Comput Optim Appl manuscript No.
(will be inserted by the editor)

A Tabu Search Approach for Proportionate Multiprocessor Open Shop Scheduling

Tamer F. Abdelmaguid · Mohamed A. Shalaby · Mohamed A. Awwad

Received: date / Accepted: date

Abstract In the multiprocessor open shop scheduling problem, jobs are to be processed on a set of processing centers -each having one or more parallel identical machines, while jobs do not have a pre-specified obligatory route. A special case is the proportionate multiprocessor open shop scheduling problem (PMOSP) in which the processing time on a given center is not job-dependent. Applications of the PMOSP are evident in health care systems, maintenance and repair shops, and quality auditing and final inspection operations in industry. In this paper, a tabu search (TS) approach is presented for solving the PMOSP with the objective of minimizing the makespan. The TS approach utilizes a neighborhood search function that is defined over a network representation of feasible solutions. A set of 100 benchmark problems from the literature is used to evaluate the performance of the developed approach. Experimentations show that the developed approach outperforms a previously developed genetic algorithm as it produces solutions with an average of less than 5% deviation from a lower bound, and 40% of its solutions are provably optimal.

Keywords scheduling · multiprocessor open shop · tabu search · makespan

T.F. Abdelmaguid
Mechanical Design and Production Dept.
Faculty of Engineering, Cairo University,
Giza 12613, Egypt
Tel.: +20106-2689333
Fax: +202-35693025
E-mail: tabdelmaguid@alumni.usc.edu

M.A. Shalaby
Mechanical Design and Production Dept.
Faculty of Engineering, Cairo University,
Giza 12613, Egypt
E-mail: mashalaby@aucegypt.edu

M.A. Awwad
Dept. of Industrial Engineering and Management Systems,
University of Central Florida,
4000 Central Florida Blvd., Orlando,
FL 32816-2993, USA.
Phone: +1 407 968 3090
E-mail: mohamed.awwad@knights.ucf.edu

1 Introduction

In the proportionate multiprocessor open shop scheduling problem (PMOSP), there are N jobs to be processed on K processing centers without having a pre-specified obligatory route. Each center has a number of parallel identical machines. Each job requires K operations, where each operation is to be processed on any of the parallel machines in one and only one processing center. Each operation requires a time $p_k > 0$ to be processed on any of the parallel machines belonging to center k. Here, the processing time is not job-dependent. It is required to find the order of processing the operations within each job, the assignment of operations to machines, and the sequence by which operations will be processed on their assigned machines such that a pre-specified objective is optimized.

As addressed in [1], the PMOSP appears in health care applications, specifically in the scheduling of diagnostic tests performed on patients in hospitals' outpatient testing centers. Other applications in service and industry include auto-repair and washing, maintenance workshops, final inspection operations and quality audits. Despite common occurrences of the PMOSP in practice, it did not receive much attention in the literature. Matta [1,2] is the first to introduce the PMOSP to the literature with the objective of minimizing the makespan. She proposed two mixed integer programming models to demonstrate the complexity of the problem and to show the difficulty of solving it via commercial MIP solvers. Therefore, a genetic algorithm (GA) approach was proposed to provide efficient solutions in a reasonable computational time. Later, Matta and Elmaghraby [3] proved that polynomial time algorithms for two special classes of the PMOSP exist; yet, the general PMOSP is hard to analyze, which is affirmed in [4].

A special case of the PMOSP is the open shop scheduling problem (OSSP) in which each processing center has exactly one machine. Most of the published work considers the OSSP with the objective of minimizing the makespan. Less interest to construct mathematical programming models for the OSSP is apparent in the literature when compared to other traditional scheduling problems. Naderi *et al.* [5] proposed four different MILP models and compared between them on the basis of their size complexities represented by the number of binary and continuous variables and the number of constraints. Regarding the complexity of the OSSP, it has been shown that for three centers or more, the OSSP is NP-hard [6]. Shmoys *et al.* [7] proved that the worst case performance ratio for any dense open shop schedule is 2. An open shop schedule is said to be dense if no machine is idle unless there is no operation ready to be processed on that machine. For the 3-machine case, Chen and Strusevich [8] proposed a linear time algorithm that transforms a dense schedule into a new schedule such that the makespan of the best of these two is at most 3/2 times worse than the optimal value.

Several metaheuristic approaches have been proposed for the OSSP. Fang *et al.* [9] introduced a so-called hybrid genetic algorithm/heuristic appraoch in which the decoding of a chromosome is incorporated with a heuristic technqiue that utilizes dispatching rules to select the next schedulable operation. A TS approach was firstly introduced by Alcaide *et al.* [10]. They used two neighborhood search functions for improving a current solution. Both functions work by defining a set of critical operations for which the start times cannot be altered without affecting the current makespan. Then either the order of two or three consecutive critical operations are reversed or the position of a critical operation is changed.

Liaw [11] introduced a different TS approach for the OSSP. He used four different neighborhood moves for improving a current solution based on reversing arcs on a critical path of a network representation of a solution. The used tabu structure keeps track of the inverses of recentely conducted arc reversals. The TS approach showed competitive results on benchmark and randomly generated instances. Liaw [12] utilized the same neighborhood moves within a simulated annealing approach. Later, Liaw [13] incorporated his previously developed TS as a local search procedure into a GA approach. Competitive results are obtained, and the hybrid approach outperformed other heuristic approaches in the literature. Prins [14] developed a competitive GA approach for the OSSP which was capable of solving hard cases known in the literature. Recently, hybrid beam search-ant colony optimization techniques are presented [15, 16]. They showed competitive results on standard benchmark problems.

