Journal of Industrial Engineering 3(2009)33-37

An Iterated Greedy Algorithm for Flexible Flow Lines with Sequence Dependent Setup Times to Minimize Total Weighted Completion Time

Bahman Naderi[a], Mostafa Zandieh [b,*], Seyed Mohammad Taghi Fatemi Ghomi[a]

[a] *Department of Indusrial Engineering, Amirkabir University of Technology, Tehran, Iran*

[b] *Department of Industrial Management, Shahid Beheshti University,G.C., Tehran, Iran*

Received 5 Oct., 2008; Revised 2 Dec., 2008; Accepted 9 May., 2009

Abstract

This paper explores the flexile flow lines where setup times are sequence- dependent. The optimization criterion is the minimization of total weighted completion time. We propose an iterated greedy algorithm (IGA) to tackle the problem. An experimental evaluation is conducted to evaluate the proposed algorithm and, then, the obtained results of IGA are compared against those of some other existing algorithms. The effectiveness of IGA is demonstrated through comparison.

Keywords: scheduling, flexible flow line, sequence dependent setup time, iterated greedy algorithm.

1. Introduction

Flexible flow line (FFL) is one of the well-known scheduling problems. In FFL, a set of n jobs is to be processed at a set of g production stages, each of which has several identical machines in parallel [3]. Some stages may have only one machine, but for the plant to be qualified as an FFL, at least one stage must have several machines. Each job is processed at only one machine in each stage. These machines are identical. In a FFL, all n jobs need to be processed at all g stages in the same order, starting at stage 1 and ending up at stage g. The jobs also might not undergo all stages (i.e. they can skip some stages). FFL has numerous applications in real industrial settings; including automobile manufacturing [9] and printed circuit board manufacture [16]. Each job i requires a fixed and pre-determined amount of processing time in each stage j. This amount is represented by P_{ij}. Additionally, we assume that all the tasks and jobs are independent and available for being processed at time 0. The m machines are continuously available.

Each machine j can only process a job i at a time. Each job i is processed on maximumally one machine at each stage j. The process of a job i on a machine j cannot be Interrupted. There are infinite buffers between all stages; if a job needs a machine that is occupied, it waits Indefinitely until it is available again. There is no transportation time between stages.

Many papers have considered a variety of practical and impractical assumptions. However, there always exists a feeling of a gap between theory and practice in the literature. Recently, however, sequence dependent setup times have become popular among researchers who intend to investigate the scheduling decisions in real manner. We consider that between the processing of two consecutive jobs on the same machine, some setup must be performed depending on the ordering of these two jobs. In many real-life situations such as chemical, printing, pharmaceutical, and automobile manufacturing [10], the setup operations, such as cleaning up or changing tools, are not only often required between jobs but they are also strongly dependent on the immediately preceding process on the same machine [3]. The hybrid flowshop is regarded as an NP-hard problem [6]. Due to the difficulties inherent in flexible flow line scheduling, no exact method has been introduced so far to be able to tackle these problems within reasonable amount of time. Hence, a variety of algorithms dividable into two main groups, namely heuristics and metaheuristics, have been applied to solve these problems as well as to find optimal or near optimal schedules [3, 4 and 10]. In this work, we intend to apply an iterated greedy algorithm. It is common in the scheduling literature to look for a sequence of jobs

---

[*] Corresponding author. Telfax.: +98-21-29902382; e-mail: m_zandieh@sbu.ac.ir

that minimizes the maximum completion time (or makespan) which coincides with the time at which the last job in the sequence is finished at the last machine. Another frequently considered criterion is the minimization of total weighted completion time (TWCT), denoted as $\sum w_i \times C_i$, C_i and w_i being the completion time of job *i* at the shop and the relative priority of job *i*. TWCT is regarded as a more realistic case of makespan [10].

Given the above explanation, in this paper we explore flexible flow line problems with sequence dependent setup times to minimize the total weighted completion time. An iterated greedy algorithm is presented to solve the problem.

