Management Science and Financial Engineering
Vol 20, No 2, November 2014, pp.33-37 http://dx.doi.org/10.7737/MSFE.2014.20.2.033
ISSN 2287-2043 | EISSN 2287-2361 |

Minimizing the Total Stretch in Flow Shop Scheduling

Suk-Hun Yoon*
Department of Industrial and Information Systems Engineering, Soongsil University

(Received: November 5, 2014 / Revised: November 19, 2014 / Accepted: November 21, 2014)

ABSTRACT

A flow shop scheduling problem involves scheduling jobs on multiple machines in series in order to optimize a given criterion. The flow time of a job is the amount of time the job spent before its completion and the stretch of the job is the ratio of its flow time to its processing time. In this paper, a hybrid genetic algorithm (HGA) approach is proposed for minimizing the total stretch in flow shop scheduling. HGA adopts the idea of seed selection and development in order to reduce the chance of premature convergence that may cause the loss of search power. The performance of HGA is compared with that of genetic algorithms (GAs).

Keywords: Scheduling, Flow Shop, Stretch, Genetic Algorithms

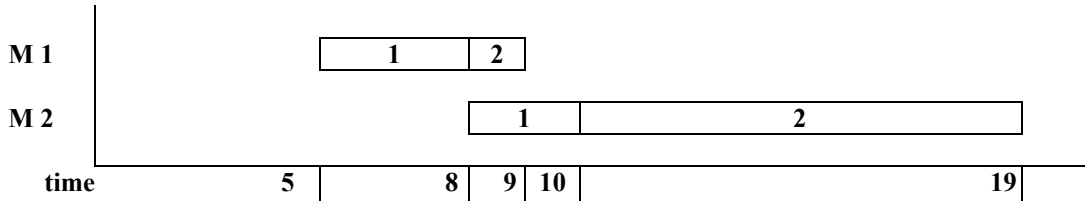
* Corresponding Author, E-mail: yoon@ssu.ac.kr

1. INTRODUCTION

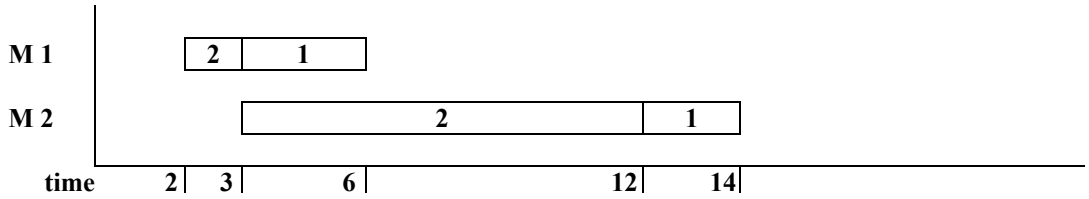
In many scheduling problems, the focus of performance measure has been on the flow time, which is defined as the amount of time that a given job spends in the system (Baker and Trietsch, 2009). Recently, alternative performance measures have been considered and among them, the stretch measure has received a lot of attention (Chan *et al*., 2006). The stretch of a job formally is defined as the ratio of its flow time to its required processing time (Bender *et al*., 2004). The stretch measure relates the jobs' waiting times to their processing times, and may reflect users' psychological expectation that, in a system with highly various job sizes, users are willing to wait longer for larger jobs (Muthukrishnan *et al*., 2005). 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 3 and 1 time units on machine 1, and 2 and 9 time units on machine 2, respectively. The release times of jobs 1 and 2 are 5 and 2 time units. If job 1 is processed before job 2, the total flow time and the total stretch will be 22 (= 5+17) and 2.7 (= 5/5+17/10) time units, respectively (schedule 1). As Figure 1(b) shows, when job 2 is processed first, the total flow time is reduced to 19 (= 10+9) time units, but the total stretch is increased to 2.8 (= 10/10+9/5) time units (schedule 2).

This paper presents a solution methodology for an n-job, m-machine flow shop scheduling problem in which the objective is to minimize the total stretch. Since the problem of minimizing the total stretch with different job release times is not tractable even for a single machine (Pinedo 2012), it is usually very difficult to find a global optimum by general local search algorithms such as the pairwise interchange methods. Recently, meta-heuristic methods such as genetic algorithms, simulated annealing, and tabu search, have been proposed to escape from local optima and search for global or near-optimal solutions and have been successful in solving combinatorial optimization problems (Dreo *et al*., 2005).