For the general case of multiprocessor open shop scheduling (MPOS) in which the processing times of jobs in a given center may vary from one job to the other, almost all of the published work studied it with the objective of minimizing the makespan [1]. A part of the literature focused on developing polynomial-time approximation algorithms that generate schedules whose makespan is provably not too far from the optimum. Here, a ρ-approximation algorithm is defined as an algorithm that is guaranteed to deliver a solution with makespan at most ρ times the optimal objective value, for some $\rho > 1$. Schuurman and Woeginger [17] presented a close to 3/2-approximation algorithm for the two-center non-preemptive MPOS problem. They also improved upon the work in [8] and presented a 2-approximation algorithm for the MPOS when the number of centers is part of the input. Sevastianov and Woeginger [18] extended their previous work on the OSSP [19] to the MPOS problem with a fixed number of centers and machines and presented a polynomial time approximation scheme with a running time linear in the number of jobs. Recently, Naderi *et al.* [20] studied the MPOS problem with the objective of minimizing the jobs' total completion time. They proposed a mixed integer linear programming model and three different metaheuristics, namely memetic algorithm, simulated annealing, and a hybrid simulated annealing incorporated with memetic algorithm as its local search engine. All three algorithms were implemented by two different encoding schemes.

Tabu Search, an iterative improvement algorithm originally developed by Glover [21, 22], has demonstrated competitive performance on various scheduling problems including the OSSP; however it has not been applied to the PMOSP. This paper investigates the performance of a TS approach on the PMOSP, and compares it with the performance of a previously developed GA for the objective of minimizing the makespan. The rest of this paper is organized as follows. In Sect. 2, a network representation and a neighborhood search function for the studied problem are described. The developed tabu search approach is presented in Sect. 3, followed by experimentations and results in Sect. 4. Finally the conclusion and future research directions are provided in Sect. 5.

2 Network representation and neighborhood search

The developed TS approach utilizes a neighborhood search function that is defined over a network representation of the solutions of the PMOSP. Network representations are common for representing solutions of many scheduling problems. The disjunctive graph representation [23] was first introduced for the job shop scheduling problem,

and later adopted for the OSSP [11]. We follow the convention of generating networks based on given permutations of operations as illustrated in [24].

Before describing the network model and the neighborhood search function, we first introduce some notations. An operation is denoted by o_{jk}, where $j = 1, \ldots, N$ is the job index and $k = 1, \ldots, K$ is the processing center index. For each processing center k, there are L_k parallel identical machines, and a machine is denoted as $m_{k,l}$ where $l = 1, \ldots, L_k$.

In the PMOSP, solutions can be defined using the pair (Π^J, Π^M), where $\Pi^J = (\pi_j^J : j = 1, ..., N)$ is the vector of operations' orders within jobs, and $\Pi^M = (\pi_{m_{k,l}}^M : l = 1, \ldots, L_k$ and $k = 1, \ldots, K)$ is the vector of processing sequences on machines. Here, Π^M represents both decisions of machine assignment to operations and the processing sequence of operations on their assigned machines.

2.1 Network representation

For a given solution (Π^J, Π^M), a directed network $G(V, A(\Pi^J, \Pi^M))$ is constructed as follows. The set of nodes V is defined so that there is a node for each operation o_{jk}, in addition to two dummy source S and terminal T nodes, i.e. $V = \{o_{jk} : j = 1, \ldots, N$ and $k = 1, \ldots, K\} \cup \{S, T\}$. A Node represents the event of starting the processing of the corresponding operation. Each node is assigned a weight that equals the corresponding operation's processing time, and the dummy nodes have zero weights. Let $\pi_j^J(i)$ denote the operation located in position i of job j's processing order vector π_j^J. Based on Π^J, the set of directed arcs $Z(\Pi^J) = \{(S, \pi_j^J(1)) : j = 1, \ldots, N\} \cup \{(\pi_j^J(i), \pi_j^J(i+1)) : i = 1, \ldots, K-1 \; \forall \; j = 1, \ldots, N\} \cup \{(\pi_j^J(K), T) : j = 1, \ldots, N\}$ is defined.

Let $\pi_{m_{k,l}}^M(i)$ denote the operation located in position i of machine $m_{k,l}$'s processing sequence vector, and let $\lambda_{k,l}$ denote the number of operations in $\pi_{m_{k,l}}^M$, i.e. the number of operations assigned to machine $m_{k,l}$. Based on Π^M, the set of directed arcs $P(\Pi^M)$ $= \{(\pi_{m_{k,l}}^M(i),\ \pi_{m_{k,l}}^M(i+1)) : i = 1, \ldots, \lambda_{k,l} - 1$ where $\lambda_{k,l} > 1 \; \forall \; l = 1, \ldots, L_k$ and $k = 1, \ldots, K\}$ is defined. Therefore, the network's set of directed arcs is defined as $A(\Pi^J, \Pi^M) = Z(\Pi^J) \cup P(\Pi^M)$.