The rest of this paper is organized as follows: Section 2 reviews the literature on the problem. Section 3 describes the iterated greedy algorithm. Section 4 evaluates the proposed algorithm. Finally, Section 5 provides some conclusions and future research.

2. Literature review

Since Johnson's pioneering work [5] on the two machine regular permutation flowshop, a lot of research has been conducted in both exact and heuristic methods for the flowshop and its other extensions. Salvador [14] first considered scheduling hybrid flowshops with no buffers between stages and no sequence dependent setup times. Branch-and-bound techniques were applied to determine the optimal permutation schedule in terms of makespan. Wittrock [15] presented a heuristic for the HFFS for the minimization of the work-in-progress (WIP) criterion. An adaptable space-based problem search method for the HFFS was proposed by Leon and Ramamoorthy [7]. Kurz and Askin [2] studied dispatching rules for flexible flow lines with identical machines and sequence dependent setup times. They explored three classes of heuristics. The first class of heuristics (cyclic heuristics) is based on the simplistic assignment of jobs to machines with little or no regard for the setup times. The second class of heuristics is based on the insertion heuristic for the traveling salesman problem (TSP). The third class of heuristics is based on Johnson's rule. They proposed eight heuristics (CH, RCH, SPTCH, FTMIH, CTMIH, MMIH, 1, *g* Johnson's rule, *g*/2, *g*/2 Johnson's rule) and compared the performances of those on a set of test problems.

Moreover, Kurz and Askin [3] formulated the sequence- dependent setup times (SDST) flexible flow lines as a mixed integer programming (MIP) model. Due to the difficulty in directly solving the MIP model, they developed a random keys genetic algorithm (RKGA). Problem data is generated to evaluate the RKGA with other dispatching rules, which they proposed aforetime. Zandieh et al. [16] proposed an immune algorithm for the same problem and showed that its algorithm outperforms the RKGA of Kurz and Askin [3] through the same benchmark. Ruiz and Stützle proposed an iterated greedy algorithm for permutation flowshop to minimize makespan [12] and SDST flowshop to minimize makespan and total weighted tardiness [13]. They compared IGA with other exiting methods to evaluate the algorithm. The effectiveness of IGA is shown in these two papers. Ruiz and Maroto [9] investigated hybrid flowshops with sequence-dependent setup times and unrelated machines. A complete survey of scheduling problems with setup times was given by Allahverdi et al [1]. Recently, Ruiz et al. [11] considered a realistic case of hybrid flexible flowshops with unrelated machines and some applied assumptions. They presented a mixed integer programming model and some heuristics for the problem.

3. Iterated greedy algorithm

Iterated greedy algorithm (IGA) is a metaheuristic approach to solve combinational optimization problems by iterating over greedy constructive heuristics. This algorithm is well-known in the computer science literature due to their simplicity with promising results. the use of a straightforward extension of an iterated local search to the context of greedy construction heuristics is the main advantage of using the IGA. It also provides very good results in a variety of applications. Jacobs and Brusco [4] applied the IGA to the set covering problem successfully. And in field of scheduling, Ruiz and Stützle [13] proposed an IGA for SDST flowshops and show that their algorithm is very effective. Therefore, we have been thinking of utilizing an iterated greedy algorithm for our problem.

The IGA generates a sequence of solutions by iterating over greedy constructive heuristics using two main phases, namely destruction and construction. In the destruction phase, some solution components are removed from a previously constructed complete candidate solution. The destruction procedure is applied and chosen randomly to a permutation *S* of *n* jobs without repeating d jobs. These jobs are then removed from *S* in the order they were chosen. This

