29]), genetic algorithms (e.g., [30]-[32]), simulated annealing (e.g., [33]-[36]) and particle swarm optimization algorithms (e.g., [37]-[40]) to find better solutions for PFSP.

Ant colony optimization (ACO) is another metaheuristic developed for solving PFSP. The ACO is a population-based approach proposed by Dorigo [41], [42] to solve discrete optimization problems. The first ACO algorithm was proposed by Stuetzle [43] for solving PFSP, which was a max-min ant system, called MMAS, for the makespan minimization problem. More recently, Rajendran and Ziegler [44] developed two ant colony algorithms for the makespan/total flowtime minimization problem (the first algorithm, called M-MMAS or extended MMAS and the second one, called PACO, developed M-MMAS). Ying and Liao [45] proposed an ant colony system, called ACS, for minimizing the makespan; they applied a different representation of pheromone trails based on a disjunctive graph. Rajendran and Ziegler [46] proposed two ant colony algorithms, called ACO1 and ACO2, for minimizing the total flowtime. Finally Gajpal and Rajendran [47] developed an ant colony algorithm, called NACO, for minimizing the completion-time variance of jobs.

In this paper, a new ant colony algorithm is presented for PFSP with the objective of minimizing makespan. Each artificial ant randomly constructs a solution based on pheromone trails and heuristic information. In order to calculate the heuristic information, two rules are developed according to the previous well-known rules. The generated solution is then improved by a local search procedure. Finally, the pheromone intensities are modified by applying the global updating rule.

The remainder of the paper is organized as follows. Next section gives the problem statement. The proposed ant colony algorithm is described in Section 3. Section 4 provides computational experiments and the conclusion is presented in Section 5.

II. PERMUTATION FLOW SHOP

The PFSP consists in scheduling N different jobs with given processing times on a set of M machines. Each job has exactly one operation to be processed on each machine. Each job has the same machine sequence and the sequence in which each machine processes all jobs is identical on all machines. Preemption is not allowed. Each machine can process at most one job at a time and each job can be processed on at most one machine at a time. Also, it is assumed that all jobs are available and ready to start at time zero and the setup times are sequence independent. Job j has processing time P_{ij} on machine i. Let C_{ij} be the completion time of job j on machine i. In this paper, the objective of minimizing the maximum completion time, or makespan, is considered. Given the job permutation $\{1,2,...,N\}$, the calculation of C_{ij} (j=1,2,…,N , i=1,2,…,M) is given as follows:

$$C_{i1} = \sum_{a=1}^{i} P_{a1} \qquad , i = 1, 2, ..., M \tag{1}$$

$$C_{1j} = \sum_{b=1}^{j} P_{1b} \qquad , j = 1, 2, ..., N \tag{2}$$

$$C_{ij} = \max\{C_{(i-1)j}, C_{i(j-1)}\} + P_{ij} \qquad , i = 2, ..., M \qquad , j = 2, ..., N \tag{3}$$

Then makespan $C_{\max}$ is obtained by $C_{\max} = C_{MN}$.

III. PROPOSED ANT COLONY ALGORITHM

In the proposed ant colony algorithm, each ant starts with an empty sequence and chooses the first job. Then

the ant iteratively appends an unscheduled job to the partial sequence constructed so far. At each step a job is chosen by applying the transition rule. The global structure of the algorithm is presented as follows:
1- The pheromone trails and the parameters are set.
2- While the stop condition is not met:
  i- Each ant constructs a complete solution using the transition rule and the solution is then improved by local search.
  ii- The pheromone trails are globally updated.
3- The best solution found is printed.

A. Transition Rule

Ants are guided, in building their solutions, by both heuristic information and pheromone intensity. In the proposed algorithm, ants choose the next job to append to the partial sequence according to ant colony system [48]: with probability q_0, an ant k on position i selects unscheduled job j for which the product between pheromone trail and heuristic information is maximum, that is,

$$j = \arg\max\left[\tau_{ij}(\eta_{ij})^{\beta}\right] \tag{4}$$

where τ_{ij} and η_{ij} are, respectively, the pheromone trail and the heuristic information between position i and job j (denoted by edge (i,j)). Also β is the relative importance of the heuristic information versus the pheromone trail ($\beta > 0$). While with probability $1 - q_0$, the ant chooses job j according to the probability distribution given in the following equation:

$$p_{ij}^{k} = \frac{\tau_{ij}(\eta_{ij})^{\beta}}{\sum_{l \in N_i^k} \tau_{il}(\eta_{il})^{\beta}} \qquad : if \;\; j \in N_i^k \tag{5}$$

where N_i^k is the feasible neighborhood of ant k on position i, that is, the set of jobs which ant k has not yet visited.

