Int J Adv Manuf Technol (2011) 52:1031–1038
DOI 10.1007/s00170-010-2792-2

ORIGINAL ARTICLE

A hybrid scatter search for the partial job shop scheduling problem

Mohammad Mahdi Nasiri · Farhad Kianfar

Received: 12 February 2010 / Accepted: 15 June 2010 / Published online: 4 July 2010

Abstract This paper presents a special case of the general shop called partial job shop problem. The partial job shop is a more realistic generalization of the mixed shop problem. The problem is formulated as a mixed integer programming model. A scatter search algorithm combined with tabu search and path relinking is used to tackle this problem with makespan criterion. The computational experiments are performed on some problem instances. The results are compared with a lower bound and the effectiveness of the algorithm is shown.

Keywords Scheduling · Partial job shop · Scatter search · Mixed shop · General shop

1 Introduction

One of the most interesting areas for researchers in the scheduling literature is the job shop scheduling problem (JSP) with makespan criterion. Since it is proven that the job shop problem is NP-hard (the proof can be found in [1]), many heuristic and metaheuristic algorithms have been developed to find a near optimal solution for JSP. Among these efforts, those that used tabu search as a part of their algorithms had better results.

Nowicki and Smutnicki [2] proposed TSAB algorithm which is well known for a solution of 10×10 benchmark within 30 s on a now dated computer. The problem remained unsolved for more than a quarter of a century. Pezzella and Merelli [3] combined tabu search method with shifting bottleneck procedure and presented TSSB. Ponnambalam et al. [4] considered tabu search to solve the job shop problem with makespan criterion and increased the number of solutions in the neighborhood. Nowicki and Smutnicki [5] proposed *i*-TSAB algorithm that generated a revolution in the literature of the job shop scheduling problem. Watson et al. [6] deconstructed *i*-TSAB algorithm and proposed several algorithms; however, none of them was more effective than *i*-TSAB. Zhang et al. [7] tried to extend the neighborhood of Balas and Vazacopoulos [8] and used it in a TS framework. Zhang et al. [9] combined TS and SA and proposed TSSA that outperformed most of the prior algorithms, except *i*-TSAB. Huang and Liao [10] presented a hybrid algorithm combining ant colony optimization with the tabu search. Eswaramurthy and Tamilarasi [11] presented an application of tabu search combined with the ant colony optimization technique to solve the job shop scheduling problems. The neighborhoods were selected based on the strategies in the ant colony optimization with dynamic tabu length strategies in the tabu search.

Genetic algorithm is another popular approach for solving JSP. Ponnambalam et al. [12] presented a method for estimating the genetic algorithm parameters for JSP. Wang and Zheng [13] used an effective crossover operator for operation-based representation. The classical mutation operator was replaced by the metropolis sample process of simulated annealing with a probabilistic jumping property to enhance the neighborhood search. Liu et al. [14] proposed an improved genetic algorithm, called the hybrid Taguchi-genetic algorithm, to solve JSP. Amirthagadeswaran and Arunachalam [15] proposed a new method of the representation of jobs and schedule deduction to be used in GA for solving the job shop

M. M. Nasiri (✉) · F. Kianfar
Department of Industrial Engineering,
Sharif University of Technology,
Tehran, Iran
e-mail: mmnasiri@gmail.com

F. Kianfar
e-mail: fkianfar@sharif.edu

problem. Amirthagadeswaran and Arunachalam [16] developed a new genetic algorithm using inversion operator. Xu and Li [17] used the framework of IGA that is a combination of the immune theory and the genetic algorithm. Zhang et al. [18] proposed a new crossover operator, called the precedence operation crossover, for the operation-based representation which can preserve the meaningful characteristics of the previous generation. Wang et al. [19] proposed a novel genetic chromosome-encoding approach; in this encoding method, the operation of crossover and mutation was done in a three-dimensional coded space.

In addition to TS and GA, many other approaches have been applied to JSP with makespan criterion. Udomsakdigool and Kachitvichyanukul [20] presented a multiple colony ant algorithm to solve JSP. In a multiple colony ant algorithm, ants cooperate to find good solutions by exchanging information among colonies which are stored in a master pheromone matrix that serves the role of global memory. Furthermore, Pardalos et al. [21] conceived a new efficient metaheuristic. The latter uses the backbone and “big valley” properties of the job shop scheduling problem. Rego and Duarte [22] proposed a heuristic algorithm that combines shifting bottleneck procedure with a filter-and-fan approach. Moreover, Lin et al. [23] presented a new hybrid swarm intelligence algorithm consisting of particle swarm optimization, simulated annealing technique, and multi-type individual enhancement scheme. The most well-known benchmark problems for the job shop scheduling with makespan criterion are Taillard problems. The newest results for these problems are due to an unpublished work of Nasiri and Kianfar and are accessible in Taillard’s web site [24].