procedure results in two subsequences: 1) the partial sequence SD with n–d jobs; and 2) the sequence of d jobs denoted as SR. SR contains jobs that have to be reinserted into SD to yield a complete candidate solution in the order they are removed from S. The construction procedure applies a greedy constructive heuristic to construct a complete candidate solution. We start with SD and insert the first job of SR, SR (1), into all possible n–d+1 positions of SD. The best position for SR (1) in the Augmented S_D sequence is the one that yields the smallest objective function. This process is iterated until S_R is

```
Procedure Iterated greedy

    $S$ := NEH_heuristic;
    $S$ := IterativeImprovment_Insertion(S);
    $S_b$ := $S$;
    while termination criterion not satisfied do
        $S'$ := $S$;                                   % Destruction phase
        for $i$ :=1 to $d$ do
            $S'$ :=remove one job at random from $S'$ and insert it in $S'_R$;
        endfor
        for $i$ := 1 to $d$ do                         % Construction phase
            $S'$ := best permutation obtained by inserting job $S_R(i)$ in all possible positions of $S'$;
        endfor
        $S''$ := Local search
        if TWCT($S''$) < TWCT($S$) then                % Acceptance Criterion
            $S$ := $S''$;
            if TWCT($S$) < TWCT($S_b$) then            % check if new best permutation
                $S_b$ := $S''$;
            endif
        elseif ($random^* \leq exp\{-(TWCT(S'')-TWCT(S))/Temperature\}$) then
            $S$ := $S''$;
        endif
    endwhile
    return $S_b$
end
```

* "*random*" is a random number distributed uniformly in [0, 1].

Fig. 1. General outline of the proposed IGA.

empty. By having done these two phases in order to improve each solution, we define a local search for the iterative greedy algorithm. There are many different alternatives for a local search algorithm to be considered. In this paper, we use a local search proposed by Ruiz and Stützle [12]. Finally, we consider whether the new sequence is accepted as an incumbent solution for the next iteration. We have two stopping criteria. The first one is the simplest acceptance criteria in order to accept new sequences, namely, if they provide a better mean completion time value. The second one is based on the simple SA-like acceptance criterion with a constant temperature. This constant temperature depending on the particular instance is computed by the formula below:

$$Temperature = \frac{\sum_{j=1}^{g}\sum_{i=1}^{n} p_{ij}}{n \times g \times 10}.T$$

Where T is a parameter that needs to be tuned.

3.1 *Encoding scheme and initial solution*

We use job-based representation to encode a solution. In job-based representation, the permutation of jobs is determined, and then by a dispatching rule the jobs are assigned to the machines. For example the first available machine. In FFL problems without considering SDST, the first available machine results in the earliest completion time, but while taking into account SDST FFL, this approach is not effective [9]. If setup times are considered in FFL, the way in which we assign the jobs to machines is modified accordingly, meaning that each job is assigned to the machine that accomplishes the job at the earliest time in a given stage. Our algorithm starts form NEH algorithm [8].

3.2 *Local search*

We applied a very simple local search. The procedure of this local search can be described as follows: The first job (x_1) in the sequence of current solution x is relocated to all possible positions in sequence. If any of these sequence v results in better makespan, current solution x is replaced by the new sequence v. This procedure iterates at most for all the subsequent jobs in sequence. If we include all the improvements in i-th < n, the local search for the current solution finishes. Figure 1 shows the general outline of the proposed IGA.

4. Experimental evaluation

In this section, the performance of the proposed IGA is evaluated by being compared with RKGA proposed by [3], SPT cyclic, FTMIH and (g/2, g/2) Johnson's rule from [2] and NEHH of [9]. We conduct an experimental evaluation. The algorithms are implemented in MATLAB 7.0 and run on a PC with 2.0 GHz Intel Core 2 Duo and 1 GB of RAM memory. We use relative percentage deviation (RPD) as performance measure to compare the methods. When the TWCT of each algorithm has been obtained for its instances, the best solution obtained for each instance (which is named Min_{sol}) is calculated by all the algorithms. Relative Percentage Deviation (RPD) is obtained by the given formula below:

$$\text{RPD} = 100 \bullet (Alg_{sol} - Min_{sol}) / Min_{sol} \quad (1)$$

where Alg_{sol} is the TWCT obtained for a given algorithm and instance. Obviously, lower values of RPD are preferable. The stopping criterion is $n^2 \times g \times 1.5$ milliseconds computational time. This stopping criterion not only permits for more time as the number of jobs or machines increases, but also is more sensitive toward a rise in number of jobs than number of stages.

