ACO-LS Algorithm for Solving No-wait Flow Shop Scheduling Problem

Ong Andre Wahyu Riyanto[1,*] and Budi Santosa[2]

[1]Department of Industrial Engineering, University of Wijaya Putra
Jl.Raya Benowo No. 1-3, Surabaya-60197, Indonesia
[2]Department of Industrial Engineering, Institut Teknologi Sepuluh Nopember,
Kampus ITS, Sukolilo, Surabaya, 60111, Indonesia
ongandre@uwp.ac.id

Abstract. In this paper, we propose a metaheuristic approach for the no-wait flow shop scheduling problem with respect to the makespan criterion. In the literature, this problem is known NP-hard type. In the literature, several algorithms have been proposed to solve this problem. We propose a hybridization of ant colony optimization (ACO) algorithm with local search (LS) in order to solve this scheduling problem, and then we call this as ACO-LS algorithm. This local search technique contributes to improve the quality of the resulting solutions. In addition, the mechanism of insert-remove technique is developed to help the searching of solution escape from the local optimum. The proposed algorithm is tested with the 31 well-known flow shop benchmark instance. The computational results based on well-known benchmarks and statistical performance comparisons are also reported. It is shown that the proposed ACO-LS algorithm is more effective than hybrid differential evolution (HDE) algorithm [Qian B., et.al, Computer & Industrial Engineering, 2009].

Keywords: Metaheuristic, No-wait flow shop scheduling, Makespan, Ant Colony Optimization, Local search.

1 Introduction

Production scheduling is one of the critical issues in the planning and manufacturing process (Pezella, et al., 2008). Scheduling problem focused on how to allocate machines to perform a collection of activities in a period of time in order to optimize a certain objective (Pinedo, 2012). In this paper, we study with the basic *n*-job *m*-machine no-wait flow shop scheduling problem. In a no-wait flow shop scheduling problem, there are *n* jobs in which each job has *m* operations and must be processed on a set of series machines continually. Once a job is started on the first machine, it must be processed through all machines without any interruption. No-wait flow shop scheduling problem is a kind of scheduling problem which has important applications including chemical processing (Rajendran, 1994), food processing (Hall and Sriskandarayah, 1996), steel production and pharmaceutical processing (Grabowski & Pempera, 2000). Given the processing time of each job in each

R. Intan et al. (Eds.): ICSIIT 2015, CCIS 516, pp. 89–97, 2015.
DOI: 10.1007/978-3-662-46742-8_8

machine, the no-wait flow shop scheduling problem is to find a set of schedules to optimize a certain objective.

In this study, we consider the makespan as optimization objective. This problem is denoted as $F|prmu, no-wait|Cmax$. The no-wait flow shop scheduling problem with single objective is NP-hard (Garey and Johnson, 1979). Therefore, many researchers are interested to investigate the finding of the near optimal solution by applying heuristic and metaheuristic algorithms which can find the near optimal solution in reasonable computational time.

The no-wait flow shop scheduling problem with makespan criterion has attracted the attention by many researchers. Qian, et. al (2009) proposed HDE for the no-wait flow shop scheduling problem with the makespan criterion and showed that the solution obtained by their HDE algorithms are superior to the ones given by other previous algorithms. Laha and Sapkal (2011) proposed constructive heuristic for the no-wait flow shop scheduling problem with the total flow time criterion and show that there is a significant improvement in solution quality over the existing heuristic. Chaudhry and Mahmood (2012) proprosed a general purpose spreadsheet based genetic algorithm (GA) for solving the no-wait flow shop scheduling problem with the minimization of makespan.

In this paper we develop a hybridization between algorithm of ant colony optimization (ACO) with technique of local search (LS) in order to solve the no-wait flow shop scheduling problem with the makespan criterion. Then we call this as ACO-LS algorithm.

The rest of the paper is organised as follows: Section 2 introduces briefly the no-wait flow shop scheduling problem, followed by problem and formulation. In Section 3, ACO is briefly presented, and Section 4, gives the proposed ACO-LS details as implemented in this paper. Experimental results are given in Section 5. The last section of the paper presents conclusions.