To illustrate the network representation described above, consider an example in which there are three processing centers and four jobs ready to be processed. The number of parallel identical machines in each processing center are $L_1 = 1$, $L_2 = 2$ and $L_3 = 2$. The processing times on the three centers are $p_1 = 2$, $p_2 = 6$ and $p_3 = 4$ time units. Consider a solution (Π^J, Π^M) in which jobs' processing orders are $\pi_1^J = (o_{11},\ o_{13},\ o_{12})$, $\pi_2^J = (o_{21},\ o_{23},\ o_{22})$, $\pi_3^J = (o_{33},\ o_{31},\ o_{32})$ and $\pi_4^J = (o_{43},\ o_{41},\ o_{42})$. Machines' processing sequences are given as $\pi_{m_{1,1}}^M = (o_{31},\ o_{21},\ o_{11},\ o_{41})$, $\pi_{m_{2,1}}^M = (o_{32},\ o_{22},\ o_{12})$, $\pi_{m_{2,2}}^M = (o_{42})$, $\pi_{m_{3,1}}^M = (o_{33},\ o_{23},\ o_{13})$ and $\pi_{m_{3,2}}^M = (o_{43})$. The network representation of this solution is presented in Fig. 1, where solid arcs correspond to $Z(\Pi^J)$ and dashed arcs correspond to $P(\Pi^M)$.

Based on the given processing orders and sequences, the Gantt chart, when the operations are scheduled based on their earliest start times, is provided in Fig. 2. The makespan of this schedule is $C_{max} = 24$, which equals the critical path length on the network representation in Fig. 1. The critical path can be determined using the same approach applied to project networks. The length of the critical path is the summation of the nodes' weights along that path. Multiple critical paths may exist. As shown in Fig. 1, the paths $(o_{33},\ o_{31},\ o_{32},\ o_{22},\ o_{12})$ and $(o_{33},\ o_{31},\ o_{21},\ o_{23},\ o_{22},\ o_{12})$ are critical.

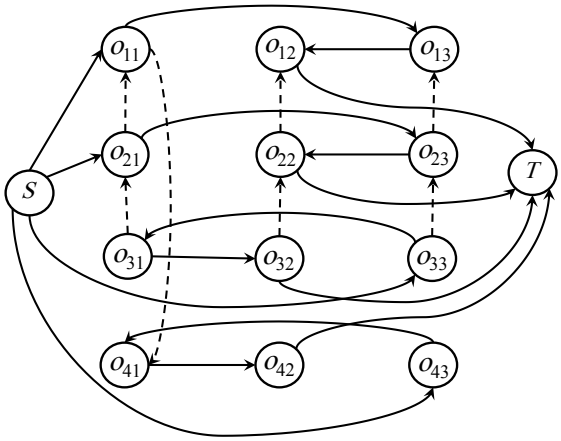

Fig. 1 Network representaion for a sample solution

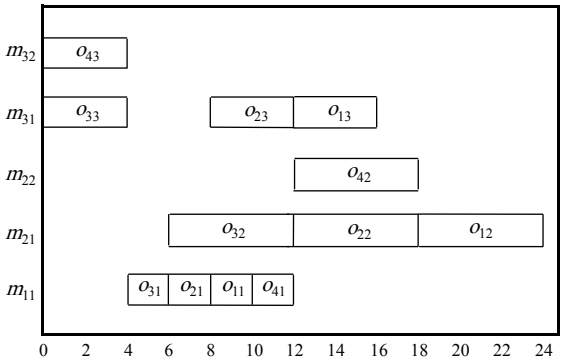

Fig. 2 Gantt chart for a sample solution

A given solution (Π^J, Π^M) may be infeasible due to logical contradictions that may exist between the operations' processing orders within jobs and the machines' processing sequences. Such an infeasibility appears in the network representation in the form of cycles. This characteristic is known for the network representations used for the job shop scheduling problem, and it applies here as it is emphasized in [11] for the open shop scheduling problem. In the above example, suppose that both π_3^J and $\pi_{m_{2,1}}^M$ are changed such that $\pi_3^J = (o_{31}, o_{32}, o_{33})$ and $\pi_{m_{2,1}}^M = (o_{22}, o_{32}, o_{12})$; while the other orders and sequences remain unchanged. This will form the cycle o_{32} - o_{33} - o_{23} - o_{22} - o_{32}, with which no feasible schedule can be constructed. The reason for that is the logical contradiction between sequencing o_{32} prior to o_{22} as a result of the route o_{32} - o_{33} - o_{23} - o_{22} and sequencing o_{22} prior to o_{32} as a result of the processing sequence on machine $m_{2,1}$. Therefore, solution approaches and neighborhood search functions should avoid the formation of cycles in the solutions' network representations.

2.2 Neighborhood search function

The neighborhood search function utilized here for the PMOSP is based on the one developed by Grabowski *et al.* [25] and extended to the open shop scheduling problem by Liaw [11]. First a critical path θ in the network representation of a given solution is identified. Then, this critical path is divided into blocks of operations. A block is a maximal subsequence of operations that either belong to the same job, denoted $\theta = (\beta_1^J, \beta_2^J, \ldots, \beta_x^J)$, or processed by the same machine, denoted $\theta = (\beta_1^M, \beta_2^M, \ldots, \beta_y^M)$. For instance, in the example presented in the previous subsection, the critical path $\theta = (o_{33}, o_{31}, o_{32}, o_{22}, o_{12})$ can be divided into three blocks $\beta_1^J = (o_{33}, o_{31}, o_{32})$, $\beta_2^J = (o_{22})$ and $\beta_3^J = (o_{12})$ based on the operations' orders within jobs. Alternatively, it can be divided into another three blocks $\beta_1^M = (o_{33})$, $\beta_2^M = (o_{31})$ and $\beta_3^M = (o_{32}, o_{22}, o_{12})$ based on machines' processing sequences.