We have developed two priority rules in order to calculate the heuristic information.

Rule 1: (Extension of Johnson's Rule)

"If a job has shorter processing times on the initial machines than those on the final machines, it must be processed first."

Rule 2: (Extension of SPT)

"If a job has short processing times on all machines, it must be processed first."

These two rules are fuzzy ones. Let $R_j^{(1)}$ and $R_j^{(2)}$ be, respectively, the priority grade (the grade of membership) of job j according to rules 1 and 2. These are defined as follows:

$$R_j^{(1)} = \frac{(P_{M-1,j} + P_{Mj}) \big/ (P_{1j} + P_{2j})}{\sum_{l=1}^{N} (P_{M-1,l} + P_{Ml}) \big/ (P_{1l} + P_{2l})} \tag{6}$$

$$R_j^{(2)} = \frac{1 \big/ \sum_{i=1}^{M} P_{ij}}{\sum_{l=1}^{N} 1 \big/ \sum_{i=1}^{M} P_{il}} \tag{7}$$

$R_j^{(1)}$ and $R_j^{(2)}$ are greater than 0 and smaller than 1, therefore the given fuzzy sets are not normal sets. This issue imposes the effect of parameter β in transition rule. Based on $R_j^{(1)}$ and $R_j^{(2)}$, the heuristic information can be calculated in several manners. In this research, six cases have been considered.

$$\begin{aligned}
&a)\ \eta_{ij} = R_j^{(1)}\\
&b)\ \eta_{ij} = R_j^{(2)}\\
&c)\ \eta_{ij} = Min(R_j^{(1)}, R_j^{(2)})\\
&d)\ \eta_{ij} = R_j^{(1)}.R_j^{(2)}\\
&e)\ \eta_{ij} = Average(R_j^{(1)}, R_j^{(2)})\\
&f)\ \eta_{ij} = Max(R_j^{(1)}, R_j^{(2)})
\end{aligned} \tag{8}$$

B. Local Search

The performance of ant colony algorithms can be greatly improved when coupled with a local search procedure. On the other hand, this increases run time of the algorithm. In order to achieve the best performance, after constructing a complete solution, the local search is applied to improve the results as follows:
1- For job j = 1 to N, do:
  With probability 0.02 for position i = 1 to N, do:
    If job j isn't in position i, insert this job in position i and calculate the makespan of the created sequence.
2- Choose the best sequence among all created sequences. If this solution is better than the initial one, replace it.

C. Global Updating of Pheromone Trails

Once all ants in the colony have constructed their solutions, the amount of pheromone on each edge (i,j) belongs to the best solution (which can be the best in the current iteration of the algorithm or up to the current iteration) is modified by applying the global updating rule as follows:

$$\tau_{ij} = (1-\rho)\,\tau_{ij} + \rho.\ z \big/ C_{\max}^{best} \qquad : if\,(i,j) \in Best \tag{9}$$

where ρ is the pheromone trail evaporation rate ($0 < \rho \le 1$), z is a parameter and $C_{\max}^{best}$ is the makespan of the best solution. Note that the initial value of the pheromone trails is firstly equal to τ_0.

IV. Computational Results

The proposed ant colony algorithm (ACA) was coded in C++ and run on a Pentium 4 CPU at 2 GHz, 256 MB RAM under Windows XP using Microsoft Visual C++6.0. In the preliminary experiment, some values were tested for the numeric parameters. With different compositions of the parameters each case in (8) was evaluated and the cases (8-c) and (8-f) yielded the best and worst results respectively and the other ones yielded fair results. Therefore, the heuristic information was set according to (8-c). Also, the best solution up to the current iteration of the algorithm in (9) prepared better performance than the best one in the current iteration. Then the best performance of the ACA was obtained with five ants, 2500 iterations of stage 2, $\tau_0 = 10^{-6}$, q_0=0.97, β =0.0001, ρ =0.01 and z=$N.M$.

The ACA was tested on the benchmark problem set of Taillard [49]. Thaillard has produced a set of problems for PFSP to minimize the makespan. There were 10 instances for each problem size. An integer processing time P_{ij} was generated from the uniform distribution [1, 99] for each instance. For each size of problem, each of problem instances was tested for five trials. Computational results show that ACA is superior when the number of machines is less than ten. Table I shows the results of three problem sizes of 20, 50 and 100 jobs with M=5, where the upper bound (UB) is the Taillard's UB. The optimal solutions are marked as bold numbers. It can be seen that ACA can effectively obtain optimal or near optimal solutions.

