Lecture Notes in Management Science (2013) Vol. **5**: 18–24
5th International Conference on Applied Operational Research, Proceedings

ISSN 2008-0050 (Print), ISSN 1927-0097 (Online)

A constructive heuristic for permutation flow-shop scheduling with the makespan criterion: in a particular case where the sum of the processing times of each job is the same on each machine

Kaveh Sheibani

Tadbir Operational Research Group Ltd, Vancouver, BC Canada
k.sheibani@tadbir.ca

Abstract. This paper describes an adaption of the fuzzy greedy heuristic (FGH) for the permutation flow-shop scheduling problem with the makespan criterion. In a particular case where the sum of the processing times of each job is the same on each machine, certain priority rule-based scheduling heuristics would assign the same priority to every job, and so in this situation, all possible permutations of the jobs would be equally likely to be selected by those heuristics. An efficient ranking method is proposed to prioritize the jobs in this particular case. Computational experiments using standard benchmark problems indicate that the proposed method is very efficient.

Keywords: combinatorial optimization; flow-shop scheduling; heuristics

Introduction

The flow-shop problem (FSP) is one of the most widely studied classical scheduling problems and reflects real operation of several industries. The problem can be stated as follows: there are n jobs to be processed by m machines in an identical sequence on each machine. The usual objective is to minimize the completion time of the last job to leave the system, commonly termed the makespan ($C_{\max}$). The FSP with m >2 belongs to the class of combinatorial optimization problems known to be NP-hard in the strong sense (Garey *et al*, 1976). Hence, approximation methods are generally considered to be the only practical way to solve most real-

life flow-shop problems. A discussion of the computational complexity of the FSP can be found in Gonzalez and Sahni (1978), Garey and Johnson (1979) and Brucker (2001). There are many variations of the FSP (Baker, 1974). In this paper, the permutation flow-shop scheduling problem (PFSP) with makespan criterion is considered. In the PFSP after completing a job on one machine, the job is processed on the next machine or joins a queue if the machine is busy. All queues are assumed to operate under the first-in-first-out (FIFO) discipline. It means that a job cannot pass another job while waiting in a queue. Therefore, clearly there are $n!$ possible schedules. Many heuristics have been proposed for this problem since Johnson's (1954) pioneering work, including Page (1961), Palmer (1965), Campbell *et al* (1970) and Nawaz *et al* (1983). The collections of survey papers (Rajendran, 1995; Framinan *et al*, 2004; Ruiz and Maroto, 2005; Stafford *et al*, 2005) and books (French, 1982; Sule, 1997; Pinedo, 2002) summarize various developments in the subject.

In this paper, we develop a straightforward adaptation of the FGH heuristic (Sheibani, 2010) for the PFSP with the makespan criterion. We examined the effectiveness and the efficiency of the proposed heuristic on a wide range of benchmark problems of varying sizes. The developed FGH gives a significantly improved performance relative to the well-known NEH heuristic (Nawaz *et al*, 1983), which has dominated the field for many years. The concluding remarks contain some suggestions for further research.

The FGH heuristic

The FGH heuristic consists of two phases: arranging the jobs in priority order and then constructing a sequence by a simple job insertion principle. The priority of the jobs is determined according to Equation (1).

$$\mu(x) = \frac{1}{1 + \lambda^2 \left(\left(\frac{1-\lambda}{\lambda} \right) x - \theta \right)^2} \tag{1}$$

In this equation, x is a generic variable associated with the data defining a particular instance of the PFSP. The parameter θ is a basic measure for evaluating the priority to be assigned to x. The parameter λ is a tuning parameter that is chosen by experimentation such that $0 \le \lambda < 1$ to adjust θ. We represent x with x_j as the coefficient of variation of the processing times (CVPT) of job j on the m machines, through Equation (2). Here, the parameter θ is the average of the coefficient of variation of processing times of jobs on the machines, through Equation (3).

