http://www.calsec.or.kr/jsebs
ISSN: 2288-3908

The Journal of Society for e-Business Studies
Vol.20, No.1, Feb. 2015, pp.79-88
http://dx.doi.org/10.7838/jsebs.2015.20.1.079

총 스트레치 최소화를 위한 분할 가능 리퀘스트 흐름 스케줄링

Minimizing the Total Stretch when Scheduling Flows of Divisible Requests without Interruption

윤석훈(Suk-Hun Yoon)*

초 록

웹 서버나 데이터베이스 서버와 같은 컴퓨터 서버들은 연속적으로 리퀘스트 스트림을 받는다. 이런 서버들은 유저들에게 최선의 서비스를 제공하기 위해 리퀘스트들을 스케줄링하여야 한다. 이 논문은 분할 가능 리퀘스트들을 스케줄링할 때 총 스트레치를 최소화하기 위해 혼합 유전자 알고리즘을 제안한다. 리퀘스트의 스트레치는 리퀘스트가 시스템에 머무는 시간에 대한 반응 시간의 비율로 정의된다. 혼합 유전자 알고리즘은 유전자 알고리즘의 활용과 탐구 능력를 개선하기 위해 시드 선택과 개발의 아이디어를 도입하였다. 혼합 유전자 알고리즘과 유전자 알고리즘의 성능을 비교하기 위하여 광범한 컴퓨터 실험이 실행되었다.

ABSTRACT

Many servers, such as web and database servers, receive a continual stream of requests. The servers should schedule these requests to provide the best services to users. In this paper, a hybrid genetic algorithm is proposed for scheduling divisible requests without interruption in which the objective is to minimize the total stretch. The stretch of a request is the ratio of the amount of time the request spent in the system to its response time. The hybrid genetic algorithm adopts the idea of seed selection and development in order to improve the exploitation and exploration power of genetic algorithms. Extensive computational experiments have been conducted to compare the performance of the hybrid genetic algorithm with that of genetic algorithms.

키워드 : 스케줄링, 흐름공정, 로트-스트리밍, 스트레치, 유전자 알고리즘
Scheduling, Flow Shop, Lot-Streaming, Total Stretch, Genetic Algorithms

* Department of Industrial and Information Systems Engineering, Soongsil University(yoon@ssu.ac.kr)
Received: 2014-11-28, Review completed: 2014-12-09, Accepted: 2014-12-30

1. Introduction

In the last decade, the problem of scheduling flows of divisible requests (jobs) without interruption in a flow manufacturing line has received an increasing amount of attention by several researchers. It can be viewed as the no-wait lot-streaming flow shop scheduling. Traditionally, the focus of performance measure has been on the maximal completion time and the flow time. Alternatively, practitioners have used stretch to measure the effect of scheduling on a job [3]. The stretch of a job formally is defined as the ratio of its flow time to its request (processing) time [1]. 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 [12]. Since the problem of minimizing the total stretch with different job release times is NP-hard even for a single machine, the computational effort to solve an instance of a problem grows remarkably quickly as the number of jobs increases [14]. Thus, for large size scheduling problems, it is necessary to consider meta-heuristic methods such as genetic algorithms (GAs), simulated annealing, and tabu search [7, 15].

This paper presents a solution methodology for an n-job, m-machine no-wait lot-streaming flow shop scheduling problem in which the objective is to minimize the total stretch. Following the literature review in the next section, the problem and notations are defined in section 3. By adopting new genetic operators (seed selection and development), a hybrid genetic algorithm (HGA) is proposed to solve the problem in section 4. In section 5, the results of extensive computational experiments comparing HGA and GA are provided. Finally, a summary of main results and conclusions are provided in Section 6.

2. Literature Review

Lot-streaming is the process of splitting a job (lot) into sublots to allow the overlapping of operations between successive machines in a multi-stage production system. These benefits include reduction of production lead times, reduction of work-in-process inventory and costs, reduction of interim storage and space requirements, reduction of material handling system capacity requirements, and improvement of product delivery. An extensive review of research in this area is presented in [4, 5].