2 No Wait Flow Shop Scheduling Problem

The no-wait flow shop scheduling problem can be described as follows: given the processing time $p(i,j)$ of job i on machine j, each of n jobs will be sequentially processed on machine 1, 2, ..., m. At any time, each machine can process at most one job and each job can be processed on at most one machine. The sequence in which the jobs are to be processed is the same for each machine. To meet the no-wait restrictions, the completion time of a job on given machine must be equal to the starting time of the job on the next machine. In other words, there must be no waiting time between the processing of any consecutive operations of each n jobs. The problem is to find a sequence that the given criteria, i.e., makespan is optimized.

Let $= (\pi_1, \pi_2, \ \pi_n)$ denote the schedule or permutation of jobs to be processed, $L(\pi_{j-1}, \pi_j)$ the minimum delay on the first machine between the start of job π_j and π_{j-1} restricted by the no-wait constraint. Then L can be calculated as follows (Qian, et. al, 2009):

$$L(\pi_{j-1}, \pi_j) = p(\pi_{j-1}, 1) + max\left[0, \underset{2\le k\le m}{max}\left\{\sum_{h=2}^{k} p(\pi_{j-1}, h) - \sum_{h=1}^{k-1} p(\pi_j, h)\right\}\right] \quad (1)$$

Thus, the makespan can defined as follows :

$$C_{max} = \sum_{j=1}^{n} L(\pi_{j-1}, \pi_j) + P_{sum}(\pi_n) \quad (2)$$

where $P_{sum}(\pi_n) = \sum_{k=1}^{m} p(\pi_n, k)$

Fig. 1 shows an example of no-wait flow shop scheduling problem with n=4 and m=4

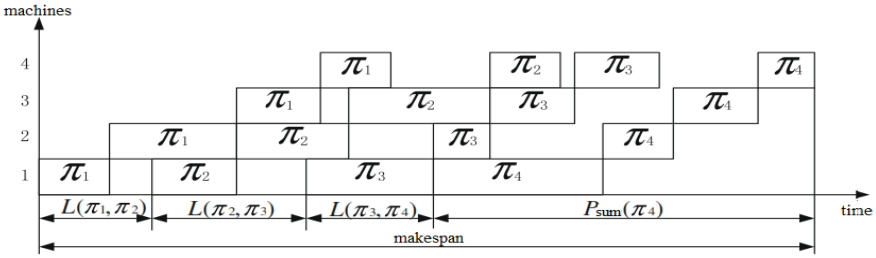

Fig. 1. No-wait Flow Shop Scheduling Problem with n=4 and m=4

3 Structure of Ant Colony Optimization

Ant Colony Optimization (ACO) is a class of metaheuristic approach proposed by Marco Dorigo in the early 1990s. The main idea ACO is to mimic the pheromone trail used by real ants for finding a nearest trail between their nest and the food sources, as shown in figure 2. Ant communication is accomplished through chemicals that called pheromones. Ants communicate to one another by laying down pheromones along their trails. Other ants perceive the presence of pheromone and tend to follow paths where pheromone concentration is higher. New pheromone will be released on the chosen path, which makes it more attractive for next ants. Shortly, all ants will select the shortest path. This analogy was applied to find the best solution to the problem of combinatorial optimization such as scheduling problem.

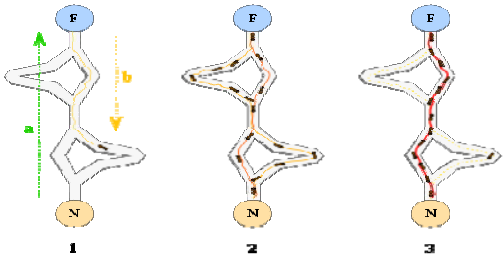

Fig. 2. Basic ant behavior: 1)Ants in a pheromone trail between nest and food. 2) Ants find paths to go around the obstacle. 3) A new pheromone trail is formed along the shorter path.

