Pavol SEMANČO[*]

MINIMIZING MAKESPAN IN GENERAL FLOW-SHOP SCHEDULING PROBLEM USING A GA-BASED IMPROVEMENT HEURISTIC

Abstract

In the paper an improvement heuristic is proposed for permutation flow-shop problem based on the idea of evolutionary algorithm. The approach employs constructive heuristic that gives a good initial solution. GA-based improvement heuristic is applied in conjunction with three well-known constructive heuristics, namely CDS, Gupta's algorithm and Palmer's Slope Index. The approach is tested on benchmark set of 10 problems range from 4 to 25 jobs and 4 to 30 machines. The results are also compared to the best-known lower-bound solutions.

1. INTRODUCTION

A flow-shop production introduces a manufacturing system where n jobs are processed by m machines in the same order. The problem of finding an optimal schedule is referred to as flow-shop scheduling problem (FSSP). In a permutation flow-shop scheduling problem, denoted as PFSSP, the same sequence, or permutation, of jobs is maintained throughout (Pinedo, 2008). The objective of the flow-shop scheduling problem is to meet optimality criterion of minimizing the makespan, total flow time or total weighted flow time. This paper investigates an optimal job sequence for flow-shop scheduling benchmark problem with objective to minimize the makespan. The general scheduling problem for a classical flow shop gives rise to $(n!)^m$ possible schedules (Gupta 1975). For flow-shop scheduling problem Johnson (1954) proposed algorithm that optimally solves a 2-machine flow-shop problem. It was later demonstrated that m-machine flow-shop scheduling problem (FSSP) is strongly NP-hard for $m \geq 3$ (Garey et al., 1976). Permutation FSSP also has to meet standard requirements like a job cannot be processed by two or more machines at a time and a machine cannot process two or more jobs at the same time.

The optimization of FSSP employs the three major types of scheduling algorithm (exact, approximation and heuristic). However, the most common type of scheduling algorithms for NP-hard FSSP is heuristic that produces near-optimal or optimal solutions in reasonable time. The heuristics can be further classified as constructive heuristic and improvement heuristic (or meta-heuristic). The improvement heuristic in contrast to constructive heuristic starts with a initial schedule trying to find an improved schedule. In this paper, the improvement-heuristic

[*] Ing. Pavol Semančo, Technical University of Kosice, Slovakia, email: pavol.semanco@tuke.sk

approach is proposed incorporating the idea of evolution. If no improvement occurs for a certain number of iterations, the algorithm backtracks to the last best result. GA-based improvement heuristic is performed by predetermined number of iterations and report of the best result.

The rest of the paper is organized as follows. The next section reviews the relevant scheduling literature for the flow-shop scheduling heuristics algorithms. In the section, namely GA-based improvement heuristic, the formal description of GA approach is covered. The Section, "Computational Experiments," discusses results obtained from the experiment. The summary of the paper and possible future research ideas are presented in the section, namely Summary and Conclusions.

2. RESEARCH BACKGROUND

The model of flow-shop scheduling problem with makespan (C_{max}) as an objective function can be specified according to 3-filed classification $\alpha/\beta/\gamma$. The first filed, namely α, stands for machine environment. For the flow-shop scheduling the machine environment is denoted as *Fm*, where *m* is the number of the machines. The β-field specifies the job constraints like for permutation of jobs the *prmu* abbreviation is used. The last field determines the optimally criterion like makespan (C_{max}). Based on this 3-field classification the general flow-shop scheduling problem can be denoted as $Fm/prmu/C_{max}$. This notation was firstly suggested by Conway et al. (1967) and until now is handy.

Hejazi and Saghafian (2005) introduced a comprehensive review of alogorithms for flow-shop scheduling problems with makespan criterion. Approaches solving flow-shop scheduling problem range from heuristics, developed, for example, by Palmer (1965), Campbell et al. (1970), Dannenbring (1977) to more complex techniques such as Branch and Bound (Brucker, 1994), Tabu Search (Gendreau, 1998), Genetic Algorithm (Murata et al., 1996), Shifting Bottleneck procedure (Balas and Vazacopoulos, 1998), Ant Colony Algorithm (Blum and Sampels, 2004) and others.

The flow-shop sequencing problem is one of the most well-known classic production scheduling problems. Focusing on the PFSSP with C_{max} objective function, first classical heuristics was proposed by Page (1961). Palmer (1965) adopted his idea and proposed the slope index to be utilized for the m-machine *n*-job permutation flow shop sequencing problem. A simple heuristic extension of Johnson's rule to m-machine flow shop problem has been proposed by Campbell et al. (1970). This extension is known in the literature as the CDS (Campbell, Dudek, and Smith) heuristic. Another method to obtain a minimum makespan is presented Gupta (1972). A significant approach to solving the FSSP proposed Nawaz et al. (1983), in which they point out that a job with larger total processing time should have higher priority in the sequence.