After identifying the blocks in the critical path, a set of candidate moves is defined. A move is defined as reversing the order of two consecutive operations, or alternatively an arc (u, v) from a block belonging to the critical path. Here, the moves defined in [26] for the job shop scheduling problem are adopted. They are:

1. In the first block β_1^J (β_1^M) of the selected critical path, reverse the arc (u, v) where u and v are the last two operations belonging to this block.
2. In the last block β_x^J (β_y^M) of the selected critical path, reverse the arc (u, v) where u and v are the first two operations belonging to this block.
3. For blocks β_2^J (β_2^M) to β_{x-1}^J (β_{y-1}^M) of the selected critical path, reverse the first two and last two operations belonging to each block.

It has been proven in [27] that such moves will not result in cycles for the job shop scheduling problem, and therefore feasible schedules will be generated. This property can be easily extended to the PMOSP since moves defined for machine blocks and job blocks do not interfere and are not conducted simultaneously.

3 Tabu search approach

Tabu search (TS) is an iterative improvement algorithm based both on the neighborhood search methods along with the use of different types of memories and strategies to guide this search. The basic form of TS is founded on ideas proposed by Glover [28]. Starting with an initial solution, neighborhood moves are examined at each iteration and the best candidate move is selected and applied to generate a new solution. This is repeatedly applied until a predetermined stopping condition occurs.

In order to prevent TS from cycling (i.e. repeating the same neighborhood moves continuously), a short term memory called the tabu list is designed to store a number of previous moves. The local search algorithm will seek a best solution in its neighborhood provided that it is not found in the tabu list. A move is not allowed for a certain number of iterations provided that it is in the tabu list. If the tabu list is full, then the move that has been in the list the longest is removed. It may happen that in certain iteration all possible moves are forbidden or tabu. In this case, the algorithm has to be follow a predetermined strategy that either selects the oldest move stored in the tabu list or stops the algorithm.

3.1 Initial solution

As indicated earlier, there are three types of decisions to be taken in the PMOSP. The first is related to ordering operations within jobs. The second is related to assigning machines to operations. And the third is related to determining the processing sequences of operations on their assigned machines. The neighborhood search function discussed in the previous section does not have moves that can alter the machine assignments to operations. This means that the assignments made at the initial solution will remain unchanged until the end of the TS run. Therefore, the construction heuristic used for generating initial solutions is designed to conduct the machine assignment decisions randomly. This entails several TS runs for a given problem instance.

The construction heuristic utilizes a dense schedule (DS) dispatching rule for selecting the next operation to be scheduled. A schedule is said to be dense if no machine is idle unless there is no operation ready to be processed on that machine. It has been shown in [17] that the worst-case performance ratio for any dense multiprocessor open shop schedule is 2, i.e. the makespan of a dense schedule of multiprocessor open shop schedule is at most twice the optimum makespan.

The following algorithm, named DS/RANDOM after the DS/RANDOM rule used in [11] for the OSSP, will be used here to generate a dense schedule as an initial solution to the problem under study. Starting at time t = 0, repeat the following steps until all operations are scheduled. Every time the set of machines μ_k in some center k are idle, check whether there exists an operation o_{ji} such that (1) o_{ji} has not been processed yet, and (2) no other operation of the corresponding job j is currently being processed on some other machine. If one or more such operations exist, start processing one of them (selected randomly) on a randomly selected machine $m_{k,l} \in \mu_k$.

3.2 Tabu list

A tabu list of fixed size ts^{max} is maintained to hold all forbidden moves. Every time a move $\nu = (u, v)$ is selected and performed, a move $\overline{\nu} = (v, u)$ will be added to the tabu list.

3.3 Search strategy

The search starts with an initial solution generated by the DS/RANDOM heuristic followed by improvement iterations. At each iteration, a single critical path is arbitrarily selected from the current solution; and operations belonging to that path are decomposed into blocks of operations. Blocks belonging to the selected critical path are classified into three types of blocks according to their position in the path, first, last and middle blocks. Using those blocks, the neighborhood of the current solution is generated as presented in Sect. 2.2.

The neighbor with the minimum makespan, after considering the tabu status of the move producing it, is selected. We call this move the selected move. As described in Sect. 3.2 the inverse of a selected move is stored in the tabu list in order to prevent repeating the current solution in the forthcoming few iterations. An aspiration criterion is defined to deal with the case in which an interesting move is tabu. A move is said

to be interesting if it leads to a new best solution. Interesting moves are allowed to be conducted even if they are in the tabu list.

At each iteration, the current solution is compared to the value of the lower bound (LB) and the algorithm terminates if the current solution's makespan is equal to the lower bound, otherwise it completes its search until another stopping criterion is met. The lower bound used in the current study is previously presented in [1]. It is defined as the maximum load among all processing centers, evaluated as LB = $\max_{k=1,..K} \left\{ \left\lceil \frac{N}{L_k} \right\rceil \times p_k \right\}$. Another stopping criterion is the limit on the number of neighborhood moves that can be conducted without resulting in any improvement to the current solution. Other stopping rules are related to a prespecified limits on the total number of iterations and the computational time. When the search stops due to one of the stopping criteria, the schedule with the minimum makespan found throughout iterations is reported. A pseudo code for the proposed TS algorithm is outlined in the following list.