Furthermore, there are several extensions of the job shop problem in the literature. The *mixed shop* is a combination of the job shop and the open shop problems. Thus, in a mixed shop, we have open shop jobs and job shop jobs ([25]). In its general form, the mixed shop problem is NP-hard. Shakhlevich et al. [26] discussed the complexity of mixed shop problems under various criteria and clarified the boundary between polynomially solvable and NP-hard problems.

The *general shop* problem is the most generalized problem in the group of shop scheduling problems (for more information, see [25]). In addition, there are other generalizations of the job shop problem. In Ramudhin and Marier [27], a problem entitled “shops with a partial ordering on operations pertaining to each job or machine” is stated and solved by an extension of the shifting bottleneck method. Furthermore, Kis [28] presents an extension of the job shop scheduling problem where the job routings are directed acyclic graphs. These graphs can model partial orders of the operations and contain sets of alternative subgraphs consisting of several operations each.

In the *partial job shop* problem that is presented in this paper, the operations of a job can have any acyclic graph for their precedence relations. If no precedence relation exists between any pair of operations of a job, the job is like an open shop job in the mixed shop problem. Similarly, if the sequence of the operations of a job is completely predetermined, the job is like a job shop job in a mixed shop problem. Consequently, the partial job shop is a generalization of the mixed shop. Furthermore, it is a special case of the general shop problem. In the general shop problem, there is no restriction in the precedence relations between the operations so that the operation of one job could be a precedent of an operation of another job. Such an assumption can complicate the graph of the problem, and we think it has rare application in the real world. Therefore, in the partial job shop, the operation of one job cannot be a predecessor of an operation of another job. This setting is very practical in the manufacturing shops that contain several different machines. In a real metal machining shop, the partial job shop is the best model that fit in. Prior to introducing the partial job shop, the mixed shop problem is developed for responding to the need for the existence of both job shop jobs and open shop jobs in a shop. In a metal machining shop, the job is the workpiece. The precedence relations of the operations that should be performed on a workpiece depend on the design of the workpiece. Generally, some of the operations have a predetermined order and some of them do not. Thus, even the mixed shop model may be inappropriate and we may need the partial job shop model.

Furthermore, scatter search is applied to project scheduling problems, and the results are astounding [29–31]. In addition, the concept of combined path relinking and tabu search is successfully utilized in [5]. Thus, we have combined the idea of scatter search, path relinking, and tabu search to develop an effective algorithm for the partial job shop scheduling problem.

The remainder of the paper is organized as follows. Section 2 gives the problem formulation and notations. The third section describes the hybrid scatter search algorithm. Computational results are included in Section 4. Finally, the last section concludes the paper.

2 Problem formulation and notations

A partial job shop consists of a set of jobs $J=\{1,\ldots,n\}$, a set of machines $M=\{1,\ldots,m\}$, and a set of operations $O=\{1,\ldots,o\}$. Each job j has a set of precedence relations R_j that determines the order of its operations. Moreover, operation i should be processed on a machine $\mu_i \in M$ during an uninterrupted processing time $p_i > 0, i \in O$. The set of operations^{O} can be decomposed into subsets $M_k = \{i \in O : \mu_i = k\}$, each of which corresponding to operations that should be processed on machine ^{k}; let $m_k = |\mathrm{M}_k|$, $k \in M$. The schedule is represented by the *processing order* of operations on machines, i.e., by m-tuple $\pi=(\pi_1,\ldots,\pi_m)$, where $\pi_k=(\pi_k(1),\ldots,\pi_k(m_k))$ is a permutation on $M_k, k\in M; \pi_k(i)$ denotes the element of M_k, which is in position i in π_k. Then, $E(\pi)$ can be defined as

$$E(\pi) = \bigcup_{k=1}^{m} \bigcup_{i=1}^{m_k-1} \{(\pi_k(i), \pi_k(i+1))\}.$$

The rest of the assumptions are as follows:

- No machine can process more than one job (operation) at a time.
- The processing of the operations cannot be interrupted.
- All jobs and all machines are available from time 0 on.
- Each job visits each machine at most once; in other words, recirculation is not permitted.
- No precedence relation between operations of different jobs is allowed.

Figure 1 shows an instance of the partial job shop problem with two jobs. For example, (1,3,2,4,5,6) is a feasible sequence for the operations of job 1, but (1,2,3,6,4,5) is not feasible.