Pan et al. [13] propose a local-best harmony search algorithm with dynamic sub-harmony memories for a lot-streaming flow shop scheduling problem with equal-size sub-lots to minimize the total weighted earliness and tardiness penalties. Edis and Ornek [8] address a single-product multistage sto-

chastic flow shop problem with consistent sublot types and discrete sublot sizes to minimize the makespan. They propose a heuristic algorithm that is combination of simulation and tabu search. Feldmann and Biskup [9] address the multi-stage flow shop problem with sublots that are allowed to intermingle by standard optimization software. They present a mixed integer programming formulation to find optimal sublot sizes and the optimal sequence simultaneously. Chiu et al. [6] present a general mathematical programming model to solve a lot streaming problem in multistage batch production systems in which transportation activities are involved. They develop two heuristic procedures to solve the mathematical programming model. Wong et al. [16] address a resource-constrained assembly job shop scheduling problem with lot streaming technique to minimize total lateness. They propose a particle swarm optimization to solve the problem.

3. Minimization of the Total Stretch in the n-job, m-machine no-wait flow Shop with Equal-size Sublots

There is a set of jobs $J=\{1, 2, \cdots, n\}$ that has to be processed on m machines in series. Job j is available at the release time τ_j and requires the processing time $p_{i,j}$ on machine i. Each job j can be divided into the number of sublots s_j. Let $r_{i,j}$ $(= p_{i,j}/s_j)$ be the sublot processing time of job j on machine i, and $c_{i,\sigma(q),k}$ represent the completion time of sublot k of the qth job on machine i. Let $p_j = p_{1,j}+p_{2,j}+\cdots+p_{m,j}$. Then the total stretch is defined as $\sum_{j=1}^{n}(C_{m,j,s_j}-\tau_j)/p_j$. For a given sequence σ, the problem can be formulated as follows:

$$\text{minimize } z(\sigma)=\sum_{j=1}^{n}\frac{c_{m,j,s_j}-\tau_j}{p_j} \tag{1}$$

Subject to

$$c_{i,\sigma(j),1}-r_{i,\sigma(j)} \geq c_{i,\sigma(j-1),s_{\sigma(j-1)}}, \quad \text{for } i=1, \cdots, m,\ j=2, \cdots, n; \tag{2}$$

$$c_{i,j,k}-r_{i,j} \geq c_{i,j,k-1}, \quad \text{for } i=1, \cdots, m,\ j=1, \cdots, n,\ k=2, \cdots, s_j; \tag{3}$$

$$c_{i,j,k}-r_{i,j} = c_{i-1,j,k}, \quad \text{for } i=2, \cdots, m,\ j=1, \cdots, n,\ k=1, \cdots, s_j; \tag{4}$$

$$c_{1,\sigma(j),1} \geq r_{1,\sigma(j)}+\tau_j, \quad \text{for } j=1, \cdots, n; \tag{5}$$

Constraint set (2) states the relationships between completion times of adjacent jobs on each machine and assures that a machine can process at most one job at the same time. Constraint set (3) provides the relationships between completion times of adjacent sublots in each job. That is, each machine can process at most one sublot at the same time. Constraint set (4) insures that once a sublot is

released from a machine, its processing on the downstream machine begins immediately. Constraint set (5) states that all jobs are available at their release times.

4. Hybrid Genetic Algorithm (HGA)

GAs are search algorithms based on the mechanics of natural selection and natural genetics. Unlike most algorithms that start with a solution, GAs start with a collection (population) of solutions (individuals). With the survival of the fittest philosophy, GAs select individuals in a population to form a gene pool according to their fitness values. Individuals in the gene pool are randomly mated to become couples and each couple goes through the crossover and mutation processes to produce two solutions (offspring). These offspring becomes the population of the next iteration (generation). The population size remains fixed in all generations and the process continues until a predetermined termination criterion is satisfied [2].