Procedure TS(receives $S_0 = (\Pi_0^J, \Pi_0^M)$, LB, $itr^{\max}$, $rt^{\max}$, and $itr_{imp}^{\max}$, ts^{max}) : returns S_{best}

1. Start with an empty tabu list, denoted $\mathcal{T}$, and let $S = S_{best} = S_0$, C_{max} = the makespan of S and $itr = itr_{imp} = 0$
2. If C_{max} = LB then return S_{best}
3. Based on the current solution S, arbitrarily determine a critical path θ from its network representation.
4. Divide θ into blocks such that $\theta = (\beta_1^J, \beta_2^J, \ldots, \beta_x^J)$, and $\theta = (\beta_1^M, \beta_2^M, \ldots, \beta_y^M)$
5. For every block β^J and β^M determine all moves based on the rules provided in Sect. 2.2. Let $\mathcal{M}$ denote the set of such moves
6. For every move $\nu \in \mathcal{M}$, construct the resultant schedule S^ν and evaluate its makespan C_{max}^ν
7. If a move $\nu \in \mathcal{M}$ is found to be in $\mathcal{T}$ then remove it from $\mathcal{M}$ unless $C_{max}^\nu < C_{max}$
8. If $\mathcal{M} = \emptyset$ then return S_{best}
9. Let $\nu^* = \arg\min_{\nu \in \mathcal{M}} \{C_{\max}^\nu\}$ and $S = S^{\nu^*}$
10. If the size of $\mathcal{T} = ts^{max}$ then remove the oldest move from $\mathcal{T}$
11. Add the inverse move $\overline{\nu^*}$ to $\mathcal{T}$
12. If $C_{max}^{\nu^*} < C_{max}$ then let $C_{max} = C_{max}^{\nu^*}$, $S_{best} = S^{\nu^*}$ and $itr_{imp} = 0$; otherwise let $itr_{imp} = itr_{imp} + 1$
13. If $itr = itr^{\max}$ or $itr_{imp} = itr_{imp}^{\max}$ or *running time* $\geq rt^{\max}$ then return S_{best}; otherwise let $itr = itr + 1$ and go to step 2

The TS procedure receives six parameters as input. These parameters are the initial solution generated by the DS/RANDOM algorithm (S_0), the evaluated lower bound (LB), the tabu list size (ts^{max}), the maximum number of iterations permitted without improving the current solution ($itr_{imp}^{\max}$), the total number of iterations allowed ($itr^{\max}$), and the computational time limit allowed for one run of the algorithm ($rt^{\max}$). The algorithm starts at step 1 by defining an empty tabu list ($\mathcal{T}$), and initializing the values of the current schedule (S), the best schedule (S_{best}), the current schedule's makespan (C_{max}), the iteration counter (itr) and the number of iterations conducted without improving the current solution (itr_{imp}).

Steps from 2 to 13 represent the main iteration loop of the algorithm. In step 2, the optimality condition of having a current makespan equal to the lower bound is checked.

If this condition is satisfied, the algorithm terminates and returns the current solution as an optimal one. Otherwise, a critical path θ is arbitrarily selected from the network representation of the current solution in step 3, followed by dividing it into job and machine blocks in step 4. Based on those blocks, the set of neighborhood moves $\mathcal{M}$ is generated in step 5 using the guidelines defined in Sect. 2.2. Then for every move in $\mathcal{M}$, the resultant schedule when a move is conducted is generated and its makespan is evaluated in step 6.

In step 7, the moves defined in set $\mathcal{M}$ are tested whether they are included in the tabu list, in which case they are removed from set $\mathcal{M}$. An exception of that is made whenever a move satisfies the aspiration criteria, which occurs when a move results in a makespan that is less than the best found makespan. After filtering the moves in step 7, step 8 checks if set $\mathcal{M}$ is found to be empty. This represents a situation in which there is apparently no move that can improve the current solution, in which case the algorithm terminates and returns the best found schedule.

In step 9, the move that results in the best makespan is selected, and its resultant schedule is assigned to the current schedule. The tabu list is then updated in steps 10 and 11 by adding the inverse of the selected move to the tabu list and removing the oldest move from it whenever the tabu list size reaches is maximum limit. The number of iterations conducted without improving the current solution (itr_{imp}), and the current best makespan value (C_{max}) are updated in step 12. Finally step 13 checks the stopping criteria that are based on the limits on the number of iterations and the computational time limit. If one of those stopping criteria is satisfied, the current best found solution is returned; otherwise, the algorithm returns back to step 2.

4 Experimentation and results

In the literature, there is only one set of benchmark problems for the PMOSP designed by Matta [1]. This set is used in the current work to assess the performance of the developed TS approach by comparing it with the GA proposed in [1,2]. The benchmark problems in [1] are designed for a balanced PMOSP case in which all centers complete their total workload at a specified due date or as close to it as possible. Each problem is defined by four main parameters: number of centers, number of machines per center, processing times, and the number of jobs.

The TS algorithm parameters are determined after conducting several pilot experiments starting with values that are commonly used in the literature. TS algorithm's parameters used in the implementation here are $itr^{\max} = 50000$, $rt^{\max} = 0.1 \times N \times K$ seconds, $itr_{imp}^{\max} = 1000$ and $ts^{max} = 15$. All experiments are conducted on a laptop supported by an Intel Core 2 Duo processor of 1.83 GHz CPU speed and 2 MB cache size. The algorithm is programmed in Visual C# 2010. Ten runs are conducted for each problem of the 100 problems as was done for Matta's GA.