2.1 MIP model

We give a unique index to each of the operations for simplicity. The parameters of the problem are as follows:

n	Number of jobs
m	Number of machines
o	Number of operations
j	Index for jobs $\{1,\ldots,n\}$
i	Index for machines $\{1,2,\ldots,m\}$
a	Index for operations $\{1,2,\ldots,o\}$
R_j	Set of precedence relations (direct or indirect) of the operations of job j
L	Set of all operations
M_i	Set of operations that should be processed on machine i
J_j	Set of operations that belongs to job j
p_a	Processing time of operation a

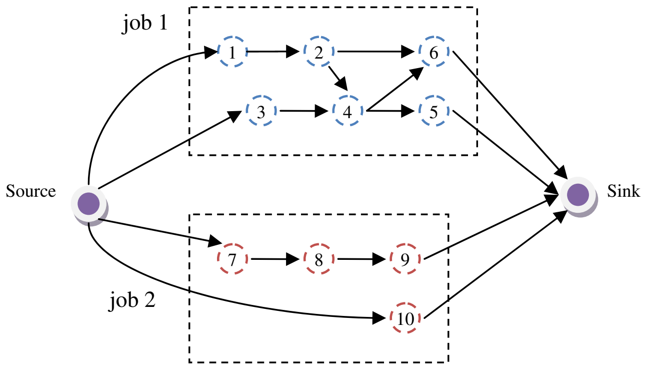


Fig. 1 Directed graph of the precedence relations for a partial job shop problem

The decision variable is y_a and indicates the start time of operation a.
Min C_{max}
s.t.

$$y_a - y_b \geq p_b \text{ or } y_b - y_a \geq p_a \qquad a, b \in M_i \tag{1}$$

$$y_a - y_b \geq p_b \text{ or } y_b - y_a \geq p_a \quad a, b \in J_j \text{ and } (a,b) \notin R_j \text{ and } (b,a) \notin R_j \tag{2}$$

$$y_b - y_a \geq p_a \qquad (a,b) \in R_j \tag{3}$$

$$C_{max} - y_a \geq p_a \qquad a \in L \tag{4}$$

$$y_a \geq 0 \qquad a \in L \tag{5}$$

Constraint set 1 ensures that some ordering exists among the operations of different jobs that have to be processed on the same machine. Constraint set 2 ensures that some ordering exists among operations that belong to the same job. Constraint set 3 ensures that the precedence relations (that make the problem a partial job shop problem) are preserved. Constraint set 4 computes $C_{\max}$, and constraint set 5 ensures that all the operations are started after the time 0.

2.2 Lower bound

A simple lower bound can be proposed similar to one of the lower bounds of the job shop that derivates from one machine problem. The lower bound can be computed by means of heads (r_l) and tails (q_l),

$$\text{LB1} = \max\{\min\{r_l : l \in M_i\} + \sum_{l \in M_i} p_l + \min\{q_l : l \in M_i\} : i \in \{1,\ldots,m\}\},$$

where r_l is the length of the longest path among the paths going to node l (excluding the node weight) and q_l is the length of the longest path among the paths going out from node l (excluding the node weight) in a graph that nodes represent the operations, arcs represent the precedence relations, and weights of nodes represent the processing times.

Since the partial job shop problem is a generalization of the job shop problem, so it is obvious that the problem is NP-hard. Consequently, we have to develop a heuristic or use metaheuristic algorithms to obtain a near optimal solution.

```
1. Generate an initial population $P$ with size of $|P|$.
2. Construct $RefSet$ using the reference set construction method.
3. Build $NewSubsets$ with the subset building method. Set $P = \emptyset$.
while ($NewSubsets \neq \emptyset$) do
    4. Select the next subset $\delta$ in $NewSubsets$.
    5. Apply path relinking to $\delta$ to obtain a new solution $\theta$.
    6. Apply tabu search to $\theta$ and add the obtained solution to $P$.
    7. Eliminate $\delta$ from $NewSubsets$.
end while
8. If one of the termination criteria is met, stop. Otherwise, go to 2.
```

Fig. 2 Our scatter search procedure

3 The scatter search algorithm

3.1 General overview

Scatter search (SS) is an evolutionary method that constructs new solutions by combining existing ones in a systematic fashion. For a general introduction to SS, the reader is referred to [32]. Figure 2 shows the structure of our algorithm. In the first step, an initial population P (containing $|P|$ solutions) is generated. In the second step, we construct the reference set *RefSet* including *RefSet*1 and *RefSet*2, the former containing b_1 solutions with low makespan, the latter containing b_2 solutions with high diversity $(b = b_1 + b_2)$. The solutions of *RefSet*1 and *RefSet*2 are called reference solutions. Next, *NewSubsets* are generated; each of them containing two reference solutions. Then, the two solutions of each subset are combined and a new solution (θ) is generated using path relinking algorithm (*NIS* procedure from [5]). Subsequently, tabu search algorithm is applied to θ and the obtained solution is added to P. Steps 2–8 are repeated until one of the termination criteria is reached.

