ICIC Express Letters
Volume **16**, Number **4**, April **2022**

 ISSN 1881-803X
pp. **419–424**

A NO-WAIT FLOW SHOP SCHEDULING WITH TOTAL STRETCH MINIMIZATION

Suk-Hun Yoon

Department of Industrial and Information Systems Engineering
Soongsil University
369 Sangdo-ro, Dongjak-gu, Seoul 06978, Korea
yoon@ssu.ac.kr

Received August 2021; accepted October 2021

Abstract. *A hybrid genetic algorithm (HGA) is proposed to minimize total stretch in a no-wait flow shop scheduling problem. The stretch of a job is the flow time of the job divided by its processing time. HGA combines local search with genetic algorithm (GA) to reduce the possibility of premature convergence. The performance of HGA is compared with that of GA and the experimental results show that HGA works well for this type of problem.*
Keywords: Scheduling, No-wait flow shop, Total stretch, Genetic algorithm, Local search

1. **Introduction.** In a no-wait flow shop scheduling problem, a job has to be continuously processed from start to completion of the job without any interruption and waiting [1]. Various manufacturing environments of no-wait process include steel, plastic, chemical, pharmaceutical, and food-processing industries [2,3].

Ye et al. [4] proposed an average idle time heuristic for no-wait flow shop to minimize the makespan. Engin and Güclü [5] developed a new hybrid ant colony algorithm for n-job m-machine no-wait flow shop scheduling problems to minimize the makespan. Zhao et al. [6] provided a discrete water wave optimization algorithm for a no-wait flow shop scheduling problem to the makespan.

Ding et al. [7] considered a no-wait flow shop scheduling problem to minimize the makespan. They presented an iterated greedy algorithm modified by introducing a Tabu-based insertion strategy for the problem. Lin and Ying [8] formulated a no-wait flow shop scheduling problem with makespan minimization as an asymmetric traveling salesman problem.

Nagano et al. [9] addressed a no-wait flow shop scheduling problem with sequence dependent setup times to minimize the total flow time. They presented a new constructive heuristic in order to obtain good approximate solutions in a short computing time. Cheng et al. [10] considered a mixed no-wait flow shop scheduling problem with sequence dependent setup times to minimize the makespan. They presented a mixed integer linear programming model and proposed a pairwise iterated greedy algorithm.

Ying and Lin [11] investigated no-wait flow shop scheduling problems with sequence independent and sequence dependent setup times to minimize the makespan. They proposed a two-phase metaheuristic algorithm. In the first phase, an approximate solution was produced by NEH heuristic and LKH algorithm. In the second phase, the solution acquired in the first phase was used as an upper bound and an optimal solution was obtained by using the Gurobi optimizer.

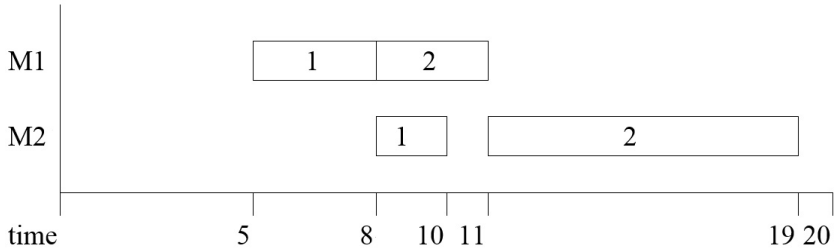
DOI: 10.24507/icicel.16.04.419

Allahverdi et al. [12] addressed a no-wait flow shop scheduling problem with separate setup times to minimize total tardiness with an upper bound on makespan. They established a dominance rule and proposed a new simulated annealing algorithm utilizing block insertion and block exchange operators.