Tables 1 to 4 present the results of the TS algorithm with those obtained from Matta's GA. The tables first show two characteristics of each problem instance: the number of jobs (N) and the lower bound (LB). Then for each algorithm, the tables show the average makespan for the 10 runs for each problem (Avg. C_{max}), best or minimum makespan obtained among the 10 runs (Best C_{max}), percentage deviation of the average makespan obtained for the 10 runs from the lower bound (Mean dev.%), and the average computational time (Avg. time) of the 10 runs in seconds. Bold face is used in tables 1 through 4 for the makespan and mean deviation results to show

Problem	N	LB	Matta's GA				Proposed TS			
			Avg. C_{max}	Best C_{max}	Mean dev. %	Avg. time	Avg. C_{max}	Best C_{max}	Mean dev. %	Avg. time
S2-P1	22	30	30	30*	0	3.187	30	30*	0	0.02
S2-P2	32	10	11	11	10	2.3	11	11	10	0.05
S2-P3	24	18	18	18*	0	2.267	18	18*	0	0
S2-P4	20	16	18	18	12.5	1.583	18	18	12.5	0.02
S2-P5	29	27	31	31	14.8	4.772	31	31	14.8	0.04
S2-P6	30	12	12	12*	0	2.373	12	12*	0	0
S2-P7	28	27	30	30	11.1	4.192	30	30	11.1	0.02
S2-P8	10	16	16	16*	0	0.534	16	16*	0	0
S2-P9	30	22	22	22*	0	4.372	22	22*	0	0
S2-P10	16	28	28	28*	0	1.606	28	28*	0	0.02
S2-P11	40	14	14	14*	0	4.797	14	14*	0	0
S2-P12	34	36	39	39	8.3	7.439	39	39	8.3	0.04
S2-P13	12	26	26	26*	0	0.966	26	26*	0	0
S2-P14	14	24	24	24*	0	1.136	24	24*	0	0
S2-P15	32	30	33	33	10	6.528	33	33	10	0.05
S2-P16	34	30	33	33	10	6.408	33	33	10	0.04
S2-P17	15	18	18	18*	0	0.981	18	18*	0	0
S2-P18	13	22	22	22*	0	0.939	22	22*	0	0.02
S2-P19	16	26	26	26*	0	1.539	26	26*	0	0.02
S2-P20	12	12	12	12*	0	0.547	12	12*	0	0
S2-P21	22	28	32	32	14.3	3.08	32	32	14.3	0.05
S2-P22	25	22	22	22*	0	2.916	22	22*	0	0.03
S2-P23	22	18	21	21	16.7	2.078	21	21	16.7	0.04
S2-P24	12	18	18	18*	0	0.73	18	18*	0	0
S2-P25	21	10	10	10*	0	1.109	10	10*	0	0.02

* Provably optimal solution

Table 1 Makespan (C_{max}) and computational time (in seconds) results for Matta's 2-center problems

the best values when different values are obtained by the two algorithms. A solution is provably optimal whenever its makespan equals the lower bound. Table 5 summarizes the performance of Matta's GA and the proposed TS algorithm in solving the 25 problems of each problem set.

For the 2-center instances provided in table 1, the computational time of the TS algorithm is either zero or close to zero. Zero computational time means that the time taken by the algorithm before it stops is less than 0.01 seconds. Here, it indicates that the TS in all or most of the 10 runs stopped at step 2 when the initial solution's makespan is found to be equal to the lower bound. A close to zero time indicates that in most of the 10 runs, few iterations are conducted and either TS stopped at step 8 when the set of moves $\mathcal{M}$ is found to be empty, or it stopped at step 13 due to reaching the limit on the number of iterations in which no improvement is achieved. In all 2-center instances the computational time of the TS algorithm is significantly smaller than GA, while both algorithms produced solutions with the same quality for all 25 instances as illustrated in table 5.

For the 4-center instances shown in table 2, the two algorithms seem to be comparable in terms of the quality of the obtained solutions with slightly better performance by GA. Among the 25 problem instances, GA is capable of producing better average and best makespan results in 12 cases. On the other hand, TS produced better average makespan results for eight cases and better best makespan results for four cases. In

Problem	N	LB	Matta's GA				Proposed TS			
			Avg. C_{max}	Best C_{max}	Mean dev. %	Avg. time	Avg. C_{max}	Best C_{max}	Mean dev. %	Avg. time
S4-P1	49	26	**31.9**	**31**	**22.7**	14.187	33	33	26.9	19.59
S4-P2	39	21	21	21*	0	5.853	21	21*	0	0
S4-P3	63	48	56.1	54	16.9	35.459	**54**	**51**	**12.5**	3.79
S4-P4	38	27	**28.8**	**27***	**6.7**	7.87	30	30	11.1	7.15
S4-P5	56	32	33.6	32*	5	16.606	**32**	32*	**0**	0
S4-P6	60	25	**27.9**	**26**	**11.6**	16.62	28.1	28	12.4	12.39
S4-P7	53	36	**36**	**36***	**0**	16.931	40	40	11.1	10.33
S4-P8	40	33	**33**	**33***	**0**	9.323	34	34	3	14.46
S4-P9	65	39	**47.8**	**46**	**22.6**	33.489	48	47	23.1	26.0†
S4-P10	53	56	57.6	56*	2.9	24.603	**56**	56*	**0**	0
S4-P11	55	40	43.7	43	9.3	22.195	**40.6**	**40***	**1.5**	0
S4-P12	58	30	**33**	**32**	**10**	19.236	37.2	36	24	1.1
S4-P13	37	40	40	40*	0	9.614	40	40*	0	0
S4-P14	42	45	**46.7**	**45***	**3.8**	14.464	48	48	6.7	16.8†
S4-P15	28	36	36	36*	0	5.347	36	36*	0	0
S4-P16	28	32	35.4	34	10.6	18.44	**34**	34	**6.3**	0.14
S4-P17	90	32	38	38	18.8	45.383	**37.6**	**36**	**17.5**	36.0†
S4-P18	30	24	**24**	**24***	**0**	4.777	28	28	16.7	3.4
S4-P19	63	36	**36.1**	**36***	**0.3**	24.237	37	37	2.8	7.64
S4-P20	62	60	68.9	63	14.8	39.212	**68.5**	63	**14.2**	3.27
S4-P21	64	32	37	34	15.6	23.458	37	34	15.6	7.32
S4-P22	58	35	37.9	37	8.3	20.778	**36.7**	**35***	**4.9**	0.3
S4-P23	61	21	24.6	24	17.1	15.574	24.6	24	17.1	5.14
S4-P24	54	44	**45.1**	**44***	**2.5**	21.536	46.6	46	5.9	17.72
S4-P25	34	21	**21.9**	**21***	**4.3**	5.034	23.5	22	11.9	8.79