3.2 Initial population

The first element of the initial population is generated using the "modified insertion" procedure of [28]. The makespan of the solution obtained from this procedure is lower than randomly generated solutions. Thus, addition of this element enhances the overall quality of the population. The other elements of the population are generated randomly without violation of the precedence relations that exists between the operations of each job.

3.3 Reference set building and subset generation methods

RefSet, the set of reference solutions, includes solutions based on both quality and diversity. The construction of high-quality solutions, *RefSet*1, initiates with the choice of the solution in P with the lowest makespan. The solution ($\mathbf{x}^1$) is added to *RefSet*1 and deleted from P. Subsequently, the next best solution $\mathbf{x}$ in P is selected and added to *RefSet*1 only if $D_{min}(\mathbf{x}) \geq dist^1_{min}$, where $D_{\min}(\mathbf{x})$ is the minimum of the distances of solution $\mathbf{x}$ to the solutions currently in *RefSet*1 and $dist^1_{min}$ is an algorithm parameter. The distance between two solution vectors $\mathbf{x}$ and $\mathbf{y}$ is calculated as

$$D(\mathbf{x},\mathbf{y}) = \left|\left\{(u,v) \in E^T(\mathbf{x}) : \mathbf{y}^{-1}(v) < \mathbf{y}^{-1}(u)\right\}\right|,$$

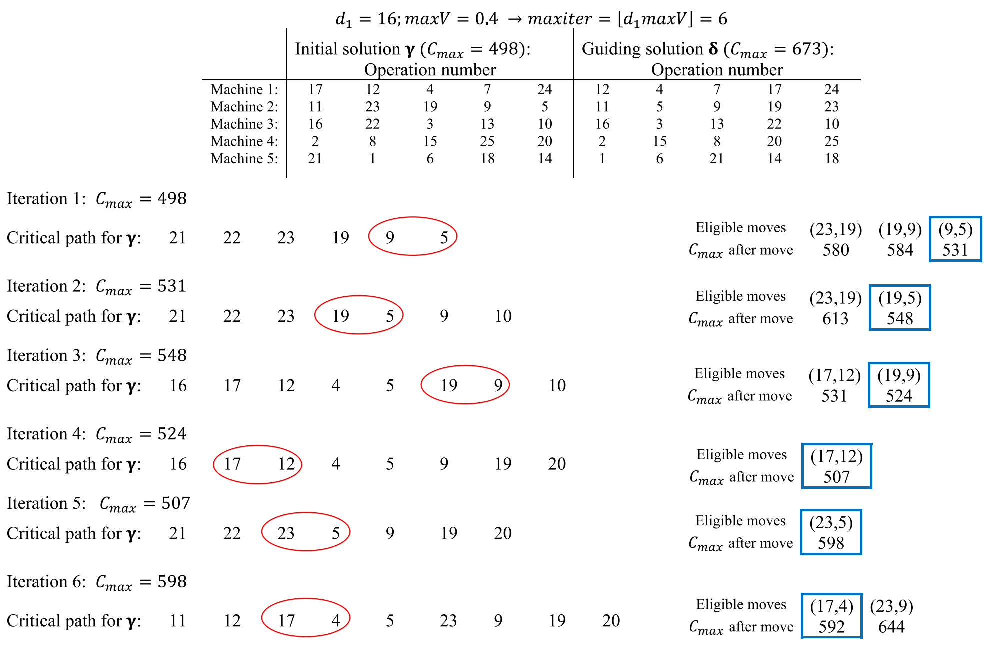
where $E^T(\mathbf{x})$ is transitive closure of $E(\mathbf{x})$ and $\mathbf{y}^{-1}(u)$ denotes the position of operation u in permutation $\mathbf{y}_{\mu_u}$; $\mathbf{y}_{\mu_u}(\mathbf{y}^{-1}(u)) = u$. In this way, *RefSet*1 contains b_1 solutions in P with best makespan, while a threshold $dist^1_{min}$ on the minimal distance between the elements in *RefSet*1 is imposed in pursuit of diversity. Besides, *RefSet*2 contains the solutions with minimum makespan from $P\backslash RefSet1$ that are sufficiently distant $(dist^2_{min} > dist^1_{min})$ from elements of both reference sets. Consequently, both *RefSet*1 and *RefSet*2 contain diversified solutions, but the diversification has more weight in *RefSet*2. When no qualified solution can be found in the population, *RefSet* is completed with randomly generated solutions. In this case, the minimum distance condition is not checked for the generated solutions.