The performance of ACA was also compared with the previous implementations of ant colony optimization, that is, MMAS [43], M-MMAS and PACO [44] and ACS [45]. The performance comparison is shown in Table II, where the solution quality is measured by the mean percentage difference from Taillard's UB as follows:

$$Quality = \frac{C_{\max} - UB}{UB} \times 100. \tag{10}$$

It does suggest that our algorithm is capable of obtaining better results for all instances where the number of machines is five. When the number of machines increases, the performance of ACA will gradually degrade. This is possibly because of the sensitivity of the proposed rules to M. We have also considered the reported running times (s,) of the methods involved (ACS was coded in Visual C++ and run on an AMD 700 MHz PC [45] which is approximately 2 to 3 times slower than our PC). The superiority of ACA over ACS is observed in the CPU times.

TABLE I
RESULTS (M=5)

N	UB	Min	Max	Average
20	**1278**	**1278**	1297	1283.4
	1359	**1359**	1366	1363.4
	1081	**1081**	1088	1085.4
	1293	1299	1304	1301.6
	1235	1244	1250	1247.2
	1195	**1195**	1210	1207
	1239	1247	1251	1250.2
	1206	**1206**	1214	1208.4
	1230	**1230**	1252	1246
	1108	**1108**	1120	1114.8
50	**2724**	**2724**	2729	2725
	2836	**2834**	2843	2838.6
	2621	**2621**	2624	2622.4
	2751	**2751**	2770	2760
	2863	**2863**	2864	2863.8
	2829	**2829**	2835	2831.2
	2725	**2725**	2736	2732.2
	2683	2694	2704	2700
	2554	2561	2569	2565.8
	2782	**2782**	2782	2782
100	**5493**	**5493**	5495	5493.4
	5274	5275	5284	5282.2
	5175	**5175**	5198	5188.2
	5018	5021	5044	5029.2
	5250	**5250**	5255	5253.6
	5135	**5135**	5135	5135
	5247	5259	5261	5260.2
	5094	5096	5113	5102
	5448	5454	5467	5463.6
	5328	5328	5346	5337

TABLE II
PERFORMANCE COMPARISION

	ACA		ACS		MMAS	M-MMAS	PACO
N|M	Quality	Time	Quality	Time	Quality	Quality	Quality
20|5	**0.184**	1.2	1.19	11	0.408	0.762	0.704
20|10	0.792	1.5	1.70	12	**0.591**	0.890	0.843
20|20	0.852	2.3	1.60	16	**0.410**	0.721	0.720
50|5	**0.061**	9.1	0.43	44	0.145	0.144	0.090
50|10	1.818	14.1	1.89	54	2.193	1.118	**0.746**
50|20	2.909	22.2	2.71	73	2.475	2.013	**1.855**
100|5	**0.046**	50.5	0.22	163	0.196	0.084	0.072
100|10	1.350	90.2	1.22	197	0.928	0.451	**0.404**
100|20	2.683	141.3	2.22	264	2.238	1.030	**0.985**
Ave. [a]	**0.097**		0.61		0.250	0.330	0.289
Average	1.188		1.46		1.065	0.801	**0.713**

[a] Average for M=5.

V. Conclusion

In several manufacturing and assembly environments, for example, ones producing integrated circuits, printed circuit boards and TV sets, a number of operations have to be done on every job in the same order. The machines are assumed to be set up in series and the environment is referred to as a flow shop. In this paper an ant colony optimization algorithm has been developed to solve the permutation flow shop scheduling problem with the objective of minimizing makespan. Each ant constructs a solution by iteratively applying a stochastic rule based on two priority rules and the pheromone trails. Then the solution is improved by a local search. Once all ants have built their solutions, pheromone trails are modified by using the global updating rule. To evaluate the performance of the proposed algorithm, it is compared with other ant colony algorithms. The algorithm has been tested on the benchmark problem and computational experiments are given to demonstrate the superiority of the proposed ant colony algorithm when the number of machines is about five.

References

[1] M.R. Garey, D.S. Johnson, R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, pp. 117-129, 1976.

[2] A.H.G. Rinnooy Kan, *Machine Scheduling Problems: Classification, Complexity, & Computations*, Martinus Nijhoff, The Hague, 1976.