Parameter tuning: It is known that the great choice of parameters of an algorithm can influence the performance of that algorithm. Our proposed IGA has two parameters, d and T. Initial instances show us that the value of $d = 2$ and $T = 0.5$ could be the best value for these parameters.

Data generation: Data required for a problem consist of the number of jobs (n), range of processing times (pt), number of stages (g) and whether all stages have the same number of machines or not. Each stage requires data defining how many machines exist at that stage ($m(t)$), range of the sequence dependent setup times (SDST), and the ready times. The ready times for stage 1 are set to 0 for all jobs. The ready times at stage $t + 1$ are the completion times at stage t, so this data need not be generated. We have $n = \{20, 50, 80, 120\}$ and $g = \{2, 4, 8\}$. The processing times are generated by the uniform distribution over range (1, 99). The duration of sequence-dependent setup times is defined as 25%, 50%, 100%, and 125% percent of processing time. The probability of skipping (Sp) a stage for each job is set at 0.10, or 0.40. The relative weights of the jobs are randomly generated from a uniform distribution over the range (1, 10). Therefore, there are 192 combinations of n, g, $m(t)$, pt, SDST and Sp, and for each combination we generate 5 instances. Factors and their levels are shown in Table 1.

Table 1
Factors and their levels

Factors	Levels	
Number of Jobs	20, 50, 80, 120	
Number of stages	2, 4, 8	
Machine distribution	Constant:	2
	Variable:	U (1,4)
Processing Time	U (1, 99)	
SDST	U (1, 25), U (1, 50), U (1, 99), U (1, 125)	
Skipping probability	0.10, 0.40	

4.1 *Experimental results*

We evaluate the algorithms in term of the selected objective function with the set of instances generated in previous subsection. The results of the experiments, averaged for each combination of n and g (80 data per average) are shown in Table 2. IGA outperforms the other algorithms with RPD (*Eq.* 1) of 0.63%. The worst performing algorithms are FTMIH and SPT cyclic with RPD of 31.4% and 25.58%.

Table 2
Average RPD for the algorithms grouped by n and g

Instance	SPT	FTIMH	John.	NEHH	IGA	RKGA
20×2	37.90	40.32	27.92	6.86	0.29	5.77
20×4	29.43	37.04	21.29	9.73	0.52	3.48
20×8	23.98	29.56	18.45	9.84	0.59	2.78
50×2	30.64	35.07	26.85	5.36	0.14	3.16
50×4	23.63	35.42	23.63	6.38	0.98	3.45
50×8	21.90	26.19	16.59	7.03	1.10	1.44
80×2	31.60	37.88	27.68	2.84	0.31	4.97
80×4	20.44	28.53	18.79	3.23	0.72	2.38
80×8	19.33	23.07	15.40	4.73	0.81	2.18
120×2	29.41	33.25	27.73	1.45	0.59	5.13
120×4	22.31	28.17	19.48	2.65	0.67	2.45
120×8	16.35	22.32	13.70	3.26	0.88	1.99
Average	25.58	31.40	21.46	5.28	0.63	3.26

For further precise analysis of the results, we carried out an analysis of variance (ANOVA). It is necessary to note that due to the considerable difference between SPT, FTMIH and (g/2, g/2) Johnson's rule and the other algorithms; we exclude them from our ANOVA. Means plot and LSD (Least Significant Difference) intervals at the 95% confidence level for the type of methods factor are shown in Figure 2. As could be seen, IGA statistically supersedes the other algorithms. To analyze the possible effects of number of jobs factor on the algorithms, we computed the performance of the algorithms in the different value of jobs. Figure 3 depicts the interaction between factors type of algorithm and number of jobs. Our IGA keeps a robust performance on various range of jobs, while NEHH outperforms RKGA in the case of n = 120.