One of the important factors that are quite frequently discussed in FSSP is the setup time (see, for instance, Allahverdi et al., 2008). The setup time represents the time required to shift from one job to another on the given machine. In the flow-shop environment, the setup time is included in the processing times of each job (Hendizadeh et al., 2007).

Modern approaches designated for larger instances are known as meta-heuristics. Approaches that combine different concepts or components of more than one meta-heuristic are named as hybrid meta-heuristic algorithms (Zobolas et al., 2009). Heuristic methods for make-span minimization have been applied, for example, by Ogbu et al. (1990) using Simulated Annealing (SA) and by Taillard (1990) applying Tabu Search (TS) algorithm. Nagar

et al. (1996) proposed a combined Branch-and-Bound (BB) and Genetic Algorithm (GA) based procedure for a flow shop scheduling problem with objectives of mean flow time and make-span minimization. Similarly, Neppalli et al. (1996) were used genetic algorithms in their approach to solve the 2-machine flow shop problem with objectives of minimizing make-span and total flow time. An atypical method based on an Artificial Immune System (AIS) approach, which was inspired from vertebrate immune system, has been presented by Engin and Doyen (2004). They used the proposed method for solving the hybrid flow shop scheduling problem with minimizing C_{max}. Obviously, there are plenty of other related approaches to this problem that are identified in survey studies, such as that of Ribas et al. (2010).

3. GA-BASED IMPROVEMENT HEURISTIC

Genetic algorithm (GA) forms one of the categories of local search method that operate with a set of solutions. GA is inspired by well-known Darvin's theory about the evolution. GA-based heuristic is started with a set of solutions, also referred to as population. Solutions (or in terms of genetic algorithm, chromosomes) from initial population are taken to form a new population with hope that the new population will be better than the old one. The selection of solutions is performed by a "survival of the fittest" principle to ensure that the overall quality of solutions increases from one generation to the next. This is repeated until some condition (for example number of generations or improvement of the best solution) is satisfied. The framework of proposed GA-based heuristic (GAH) is introduced below.

NOTATION OF GAH ALGORITHM

The following notation was used:
G number of generations
P population size
$F(s)$ fitness function
C_{max} makespan
s solution represented by a job sequence
s_i initial solution
p_c crossover probability parameter
p_m mutation probability parameter
c chromosome string
c_p parent chromosome
c_o offspring

GA OPERATORS

The most important parts of the genetic algorithm are genetic operators, referred to as encoding, selection, crossover and mutation operator that impact the whole performance. Proposed GA-based improvement heuristic employs permutation encoding of chromosomes, where each chromosome is a string of numbers (genes), which represents number in a sequence.

For the selection of best chromosomes the roulette wheel method was used. Proposed GAH employs also a method, called elitism, before roulette wheel selection to ensure that at least one best solution is copied without changes to a new population, so the best solution found can survive to end of run.

The crossover operator is carried out with a crossover probability. Crossover selects genes from parent chromosomes and creates a new offspring. It randomly selects a crossover point and everything before this point is copied from the first parent. Then the second parent is scanned and if the scanned gene is not yet in the offspring, it is appended. This method is also called as Single point crossover.

Mutation is also done randomly for each gene and it depends upon another parameter called mutation probability. In this method inversion mutation is adopted where one gene is selected at random and exchanged with another gene mutually. Basically it is an order changing where two numbers are exchanged.

PSEUDO CODE OF GA FOR MINIMIZING THE MAKESPAN

In the paper GAH is used to search for solution of minimal make-span. Figure 1 introduces the pseudo code of proposed GA-based improvement heuristic in conjunction with constructive heuristic. The constructive heuristic gives a good initial solution to be improved by GA-based heuristic. The objective of the fitness function is to minimize a makespan. The best solution is represented by minimal makespan.

Step 1 Find initial solution (s_i) by selected constructive heuristic
Step 2 Generate initial population (P) based on initial solution and randomness
Step 3 Apply selection with elitism
Step 4 Apply crossover with crossover probability (c_p)
Step 5 Apply mutation with mutation probability (m_p)
Step 6 Compute the fitness value for new offspring
Step 7 Evaluate and save the best chromosome
Step 8 Go to *Step 2* until the generation value reaches G

Fig. 1. Pseudo code of proposed algorithm

4. COMPUTATIONAL EXPERIMENTS

The experiment was run with objective of minimizing makespan on benchmark dataset that has 10 instances. The dataset ranges from 20 to 500 jobs and 5 to 20 machines.