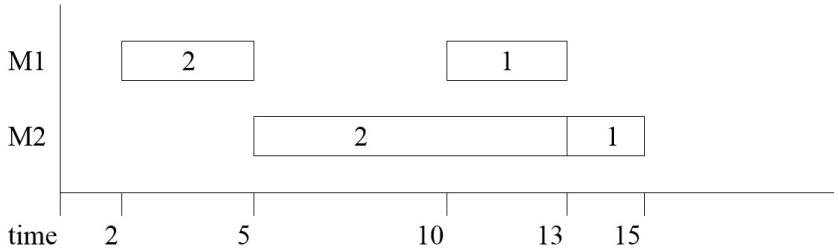
Samarghandi [13] developed a particle swarm optimization for a no-wait flow shop scheduling problem with due dates to minimize makespan. Gao et al. [14] considered a two-machine no-wait permutation flow shop scheduling problem with common due date and controllable job processing times to minimize total earliness, tardiness, common due date cost and total resource cost. They showed that the problem could be solved in polynomial time.

Dong et al. [15] introduced a no-wait two-stage flow shop scheduling problem with the first stage machine having multitask flexibility to minimize the makespan. They presented a linear time algorithm with an approximation ratio 13/8. Koulamas and Kyparisis [16] considered a no-wait flow shop scheduling problem with rejection. They presented a backward $O(n^3)$ dynamic programming to minimize the sum of total completion time and total rejection cost with ordered jobs.

This paper addresses an n-job, m-machine no-wait flow shop scheduling problem in which the objective is to minimize the total stretch. Stretch (response time or slowdown) of a job is defined as the ratio of its flow time to its processing time [17,18]. Stretch is a fairly natural criterion in that jobs of large processing time must be prepared to wait longer than the ones necessary for less time [19]. The idea of stretch is illustrated by the two-job, two-machine flow shop shown in Figure 1. The processing times of jobs 1 and 2 are all 3 time units on machine 1. The processing times of jobs 1 and 2 are 2 and 8 time units on machine 2, respectively. The release times of jobs 1 and 2 are 5 and 2 time units, respectively. If job 1 is processed before job 2, total flow time and total stretch will be 22 and 2.55 time units, respectively (schedule 1). On the contrary, when job 2 is processed first, total flow time is reduced to 21 time units, but total stretch is increased to 3 time units (schedule 2).



(a) Schedule with smaller total stretch (schedule 1)



(b) Schedule with smaller total flow time (schedule 2)

FIGURE 1. Two schedules for a two-machine flow shop (the numerals represent jobs)

For job j, $j = 1, \ldots, n$, let r_j be the release time, p_{ij} the processing time on machine i, $i = 1, \ldots, m$. Let $p_j = p_{1j} + p_{2j} + \cdots + p_{mj}$. If the completion time of job j on machine i is $c_{i,j}$, then the stretch is $p_j = s_j = (c_{mj} - r_j)/p_j$.

For a given sequence σ, the problem can be formulated as follows:

$$\text{minimize} \quad z(\sigma) = \sum_{j=1}^{n} \frac{(c_{mj} - r_j)}{p_j}$$

$$\text{subject to} \quad c_{i,\sigma(j)} - p_{i,\sigma(j)} = c_{i-1,\sigma(j)}, \quad i = 2, 3, \ldots, m,\ j = 1, 2, \ldots, n \tag{1}$$

$$c_{i,\sigma(j)} - p_{i,\sigma(j)} \geq c_{i,\sigma(j-1)}, \quad i = 1, 2, \ldots, m,\ j = 2, 3, \ldots, n \tag{2}$$

$$c_{i,\sigma(j)} - p_{i,\sigma(j)} \geq \gamma_{\sigma(j)}, \quad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n \tag{3}$$

where $\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(n)\}$ is a job sequence, and $\sigma(j)$ represents the jth job in the sequence.

Constraint set (1) insures that once a job is released from a machine, its processing on the next machine begins immediately. Constraint set (2) indicates that only one job at most can be processed on each machine at a time. Constraint set (3) states that jobs are available after their release times.

The problem of minimizing the total stretch with different job release times is NP-hard even for a single machine [20]. Thus, the problem to be solved seems to require an amount of time that grows exponentially with the problem size and it becomes quickly impractical even for current-generation computers. Metaheuristics such as simulated annealing, tabu search and genetic algorithms are useful for solving difficult problems. They have been successful to search for global or near-optimal solutions in many application areas [21]. GA is a stochastic search method designed to search large and complex spaces by exploiting best solution and exploring the search space. GA may fail for various reasons including premature convergence regardless of its desirable characteristics [22].

