Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 8 (16): 2843-2850, 2008
ISSN 1812-5654

A Variable Neighborhood Search for Hybrid Flexible Flowshops with Setup Times Minimizing Total Completion Time

[1]B. Naderi, [2]M. Khalili, [2]M.T. Taghavifard and [1]V. Roshanaei
[1]Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Avenue, Tehran, Iran
[2]Department of Industrial Engineering, Azad University, South Tehran Branch, Tehran, Iran

Abstract: In this study we investigate a complex variant of scheduling environments known as hybrid flexible flowshops to minimize total completion time where setup times are sequence dependent. Since this problem belongs to NP-hard class, we propose an effective Variable Neighborhood Search (VNS) metaheuristic algorithm to tackle the problem considered. Present proposed VNS embodies three advanced insertion Neighborhood Search Structures (NSS) together with the framework of Variable Neighborhood Descent (VND) that is a special version of VNS. For the evaluation of our proposed VNS, the standard benchmark of Taillard is used to compare the VNS against some other high-performing algorithms in the literature which have already proven their effectiveness. Computational results clearly reveal that our proposed VNS is competitive when compared to many other well known algorithms.

Key words: Scheduling, hybrid flexible flowshop, sequence dependent setup time, variable neighborhood search, variable neighborhood descent

INTRODUCTION

After more than 50 years of research in the field of scheduling, we still observe a remarkable gap between theory and practice in this area. Scheduling optimization normally aims at minimizing the total or maximum completion times of operations, leading to overall optimization of production cost. Hybrid Flexible Flowshop (HFFS) is one of the most important and realistic scheduling environments applicable to a wide variety of industries. For a process to qualify to be called hybrid, it must possess at least two machines in parallel in at least one stage while the term flexible indicates that each job can possibly skip some stages. Companies adopt many cost-effective policies to reduce their shop costs and remain competitive. One of the most effective strategies is to have a proper design of production environment and this is achieved by considering many applied realistic industrial assumption like for example sequence dependent setup times to obtain optimal or near optimal production sequence in the existence of the above assumption. In this study, intend to span the gap between theory and practice of scheduling by investigating a realistic and complex variant of flowshop known as hybrid flexible flowshops with the consideration of sequence dependent setup times to end up with an optimal or near optimal schedule. For this purpose, we introduce a novel metaheuristic algorithm, in form of Variable Neighborhood Search (VNS) to tackle the problem. In HFFS, a set of n jobs with the same operational sequence must go through a set of g production stages, each of which has several identical machines in parallel (Ruiz *et al*., 2008).

In HFFS, all n jobs need to be processed in the same order, starting at stage 1 until finishing in stage *g* with this realistic assumption that some of the jobs might skip some of the stages (Kurz and Askin, 2003). A machine cannot process simultaneously more than one job at a time and a job cannot be at the same time processed by more than one machine at a time. Additionally, we assume that all the tasks and jobs are independent and available for their process at time 0. The m machines are continuously available. Each job i is processed on at most one machine at each stage t. The process of a job i in 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. There is no transportation time between stages. The objective is to find a production sequence of the jobs on the machines such that one or some selected criteria are optimized.

Flowshop problems and its extensions have recently gained momentum toward perfection in application and been extremely favored by researchers after appearing the first methodical study in the 1950s with the study of Johnson (1954). A variety of applied assumptions have

Corresponding Author: B. Naderi, Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Avenue, Tehran, Iran

been taken into account by many researchers since then. Among them, considering sequence dependent setup times have recently become popular among researchers those intend to investigate the scheduling decisions in real manner (Allahverdi *et al*., 2008). The motivation behind the utilization of this assumption is to obtain an optimal sequence of jobs as well as tremendous savings when set up times are explicitly included in scheduling decisions (Allahverdi *et al*., 2008). With respect to the corresponding explanation, we consider that there exist sequence dependent setup times in our problem. Since Johnson's (1954) pioneering work on the two machine regular permutation flowshops, a lot of innovatively effective methods including both exact and heuristic methods for the flowshop and its other extensions have been propounded to find a better solution (optimal scheduling sequence). Johnson (1954) introduces a heuristic for two-machine flow shop with setup times included to minimize makespan. Kurz and Askin (2003) consider dispatching rules for SDST HFFS. They investigate three classes of heuristics based on naïve greedy heuristics, the insertion heuristics and Johnson's rule.