After the *RefSet* construction, *NewSubsets* are generated which contain all pairs that are composed of two elements from *RefSet*1 and all pairs that are composed of one element from *RefSet*1 and one element from *RefSet*2. Therefore, $|NewSubsets| = \frac{b_1(b_1-1)}{2} + b_1 b_2$.

3.4 Path relinking procedure

The path relinking procedure that is used in this research is an extension of the new initial solution generator (*NIS*) procedure of [5]. *NIS* takes two processing orders initial solution γ and guiding solution δ with $D(\gamma,\delta)=d_1$ and swaps the adjacent operations of the critical path that should be performed on the same machine in γ to reduce the distance between γ and δ while trying to decrease the makespan as much as possible. Subsequently, the procedure terminates when $D(\gamma,\delta) < \lceil (1 - maxV)d_1 \rceil$ where $maxV$ is an algorithm parameter. Finally, γ is returned as the combined solution.

Figure 3 illustrates *NIS* by an example. The problem is a job shop problem with 5 jobs, 5 machines and 25 operations. An initial solution γ with $C_{\max}$=498 and a guiding solution δ with $C_{\max}$=673 are available. The sequence of the operations of the two solutions on each machine is shown. The aim is to have a new solution θ between γ and δ to start the local search. The distance between two solutions is $D(\gamma,\delta) = d_1 = 16$ and $maxV$= 0.4. Therefore, *NIS* will be terminated in $\lceil 16 \times 0.4 \rceil = 6$ iterations. Now, the sequence of every two adjacent



$d_1 = 16; maxV = 0.4 \rightarrow maxiter = \lfloor d_1 maxV \rfloor = 6$

	Initial solution $\boldsymbol{\gamma}$ ($C_{max} = 498$): Operation number					Guiding solution $\boldsymbol{\delta}$ ($C_{max} = 673$): Operation number				
Machine 1:	17	12	4	7	24	12	4	7	17	24
Machine 2:	11	23	19	9	5	11	5	9	19	23
Machine 3:	16	22	3	13	10	16	3	13	22	10
Machine 4:	2	8	15	25	20	2	15	8	20	25
Machine 5:	21	1	6	18	14	1	6	21	14	18

Iteration 1: $C_{max} = 498$
Critical path for $\boldsymbol{\gamma}$: 21 22 23 19 9 5
Eligible moves (23,19) (19,9) $\boxed{(9,5)}$
C_{max} after move 580 584 $\boxed{531}$

Iteration 2: $C_{max} = 531$
Critical path for $\boldsymbol{\gamma}$: 21 22 23 19 5 9 10
Eligible moves (23,19) $\boxed{(19,5)}$
C_{max} after move 613 $\boxed{548}$

Iteration 3: $C_{max} = 548$
Critical path for $\boldsymbol{\gamma}$: 16 17 12 4 5 19 9 10
Eligible moves (17,12) $\boxed{(19,9)}$
C_{max} after move 531 $\boxed{524}$

Iteration 4: $C_{max} = 524$
Critical path for $\boldsymbol{\gamma}$: 16 17 12 4 5 9 19 20
Eligible moves $\boxed{(17,12)}$
C_{max} after move $\boxed{507}$

Iteration 5: $C_{max} = 507$
Critical path for $\boldsymbol{\gamma}$: 21 22 23 5 9 19 20
Eligible moves $\boxed{(23,5)}$
C_{max} after move $\boxed{598}$

Iteration 6: $C_{max} = 598$
Critical path for $\boldsymbol{\gamma}$: 11 12 17 4 5 23 9 19 20
Eligible moves $\boxed{(17,4)}$ (23,9)
C_{max} after move $\boxed{592}$ 644

Fig. 3 An example for the path relinking procedure (*NIS*)

operations in the critical path of γ that should be performed on the same machine is considered. For example, in iteration 1, none of the pairs (21,22) and (22,23) is eligible for move since the operations in each pair do not belong to the same machine. The pairs that can be selected are (23,19), (19,9), and (9,5) since in machine 2 of the guiding solution δ, 23 is after 19, 19 is after 9, and 9 is after 5. Subsequently, the makespan is computed for the eligible moves and the move with the minimal makespan is chosen. Now, the move is implemented and the new critical path is obtained. This process is continued for six iterations, and finally, a solution θ with $C_{\max}$=592 is the result of *NIS*.

We extend *NIS* so that the swap between two adjacent operations in a job is also possible whenever no precedence relation (direct or indirect) exists between the two operations. In this way, two elements of each subset are combined using the path relinking procedure, and in the next step, the tabu search procedure is applied to the resulting solution.