HGA is a GA-based heuristic method which incorporates seed individuals and development processes. HGA uses a permutation representation for individuals (sequences), where a sequence of n jobs is defined by a permutation of integers $\{1, \cdots, n\}$. Most individuals of an initial population are generated randomly. To generate the rest of the initial population, the seed selection process is used. The seed selection process is based on rules to generate initial sequences and neighborhood search methods. The initial sequences are generated by the shortest release time (SRT), the shortest processing time on the first machine (SPTF), and the shortest total processing times (SPTT). The initial sequence $\sigma = \{\sigma(1), \sigma(2), \cdots, \sigma(n)\}$ can be obtained as follows:

For $u = 2, \cdots, n$,

SRT : $\tau_{\sigma(u-1)} \leq \tau_{\sigma(u)}$,

SPTF : $p_{1,\sigma(u-1)} \leq p_{1,\sigma(u)}$,

SPTT : $\sum_{i=1}^{m} p_{i,\sigma(u-1)} \leq \sum_{i=1}^{m} p_{i,\sigma(u)}$,

The initial sequences are modified by the neighborhood search methods such as the non-adjacent pairwise interchange (NAPI), the extraction and forward shifted reinsertion (EFSR), and the extraction and backward shifted reinsertion (EBSR). NAPI switches two jobs in a sequence. EFSR and EBSR choose two jobs and move a job in the front just after another job (EFSR) or a job in the back just before another job (EBSR) in a job sequence. Use of randomly generated individuals and seed individuals enhances search for unbiased sampling of the space, which provides robust exploitation of the current solutions, and exploitation of good quality of solutions.

Individuals in the population have been evaluated by the fitness values that can be obtained by several methods. Two methods are mostly used in the literature [11]: fitness value by normalization (f_{scale}) and fitness value by rank (f_{rank}). These can be obtained as follows:

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

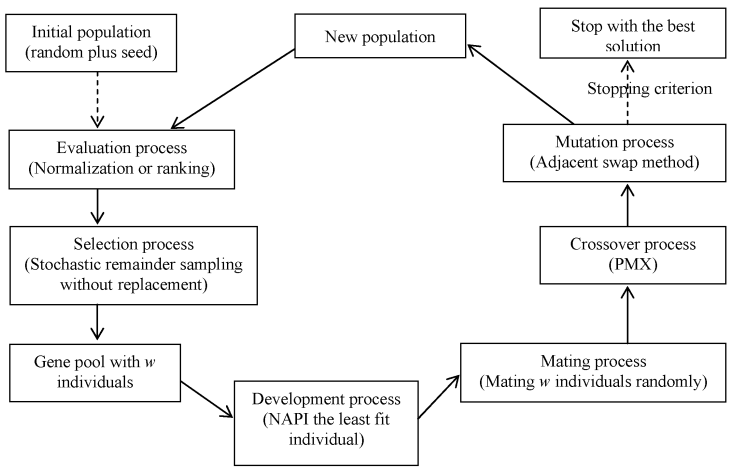
where z_{max}, z_{min} and z_{avg} are the maximum, minimum and average objective values in the current population, respectively, and $z(\sigma_l)$ is the objective value of individual l. In the ranking procedure, the population is sorted by a descending order of objective function values. The first individual in the order has fitness value of one, the second one has fitness value of two, and so on.

Individuals with their fitness values are selected to form the mating pool by using the stochastic remainder selection procedure without replacement. To determine the expected number of copies of individual l, $E(l)$, $l = 1, \cdots, w$, in the mating pool, $E(l)$ can be calculated as follows:

$$E(l) = w f(\sigma_l) / \sum_{j=1}^{w} f(\sigma_j),$$

where $f(\sigma_l)$ is the fitness value of individual l, $l = 1, \cdots, w$. 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, $l = 1, \cdots, w$, one by one until the mating pool is full. When individual l, $l = 1, \cdots, w$, is selected, $P_s(l)$ is reduced to 0.