[3] S.P. Bansal, "Minimizing the sum of completion times of n-jobs over M-machines in a flowshop - a branch and bound approach," *AIIE Trans.*, vol. 9, pp. 306-311, 1977.

[4] E. Ignall, L. Schrage, "Application of the branch-and-bound technique to some flowshop scheduling problems," *Oper. Res.*, vol. 13, pp. 400-412, 1965.

[5] Z.A. Lomnicki, "A branch and bound algorithm for the exact solution of the three-machine scheduling problem," *Oper. Res. Quart.*, vol. 16, no. 1, pp. 89-100, 1965.

[6] E.F. Stafford, "On the development of a mixed integer linear programming model for the flowshop sequencing problem," *J. Oper. Res. Soc.*, vol. 39, pp. 1163-1174, 1988.

[7] A. Allahverdi, T. Aldowaisan, "New heuristics to minimize total completion time in m-machine flowshops," *Int. J. Prod. Econ.*, vol. 77, pp. 71-83, 2002.

[8] H.G. Campbell, R.A. Dudek, M.L. Smith, "A heuristic algorithm for the n job, m machine sequencing problem," *Manag. Scie.*, vol. 16, no. 10, pp. B630-B637, 1970.

[9] J.M. Framinan, R. Leisten, "An efficient constructive heuristic for flowtime minimisation in permutation flow shops," *OMEGA*, vol. 31, pp. 311-317, 2003.

[10] J.M. Framinan, R. Leisten, "A heuristic for scheduling a permutation flowshop with makespan objective subject to maximum tardiness," *Int. J. Prod. Econ.*, vol. 99, pp. 28-40, 2006.

[11] J.M. Framinan, R. Leisten, R. Ruiz-Usano, "Efficient heuristics for flowshop sequencing with the objectives of makespan and flowtime minimisation," *Eur. J. Oper. Res.*, vol. 141, pp. 559-569, 2002.

[12] L.F. Gelders, N. Sambandam, "Four simple heuristics for scheduling a flow-shop," *Int. J. Prod. Res.*, vol. 16, pp. 221-231, 1978.

[13] J.N.D. Gupta, "Heuristic algorithms for multistage flowshop scheduling problem," *AIIE Trans.*, vol. 4, pp. 11-18, 1972.

[14] J.C. Ho, "Flowshop sequencing with mean flow time objective," *Eur. J. Oper. Res.*, vol. 81, pp. 571-578, 1995.

[15] J.C. Ho, Y.L. Chang, "A new heuristic for the n-job, M-machine flowshop problem," *Eur. J. Oper. Res.*, vol. 52, pp. 194-202, 1991.

[16] S. Miyazaki, N. Nishiyama, "Analysis for minimizing weighted mean flowtime in flowshop scheduling," *J. Oper. Res. Soc. of Japan*, vol. 23, pp. 118-132, 1980.

[17] S. Miyazaki, N. Nishiyama, F. Hashimoto, "An adjacent pairwise approach to the mean flowtime scheduling problem," *J. Oper. Res. Soc. of Japan*, vol. 21, pp. 287-299, 1978.

[18] M. Nawaz, E. Enscore Jr., I. Ham, "A heuristic algorithm for the m-machine, n-job flow shop sequencing problem," *OMEGA*, vol. 11, no. 1, pp. 91-95, 1983.

[19] D.S. Palmer, "Sequencing jobs through a multistage process in the minimum total time: A quick method of obtaining a near-optimum," *Oper. Res. Quart.*, vol. 16, pp. 101-107, 1965.

[20] C. Rajendran, "Heuristic algorithm for scheduling in a flowshop to minimize total flowtime," *Int. J. Prod. Econ.*, vol. 29, pp. 65-73, 1993.

[21] C. Rajendran, D. Chaudhuri, "A flowshop scheduling algorithm to minimize total flowtime," *J. Oper. Res. Soc. of Japan*, vol. 34, pp. 28-45, 1991.

[22] C. Rajendran, D. Chaudhuri, "An efficient heuristic approach to the scheduling of jobs in a flowshop," *Eur. J. Oper. Res.*, vol. 61, pp. 318-325, 1991.

[23] C. Rajendran, H. Ziegler, "An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs," *Eur. J. Oper. Res.*, vol. 103, pp. 129-138, 1997.

[24] S. Sarin, M. Lefoka, "Scheduling heuristic for the n-job m-machine flow shop," *OMEGA*, vol. 21, no. 2, pp. 229-234, 1993.