3.5 Tabu search procedure

Many effective algorithms use a local search procedure for the intensification aims. In our algorithm, a modified version of TSAB (a tabu search algorithm) of [2] is utilized as local search procedure. In TSAB, the move is a swap between two adjacent operations that should be performed on the same machine. In the modified version, the swap of two adjacent operations that belonged to the same job is also possible whenever no precedence relation (direct or indirect) exists between these two operations. The solution obtained from modified TSAB is added to the population. When the set *NewSubsets* is emptied, the process of building the reference sets and generating subsets is started again and this process is repeated until one of the termination criteria is met.

3.6 Termination criteria

The algorithm stops when optimal solution is found or the total number of iterations is greater than an algorithm parameter *TotIter*.

4 Computational results

4.1 Problem instances

We were not aware of any data set for the partial job shop problem. Therefore, to produce the problem instances, Taillard benchmarks (TA01–TA50) are used, which are the well-known job shop benchmarks [33]. The instances TA51–TA80 are not considered since they are easy to solve. The processing times of the operations and machines that the operations should be performed on them are the same as Taillard instances, but the processing order of operations in a job is according to the partial job shop setting instead of the job shop conditions. For each series of problems (ten problems that have the same number of jobs and the same number of machines), the precedence relations set for each job is the same, i.e., the precedence relations set for job 1 is the same for ten problems and so on. Thus, to produce the set of precedence relations for job j, first, a random number (nr_j) between 0 and m is produced for the number of precedence relations. Subsequently, nr_j random precedence relations are produced one by one.

Table 1 Parameter setting for different problem sets

Problem group	Size	$dist_{min}^1$	$dist_{min}^2$	b_1	b_2
PTA01–10	15×15	10	20	3	2
PTA11–20	20×15	20	40	4	2
PTA21–30	20×20	150	200	5	2
PTA31–40	30×15	150	250	5	2
PTA41–50	30×20	300	550	5	3

After the production of each precedence relation, the precedence relations set is checked for cycle. If a cycle is detected, the relation is reversed. In this way, a set of precedence relations is assigned to each job of the problem and the partial job shop instance is generated. The instances which are generated for the partial job shop problem by this method are called PTA01–PTA50.

4.2 Parameter setting

Using fine tuning, the parameters of the algorithm are set. Prior to starting the computational experiments, some tests were performed that show the low interaction between algorithm parameters. Therefore, the values of parameters are obtained independently. For each of the parameters, about five different values are tested. These tests continue until a local optimum is found for the parameter. Concerning the briefness in the paper, we ignored the explanation of the details. Many well-known papers [2, 5, 9, 31, 34] that presented an algorithm for the scheduling problems implemented similar procedures for parameter setting. The thresholds on minimal distance ($dist_{\min}^1$ and $dist_{\min}^2$) and the cardinality of reference sets (b_1 and b_2) are according to Table 1. The initial population is set to $|P|=10b$, and the parameter of the procedure *NIS* is set to $maxV=0.5$. In addition, maximum number of iterations for instances PTA01–PTA20 is set to $TotIter=5{,}000{,}000$ and for instances PTA21–PTA50 is set to $TotIter=10{,}000{,}000$. Furthermore, the parameters of tabu search procedure are the same as the parameters of TSAB ([2]).

4.3 Results

The hybrid scatter search algorithm is executed in VC++ on a Dell XPS M1210 Laptop with 1.83-GHz CPU. In order to

Table 2 Summary of the results

Problem group	TA01–10	TA11–20	TA21–30	TA31–40	TA41–50
Mean RE	22.54	7.57	17.27	0.97	2.37
σ of RE	4.75	4.15	3.89	1.04	2.75