A one-machine sequence dependent setup time scheduling problem is equivalent to a traveling-salesman problem and is NP-hard (Luh *et al*., 1998). Even for a small system, the complexity of this problem is beyond the exact theories (Luh *et al*., 1998). Hybrid flexible flowshop problems are considerably more complex than the regular single machine scheduling. On the other hand, Gupta (1988) shows the flowshop with multiple processors (FSMP) problem with only two stages (m = 2) is NP-hard even when one of the two stages contains a single machine. Since the FSMP problem is a specific variant of the hybrid flexible flowshop, we can conclude then that this latter problem is also NP-hard (Ruiz *et al*., 2005).

Since hybrid flexible flowshops scheduling is essentially complex; exact method does not possess the potentiality to overcome the complexity of these problems and are not capable to tackle them within reasonable amount of time. Hence, a variety of algorithms into two main groups, heuristics and metaheuristics, have been applied to solve the problems to find optimal or near optimal schedule (Kurz and Askin, 2003, 2004; Zandieh *et al*., 2006).

Kurz and Askin (2004) formulate the SDST HFFS as an integer programming model. Because of the difficulty in solving the IP model directly, they develop a Random Keys Genetic Algorithm (RKGA). Problem data is generated to evaluate the RKGA with other scheduling heuristics rules, which they propose aforetime. They create a lower bound to evaluate the heuristics. Zandieh *et al*. (2006) propose an immune algorithm and show the outperformance of its algorithm over the RKGA of Kurz and Askin (2004). A complete survey of scheduling problems with setup times is given by Allahverdi *et al*. (2008). Recently, Ruiz *et al*. (2008) considered a realistic case of hybrid flexible flowshops with unrelated machines and some applied assumptions. They presented a mixed integer programming and some heuristics for the problem. Although some of the existing algorithms are effective, they usually obtain the outstanding results at the expense of being entirely sophisticated. In this study, it aims at overcoming this drawback by proposing a Variable Neighborhood Search (VNS) metaheuristic that is very effective, yet simple to understand. The proposed metaheuristic employs three advanced neighborhood search structures based on insertion neighborhoods in framework of a specific case of VNS, called Variable Neighborhood Descent (VND).

In a nutshell, we consider a realistic shop scheduling known as hybrid flexible flowshop with sequence dependent setup times to minimize Total Completion Time (TCT) for the purpose of narrowing the slot between theory and practice of scheduling. Meanwhile, we code some other effective metaheuristic algorithms with the same benchmark to draw analogies between the performances of the algorithms with our proposed VNS.

VARIABLE NEIGHBORHOOD SEARCH (VNS) METAHEURISTIC

The background of VNS: Recently, an effective algorithm, variable neighborhood search (hereinafter referred to as VNS), has been gained popularity among researchers. Considerable number of papers reporting the successful performance of VNS in various fields attests to (acknowledge) our statement (Flesza and Hindi, 2004; Liao and Cheng, 2007; Gao *et al*., 2008). The term variable neighborhood search is referred to all local search based approaches that are centered on the principle of systematically exploring more than one type of neighborhood structure during the search.

VNS is closely related to Iterated Local Search (ILS): instead of iterating over one constant type of neighborhood structure (i.e., local search) as done in ILS, VNS iterates in an analogous way over some neighborhood structures until some stopping criterion is met. The central observations of VNS are: (1) A local minimum one neighborhood structure is not necessarily the locally minimal with respect to another neighborhood structure. (2) A global optimum is the locally optimal with respect to all neighborhood structures.