Individuals in a mating pool are mated randomly go through crossover and mutation process. HGA uses the partially matched crossover (PMX) for crossover [10]. Under PMX, two crossover points are picked at random. The genes between the points are swapped and used to construct the match table. According to the match table, the genes before the first and after the second crossover points are exchanged if the gene has the same value (allele) in the table. The process to construct the table is explained below. For example, suppose that A and B are the individuals chosen for crossover such that $A = (6\ 5\ \mathbf{2}\ 7\ 4\ 3\ 1)$ and $B = (5\ 3\ \mathbf{6}\ \mathbf{2}\ \mathbf{4}\ \mathbf{1}\ 7)$, and the two crossover points are 2 and 6. First, the genes between two crossover points are swapped (2, 7, 4, 3 of A and 6, 2, 4, 1 of B). Second, the genes in the same positions (loci) between two crossover points are compared to construct the match table. The comparison ($2 \leftrightarrow 6$, $7 \leftrightarrow 2$, $3 \leftrightarrow 1$) results in the match table ($7 \leftrightarrow 6$, $3 \leftrightarrow 1$). According to the table, if genes have the value 7, 6, 3,



〈Figure 1〉 HGA cycle

and 1, they are exchanged into 6, 7, 1, and 3, respectively. Thus, the resulting individuals by PMX are A' = (7 5 6 2 4 1 3) and B' = (5 1 2 7 4 3 6). HGA adopts the adjacent swap method for the mutation process, 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.

GAs propagate the characteristics of high fit individuals to the next generation by giving high probability of selection to these individuals, but sometimes it may produce dominance of high fit individuals in the early generations (premature convergence). On the contrary, restraining the maximum number of copies of high fit individuals in the selection process in order to reduce the chance of premature convergence, may cause the loss of GA search power. This contradiction could not be solved by traditional GAs. HGA is proposed to overcome this problem by applying NAPI to the least fit individual after the stochastic remainder sampling without replacement.

<Figure 1> shows the cycle of HGA. The development process and adoption of seed individual are new in order to reduce the premature convergence of GAs.

5. Computational Study

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. Release times, processing times of jobs, and the number of sublots per each job, 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
Number of sublots (NT)	Uniform(1, 6)
sublot processing times (SP)	Uniform(1, 31)
Job processing times	NT×SP
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, the preliminary test is necessary to determine the best set of parameters for each algorithm. In the preliminary test, five test problems of different sizes were generated according to all combinations of number of jobs and number of machines provided in <Table 1>. Each problem was solved by both algorithms with different combinations of control parameters. The control parameters of GA such as he fitness function, the population size (PPSZ), the number of generations (XGEN), the crossover rate (Pc), and the mutation rate (Pm) are investigated in the preliminary test. The control parameters of HGA are the fitness function, the number of seed individuals generated for an initial population (Ns), PPSZ, XGEN, Pc, and Pm that provide the best performance. The best choices of the control parameters for GA are the stochastic remainder selection procedure without replacement, the fitness function by rank, a population size of 100, a total of 100 generations, PMX with a crossover rate of 1.0, and the adjacent swap method with a mutation rate of 0.01. The best choices of the control parameters for HGA are he stochastic remainder selection procedure without replacement, the fitness function by rank, a population size of 100, a total of 100 generations, a total of 10 seed individuals, a crossover rate of 1.0, and a mutation rate of 0.01. These parameters were used in the main test.

The test problems for the main test were generated in a similar way. Ten different test problems were generated for each problem size. For small size lot-streaming flow shop problems (5 and 7 jobs and 2~5 machines), the results of the HGA were compared with the optimal solutions obtained by exhaustive search. HGA achieved optimal solutions for all eighty small size problems. HGA was also 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 those provided by GA. The results of HGA and GA for medium and large size problems are shown in <Table 2>. Based on these results, HGA provides 3.94, 7.54, 10.46 and 12.30% improvements, on the average, for 10, 15, 20, and 30 jobs, respectively, in regard to GA.