In spite of their desirable properties of GAs, basic GAs can still fail for a variety of reasons including convergence to local optima (premature convergence). The problem of premature convergence is related with the loss of genetic diversity of the individuals (solutions), which can be the cause of poor quality of the individuals (Liepins and Hilliard, 1989). In this paper, a hybrid genetic algorithm (HGA) is proposed to lessen the problem of premature convergence through the aid of more



(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 (Numerical Entries Represent Jobs)

rational genetic operators. In section 2, the minimization problem of the total stretch in the *n*-job, *m*-machine flow shop scheduling is defined. The development of HGA and its application to the flow shop scheduling problem are presented in section 3. In section 4, the results of extensive computational experiments comparing HGA and GAs are provided. Finally, a summary of main results and conclusions are provided in Section 5.

2. MINIMIZATION OF THE TOTAL STRETCH IN THE *n*-JOB, *m*-MACHINE FLOW SHOP

There is a set of jobs $J = \{1, 2, \cdots, n\}$ that has to be processed in the system consisting of m machines in series. Each machine processes jobs in the same order. For job j, $j = 1, \cdots, n$, let r_j be the release time, p_{ij} the processing time on machine i, $i = 1, \cdots, m$. Let p_j be the sum of processing times of job j ($= p_{1j} + p_{2j} + \cdots + p_{mj}$). If the completion time of job j on machine i is c_{ij}, then the stretch $s_j = (c_{mj} - r_j)/p_j$.

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

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

subject to

$$c_{i,\sigma(j)} - p_{i,\sigma(j)} \geq c_{i-1,\sigma(j)}, \quad \text{for } i = 2, \cdots, m,\ j = 1, \cdots, n \qquad (1)$$

$$c_{i,\sigma(j)} - p_{i,\sigma(j)} \geq c_{i,\sigma(j-1)}, \quad \text{for } i = 1, \cdots, m,\ j = 2, \cdots, n \qquad (2)$$

$$c_{i,\sigma(j)} - p_{i,\sigma(j)} \geq r_{\sigma(j)}, \quad \text{for } i = 1, \cdots, m,\ j = 1, \cdots, n \qquad (3)$$

Constraint set (1) states that each machine can process at most one job at the same time. Constraint set (2) establishes the relationships between completion times of any two jobs on each machine. That is, only one job at most can be processed on each machine at the same time. Constraint set (3) states that jobs are available after their release times.

3. HYBRID GA APPROACH

GAs are stochastic search methods designed to search large and complex spaces by exploitation of currently known solutions and a robust exploration of the space (Lee *et al*., 1997). GAs start with a collection (or population) of randomly selected solutions (or individuals). With the survival of the fittest philosophy, GAs select individuals in a population to form a gene pool according to their fitness values. Two individuals in the gene pool are randomly mated and go through the crossover and mutation process to produce two new solutions (or offsprings), and the solutions steadily improve from iteration (or generation) to iteration. The population size (the number of individuals in a population) remains fixed in all generations (Bhattacharyya, 1999).

The proposed HGA adopts three basic operators of GAs (selection, crossover, mutation) and combines the idea of seed selection and development, which prevents the premature convergence of many GAs and also maintains the search power of GAs. The detail procedure of HGA is explained below in comparison with GAs.

Utilization of Seed Selection to Generate Initial Population

Most scheduling problems use a permutation representation for individuals where a sequence of n jobs is defined by a permutation of integers $\{1, \cdots, n\}$. In the literature, GAs begin with randomly generated populations (Liepins and Hilliard, 1989). HGA generates most

of individuals in an initial population randomly in order to search unbiased sampling of the space, which provides robust exploitation of the current solutions. To start with some good quality of solutions, HGA fills the rest of the initial population with individuals generated by various types of local search methods based on two steps: (1) rules to generate the initial sequences such as the shortest release time (SRT), the shortest processing time on the first machine (SPTF), and the shortest total processing times (SPTT), (2) neighborhood search mechanisms such as the non-adjacent pairwise interchange (NAPI), the extraction and forward shifted reinsertion (EFSR), and the extraction and backward shifted reinsertion (EBSR). Generating too many individuals by seed selection may prevent the robust search of HGA.

Calculation of Individual's Fitness

An individual with high objective value should have a low fitness value, and vice versa. HGA transforms an objective value of individual l into its fitness value (f_{scale}) by the following equation:

$$f_{scale}(\sigma_l) = \frac{z_{\max} - z(\sigma_l) + z_{\min}}{z_{avg}}, \quad \text{for } l = 1, \cdots, w,$$

where z_{max}, z_{min} and z_{avg} are the maximum, minimum and average objective values in the current population, respectively, and $z(\sigma_l)$ is the objective value of individual l. When the difference between normalized objective values of individuals in a population is relatively small, the selection process may become a random walk. In this case, the fitness value of individual l (f_{rank}) by rank can be calculated below:

$$f_{rank}([\sigma_l]) = \frac{2l}{w(w+1)}, \quad \text{for } l = 1, \cdots, w,$$

where $[\sigma_l]$ is the l^{th} individual in a descending order of objective function values.