Table 3 Results for PTA01-50 instances

Problem	Size	LB1	Scatter search partial JSP	Relative error (RE)	Scatter search CPU time (s)
PTA01	15×15	998	1,249	25.15	59.5
PTA02	15×15	925	1,157	25.08	57.7
PTA03	15×15	927	1,087	17.26	57.8
PTA04	15×15	885	1,131	27.80	59.0
PTA05	15×15	908	1,161	27.86	58.7
PTA06	15×15	906	1,162	28.26	59.9
PTA07	15×15	938	1,100	17.27	58.6
PTA08	15×15	881	1,086	23.27	59.6
PTA09	15×15	972	1,139	17.18	60.3
PTA10	15×15	925	1,076	16.32	59.1
PTA11	20×15	1,143	1,174	2.71	81.1
PTA12	20×15	1,269	1,400	10.32	82.9
PTA13	20×15	1,181	1,267	7.28	80.0
PTA14	20×15	1,135	1,152	1.50	77.8
PTA15	20×15	1,165	1,289	10.64	81.2
PTA16	20×15	1,195	1,341	12.22	79.6
PTA17	20×15	1,261	1,344	6.58	85.5
PTA18	20×15	1,166	1,293	10.89	76.6
PTA19	20×15	1,209	1,225	1.32	79.6
PTA20	20×15	1,198	1,345	12.27	79.8
PTA21	20×20	1,186	1,452	22.43	170.0
PTA22	20×20	1,244	1,365	9.73	192.4
PTA23	20×20	1,188	1,332	12.12	179.1
PTA24	20×20	1,282	1,550	20.90	185.5
PTA25	20×20	1,260	1,519	20.56	180.9
PTA26	20×20	1,226	1,403	14.44	188.5
PTA27	20×20	1,351	1,583	17.17	193.9
PTA28	20×20	1,275	1,485	16.47	183.9
PTA29	20×20	1,271	1,517	19.35	193.9
PTA30	20×20	1,170	1,398	19.49	187.0
PTA31	30×15	1,766	1,780	0.79	239.2
PTA32	30×15	1,780	1,829	2.75	257.3
PTA33	30×15	1,733	1,733	0.00	125.3
PTA34	30×15	1,838	1,849	0.60	227.4
PTA35	30×15	1,730	1,745	0.87	224.7
PTA36	30×15	1,780	1,780	0.00	95.5
PTA37	30×15	1,774	1,774	0.00	28.9
PTA38	30×15	1,674	1,674	0.00	199.1
PTA39	30×15	1,645	1,686	2.49	256.8
PTA40	30×15	1,607	1,642	2.18	225.8
PTA41	30×20	1,832	1,942	6.00	281.5
PTA42	30×20	1,766	1,766	0.00	151.4

Table 3 (continued)

Problem	Size	LB1	Scatter search partial JSP	Relative error (RE)	Scatter search CPU time (s)
PTA43	30×20	1,695	1,712	1.00	302.0
PTA44	30×20	1,789	1,789	0.00	223.2
PTA45	30×20	1,732	1,732	0.00	241.9
PTA46	30×20	1,860	1,860	0.00	227.0
PTA47	30×20	1,693	1,768	4.43	300.3
PTA48	30×20	1,747	1,854	6.12	284.9
PTA49	30×20	1,763	1,763	0.00	185.7
PTA50	30×20	1,682	1,786	6.18	301.8

evaluate the performance of the proposed algorithm, some problem instances are generated and solved by the hybrid scatter search algorithm. The results are summarized in Table 2.

Table 3 contains six columns and presents the results in more details. "LB1" stands for the lower bound (see Section 2.2) that is computed for all of the problem instances, and the results are compared to it. The "relative error (RE)" stands for a measure of the effectiveness of the algorithm that can be calculated using the formula $\mathrm{RE} = 100 \times \left(\mathrm{UB}_{\mathrm{alg}} - \mathrm{LB1}\right)/\mathrm{LB1}$, where $\mathrm{UB}_{\mathrm{alg}}$ is the makespan of the best solution obtained by the algorithm. In this way, mean relative error of the algorithm is 10.15 and the standard deviation of the relative error is 9.16. In addition, 9 out of 50 instances have been solved to optimality.

5 Conclusions

This research considers the partial job shop problem (an extension of the mixed shop problem) with makespan criterion. A scatter search algorithm combined with path relinking and tabu search is proposed to solve the problem. In order to evaluate the performance of the proposed algorithm, Taillard benchmarks are used. For each of Taillard's problem instances, a set of precedence relations are produced to convert the job shop problem to the partial job shop problem. Then a series of computational experiments are performed and the results are compared with a lower bound.

The outcomes show that by increasing the size of the problem, the effectiveness of the proposed lower bound is improved. Using tabu search improves the intensification characteristics of the algorithm, and generating diverse initial solutions and applying the path relinking procedure improves the diversification attributes of the algorithm. As a result, the combination of these meta-heuristic algorithms is very effective in solving the partial job shop problem and may also be effective for solving other scheduling problems.

In addition, solving the partial job shop problem with other criteria and more restrictive assumptions seems to be a fruitful area for future research. Finally, the research can be focused on finding better lower bounds for the problem.

Acknowledgments The authors would like to thank the anonymous referees for their invaluable suggestions.