The CDS, Palmer's Slope Index, Gupta's algorithms and GAH were coded in PHP script, running on a PC with 1.6 GHz Intel Atom and 1GB of RAM. All PHP-coded algorithms has user-friendly interface with eventuality to select whether to run each heuristic itself or all together. It has also an option to draw a Gantt chart. Table 1 contains the input parameters of GAH approach for the experiment purposes.

Table 1. GA constraints

Parameter	Value
P	20
G	500
p_c	0.6
p_m	0.05
F(s)	*makespan*

RESULTS

Results of GA-based heuristic are represented by use of percentage improvement from solution of constructive heuristic and gap from lower-bound solution (LB).

The paper will refer to the 3-heuristic GAH versions, namely P-GAH (Palmer-GAH), CDS-GAH and G-GAH (Gupta-GAH). Table 2 summarizes the results for all 10 instances and also shows percentage improvement of GAH over constructive heuristic. Table 1 also introduces the best-known lower bounds and percentage gap from the best-known bound for the best GAH result. In the table the results are displayed for Palmer alone, CDS alone, NEH alone, P-GAH, CDS-GAH and G-GAH.

Table 2. Makespans and improvements for 10 benchmark problems

No.	Problem Size	LB	Gupta			CDS			Palmer			Best GAH	% gap from LB
			Single pass	G-GAH	% Imprv GAH	Single pass	CDS-GAH	% Imprv GAH	Single pass	P-GAH	% Imprv GAH		
1.	4x4	156	157	156	0.64	156	156	0.00	157	156	0.64	156	**0.00**
2.	5x4	51	51	51	0.00	51	51	0.00	53	51	3.77	51	**0.00**
3.	6x5	7.7	7.7	7.7	0.00	7.7	7.7	0.00	8.35	7.7	7.78	7.7	**0.00**
4.	7x7	65	65	65	0.00	67	65	2.99	75	65	13.33	65	**0.00**
5.	8x7	69	69	66	4.35	66	66	0.00	70	69	1.43	66	**-4.55***
6.	10x12	93	106	97	8.49	104	100	3.85	104	96	7.69	96	3.13
7.	12x12	104	111	110	0.90	114	107	6.14	115	108	6.09	107	2.80
8.	15x18	141	163	150	7.98	153	149	2.61	146	142	2.74	142	0.70
9.	23x25	219	264	233	11.74	259	232	10.42	241	225	6.64	225	2.67
10.	30x25	249	285	260	8.77	271	258	4.80	274	261	4.74	258	3.49

LB – Best-known lower bound solution
Single pass – makespan of constructive heuristic
* new lower-bound solution

Overall neither of 3-heuristic GAH versions performed significantly better, although all of them gave feasible improved solutions. For flow-shop scheduling problem sizes range from 4 to 7 machines and jobs, GAH matched the best-known lower bound solutions. for 24 of the 30 problems and found a new upper bound for one problem. For the fifth problem the new lower bound was found by the GA-based improvement heuristic.

Average computational times (CPU) for each size of the problem are summarized and depicted in Figure 2. The computation times of course vary by the size of the problem. The variance, within three versions of GAH was not significant.

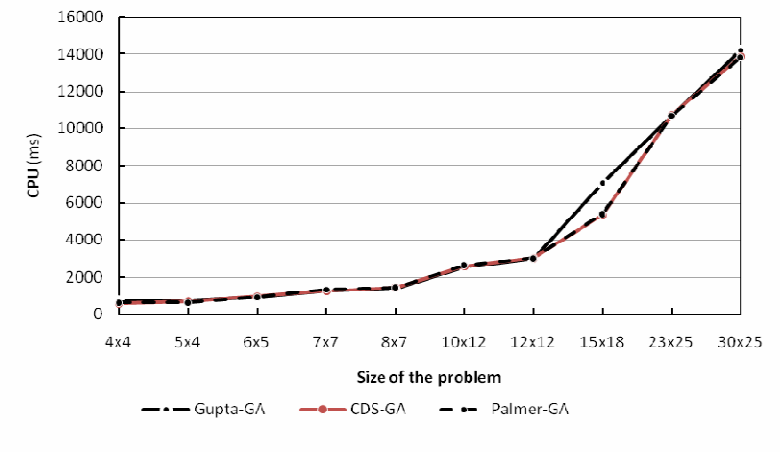

Figure 2. Computational times of GAH algorithm for each size of the problem.

5. SUMMARY AND CONCLUSIONS

In presented study, the scheduling problem with sequence-dependent operations was dealt. The main idea is to minimize the make-span time and thereby reducing the idle time of both jobs and machines since these criteria are often applied for operational decision-making in scheduling. Under above mentioned consideration an improvement heuristic based on evolutionary algorithm (GAH) is proposed and applied to the permutation flow-shop scheduling problem. The GA-based heuristic approach uses a constructive heuristic to get an initial solution that tries to find improvements iteratively.