Selection

HGA adopts the stochastic remainder selection procedure without replacement to determine the expected number of copies of individual l, $E(l)$, l = 1, ⋯, w, in the mating pool. $E(l)$ can be calculated as follows:

$$E(l) = wf(\sigma_l) / \sum_{j=1}^{w} f(\sigma_j), \quad \text{for } l = 1, \cdots, w,$$

where $f(\sigma_l)$ is the fitness value of individual l. If $E(l)$ is non-integer, for some l, then only $\lfloor E(l) \rfloor$ copies of individual l are assigned to the mating pool, and additional individuals need to be selected from the current population. Bernoulli trials (weighted coin tosses) with success probabilities $P_s(l) = E(l) - \lfloor E(l) \rfloor$ are performed to individual l one by one until the mating pool is full. When individual l is selected, $P_s(l)$ is reduced to 0.

Recombination Operators (Crossover and Mutation)

HGA adopts PMX with a constant crossover probability (rate). Under PMX, two crossover points are picked at random. The genes in an individual that are positioned in the section bounded by the two crossover points are matched (connected) to genes in the corresponding positions of the other individual. If two pairs of connected genes share the same value (or allele), the common gene is eliminated and the two remaining genes are combined into a single connection. After all the reductions of this type are completed, the remaining pairs of connected genes form the match table. Finally, 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 individuals chosen for crossover such that A = (7 3 **2 6 4** 5 1) and B = (4 3 **7 2 5** 1 6), and the two crossover points are 2 and 5. First, the genes between two crossover points are swapped (2, 6, 4 of A and 7, 2, 5 of B). Second, the genes before the first crossover point and after the second crossover points are exchanged according to the match table ($7 \leftrightarrow 6$, $5 \leftrightarrow 4$) Then, the resulting individuals by PMX are A' = (6 3 7 2 5 4 1) and B' = (5 3 2 6 4 1 7).

HGA adopts the adjacent swap method with a constant mutation rate in which a job is exchanged with the next job in the job sequence. If the last job is to be mutated, it is exchanged with the first job in the job sequence.

Development

The premature convergence of many GAs can be overcome by using the stochastic remainder selection procedure without replacement (Goldberg, 1989), since the maximum number of copies of individual l, l = 1, ⋯, w, in the mating pool is bounded and thus, high fit individuals cannot prevail in the early generations. However, using this selection scheme may increase the probability of selecting the least fit individuals in comparison with other selection schemes such as the roulette wheel selection method. The selection of the least fit individuals may provide low fit offsprings and, in turn, severely decrease the search power for the best solutions. HGA applies a non-adjacent pairwise interchange (NAPI) method to the least fit individual in the mating pool, and replaces the least fit individual by the best individual in its neighborhood. The combination of the stochastic remainder selection procedure without replacement and NAPI prevents high fit individuals from dominating the population in the early generations and also maintains the GA search power to reach for the best solutions.

Hybrid Genetic Algorithm (HGA)

Step 0 (Initialization)

In a preliminary test, the best set of following parameters is determined before the main test: Method of fitness value calculation, population size, number of generation, crossover probability, mutation probability, number of individuals generated by seed selection

Step 1 (Construction of an initial population)
(a) Generate predetermined number of individuals using a random number generator.
(b) Generate the rest of individuals by seed selection

Step 2 (Evaluation and selection)
(a) Obtain objective values of individuals in the population
(b) Compute the fitness values of individuals in the population.
(c) Use the stochastic remainder sampling without replacement to select individuals from the population to form a mating pool.

Step 3 (Development)
Apply the NAPI method to find the best individual in the neighborhood of the least fit individual selected in Step 2(c). Replace the least fit individual in the mating pool by the best individual in the neighborhood.

Step 4 (Reproduction)
(a) Mate individuals in the mating pool randomly.
(b) Apply PMX with a constant crossover rate to the couples.
(c) Apply the adjacent swap method with a constant mutation rate to the offsprings produced by PMX.

Step 5 (Termination test)
If HGA reaches the maximum number of generations, stop. Otherwise, go to Step 2.

4. COMPUTATIONAL STUDY

The HGA and GA were coded in Visual FORTRAN and ran on an Intel Core i7 CPU@3.4 GHz PC. Since no sample problems were found in the literature that could be used as a benchmark for testing the proposed HGA, the test problems were generated randomly. Processing times and release times of jobs were generated according to the integer uniform distributions provided in Table 1.