Since, all metaheuristics majority make use of just one type neighborhood structure, high probability for them to get trapped in local optima after a certain number of iterations is supposed. Therefore, it necessitates developing a strong algorithm enjoying diverse systematic neighborhood search structure and sufficient potentiality to escape from local optima. The reasons why VNS has obtained its acceptability and popularity among researchers are due to the utilization of several neighborhood structures, easy to implement and high flexibility and brilliant adaptability of VNS for different problems. A variant of VNS is Variable Neighborhood Descent (VND) with these two main differences: (1) In VND, change of the neighborhoods is performed in a deterministic way while in basic VNS, the neighborhoods are explored randomly. (2) In VND, usually the first improvement is accepted instead of the best improvement while in basic VNS, it is conversely applied. In subsequent subsection, we are applying a VNS with advanced and powerful neighborhood structures under the framework of VND.

The proposed VNS: Making a solution recognizable for an algorithm (encoding scheme) is the first thing to do to apply a metaheuristic to the scheduling (Ruiz and Maroto, 2006). By making use of job-based representation we determine the permutation of the jobs and then by a dispatching rule the jobs are assigned to the machines, for example the first available machine. In HFFS problems without considering SDST, the first available machine results in the earliest completion time, but while taking into account SDST HFFS, this approach lose its effectiveness. If setup times are incorporated in HFFS, 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.

In a nutshell, variable neighborhood search starts from an initial solution. The significant role of initial solution on the quality of final results of algorithms has already been recognized by many researchers in recent years. Since almost all metaheuristics providing the best results so far start from NEH (first proposed by Nawaz *et al*., 1983) and its variants (Ruiz and Maroto, 2006; Kalczynski and Kamburowski, 2008; Osman and Potts, 1989), the initial sequence of the VNS is determined by this heuristic. The initial solution of our VNS is produced by an extension of NEH. In this study the application of NEH to hybrid flowshop is used, called NEHH (proposed by Ruiz and Maroto, 2006).

Since, VNS improves its current solution by the means of different neighborhood search structures, study proposed VNS utilizes three different types of advanced neighborhood structures (NS) to comply with the framework of VND version of VNS. Systematic switch of one type NSs to another one is done to purposefully lead us to maintain the probability of visiting the better solutions.

All three types of NSs that we define are based on insertion neighborhood. Many researchers have concluded that the insertion neighborhood is superior to the swap or exchange neighborhoods in the field of scheduling (Osman and Potts, 1989; Naderi *et al*., 2008). The insertion neighborhood of a sequence comprises all those sequences that can be obtained by removing some jobs and inserting them in other positions. With regard to the given explanation, in each insertion neighborhood, three main decisions should be made:

- No. of jobs to be removed from a complete sequence π
- How to choose these job (s)
- How to relocate the removed job(s) to construct a new complete sequence π'

First NS of study is defined as such: a job is removed from the sequence at random and without repetition and then inserted in all possible n positions. Considering the above definition, the three main decisional factors are adopted as such: number of removed job is one. All n jobs are selected one by one without repetition at a random order. That certain removed job is inserted in all n positions obtained from the n-1 remaining jobs. The procedure is illustrated by applying it to an example. Suppose n = 4 and current solution is {3-4-1-2}. The first removed job is randomly chosen to be 4. It can be inserted into 4 possible positions (the first, second, third or fourth position in the sequence). We start inserting job 4 in all possible positions from left to right. If you observe the first improvement, the associated sequence is accepted and the procedure restarts. If not, this mechanism repeats for the n-1 subsequent jobs. As soon as the first improvement is observed the procedure restates. The procedure of NS type 1 is shown in Fig. 1.