* Provably optimal solution
† All ten runs reached the running time limit $rt^{\max}$

Table 2 Makespan (C_{max}) and computational time (in seconds) results for Matta's 4-center problems

nine cases, both algorithms produced the same best makespan values. As shown in table 5, the average performance among the 4-center 25 instances for GA is 1.66% better than TS in the case of average makespan, and 2.67% better in the case of best makespan. The computational time of TS is higher than GA in only four cases. In three cases, the TS algorithm reached the time limit $rt^{\max}$ in all 10 runs. Zero computational times are associated with cases in which the TS algorithm is capable of achieving provably optimal solutions, which is an indication that the makespan of the initial solution equals the lower bound in all or most of the 10 runs, or few iterations are conducted before finding an early provably optimal solution. This is reported in six cases. As shown in table 5, the average computational time of the TS algorithm for all 25 4-center instances is less than that of Matta's GA by more than 10 seconds.

For both the 8 and 16-center instances shown in tables 3 and 4 respectively, the TS algorithm is producing better average and best makespan values for all instances. The TS algorithm reached the time limit $rt^{\max}$ in 13 cases of the 8-center instances and 17 cases of the 16-center instances. This indicates that the proposed approach is capable of producing provably optimal solutions at an early stage even for large problem instances. Table 5 clearly shows that the proposed TS approach completely outperforms GA in terms of the quality of solutions and computational time for the 8 and 16-center instances.

Problem	N	LB	Matta's GA				Proposed TS			
			Avg. C_{max}	Best C_{max}	Mean dev. %	Avg. time	Avg. C_{max}	Best C_{max}	Mean dev. %	Avg. time
S8-P1	146	48	60.4	57	25.8	197.87	**55.9**	**53**	**16.5**	116.8†
S8-P2	144	32	39.6	39	23.8	136.947	**35**	**35**	**9.4**	115.2†
S8-P3	87	32	37.5	37	17.2	52.297	**34.9**	**34**	**9.1**	69.6†
S8-P4	161	108	129.9	126	20.3	462.42	**120.3**	**120**	**11.4**	128.8†
S8-P5	117	78	87.2	85	11.8	182.594	**78**	**78***	**0**	0
S8-P6	99	63	71.9	70	14.1	111.576	**64.3**	**64**	**2.1**	79.2†
S8-P7	84	36	40.2	39	11.7	52.837	**36.3**	**36***	**0.8**	26.88
S8-P8	110	55	68.1	66	23.8	131.494	**64.3**	**59**	**16.9**	88†
S8-P9	128	42	49.1	48	16.9	131.339	**43.6**	**43**	**3.8**	61.44
S8-P10	90	32	38.8	38	21.3	56.642	**37.3**	**36**	**16.6**	72†
S8-P11	102	45	47.8	47	6.2	83.839	**45**	**45***	**0**	0
S8-P12	92	60	68.8	67	14.7	93.427	**62.8**	**61**	**4.7**	73.6†
S8-P13	101	35	41.6	41	18.9	73.627	**38.1**	**36**	**8.9**	80.8†
S8-P14	72	84	93.9	92	11.8	78.312	**85.1**	**84***	**1.3**	28.8
S8-P15	100	60	69.3	68	15.5	113.008	**61.9**	**61**	**3.2**	32
S8-P16	81	70	84.4	83	20.6	86.977	**76.2**	**76**	**8.9**	64.8†
S8-P17	100	60	65.5	64	9.2	107.352	**60.2**	**60***	**0.3**	16
S8-P18	106	56	62.6	62	11.8	112.9	**58**	**56***	**3.6**	25.44
S8-P19	108	75	84.3	81	12.4	150.233	**78**	**78**	**4**	86.4†
S8-P20	105	49	55.5	53	13.3	99.375	**49.9**	**49***	**1.8**	25.2
S8-P21	152	42	47.3	46	12.6	172.205	**42.5**	**42***	**1.2**	36.48
S8-P22	104	30	31	30*	3.3	62.608	**30**	**30***	**0**	0
S8-P23	97	75	77.4	75*	3.2	113.772	**75**	**75***	**0**	0
S8-P24	104	35	42.2	41	20.6	78.831	**38**	**36**	**8.6**	83.2†
S8-P25	101	35	41.6	40	18.9	73.744	**37.2**	**36**	**6.3**	80.8†

* Provably optimal solution
† All ten runs reached the running time limit $rt^{\max}$

Table 3 Makespan (C_{max}) and computational time (in seconds) results for Matta's 8-center problems

Table 6 presents a summarized comparison between both algorithms for all 100 instances. Among all 100 instances, the TS algorithm results in a best makespan that is not worse than Matta's best makespan in 88 cases. The TS algorithm, as well, outperformed Matts'a GA in terms of the average of the percentage mean deviation of the makespan from the lower bound for all 100 problems, with 5.96% in the case of the TS algorithm and 11.36% for Matta's GA. On the other hand, the TS algorithm results in 4.36% average percentage deviation of the best makespan from the lower bound for all 100 problems. This figure is more than twice as better as Matta's GA.