$$x_j = \left(\frac{m\sum_{i=1}^{m} P_{ij}^2 - \left(\sum_{i=1}^{m} P_{ij}\right)^2}{m(m-1)} \right)^{1/2} \times \left(\frac{m}{\sum_{i=1}^{m} P_{ij}} \right), \quad j = 1, \ldots, n \tag{2}$$

$$\theta = \frac{1}{n}\sum_{j=1}^{n} x_j \tag{3}$$

The evaluation function in Equation (1) has the following properties: $\mu(\lambda\theta / (1 - \lambda)) = 1$ and $0 < \mu(x_j) < 1$ for all $x_j \neq \lambda\theta / (1 - \lambda)$. This implies that scheduling the job with x_j closest numerically to $\lambda\theta / (1 - \lambda)$ should be given higher priority. Let $x_{\min}$ and $x_{\max}$ be the smallest and the largest values of x_j, respectively. We define a small enough value $\lambda_{\min} = x_{\min} / (x_{\min} + \theta)$ for $\theta > 0$ and any $\lambda \leq \lambda_{\min}$, for which inequality $x_j \geq \lambda_{\min}\theta / (1 - \lambda_{\min})$ holds. We also define a big enough value $\lambda_{\max} = x_{\max} / (x_{\max} + \theta)$ for $\theta > 0$ and any $\lambda \geq \lambda_{\max}$, for which inequality $x_j \leq \lambda_{\max}\theta / (1 - \lambda_{\max})$ holds. For more details on the fuzzy greedy evaluation methodology, we refer the interested reader to Sheibani (2008). The steps of the FGH heuristic are as follows:

The FGH heuristic

(1) Calculate $\mu(x_j)$ for each job j.
(2) Arrange the jobs by descending order of $\mu(x_j)$.
(3) Select the next job and insert it in all possible positions in the partial sequence and keep the best one (i.e. minimum makespan) as the current partial sequence.
(4) Repeat step 3 until all jobs are scheduled.

The heuristic has the computational complexity same as the NEH heuristic and so can be implemented as $O(n^2m)$, though it has to be invoked for a fixed number of times for any given problem instance. This number, say t, depends on whether we decide to let λ take all possible values to 2, 3, 4,…decimal places over the range from $\lambda_{\min}$ to $\lambda_{\max}$. The choice of t is, therefore, independent of both m and n. Hence, our experimental results have a computation time that is t times that of the NEH heuristic. In this sense, our approach is of course more computationally expensive than that of the NEH heuristic. However, because t is a fixed number that is independent of m and n, the theoretical computational complexity of the approach remains unaffected by the value of t. We would argue that the additional computational expense incurred by this approach, compared to the NEH heuristic, is justified by the significant reduction obtained in the average error.

It might be supposed that, instead of using t different λ values for the FGH heuristic, a possible alternative approach that is equally expensive computationally would be to repeat the NEH heuristic t times, in each case using a different randomly generated initial job order. In fact this is unlikely to produce an improvement. Framinan *et al* (2003) showed that such randomization does not, in general, improve the NEH heuristic. Randomization was included in 177 different systematic approaches to modifying the NEH heuristic. None of these 177 approaches gave an improvement over the NEH heuristic. This is in direct contrast to the FGH heuristic proposed in this paper.

Computational results

The proposed heuristic was implemented in C++ code. The test problems are all from an extensive set of Taillard's (1993) standard benchmark instances (http://mistic.heig-vd.ch/taillard/problemes.dir/problemes.html, accessed 15 March 2010). The solution quality is measured by the percentage deviation of the obtained solution from the best-known solution. We introduced the tuning parameter λ to obtain a good performance of the proposed heuristic. The efficiency of the FGH heuristic depends greatly on the choice of the parameter λ in an effective range. If the range is too small, the probability that it includes the best λ value will be low. If it is too large, the algorithm may waste computational resources. In this experimentation, we evaluate the results obtained for all values of λ between λ_{min} and λ_{max} up to four decimal places (i.e. with increments of λ equal to 10^{-4}) for each instance of the benchmark problems considered. Where there are several such λ values, we select one that corresponds to a minimum makespan. Figure 1 shows the distribution of these λ values as a histogram. It was experimented that setting λ up to 4 decimal places is sufficiently extensive for the benchmark instances considered (Sheibani, 2005). Figure 2 exemplifies the effect of different values of λ on the computational performance of the proposed heuristic.