In the next section, a hybrid genetic algorithm is proposed for the no-wait flow shop scheduling problem to minimize total stretch. HGA incorporates local search into GA in order to restrain tremendous copies of a super-chromosome and relieve the premature convergence. In Section 3, the results of computational experiments of HGA and GA are provided. Summary and future researches are given in Section 4.

2. Hybrid Genetic Algorithm.

GA starts by generating an initial population of m chromosomes, which are abstract representation of solutions. The fitness of each chromosome is determined by a fitness function. Selection is done randomly with a probability depending on the relative fitness of the chromosomes to construct a mating pool of m chromosomes. A pair of chromosomes from the mating pool is chosen and goes through recombination (crossover) and mutation to produce two offspring. A new population of m chromosomes created in this manner constitutes the next generation and this process is repeated until a stopping criterion is reached [23,24].

HGA randomly generates an initial population of m chromosomes in order to search unbiased sampling of the space. The chromosomes are encoded by a permutation representation. HGA uses the stochastic remainder selection procedure without replacement. The procedure determines $E(l)$, the expected number of copies of chromosome l. Only $\lfloor E(l) \rfloor$ copies of chromosome l are assigned to the mating pool. If the number of chromosomes in the mating pool is less than m, Bernoulli trials with success probabilities $P_s(l) = E(l) - \lfloor E(l) \rfloor$ are performed to chromosome l one by one. When chromosome l is selected, $P_s(l)$ is reduced to 0. This process continues until m chromosomes are selected.

With a high selection pressure, the copies of a super-chromosome reproduce much more quickly than the others. The exploration of the search space seems to be limited to a search randomly centered on the super-chromosome and there will be a great risk of premature convergence [25]. To prevent this situation, HGA adopts a generalized

pairwise interchange (GPI) to be applied to worst chromosome to find best neighborhood. Replacing the worst chromosome by the best neighborhood inhibits increasing the rate of premature convergences.

HGA adopts PMX, in which two crossover points are picked at random. Genes between the points are exchanged and alleles of the genes are used to construct the match table. The genes in the first and last sections are exchanged if they are included in the match table. For example, suppose that A and B are the chromosomes chosen for crossover such that A = (6 3 4 5 1 2) and B = (3 2 6 1 4 5), and the two crossover points are 2 and 5. First, the genes between two crossover points are swapped (4 5 1 of A and 6 1 4 of B). Second, the genes before the first crossover point and after the second crossover points are exchanged according to the match table ($4 \leftrightarrow 6$), ($5 \leftrightarrow 1$) and ($1 \leftrightarrow 4$), which results in ($5 \leftrightarrow 6$). Then, the resulting chromosomes by PMX are $A' = (5\ 3\ 6\ 1\ 4\ 2)$ and $B' = (3\ 2\ 4\ 5\ 1\ 6)$.

HGA adopts the adjacent swap method as mutation operator, which exchanges a job with the next job in the job sequence if the job is selected with mutation rate. If the last job is to be mutated, it is exchanged with the first job in the job sequence. A flow chart of the HGA is shown in Figure 2.