The general procedure of ACO algorithm is described as follows :

3.1 Heuristic Information

The heuristic information value is also initialized at the initialization step. The use of heuristic information to direct the ants' probabilistic solution construction is important because it provide problem specific knowledge. Heuristic information used in this study is the distance between two jobs SPIRIT (Sequencing Problem Involving a Resolution by Integrated Taboo Search Techniques) rule presented by Widmer and Hertz (1989). We modify SPIRIT method for no-wait flow shop scheduling problem. The distance between the start of job π_j and π_{j-1} , $d(\pi_{j-1}, \pi_j)$ is given by the following equation:

$$d(\pi_{j-1}, \pi_j) = L(\pi_{j-1}, \pi_j) \tag{3}$$

$$\eta(\pi_{j-1}, \pi_j) = 1/d(\pi_{j-1}, \pi_j) \tag{4}$$

3.2 Solution Construction

In the iterative step, an ants colony determines starting jobs. Each ant repeatedly applies the state transition rule to select the next processing job up to a complete schedule is formed. When building a schedule, both the heuristic information and pheromone amount are used to choose the next job. While constructing the schedule, an ant also decreases the amount of pheromone between selected jobs by applying the local updating rule to vary other ants schedule and to avoid in leading to local optima.

While finding appropriate solution, after the ant *k* chooses the next job to move to by applying the state transition rule, selected job is added into tabu list. Until the last job is selected, the procedure is repeated.

3.3 State Transition Rule

In the process of schedule constructed, the ant *k* in job *i* selects the job *j* ($i \neq j$) to move by applying the following state transition rule:

$$j = \begin{cases} \arg\max_{u \in S_k(i)} \left\{ [\tau(i,u)]^{\alpha} \left[\eta(i,u) \right]^{\beta} \right\} & \text{if } q \leq q_0 \\ J & \text{otherwise} \end{cases} \tag{5}$$

where, $\tau(i,u)$ is the amount of pheromone trail on edge (i,u). $Where\ \eta(i,u) = 1/\delta(i,u)$ is the inverse of the distance $\delta(i,u)$ between job *i* and job *u* denotes the reciprocal of a cost measure between nodes *i* and *u*. In the no-wait flow shop scheduling problem, $\delta(i,u)$ is identical with $d(\pi_{j-1}, \pi_j)$. $S_k(i)$ is the set of feasible jobs to be selected by ant *k* in job *i*. It is clear that the set of feasible jobs not contained in tabu. α is a parameter that allow a user to control the relative importance of pheromone trail $(\alpha > 0)$. β is a parameter that determines the relative importance of heuristic information. $(\beta > 0)$, q is a value chosen randomly with uniform probability in

[0,1] and q_0 is a parameter that determines the relative importance of exploitation versus exploration $(0 \leq q_0 \leq 1)$. J is a random variable selected according to the following random-proportional rule probability distribution, which is the probability with that ant k chooses to move from job i to job j:

$$p_{k(i,j)} = \begin{cases} \frac{[\tau(i,j)]^{\alpha}\left[\eta(i,j)\right]^{\beta}}{\sum_{u \in S_k(i)}[\tau(i,u)]^{\alpha}\left[\eta(i,u)\right]^{\beta}} & \text{if } j \in S_k(i) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

3.4 Pheromone Trail Update

While an ant construct a schedule, an ant decreases the level of pheromone trail between selected jobs by applying the update rule. The update rules consists of two terms: the first, is the evaporation of the existing pheromone (local updating rule) ; the second, is the amount of added pheromone on the trail (global updating rule). The local updating rule is formulated as follows :

$$\tau(i,j) = (1 - \rho l) \cdot \tau(i,j) + \rho l \cdot \tau_0 \quad (7)$$

where, τ_0 is the initial pheromone level and ρl $(0 < \rho l < 1)$ is the local pheromone evaporation parameter.