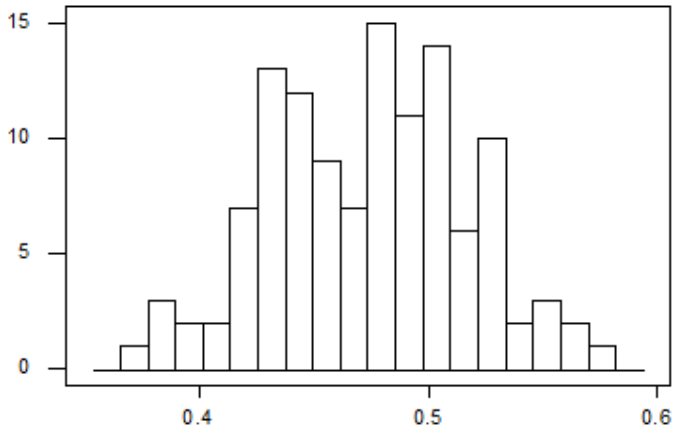

Fig. 1. The distribution of the best considered λ values as a histogram

In Table 1, we compared the performance of the FGH heuristic (Sheibani, 2010) and its new adaptation (FGH.cvpt) with Page (1961), Palmer (1965) and Campbell *et al* (1970) heuristics as well as the NEH heuristic which has been the best heuristic known in the literature. The results show that the FGH heuristic is clearly superior to all of those methods for all the problems of varying sizes in terms of the percentage error obtained. These problems are all from Taillard's standard benchmark instances for the PFSP. They were chosen in order to compare our results with those obtained by others.

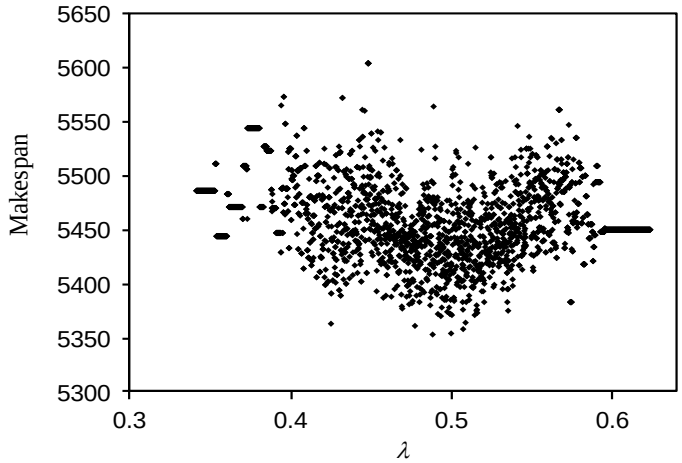

Fig. 2. The FGH heuristic on the ta076 (100×10) problem with different values of λ

Table 1. A comparison of the FGH heuristic with other heuristics

Problem (Job × Machine)	*FGH*	*FGH.cvpt*	*NEH*	*Campbell*	*Palmer*	*Page*
20 × 5	0.88	1.21	2.81	9.54	10.58	15.15
20 × 10	1.97	1.91	4.58	12.13	15.28	20.43
20 × 20	1.94	1.82	3.60	9.64	16.34	16.18
50 × 5	0.18	0.35	1.09	6.10	5.34	10.14
50 × 10	3.03	3.11	6.00	12.98	14.01	20.47
50 × 20	4.15	4.47	6.09	13.85	15.99	23.12
100 × 5	0.16	0.15	0.49	5.01	2.38	7.98
100 × 10	1.10	1.19	2.22	9.15	9.20	15.79
100 × 20	3.99	4.15	5.61	13.12	14.41	21.68
200 × 10	0.65	0.82	1.24	7.38	5.13	12.74
200 × 20	3.27	3.45	4.56	12.08	13.17	19.43
500 × 20	1.61	1.79	2.23	8.55	7.09	14.05
Average	**1.91**	**2.04**	**3.38**	**9.96**	**10.74**	**16.43**