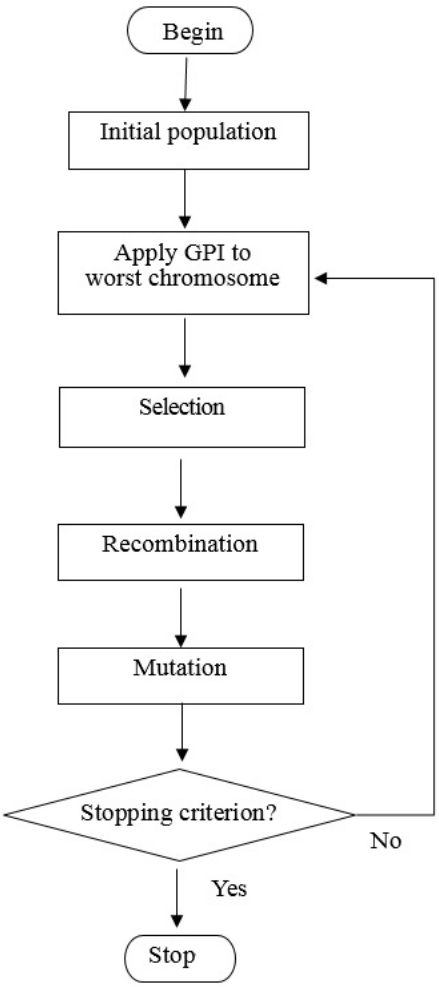


FIGURE 2. Flow chart of HGA

3. **Computational Experiments.** The HGA and GA were coded in Visual FORTRAN and ran on an Intel Core i7 CPU@3.4 GHz PC. The test problems were generated randomly and provided in Table 1. Processing times and release times of jobs were generated according to the integer uniform distributions.

Table 1. Data used to generate test problems (all data are integers)

Data	Value
Number of jobs (NJ)	5, 7, 10, 15, 20, 30
Number of machines (NM)	2, 3, 4, 5
Job processing times on machines	Uniform$(1, 31)$
Job release times	Uniform$(1, 6)$

The experiments were done by a preliminary test and a main test. In a preliminary test, 5 test problems of different sizes were solved to tune several control parameters of GA and HGA. The best combination of the parameters was a population size of 100, a total of 100 generations, a crossover rate of 1.0, and a mutation rate of 0.01.

In the main test, 9 test problems were generated for each problem size. HGA achieved optimal solutions for all small size problems (5 and 7 jobs and 2-5 machines). For medium size (10 and 15 jobs and 2-5 machines) and large size (20 and 30 jobs and 2-5 machines) problems, the results of HGA and GA were shown in Table 2. Based on these results, HGA provided 5.12%, 10.52%, 15.53%, and 25.62% improvements, on the average, respectively in regard to GA.

Table 2. Results for medium and large size flow shop problems

No. of jobs	No. of machines	GA method Avg. obj. value (z_g)	HGA method Avg. obj. value (z_h)	%Dev $(z_g - z_h/z_g) \times 100$
10	2	30.53	29.13	4.59
	3	26.80	24.27	9.44
	4	23.30	22.37	3.99
	5	20.81	20.30	2.45
15	2	68.04	56.07	17.59
	3	49.20	43.94	10.69
	4	41.12	37.36	9.14
	5	39.18	37.36	4.65
20	2	104.25	82.40	20.96
	3	91.21	75.29	17.45
	4	72.01	62.81	12.78
	5	59.74	53.21	10.93
30	2	237.91	169.52	28.75
	3	176.31	136.72	22.45
	4	146.82	115.45	21.37
	5	146.82	102.90	29.91
Average		86.90	66.82	14.20

4. **Conclusions.** This paper has considered the problem of minimizing the total stretch in the n-job, m-machine no-wait flow shop. Since this problem is intractable, it requires significant computational effort to solve the problems with large n. HGA is proposed to prohibit the premature convergence of GA and maintain the search power by adopting GPI. Extensive computational experiments have been conducted to compare the performance of HGA with that of GA. These results show that the average improvement of HGA over GA is 14.2% for different size problems.

Over the last years, several metaheuristics such as iterated local search, guided local search and variable neighborhood search have been developed. Hybrid optimization approaches have become increasingly popular for addressing hard optimization problems.

Development of most appropriate hybridizing these metaheuristics in a particular situation will be a future research.

REFERENCES

[1] H. H. Miyata, M. S. Nagano and J. N. D. Gupta, Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization, *Computers & Industrial Engineering*, vol.135, pp.79-104, 2019.

[2] A. Allahverdi, A survey of scheduling problems with no-wait in process, *European Journal of Operational Research*, vol.255, no.3, pp.665-686, 2016.

[3] S. U. Sapkal and D. Laha, A heuristic for no-wait flow shop scheduling, *International Journal of Advanced Manufacturing Technology*, vol.68, pp.1327-1338, 2013.