After all ants completed their schedules then global updating rule is performed. Global updating rule provides a greater amount of pheromone trail for between adjacent jobs of the best schedule. The pheromone trail level is updated as follows:

$$\tau(i,j) = (1 - \rho g) \cdot \tau(i,j) + \rho g \cdot \Delta\tau(i,j) \quad (8)$$

where,

$$\Delta\tau(i,j) = \begin{cases} (L_b)^{-1} & if\ (i,j) \epsilon \text{ best schedule} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

In the Eq. (8), $\rho g (0 < \rho g < 1)$ is the evaporation parameter of global updating rule and L_b is the objective function value of the best schedule until the current iteration.

4 Proposed Ant Colony Optimization – Local Search (ACO-LS)

The main advantages such as ACO metaheuristic approach is able to find the near optimal solution in reasonable computational time.

The structure of proposed ACO can be describe as follows :

1. **Initialize** the pheromone trails, heuristic information, and parameters
2. **Iteration :**
   2.1 Ant colonies determine the starting jobs ;
   2.2 Each ant constructs a complete solution ;

Repeat
Applying state transition rule for selecting the next jobs
Until complete jobs

2.3 Improve the solution by local search

3. **Cycle**. If the maximum number of iterations is reached, then the iteration stops. If not then go back to step 2.
4. Return best solution found

Local Search

ACO algorithm can perform better to find solution when combined with a local search algorithm (Yagmahan and Yenisey, 2010). In this study, we propose a local search procedure as follows:

Step 1:

Determine one best solution of job sequence π_{i_0}

Step 2:

Choose randomly u and v, where $u \neq v$; $\pi_i = insert(\pi_i, u, v)$.

Step 3:

Set loop=0

While $loop < n * (n-1)$

$count = 0$;

max_$method = 3$;

While $count <$ max _$method$

Choose randomly u and v, $u \neq v$;

$\pi_{i\_1} = insert(\pi_i, u, v)$;

if $f(\pi_{i\_1}) < f(\pi_i)$,

then $\pi_i = \pi_{i\_1}$; $count = 0$;

if $f(\pi_{i\_1}) > f(\pi_i)$,

then $\pi_i = \pi_i$; $count = count + 1$;

$choose\ randomly \quad u, v, r, s$; $u \neq v \neq r \neq s$;

$\pi_{i\_1} = remove(\pi_i, u, v) \rightarrow remove(\pi_i, r, s)$;

if$(\pi_{i_1}) < f(\pi_i)$,

then $\pi_i = \pi_{i\_1}$; $count = 0$;

if $f(\pi_{i\_1}) > f(\pi_i)$,

then $\pi_i = \pi_i$; $count = count + 1$;

choose randomly t, u, v, q, r, s; t≠ u≠ $v \neq q \neq$ r≠ s;

$\pi_{i\_1} = remove(\pi_i, t, u, v) \rightarrow remove(\pi_i, q, r, s)$;

if $f(\pi_{i\_1}) < f(\pi_i)$,

then $\pi_i = \pi_{i\_1}$; $count = 0$;

if $f(\pi_{i\_1}) > f(\pi_i)$,

then $\pi_i = \pi_i$; $count = count + 1$;

end

end

Step 3:

if $f(\pi_i) < f(\pi_{i\_0})$,

then $\pi_{i\_0} = \pi_i$;

if $f(\pi_i) < f(\pi_{i\_0})$,

then $\pi_{i\_0} = \pi_{i\_0}$;

Where : n is set of all the job. π_i is sequence of n job. While t, u, v, q, r, s is the position of a job in the sequence π_i .