〈Table 2〉 Results for Medium and Large Size Problems

No. of Jobs	No. of machines	GA method	HGA method	%Dev
		Avg. obj. value (z_g)	Avg. obj. value (z_h)	(z_g-z_h/z_g)×100
10	2	4.67	4.34	7.15
	3	4.58	4.38	4.36
	4	4.65	4.55	2.15
	5	4.86	4.76	2.07
15	2	7.13	5.90	17.30
	3	6.60	6.28	4.80
	4	6.87	6.45	6.12
	5	6.28	6.15	1.95
20	2	8.65	6.94	19.81
	3	8.04	7.63	5.09
	4	8.44	7.65	9.36
	5	8.27	7.65	7.56
30	2	11.68	10.11	13.43
	3	12.54	10.27	18.11
	4	12.04	11.17	7.26
	5	12.22	10.95	10.41
Average		7.97	7.20	8.56

6. Conclusions

In this paper, HGA has been proposed to lessen the premature convergence of GAs and maintain the search power by adopting the seed selection and the development process. By these new processes, HGA restrains the high fitness individuals from dominating populations in early generations. Extensive computational experiments have been conducted to compare the performance of HGA and that of GA. These results show that the average improvement of HGA over GA is 8.56% for medium to large size problems.

References

[1] Bender, M. A., Muthukrishnan, S., and Rajaraman, R., “Approximation algorithms for average stretch scheduling,” Journal of Scheduling, Vol. 7, pp. 195-222, 2004.

[2] Bhattacharyya, S., “Direct marketing performance modeling using genetic algorithms,” INFORMS Journal on Computing, Vol. 11, pp. 248-257, 1999.

[3] Chan, W.-T., Lan, T.-W., Liu, K.-S., and Wong, P. W. H., “New resource augmen-

tation analysis of the total stretch of SRPT and SJF in multiprocessor scheduling," Theoretical Computer Science, Vol. 359, pp. 430-439, 2006.

[4] Chang, J. H. and Chiu, H. N., "A comprehensive review of lot streaming," International Journal of Production Research, Vol. 43, No. 8, pp. 1515-1536, 2005.

[5] Cheng, M., Mukherjee, H. J., and Sarin, S. C., "A review of lot streaming," International Journal of Production Research, Vol. 51, No. 23-24, pp. 7023-7046, 2013.

[6] Chiu, H.-N., Chang, J.-H., and Lee, C.-H., "Lot streaming models with a limited number of capacitated transporters in multistage batch production system," Computers and Operations Research, Vol. 31, pp. 2003-2020, 2004.

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

[8] Edis, R. S. and Ornek, M. A., "A tabu search-based heuristic for single-product lot streaming problems in flow shops," International Journal of Advanced Manufacturing Technology, Vol. 43, pp. 1202-1213, 2009.

[9] Feldmannn, M. and Biskup, D., "Lot streaming in a multiple product permutation flow shop with intermingling," International Journal of Production Research, Vol. 46, No. 1, pp. 197-216, 2008.

[10] Goldberg, D. E., Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, New York, 1989.

[11] Liepins, G. E. and Hilliard, M. R., "Genetic algorithms : Foundation and applications," Annals of Operations Research, Vol. 21, No. 1-4, pp. 31-58, 1989.

[12] Muthukrishnan, S., Rajaraman, R., Shaheen, A., and Gehrke, J. F., "Online scheduling to minimize average stretch," Siam Journal on Computing, Vol. 34, No. 2, pp. 433-452, 2005.

[13] Pan, Q.-K., Suganthan, P. N., Liang, J. J., and Tasgetiren, M. F., "A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming lfow shop scheduling problem," Expert Systems with Applications, Vol. 38, pp. 3252-3259, 2011.

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

[15] Song, B. D. and Oh, Y., "A study on network redesign for supply chain expansion," The Journal of Society for e-Business Studies, Vol. 17, No. 4, pp. 141-153, 2012.

[16] Wong, T. C., Chan, F. T. S., and Chan, L. Y., "A resource-constrained assembly job shop scheduling problem with lot streaming technique," Computers and Industrial Engineering, Vol. 57, 983-995, 2009.

저 자 소 개

윤석훈	(E-mail : yoon@ssu.ac.kr)
1988년	서울대학교 산업공학과 (학사)
1995년	NC A&T State University (석사)
2002년	Pennsylvania State University (박사)
2002년~현재	숭실대학교 산업정보시스템공학과 부교수
관심분야	빅데이터, Scheduling