[4] H. Ye, W. Li and A. Abedini, An improved heuristic for no-wait flow shop to minimize makespan, *Journal of Manufacturing Systems*, vol.44, pp.273-279, 2017.

[5] O. Engin and A. Güclü, A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems, *Applied Soft Computing*, vol.72, pp.166-176, 2018.

[6] F. Zhao, H. Liu, Y. Zhang, W. Ma and C. Zhang, A discrete Water Wave Optimization algorithm for no-wait flow shop scheduling problem, *Expert Systems with Applications*, vol.91, pp.347-363, 2018.

[7] J.-Y. Ding, S. Song, J. N. D. Gupta, R. Zhang, R. Chiong and C. Wu, An improved iterated greedy algorithm with a Tabu-based reconstruction strategy for the no-wait flowshop scheduling problem, *Applied Soft Computing*, vol.30, pp.604-613, 2015.

[8] S.-W. Lin and K.-C. Ying, Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics, *Omega*, vol.64, pp.115-125, 2016.

[9] M. S. Nagano, H. H. Miyata and D. C. Araújo, A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times, *Journal of Manufacturing Systems*, vol.36, pp.224-230, 2015.

[10] C.-Y. Cheng, K.-C. Ying, S.-F. Li and Y.-C. Hsieh, Minimizing makespan in mixed no-wait flowshops with sequence-dependent setup times, *Computers & Industrial Engineering*, vol.130, pp.338-347, 2019.

[11] K.-C. Ying and S.-W. Lin, Minimizing makespan for no-wait flowshop scheduling problems with setup times, *Computers & Industrial Engineering*, vol.121, pp.73-81, 2018.

[12] A. Allahverdi, H. Aydilek and A. Aydilek, No-wait flowshop scheduling problem with separate setup times to minimize total tardiness subject to makespan, *Applied Mathematics and Computation*, vol.365, DOI: 10.1016/j.amc.2019.124688, 2020.

[13] H. Samarghandi, A particle swarm optimisation for the no-wait flow shop problem with due date constraints, *International Journal of Production Research*, vol.53, no.9, pp.2853-2870, 2015.

[14] F. Gao, M. Liu, J.-J. Wang and Y.-Y. Lu, No-wait two-machine permutation flow shop scheduling problem with learning effect, common due date and controllable job processing times, *International Journal of Production Research*, vol.56, no.6, pp.2361-2369, 2018.

[15] J. Dong, H. Pan, C. Ye, W. Tong and J. Hu, No-wait two-stage flowshop problem with multi-task flexibility of the first machine, *Information Sciences*, vol.544, pp.25-38, 2021.

[16] C. Koulamas and G. J. Kyparisis, The no-wait flow shop with rejection, *International Journal of Production Research*, vol.59, no.6, pp.1852-1859, 2021.

[17] B. Berg, R. Vesilo and M. Harchol-Balter, heSRPT: Parallel scheduling to minimize mean slowdown, *Performance Evaluation*, vol.144, DOI: 10.1016/j.peva.2020.102147, 2020.

[18] K. M. Kobayashi, Improved lower bounds for online scheduling to minimize total stretch, *Theoretical Computer Science*, vol.705, pp.84-98, 2018.

[19] M. A. Bender, S. Muthukrishnan and R. Rajaraman, Approximation algorithms for average stretch scheduling, *Journal of Scheduling*, vol.7, pp.195-222, 2004.

[20] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 5th Edition, Springer, New York, 2016.

[21] B. Chopard and M. Tomassini, *An Introduction to Metaheuristics for Optimization*, Springer, New York, 2018.

[22] J. Dreo, P. Siarry, A. Petrowski and E. Taillard, *Metaheuristics for Hard Optimization*, Springer, New York, 2006.

[23] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, New York, 2008.

[24] M. Afzalirad and M. Shafipour, Design of an efficient genetic algorithm for resource-constrained unrelated parallel machine scheduling problem with machine eligibility restrictions, *Journal of Intelligent Manufacturing*, vol.29, pp.423-437, 2018.

[25] M. Gendreau and J.-Y. Potvin, *Handbook of Metaheuristics*, 3rd Edition, Springer, New York, 2019.