The whole procedure repeats so long as no improvement is obtained through inserting of all the n jobs in all the n possible positions. Now, with respect to the fact that all the possible sequences acquired from changing only one job have been exploited, we definitely believe that there is no hope for further improvement. Hence, it necessitates presenting another NS to escape from this local optima of the first NS. According to this research finding, we need to introduce a NS to generate farther neighbors than just changing the position of one job.

```
Procedure: Neighborhood_stucture_type_1

improvement = yes
while (improvement = yes) do
  improvement = no
  i = 1
  while, i < n + 1 do
        remove job h at random and without repetition from the current sequence π
        ii = 0
        while, ii = < n + 1 do
          π′ = put job h into the ii-th position of current sequence π
          if TCT( π′ ) < TCT(π) do
                A = B
                ii = n
                i = n
                improvement = yes
           endif
          ii = ii + 1
        endwhile
        i = i + 1
  endwhile
endwhile
```

Fig. 1: Procedure of neighborhood structure type 1

```
Procedure: Neighborhood_stucture_type_2

i = 1
while, i < n + 1 do
     ii = i + 1
     while, ii = < n + 1 do
          remove i and ii-th jobs of current sequence (A) from the sequence
          (B) = relocate these two jobs into two random positions
          if TCT(B) < TCT(A) do
               A = B
               ii = n
               i = n
          endif
          ii = ii + 1
    endwhile
    i = i + 1
endwhile
```

Fig. 2: Procedure of neighborhood structure type 2

In second NSS the number of removed job is set equal to 2. The manner of choosing these 2 jobs is all the combinations of two-out-of-n jobs. Due to the large number of sequences created by inserting two selected jobs into all the possible positions, we decide to relocate the two jobs into two randomly selected positions. Suppose that this random relocation represents all the possible simultaneous changes in the two corresponding

Procedure: *Neighborhood_stucture_type_3*

s = *TCT*(A)
for i = 1 to ϕ **do**
    remove three randomly selected jobs from the current sequence π
    π'_i = relocate these three jobs into three random positions
endfor
A = Find Best (π'_i) % even if sequence *Best* (π'_i) does not improve TCT (π')

Fig. 3: Procedure of neighborhood structure type 3

Procedure: *The_proposed_variable_neiborhood_search*

A = *initialization* (By NEHH heuristic)
k = 1
while, *the stopping criterion is not met* **do**
    apply NSS type k
    if *A is improved* **do**
        k = 1
    else
        $k = k + 1$
        if $k = 4$ **do**
            k = 1
        endif
    endif
endwhile

Fig. 4: General outline of the proposed VNS

jobs because it is very costly to check all the possible insertions produced by the two selected jobs. Figure 2 reports the general outline of NS type 2.

Having observed the first improvement, the associated sequence is accepted and the procedure restarts from NS type 1. If not, this mechanism repeats for the subsequent combinations. By the same reasoning of switching from NS type 1 to type 2, NS type 2 terminates and NS type 3 starts. In the third NS, the three randomly selected jobs are randomly relocated into three other randomly selected positions. As can be seen, we randomly choose three jobs as well as randomly relocating the jobs in three different positions. In the NS type 3, the number of sequences (ϕ) which are supposed to be generated during NS type 3 is the parameter that needs to be tuned. After observing the first improvement, the corresponding sequence is accepted to move and the procedure restarts from NS type one. In addition to the mentioned decisional difference between NSs, in NS type 3, algorithm is obliged to accept one move (the best sequence among ϕ produced sequences), even if it is inferior with respect to the current solution. This is so because returning to NSS type 1 and 2 with the former candidate solution most likely does not improve the best solution. This mechanism can be considered as an acceptance criterion of our proposed VNS to avoid stagnation situations during search. The procedure of NS type 3 and the general outline of the proposed VNS are shown in Fig. 3 and 4, respectively.

EXPERIMENTAL DESIGN

Here, it is aimed to compare the results of the proposed VNS with the other existing algorithms. Among the existing methods, we choose some well-known heuristics including SPT Cyclic, FTMIH and The g/2, g/2 Johnson's Rule from Kurz and Askin (2003), NEHH of Ruiz and Maroto (2006) as well as some other metaheuristics including RKGA proposed by Kurz and Askin (2003) and the immune algorithm of

Zandieh *et al.* (IA_MZ) (2006). These algorithms have shown the superior performance in the papers they have been applied.