Termination Criteria

Termination criteria using the maximum number of iterations

Table 1. Comparison of HDE and ACO-LS

		RAJ	HDE			ACO-LS		
Problem	**n,m**	**C***	**BRE**	**ARE**	**WRE**	**BRE**	**ARE**	**WRE**
Car1	11,5	8142	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.27
Car2	13,4	8242	**0.00**	**0.00**	0.06	**0.00**	**0.00**	**0.00**
Car3	12,5	8866	**0.00**	0.03	**0.08**	**0.00**	**0.00**	**0.08**
Car4	14,4	9195	**0.00**	1.15	2.41	**0.00**	**0.06**	**0.68**
Car5	10,6	9159	**0.00**	**0.00**	**0.00**	0.39	0.64	2.23
Car6	8,9	9690	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Car7	7,7	7705	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Car8	8,8	9372	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Rec01	20,5	1590	-3.71	-3.50	-3.14	**-4.03**	**-3.88**	**-3.83**
Rec03	20,5	1457	**-6.59**	-5.33	-3.77	**-6.59**	**-6.23**	**-4.99**
Rec05	20,5	1637	**-7.70**	-6.76	-5.86	-7.64	**-7.35**	**-7.20**
Rec07	20,10	2119	-3.59	-3.19	-1.27	**-3.63**	**-3.55**	**-3.44**
Rec09	20,10	2141	-4.58	-4.30	-3.60	**-4.62**	**-4.56**	**-4.31**
Rec11	20,10	1946	**-3.34**	**-3.00**	**-2.21**	**-3.34**	-2.90	**-2.21**
Rec13	20,15	2709	-5.76	-4.72	-3.32	**-6.05**	**-5.73**	**-4.89**
Rec15	20,15	2691	**-6.02**	-5.89	-5.39	**-6.02**	**-5.98**	**-6.25**
Rec17	20,15	2740	**-5.58**	**-5.55**	**-5.47**	**-5.58**	-5.50	-5.44
Rec19	30,10	3157	-7.73	-6.76	-6.34	**-9.72**	**-8.78**	**-8.01**
Rec21	30,10	3015	-4.61	-3.74	-3.32	**-6.27**	**-6.10**	**-6.07**
Rec23	30,10	3030	-8.71	-7.69	-7.19	**-10.89**	**-10.16**	**-9.92**
Rec25	30,15	3835	-4.64	-4.02	-3.60	**-6.31**	**-5.67**	**-5.44**
Rec27	30,15	3655	-3.97	-3.07	-2.71	**-6.10**	**-5.39**	**-4.89**
Rec29	30,15	3583	-6.61	-5.98	-5.44	**-8.15**	**-7.60**	**-7.23**
Rec31	50,10	4631	-3.95	-3.22	-2.92	**-5.92**	**-4.95**	**-3.67**
Rec33	50,10	4770	-2.37	-1.82	**-1.26**	**-3.86**	**-2.48**	-0.76
Rec35	50,10	4718	-3.98	-3.77	**-3.39**	**-5.26**	**-3.86**	-1.82
Rec37	75,20	8979	-7.57	-7.25	**-6.97**	**-8.56**	**-7.39**	-6.57
Rec39	75,20	9158	-3.53	-3.07	-2.86	**-5.43**	**-4.36**	**-2.89**
Rec41	75,20	9344	-5.15	-4.84	-4.54	**-6.93**	**-6.09**	**-4.89**
Hel1	100,10	780	-5.26	-4.73	-4.36	**-6.92**	**-5.81**	**-5.09**
Hel2	20,10	189	**-5.29**	-4.23	-1.59	**-5.29**	**-4.92**	**-4.30**
Average			-3.88	-3.40	-2.84	**-4.60**	**-4.15**	**-3.58**