Concluding remarks

In this paper a straightforward adaptation of the fuzzy greedy heuristic (FGH) was developed for the permutation flow-shop scheduling problem with the makespan criterion. In a particular case where the sum of the processing times of each job is

the same on each machine, certain priority rule-based scheduling heuristics would assign the same priority to every job, and so in this situation, all possible permutations of the jobs would be equally likely to be selected by those heuristics. An efficient ranking method was proposed to prioritize the jobs in this particular case.

For future research we believe that the following topics are potentially useful: (1) developing efficient adaptations of the proposed heuristic; (2) extending our method to other objectives; (3) developing efficient methods using the fuzzy greedy evaluation concept in other areas of combinatorial optimization.

References

Baker KR (1974). Introduction to Sequencing and Scheduling. John Wiley & Sons: New York.

Brucker P (2001). Scheduling Algorithms. Springer-Verlag: Heidelberg.

Campbell HG, Dudek RA and Smith ML (1970). A heuristic algorithm for the n-job m-machine sequencing problem. Mngt Sci 16: B630–B637.

Framinan JM, Leisten R and Rajendran C (2003). Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. Int J Prod Res 41: 121–148.

Framinan JM, Gupta JND and Leisten R (2004). A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. J Opl Res Soc 55: 1–13.

French S (1982). Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop. Ellis Horwood: England.

Garey MR and Johnson DS (1979). Computers and Intractability a Guide to the Theory of NP-Completeness. WH Freeman & Company: San Francisco.

Garey MR, Johnson DS and Sethi R (1976). The complexity of flowshop and jobshop scheduling. Math Opns Res 1: 117–129.

Gonzalez T and Sahni S (1978). Flowshop and jobshop schedules: Complexity and approximation. Opns Res 26: 36–52.

Johnson SM (1954). Optimal two-and-three-stage production schedules with set-up times included. Nav Res Log 1: 61–68.

Nawaz M, Enscore Jr EE and Ham I (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega—Int J Mngt Sci 11: 91–95.

Page ES (1961). An approach to the scheduling of jobs on machines. J R Stat Soc 23: 484–492.

Palmer DS (1965). Sequencing jobs through a multi-stage process in the minimum total time—A quick method of obtaining a near optimum. Opns Res Quart 16: 101–107.

Pinedo M (2002). Scheduling: Theory, Algorithms, and Systems. Prentice Hall: Upper Saddle River, NJ.

Rajendran C (1995). Heuristics for scheduling in flowshop with multiple objectives. Eur J Opl Res 82: 540–555.

Ruiz R and Maroto C (2005). A comprehensive review and evaluation of permutation flowshop heuristics. Eur J Opl Res 165: 479–494.

Stafford EF, Tseng FT and Gupta JND (2005). Comparative evaluation of MILP flowshop models. J Opl Res Soc 56: 88–101.

Sheibani K (2005). Fuzzy greedy evaluation in search, optimisation, and learning. PhD thesis, London Metropolitan University, London, UK.

Sheibani K (2008). Fuzzy Greedy Search in Combinatorial Optimisation. Tadbir Institute for Operational Research, Systems Design and Financial Services: Tehran.

Sheibani K (2010). A fuzzy greedy heuristic for permutation flow-shop scheduling. J Opl Res Soc 61: 813–818.
Sule DR (1997). Industrial Scheduling. PWS Publishing Company: Boston, MA.
Taillard E (1993). Benchmarks for basic scheduling problems. Eur J Opl Res 64: 278–285.