Since, SPTcyclic, FTMIH and (g/2, g/2) Johnson rule only give one sequence, we expect them to be inferior to the NEHH and the other metaheuristics which visit many sequences iteratively. Study purpose is to bring them into the comparison is to make use of them as the upper bounds for a given instance.

In this study, implement the algorithms in MATLAB 7.0 and run on a PC with 2.0 GHz Intel Core 2 Duo and 2 GB of RAM memory. The stopping criterion used when testing all instances with the algorithms is set to a CPU time limit fixed to $n^2 \times g \times 1.5$ millisecond. This stopping criterion is chosen because it not only permits for more time as the number of jobs or stages increases, but also is more sensitive toward a rise in number of jobs than number of stages.

Study used relative percentage deviation (RPD) as a common performance measure to compare the methods (Ruiz and Stützle, 2008). The best solutions obtained for each instance (which is named Min_{sol}) are calculated by any of the algorithms. RPD is obtained by given formula below:

$$RPD = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100 \quad (1)$$

where, Alg_{sol} is the objective function value obtained for a given algorithm and instance. Clearly, lower values of RPD are preferable.

Data generation: Data required for this problem consist of number of jobs (n), number of stages (g), number of identical machines in each stage (m(j)), range of processing times (p), range of the sequence dependent setup times (sdst) and ready times. We have n = {20, 50, 80, 120} and g = {2, 4, 8}. To define the number of machines at each stage, we have to sets. In the first one, we have a random uniformly distribution number of machines of between one and three machines per stage and in the second one, we have a fixed number of two machines per stage. The ready times for stage 1 are set to 0 for all jobs. The ready times at stage (j+1) are the completion times at stage j, so this data need not be generated. The processing times are uniformly distributed over the range (1, 99). The sequence dependent setup times are randomly generated from four uniform distributions over the ranges (1, 25), (1, 50), (1, 99) and (1, 125). The flexibility is considered by allowing some jobs to skip some stages. The probability of skipping a stage is set at 0.1 and 0.4. Factors and their levels are shown in Table 1.

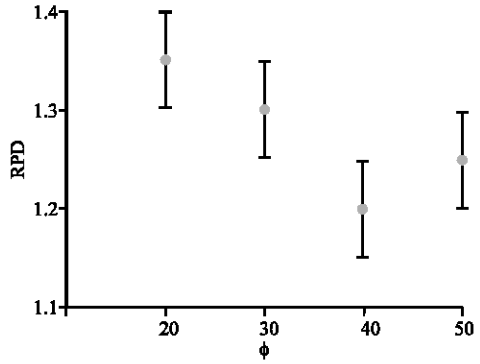

Fig. 5: Means plot and LSD intervals (at the 95% confidence level) for the different levels of parameter ϕ

Table 1: Factors and their levels

Factors	Levels			
No. of jobs	20	50	80	120
No. of stages	2	4	8	
Machine distribution	a. Constant: 2			
	b. Variable: U (1, 3)			
Processing time	U (1, 99)			
SDST	U (1, 25)	U (1, 49)	U (1, 99)	U (1, 125)
Skipping probability	0.1	0.4		

Parameter tuning: One of the advantages of our proposed VNS is that it has only one parameter (ϕ) to be tuned. The considered levels are 20, 30, 40 and 50. A set of 90 instances including 10 instances for each combination of n and g is randomly generated. All the 90 instances are solved by 4 VNSs produced by the cited levels. It is necessary to notice that for using ANOVA, three main hypotheses, normality, homogeneity of variance and independence of residuals, must be checked. We did that and found no bias for questioning the validity of the experiment. The means plot and LSD intervals for different levels of parameter ϕ are shown in Fig. 5. As it could be seen, ϕ of 40 provides the best results among the levels.