The GAH algorithm was used to improve upon heuristics, namely, Palmer, CDS and Gupta. For all three heuristics, GAH showed significant improvements. The best improvements were compared well with the best-known lower bounds. The average gap from the best-known lower bound was 0.82% for all ten problems.

Future research should look at this heuristic for the more difficult flow-shop scheduling problems involving sequence-dependent setup times. Different objective functions can also be tested.

REFERENCES

1. ALLAHVERDI. A.. NG. C.T.. CHENG. T.C.E.. & KOVALYOV. M.Y. (2008). *A survey of scheduling problems with setup times or costs*. European Journal of Operational Research. 187. 985-1032.
2. BALAS. E.. & VAZACOPOULOS. A. (1998). *Guided local search with shifting bottleneck for job shop scheduling*. Management Science. 44 (2). 262-275.
3. BLUM. C.. & M. SAMPELS. (2004). *An ant colony optimization algorithm for shop scheduling problems*. Journal of Mathematical Modelling and Algorithms. 3(3). 285-308.
4. BRUCKER. P.. JURISCH. B.. & SIEVERS. B. (1994). *A branch and bound algorithm for the job shop scheduling problem*. Discrete Applied Mathematics. 49(1). 109-127.

5. CAMPBELL. H.G.. DUDEK. R.A.. & SMITH. M.L. (1970). *A heuristic algorithm for the n job. m machine sequencing problem.* Management Science. 16(10). 630-637.
6. DANNENBRING. D.G. (1977). *An evaluation of flow shop sequencing heuristics.* Management Science. 23(11). 1174-1182.
7. ENGIN. O.. & DOYEN. A. (2004). *A new approach to solve hybrid flow shop scheduling problems by artificial immune system*. Future Generation Computer Systems. 20. 1083-1095.
8. GAREY. M.R.D.. JOHNSON. D.S.. & SETHI. R. (1976). *The complexity of flowshop and jobshop scheduling*. Mathematics of Operations Research. 1. 117-129.
9. GENDREAU. M.. LAPORTE. G.. & SEMET. F. (1998). *A tabu search heuristic for the undirected selective travelling salesman problem*. European Journal of Operational Research. 106(2-3). 539-545. Elsevier.
10. GUPTA. J.N.D. (1972). *Heuristic algorithms for multistage flowshop scheduling problem.* AIIE Transactions. 4 (1). 11-18.
11. GUPTA. J.N.D. (1975). *Analysis of combinatorial approach to flowshop scheduling problems*.
12. HEJAZI. S.R.. & SAGHAFIAN. S. (2005). *Flowshop scheduling problems with makespan criterion: a review*. International Journal of Production Research. 43(14). 2895-2929.
13. HENDIZADEH. S.H.. ELMEKKAWY. T.Y.. & WANG. G.G. (2007). *Bi-criteria scheduling of a flowshop manufacturing cell with sequence dependent setup time*.. European Journal of Industrial Engineering. 1. 391-413.
14. JOHNSON. S. M. (1954). *Optimal two and three stage production schedules with set-up times*. Naval Research Logistics Quarterly. 1. 61-68.
15. NAGAR. A.. HERAGU. S.S.. & HADDOCK. J. (1996). *A combined branch-and-bound and genetic algorithm based approach for a flowshop-scheduling problem.* Annal. Oper. Res.. 63. 397-414.
16. NAWAZ. M.E.. ENSCORE. I.. & HAM. I. (1983*). A heuristic algorithm for the m machine. n job flow shop sequence problem*. OMEGA. 11 (1). 91-95.
17. NEPPALLI. V.R.. CHEN. C.L. & GUPTA. J.N.D. (1996). *Genetic algorithms for the two-stage bicriteria flowshop problem*. Eur. J. Oper. Res.. 95. 356–373.
18. OGBU. F.A.. & SMITH. D.K. (1990). *The application of the simulated annealing algorithm to the solution of the n/m/Cmax owshop problem*. Computers & Operations Research. 17. 3243-253.
19. PALMER. D. S. (1965). *Sequencing jobs through a multi-stage process in the minimum total time - a quick method of obtaining a near optimum.* Opers Res. Q.. 16. 101-107.
20. PINEDO. M. (2008). *Scheduling: Theory. algorithms and Systems*. Prentice Hall. New Jersey: Springer.
21. RIBAS. R.. LEISTEN. J.M. (2010). *Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective*. Computers and Operations Research. 37(8).1439-1454.
22. ZOBOLAS. G. I.. TARANTILIS. C. D.. & IOANNOU. G. (2009) *Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm*. Computers and Operations Research. 36 (4). 1249-1267.