5 Experimental Result

In this section, the results of computational experiments performed are presented to evaluate the performance of proposed ACO-LS. The performance of the proposed ACO-LS is tested with numerical simulations are carried out with 31 well-studied benchmark contributed to the OR-Library (http://www.people.brunel.ac.uk/-mastjjb/jeb/info.html). The first 8 problems are called Car1, Car 2 through Car8 by Carlier (1978). The second 21 problem are called Rec01, Rec03 through Rec41by Reeves (1995). The last two problem Hel1 and Hel2 by Heller (1960). The performance of ACO-LS using test problem is compared with HDE algorithm proposed by Qian, et. al (2009). All algorithms are coded in matlab 2009a and run on PC with 2 GHz Intel Core 2 Duo processor and 2 GB RAM memory. For fair comparison, HDE algorithm are executed on the same PC and let HDE algorithm run at the same time as ACO-LS. Each benchmark is independently run 20 time for comparison. Where C* denotes the references makespan produced by the famous RAJ heuristic (Rajendran, 1994). BRE denotes the best relative percentage error to C*. ARE denotes the average relative percentage error to C*, and WRE denotes the worst relative percentage error to C*. The statistical results are reported in table 1.

It can be seen from table 1 that the BRE, ARE, and WRE values performed by ACO-LS are much better than those obtained by HDE almost for all benchmark.

6 Conclusion

In this paper, we have presented an ACO-LS algorithm for the no-wait flow shop scheduling problem with the objective of minimizing makespan criterion. Based on the computational experimentation, the proposed ACO-LS algorithm gives comparable performance as that of HDE algorithm for small problem sizes, whereas, there is significant improvement in solution quality for large problem sizes. To the best our knowledge, this is the first report to apply ACO-LS algorithm for no-wait flow shop scheduling problems with makespan. In future work, we will extend the ACO-LS algorithm to the scheduling problem with two or more objectives.

Acknowledgments. This research is funded by DP2M Ditjen Dikti with the grants scheme of Pekerti (Nomor: 009/SP2H/P/K7/KM/2014, 19 May 2014). We would like to thank you for DP2M Ditjen Dikti, Kopertis 7, and LPPM-UWP.

References

1. Chaudhry, I.A., Mahmood, S.: No-wait Flowshop Scheduling Using Genetic Algorithm. In: Proceedings of the World Congress on Engineering, vol. 3 (2012)
2. Carlier, J.: Ordonnancements a contraintes disjonctives. RAIRO-Operations Research-Recherche Opérationnelle 12(4), 333–350 (1978)
3. Grabowski, J., Pempera, J.: Sequencing of jobs in some production system. European Journal of Operational Research 125(3), 535–550 (2000)

4. Garey, M.R., Johnson, D.S.: Computers and intractability, aguide to the theory of NP-completeness. Freeman, San Francisco (1979)
5. Hall, N.G., Sriskandarajah, C.: A survey of machine scheduling problems with blocking and no-wait in process. Operations Research 44(3), 510–525 (1996)
6. Heller, J.: Some numerical experiments for an M× J flow shop and its decision-theoretical aspects. Operations Research 8(2), 178–184 (1960)
7. Laha, D., Sapkal, S.U.: An efficient heuristic algorithm for m-machine no-wait flow shops. In: Proceedings of the International MultiConference of Engineers and Computer Scientists, vol. 1 (2011)
8. Pinedo, M.L.: Scheduling: theory, algorithms, and systems. Springer (2012)
9. Pezzella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the flexible job-shop scheduling problem. Computers & Operations Research 35(10), 3202–3212 (2008)
10. Qian, B., Wang, L., Hu, R., Huang, D.X., Wang, X.: A DE-based approach to no-wait flow-shop scheduling. Computers & Industrial Engineering 57(3), 787–805 (2009)
11. Rajendran, C.: A no-wait flowshop scheduling heuristic to minimize makespan. Journal of the Operational Research Society, 472–478 (1994)
12. Reeves, C.R.: A genetic algorithm for flowshop sequencing. Computers & Operations Research 22(1), 5–13 (1995)
13. Rajendran, C.: A no-wait flowshop scheduling heuristic to minimize makespan. Journal of the Operational Research Society, 472–478 (1994)
14. Widmer, M., Hertz, A.: A new heuristic method for the flow shop sequencing problem. European Journal of Operational Research 41(2), 186–193 (1989)
15. Yagmahan, B., Yenisey, M.M.: A multi-objective ant colony system algorithm for flow shop scheduling problem. Expert Systems with Applications 37(2), 1361–1368 (2010)