EXPERIMENTAL RESULTS

We research, it is intended to evaluate our proposed VNS and the other methods. The objective is to find a sequence which minimizes total completion time. Here is used RPD (Eq. 1) measure to compare the performances of algorithms. The results of the experiments, averaged for each combination of n and g (160 data per average) are reported in Table 2. As expected, the metaheuristics algorithms dramatically outperform the heuristics. The best algorithm is proposed VNS with the RPD of 1.59%. After VNS, IA_MZ and RKGA demonstrate better performances with RPD of 3.28 and 4.21, respectively. The worst performing algorithms

Table 2: Average relative percentage deviation (RPD) for the algorithms grouped by *n* and *g*

		Algorithm						
n	g	John.	SPTcyclic	FTIMH	NEHH	VNS	RKGA	IA_MZ
20	2	24.92	34.90	37.32	8.77	0.82	4.20	3.20
	4	18.29	26.43	34.04	9.48	1.28	5.94	3.63
	8	15.45	20.98	26.56	7.78	1.14	3.50	3.57
50	2	23.85	27.64	32.07	8.16	1.20	3.06	3.48
	4	20.63	20.63	32.42	5.45	1.79	4.44	2.61
	8	13.59	18.90	23.19	6.44	1.87	3.57	3.25
80	2	24.68	28.60	34.88	6.97	1.36	3.95	2.93
	4	15.79	17.44	25.53	7.38	0.91	3.90	3.34
	8	12.40	16.33	20.07	6.18	1.66	3.80	2.31
120	2	24.73	26.41	30.25	5.83	2.48	5.48	3.76
	4	16.48	19.31	25.17	6.45	2.34	4.18	3.38
	8	10.70	13.35	19.32	5.49	2.27	4.49	3.86
Average	18.46	22.58	28.40	7.03	1.59	4.21	3.28	

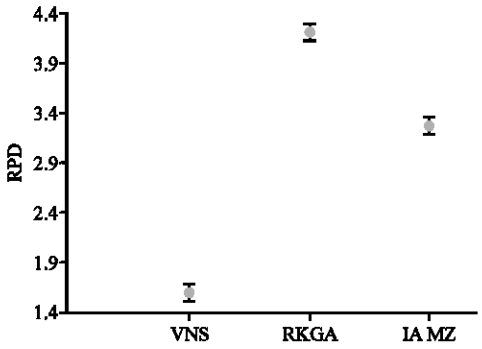

Fig. 6: Means plot and LSD intervals (at the 95% confidence level) for the type of algorithm factor

are FTIMH, SPTcyclic and John with RPD of 28.4, 22.58 and 18.46, respectively. Among heuristic algorithms, NEHH yields more acceptable performance with RPD of 7.03 and is very close to the performances of metaheuristic algorithms. As the Table 2 reveals the performances of heuristic algorithms improve as the number of jobs increases whereas metaheuristics sustain their robustness in almost all instances. The sustainable behavior of metaheuristic algorithms are quite tangible but the strength of proposed VNS to find better results due to its several systematic neighborhood search structure is more dazzling. Comparing the results of the metaheuristic and well-known dispatching rules shows that the applied metaheuristics provide convincing results in general assessment.

For further precise analysis of the results, conduct an analysis of variance (ANOVA) for each subset. Due to the considerable difference between the performance of the heuristics, (i.e., SPTcyclic, FTMIH, (g/2, g/2) Johnson's rule and NEHH) and the metaheuristics (i.e., RKGA, VNS and IA_MZ); we take the heuristics out of the experiment. This way we have 4 different algorithms and 1920 treatments.

The results indicate that there are statistically significant differences between the different algorithms with a p-value very close to zero. The means plot and LSD intervals (at the 95% confidence level) for the different metaheuristics are shown in Fig. 6. Clearly, the VNS is, by far, the best metaheuristic among those tested.