Table 1. Data Used to Generate Test Problems (All Data are Integers)

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

The experiments were divided into two parts: a preliminary test and a main test. Since the performances of GA and HGA are influenced by several control parameters, a preliminary test is necessary to achieve the best parameter set for GA and HGA. In the preliminary test, 5 test problems of different sizes generated according to the data in Table 1 were solved. The best average objective function value was obtained by using the fitness function by rank, a total of 10 seed individuals, a population size of 100, a total of 100 generations, a crossover rate of 1.0, and a mutation rate of 0.01.

The test problems for the main test were generated in a similar way. Nine different test problems were generated for each problem size. These 216 problems were solved by HGA. For small size flow shop problems (5 and 7 jobs and 2-5 machines), the results of HGA were compared with the optimal solutions obtained by ex-

Table 2. Results for Medium and Large Size Problems

No. of Jobs	No. of Machines	HGA		GA		% Dev (z_g-z_h/z_g)x100
		Avg. obj. value (z_h)	Max obj. value	Avg. obj. value (z_g)	Max obj. value	
10	2	26.45	30.34	27.04	31.00	2.19
	3	22.15	24.49	23.49	25.61	5.71
	4	19.93	21.87	20.59	22.79	3.19
	5	18.49	19.78	19.10	20.47	3.20
15	2	51.87	58.59	56.52	61.88	8.22
	3	42.75	46.30	45.34	48.08	5.71
	4	37.17	39.63	40.03	43.23	7.15
	5	34.25	36.01	36.37	38.09	5.83
20	2	85.26	89.98	100.92	110.82	15.52
	3	72.48	77.14	80.59	88.00	10.06
	4	61.49	65.32	68.79	73.19	10.60
	5	56.04	60.06	61.45	64.32	8.80
30	2	182.20	194.87	226.21	252.73	19.46
	3	146.94	163.85	170.22	178.63	13.68
	4	125.82	137.30	146.67	154.85	14.22
	5	112.20	120.81	123.68	133.13	9.29
Average		68.52	74.47	77.94	84.18	8.93

haustive search. HGA achieved optimal solutions for all small size problems. HGA was applied to medium size (10 and 15 jobs and 2-5 machines) and large size (20 and 30 jobs and 2-5 machines) problems. To evaluate the performance of HGA, the solutions obtained by HGA were compared with the solutions provided by GA. The results of HGA and GA for medium and large size problems are shown in Table 2. The average objective function values reported in Table 2 are the average values of nine instances for each problem size. Based on these results, HGA provides an 8.93% improvement with respect to GA on the average.

5. CONCLUSIONS

This paper addresses the problem of minimizing the total stretch in the *n*-job, *m*-machine flow shop. Since this problem is NP-hard even for a single machine, it requires significant computational effort to solve the problems with large *n*. Meta-heuristic algorithms such as GAs have been used for many scheduling problems. HGA is proposed to reduce the premature convergence of GAs and maintain the search power by adopting the new idea of seed selection and development. The performance of the HGA was compared with that of GA and the results of the computational experiments show that the HGA works well for this type of problem.

REFERENCES

Baker, K. R. and D. Trietsch, *Principle of Sequencing and Scheduling*, Wiley, New Jersey, 2009.

Bender, M. A., S. Muthukrishnan, and R. Rajaraman, "Approximation algorithms for average stretch scheduling," *Journal of Scheduling* 7 (2004), 195-222.

Bhattacharyya, S., "Direct marketing performance modeling using genetic algorithms." *INFORMS Journal on Computing* 11 (1999), 248-257.

Chan, W.-T., T.-W. Lan, K.-S. Liu, and P. W. H. Wong, "New resource augmentation analysis of the total stretch of SRPT and SJF in multiprocessor scheduling," *Theoretical Computer Science* 359 (2006), 430-439.

Dreo, J., A. Petrowski, P. Siarry, and E. Taillard, Metaheuristics for Hard Optimization: Methods and Case Studies, Springer, New York, 2005.

Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

Lee, C.-Y., S. Piramuthu, and Y.-K. Tsai, "Job shop scheduling with a genetic algorithm and machine learning," *International Journal of Production Research* 35 (1997), 1171-1191.

Liepins, G. E. and M. R. Hilliard, "Genetic algorithms: Foundation and applications," *Annals of Operations Research* 21, 1~4 (1989), pp. 31-58.

Muthukrishnan, S., R. Rajaraman, A. Shaheen, and J. F. Gehrke, "Online scheduling to minimize average stretch," *Siam Journal on Computing* 34, 2 (2005), 433-452.

Pinedo, M. L,, *Scheduling: Theory, Algorithms, and Systems (4th Ed.)*, Springer, New York, 2012.