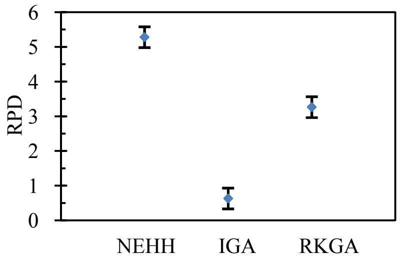

Fig. 2. Means plot and LSD intervals for the type algorithm

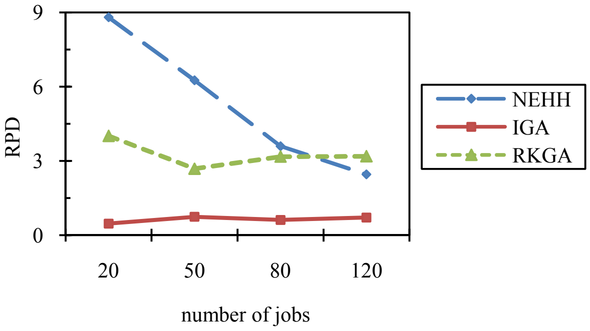

Fig. 3. Means plot for the interaction between factors type algorithm and number of jobs

5. Conclusion and future research

In this paper, we investigated flexible flow line scheduling problems where setup times were sequence dependent. Our optimization criterion was total weighted completion time. An effective iterated greedy algorithm was applied to tackle the problem. To evaluate the performance of IGA, we compared it with some existing algorithms in the literature using a standard. The results supported the effectiveness of our IGA.

As future research, it could be interesting to work on a population-based IGA for the problem and to compare its performances with the IGA proposed here. Another direction is to apply the IGA to other scheduling problems like job shops and open shops.

Reference

[1] A. Allahverdi, C.T. Ng, T.C.E. Cheng, Y.M. Kovalyov, A survey of scheduling problems with setup times or costs, European Journal of Operational Research, vol. 187(3), 985–1032, 2008.

[2] M. E. Kurz, R.G. Askin, Comparing scheduling rules for flexible flow lines. International Journal of Production Economics, vol. 85, 371–388, 2003.

[3] M. E. Kurz, R.G. Askin, Scheduling flexible flow lines with sequence-dependent setup times, European Journal of Operational Research, vol. 159(1), 66–82, 2004.

[4] L.W. Jacobs, M.J. Brusco, A local search heuristic for large set-covering problems, Naval Research Logistics Quarterly, vol. 42(7), 1129–1140, 1995.

[5] S.M. Johnson, Optimal two and three-stage production schedules with setup times included, Naval Research Logistics Quarterly, vol. 1, 61–67, 1954.

[6] Z. Jin, Z. Yang, T. Ito, Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem, International Journal of Production Economics, vol. 100: 322–334, 2006.

[7] V.R. Leon, B. Ramamoorthy, An adaptable problem space-based search method for flexible flow line scheduling, IIE Transactions, vol. 29, 115–125, 1997.

[8] M. Nawaz, E.E. Enscore Jr, I. Ham, A heuristic algorithm for the m-machine, *n*-job flowshop sequencing problem, Omega, vol. 11(1): 91–95, 1983.

[9] R. Ruiz, C. Maroto, A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility, European Journal of Operational Research, vol. 169: 781–800, 2006.

[10] R. Ruiz, A. Allahverdi, Some effective heuristics for no-wait flowshops with setup times to minimize total completion time, Annals Operations Research vol. 156: 143-171, 2007.

[11] R. Ruiz, F. Sivrikaya Serifoglu, T. Urlings, Modeling realistic hybrid flexible flowshop scheduling problems, Computers and Operations Research, vol. 35 (4), 1151–1175, 2008.

[12] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European Journal of Operational Research, vol. 177, 2033–2049, 2007.

[13] R. Ruiz, T. Stützle, An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives, European Journal of Operational Research, vol. 187(3), 1143-1159, 2008.

[14] M.S. Salvador, A solution to a special case of flow shop scheduling problems, In: S.E. Elmaghraby (Ed.), Symposium on the Theory of Scheduling and its Applications, 83–91, 1973.

[15] R.J. Wittrock, Scheduling algorithms for flexible flow lines, IBM Journal of Research and Development, vol. 29(4), 401–412, 1985.

[16] M. Zandieh, M., S.M.T. Fatemi Ghomi, S.M. Moattar Husseini, An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times, Applied Mathematics and Computation, vol. 180, 111–127, 2006.