CONCLUSION AND FUTURE RESEARCH

In this research, it has investigated the problem of hybrid flexible flowshops scheduling with sequence dependent setup times under minimization of total completion time. To tackle such a NP-hard problem, here propounded a Variable Neighborhood Search (VNS) implemented in a specific framework of VNS, called Variable Neighborhood Descent (VND) because switching from one neighborhood search structure (NSS) to another is purposefully done. The VNS has employed three advanced NSSs based on different applications of insertion neighborhood. By these NSSs, it has tried to make a compromise between small and large lengths of NSSs and consequently enhance the competitiveness of our proposed algorithm in a big way. Another advantage of the VNS is its conceptual simplicity to understand and implement. In order to evaluate the effectiveness of our proposed VNS, we have compared the VNS with some high-performing metaheuristics in the literature as well as some well-known dispatching rules. The comparison among the performances of the algorithms revealed the absolute superiority of our proposed VNS over the other algorithms.

As a direction for future research, it could be interesting to work on some other metaheuristics, such as particle swarm optimization and electromagnetic-like algorithms and compare them with our VNS, or to examine the performance of our algorithm in some other complex scheduling problems, such as job shop and open shop, to see as to whether the high performance of our VNS is

transferable to the other scheduling problems. Another clue for future research is the consideration of some other realistic assumptions, such as machine availability constraints and unrelated machines. Another opportunity for research is considering the problem with the other optimization objectives, such as total completion time and total tardiness, or even multi objective cases.

REFERENCES

Allahverdi, A., C.T. Ng, T.C.E. Cheng and Y.M. Kovalyov, 2008. A survey of scheduling problems with setup times or costs. Eur. J. Operat. Res., 187: 985-1032.

Flesza, K. and K.S. Hindi, 2004. Solving the resource-constrained project scheduling problem by a variable neighborhood search. Eur. J. Operat. Res., 155: 402-413.

Gao, J., L. Sun and M. Gen, 2008. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Comput. Operat. Res., 35: 2892-2907.

Gupta, J.N.D., 1988. Two-stage hybrid flowshop scheduling problem. J. Operat. Res. Soc., 39: 359-364.

Johnson, S.M., 1954. Optimal two and three-stage production schedules with setup times included. Naval Res. Logistics Q., 1: 61-67.

Kalczynski, P.J. and J. Kamburowski, 2008. An improved NEH heuristic to minimize makespan in permutation flow shops. Comput. Operat. Res., 35: 3001-3008.

Kurz, M.E. and R.G. Askin, 2003. Comparing scheduling rules for flexible flow lines. Int. J. Prod. Econ., 85: 371-388.

Kurz, M.E. and R.G. Askin, 2004. Scheduling flexible flow lines with sequence-dependent setup times. Eur. J. Operat. Res., 159: 66-82.

Liao, C.J. and C.C. Cheng, 2007. A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. Comput. Ind. Eng., 52: 404-413.

Luh, P.B., L. Gou, Y. Zhang, T. Nagahora, M. Tsuji, K. Yoneda, T. Hasegawa, Y. Kyoya and T. Kano, 1998. Job shop scheduling with group dependent setups, finite buffers and long time horizon. Ann. Operat. Res., 76: 233-259.

Nawaz, M., E.E. Enscore Jr. and I. Ham, 1983. A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. Omega, 11: 91-95.

Osman, I. and C. Potts, 1989. Simulated annealing for permutation flowshop scheduling. Omega, 17: 551-557.

Ruiz, R., C. Maroto and J. Alcaraz, 2005. Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. Eur. J. Operat. Res., 165: 34-54.

Ruiz, R. and C. Maroto, 2006. A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. Eur. J. Operat. Res., 169: 781-800.

Ruiz, R. and T. Stützle, 2008. An Iterated Greedy heuristic for the sequence-dependent setup time flowshop problem with makespan and weighted tardiness objectives. Eur. J. Operat. Res., 187: 1143-1159.

Ruiz, R., F.S. Serifoglu and T. Urlings, 2008. Modeling realistic hybrid flexible flowshop scheduling problems. Comput. Operat. Res., 35: 1151-1175.

Zandieh, M., S.M.T. Fatemi Ghomi and S.M. Moattar Husseini, 2006. An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. J. Applied Math. Comput., 180: 111-127.