References

1. Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and job-shop scheduling. Math Oper Res 1:117–129
2. Nowicki E, Smutnicki C (1996) A fast taboo search algorithm for the job shop scheduling problem. Manage Sci 42(6):797–813
3. Pezzella F, Merelli E (2000) A tabu search method guided by shifting bottleneck for job shop scheduling problem. Eur J Oper Res 120:297–310
4. Ponnambalam SG, Aravindan P, Rajesh SV (2000) A tabu search algorithm for job shop scheduling. Int J Adv Manuf Technol 16:765–771
5. Nowicki E, Smutnicki C (2005) An advanced tabu search algorithm for the job shop problem. J Sched 8(2):145–159
6. Watson J, Howe AE, Whitley LD (2006) Deconstructing Nowicki and Smutnicki's i-TSAB tabu search algorithm for the job-shop scheduling problem. Comput Oper Res 33:2623–2644
7. Zhang CY, Li PG, Guan ZL, Rao YQ (2007) A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. Comput Oper Res 34:3229–3242
8. Balas E, Vazacopoulos A (1998) Guided local search with shifting bottleneck for job shop scheduling. Manage Sci 44 (2):262–275
9. Zhang CY, Li PG, Rao YQ, Guan ZL (2008) A very fast TS/SA algorithm for the job shop scheduling problem. Comput Oper Res 35:282–294
10. Huang KL, Liao CJ (2008) Ant colony optimization combined with taboo search for the job shop scheduling problem. Comput Oper Res 35:1030–1046
11. Eswaramurthy VP, Tamilarasi A (2009) Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems. Int J Adv Manuf Technol 40:1004–1015
12. Ponnambalam SG, Jawahar N, Kumar B (2002) Estimation of optimum genetic control parameters for job shop scheduling. Int J Adv Manuf Technol 19:224–234
13. Wang L, Zheng DZ (2002) A modified genetic algorithm for job shop scheduling. Int J Adv Manuf Technol 20:72–76
14. Liu TK, Tsai JT, Chou JH (2006) Improved genetic algorithm for the job-shop scheduling problem. Int J Adv Manuf Technol 27:1021–1029
15. Amirthagadeswaran KS, Arunachalam VP (2006) Improved solutions for job shop scheduling problems through genetic algorithm with a different method of schedule deduction. Int J Adv Manuf Technol 28:532–540
16. Amirthagadeswaran KS, Arunachalam VP (2007) Enhancement of performance of genetic algorithm for job shop scheduling problems through inversion operator. Int J Adv Manuf Technol 32:780–786

17. Xu X, Li C (2007) Research on immune genetic algorithm for solving the job-shop scheduling problem. Int J Adv Manuf Technol 34:783–789
18. Zhang C, Rao Y, Li P (2008) An effective hybrid genetic algorithm for the job shop scheduling problem. Int J Adv Manuf Technol 39:965–974
19. Wang YM, Yin HL, Wang J (2009) Genetic algorithm with new encoding scheme for job shop scheduling. Int J Adv Manuf Technol 44:977–984
20. Udomsakdigool A, Kachitvichyanukul V (2008) Multiple colony ant algorithm for job-shop scheduling problem. Int J Prod Res 46:4155–4175
21. Pardalos PM, Shylo OV, Vazacopoulos A (2010) Solving job shop scheduling problems utilizing the properties of backbone and "big valley". Comput Optim Appl. doi:10.1007/s10589-008-9206-5
22. Rego C, Duarte R (2009) A filter-and-fan approach to the job shop scheduling problem. Eur J Oper Res 194:650–662
23. Lin TL, Horng SJ, Kao TW et al (2010) An efficient job-shop scheduling algorithm based on particle swarm optimization. Expert Syst Appl 37:2629–2636
24. Taillard ED (2009) http://mistic.heig-vd.ch/taillard, November
25. Brucker P (2004) Scheduling algorithms, 4th edn. Springer, Berlin
26. Shakhlevich NV, Sotskov YN, Werner F (2000) Complexity of mixed shop scheduling problems: a survey. Eur J Oper Res 120:343–351
27. Ramudhin A, Marier P (1996) The generalized shifting bottleneck procedure. Eur J Oper Res 93:34–48
28. Kis T (2003) Job-shop scheduling with processing alternatives. Eur J Oper Res 151:307–332
29. Debels D, De Reyck B, Leus R, Vanhoucke M (2006) A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. Eur J Oper Res 169:638–653
30. Yamashita DS, Armentano VA, Laguna M (2006) Scatter search for project scheduling with resource availability cost. Eur J Oper Res 169:623–637
31. Ranjbar M, De Reyck B, Kianfar F (2009) A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. Eur J Oper Res 193:35–48
32. Laguna M, Marti R (2003) Scatter search—methodology and implementations in C. Kluwer, Boston
33. Taillard ED (1993) Benchmarks for basic scheduling problems. Eur J Oper Res 64:278–285
34. Grabowski J, Wodecki M (2004) A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. Comput Oper Res 31:1891–1909