[25] M. Ben-Daya, M. Al-Fawzan, "A tabu search approach for the flow shop scheduling problem," *Eur. J. Oper. Res.*, vol. 109, pp. 88-95, 1998.

[26] J. Grabowski, J. Pempera, "New block properties for the permutation flow-shop problem with application in TS," *J. Oper. Res. Soc.*, vol. 52, pp. 210-220, 2001.

[27] J. Grabowski, M. Wodecki, "A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion," *Comp. & Oper. Res.*, vol. 31, no. 11, pp. 1891-1909, 2004.

[28] E. Nowicki, C. Smutnicki, "A fast tabu search algorithm for the permutation flowshop problem," *Eur. J. Oper. Res.*, vol. 91, pp. 160-175, 1996.

[29] C. Reeves, "Improving the efficiency of tabu search for machine sequencing problem," *J. Oper. Res. Soc.*, vol. 44, no. 4, pp. 375-382, 1993.

[30] T. Murata, H. Ishibuchi, H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Comp. & Ind. Eng.*, vol. 30, no. 4, pp. 1061-1071, 1996.

[31] C. Reeves, "A genetic algorithm for flowshop sequencing," *Comp. & Oper. Res.*, vol. 22, no. 1, pp. 5-13, 1995.

[32] C. Reeves, T. Yamada, "Genetic algorithms, path relinking and the flowshop sequencing problem," *Evolu. Comp.*, vol. 6, pp. 45-60, 1998.

[33] H. Ishibuchi, S. Misaki, H. Tanaka, "Modified simulated annealing algorithms for the flow shop sequencing problems," *Eur. J. Oper. Res.*, vol. 81, pp. 388-398, 1995.

[34] F. Ogbu, D. Smith, "The application of the simulated annealing algorithm to the solution of the n/m/C_{max} flowshop problem," *Comp. & Oper. Res.*, vol. 17, no. 3, pp. 243-253, 1990.

[35] F. Ogbu, D. Smith, "Simulated annealing for the permutation flow-shop problem," *OMEGA*, vol. 19, pp. 64-77, 1991.

[36] I. Osman, C. Potts, "Simulated annealing for permutation flow shop scheduling," *OMEGA*, vol. 17, no. 6, pp. 551-557, 1989.

[37] Z. Lian, X. Gu, B. Jiao, "A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan," *Chaos, Soli. & Frac.*, in press.

[38] Z. Lian, X. Gu, B. Jiao, "A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan," *Appl. Math. & Comp.*, vol. 175, pp. 773-785, 2006.

[39] C.J. Liao, C.T. Tseng, P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Comp. & Oper. Res.*, vol. 34, pp. 3099-3111, 2007.

[40] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total

flowtime minimization in the permutation flowshop sequencing problem," *Eur. J. Oper. Res.*, vol. 177, pp. 1930–1947, 2007.

[41] M. Dorigo, V. Maniezzo, A. Colorni, "The ant system: Optimization by a colony of cooperating agents," *IEEE Trans. on Sys., Man and Cyber.*, Part B, vol. 26, pp. 29-41, 1996.

[42] M. Dorigo, "Optimization, learning and natural algorithm," (in Italian), Ph.D. thesis, DEI, Politecnico di Milano, Itally, 1992.

[43] T. Stuetzle, "An ant approach for the flow shop problem," in *Proc. of the sixth Eur. Congress on intelligent techq. & soft comp., (EUFIT '98)*, Aachen: Verlag Mainz 3, pp. 1560-1564, 1998.

[44] C. Rajendran, H. Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs," *Eur. J. Oper. Res.*, vol. 155, pp. 426-438, 2004.

[45] K.C. Ying, C.J. Liao, "An ant colony system for permutation flow-shop sequencing," *Comp. & Oper. Res.*, vol. 31, pp. 791-801, 2004.

[46] C. Rajendran, H. Ziegler, "Two ant-colony algorithms for minimizing total flowtime in permutation flowshops," *Comp. & Ind. Eng.*, vol. 48, pp. 789-797, 2005.

[47] Y. Gajpal, C. Rajendran, "An ant-colony optimization algorithm for minimizing the completion-time variance of jobs in flowshops," *Int. J.Prod. Econ.*, vol. 101, pp. 259-272, 2006.

[48] M. Dorigo, L.M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. on Evolu. Comp.*, vol. l, pp. 53-66, 1997.

[49] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, pp. 278-285, 1993.