From the above results, it is evident that the performance of TS is comparable to GA in terms of the quality of solutions for small size instances of 2 and 4 centers, while it outperforms GA in terms of the computational time. For large instances of 8 and 16 centers, TS outperforms GA in both the quality of solutions and the computational time. For small size instances, specifically the 4-center cases, the proposed TS approach seems to provide less quality of solutions compared to GA. By carefully examining the computational time results of TS, it is evident that most of the runs except only three cases didn't reach the maximum running time limit in most of the ten runs conducted for each problem instance. This indicates that the TS algorithm stopped at an early stage due to either reaching a makespan that equals the lower bound (such as in S4-P2, S4-P5, S4-P10, S4-P11, S4-P15 and S4-P22), or a number of iterations are conducted

Problem	N	LB	Matta's GA				Proposed TS			
			Avg. C_{max}	Best C_{max}	Mean dev. %	Avg. time	Avg. C_{max}	Best C_{max}	Mean dev. %	Avg. time
S16-P1	88	135	162.6	159	20.4	211.297	**144.6**	**144**	**7.1**	140.8†
S16-P2	102	99	112.7	111	13.8	215.083	**100**	**99***	**1**	16.32
S16-P3	99	140	164.7	163	17.6	272.122	**146.2**	**144**	**4.4**	158.4†
S16-P4	90	104	125.9	124	21.1	184.492	**108.2**	**106**	**4**	144†
S16-P5	96	100	119.8	118	19.8	201.75	**108.2**	**104**	**8.2**	153.6†
S16-P6	104	143	167.2	165	16.9	303.741	**147.1**	**145**	**2.9**	166.4†
S16-P7	106	121	142.4	141	17.7	278.149	**124.2**	**123**	**2.6**	169.6†
S16-P8	81	63	70.2	69	11.4	101.491	**63**	**63***	**0**	0
S16-P9	101	121	140.9	140	16.4	251.225	**124.2**	**121***	**2.6**	129.28
S16-P10	96	120	140.1	139	16.8	227.528	**122.2**	**121**	**1.8**	153.6†
S16-P11	93	80	92.2	90	15.3	160.841	**80**	**80***	**0**	0
S16-P12	110	154	194.6	192	26.4	376.427	**166.7**	**166**	**8.2**	176†
S16-P13	180	180	213.6	210	18.7	514.925	**188.1**	**183**	**4.5**	180†
S16-P14	97	84	99.7	98	18.7	179.886	**88**	**88**	**4.8**	155.2†
S16-P15	86	126	147.8	146	17.3	188.366	**130.3**	**127**	**3.4**	137.6†
S16-P16	106	56	67.1	66	19.8	167.417	**59.2**	**57**	**5.7**	169.6†
S16-P17	94	70	81.5	80	16.4	148.234	**71.3**	**70***	**1.9**	45.12
S16-P18	102	110	128.2	126	16.5	238.664	**115**	**112**	**4.5**	163.2†
S16-P19	80	112	136.6	135	22	155.491	**124.6**	**117**	**11.3**	128†
S16-P20	84	90	105.4	104	17.1	141.581	**94.2**	**92**	**4.7**	134.4†
S16-P21	78	88	105.8	103	20.2	124.656	**94.3**	**90**	**7.2**	124.8†
S16-P22	79	60	67.8	67	13	95.359	**62.2**	**60***	**3.7**	50.56
S16-P23	97	70	85.5	84	22.1	164.134	**74.7**	**73**	**6.7**	155.2†
S16-P24	93	60	67.6	66	12.7	128.911	**60.5**	**60***	**0.8**	14.88
S16-P25	96	120	140.1	137	16.8	226.203	**122.6**	**121**	**2.2**	153.6†

* Provably optimal solution

† All ten runs reached the running time limit $rt^{\max}$

Table 4 Makespan (C_{max}) and computational time (in seconds) results for Matta's 16-center problems

	2-center		4-center		8-center		16-center	
	GA	TS	GA	TS	GA	TS	GA	TS
Average deviation of mean makespan from LB	4.31%	4.31%	8.14%	9.8%	15.18%	5.56%	17.8%	4.17%
Average deviation of best makespan from LB	4.31%	4.31%	4.96%	7.63%	12.14%	3.41%	15.94%	2.08%
No. of provably optimal solutions (out of 25)	16	16	13	7	2	10	0	7
Average computational time (seconds)	2.74	0.02	18.81	8.05	120.65	55.66	210.32	120.81

Table 5 Average performance of Matta's GA and the proposed TS for each problem set

	Matta's GA	Proposed TS
Average deviation of Mean Makespan from LB	11.36%	5.96%
Average deviation of the best Makespan from LB	9.34%	4.36%
No. of provably optimal solutions found	31	40

Table 6 Comparison between Matta GA's performance and TS algorithm's performance

and the algorithm stopped at step 8 when no non-tabu move that can improve a current solution is found (such as in S4-P4, S4-P6, S4-P7, S4-P8, S4-P12, S4-P16..etc). The latter stopping condition seems to be the main reason of not being able to improve a current solution and consequently to get better results compared to GA in some cases. This is the condition of being stuck in a local minimum which is unavoidable in the currently proposed TS approach. The GA on the other hand is based on a random search approach that can avoid such conditions and therefore provide better results for the 4-center instances. On the other hand, the random search approach of GA is apparently consuming more time and follows an unguided search that is not capable of improving the quality of the obtained solutions, especially for larger problem instances.

5 Conclusion and future research

In this research, the proportionate multiprocessor open shop scheduling problem, a more realistic variant of the open shop scheduling problems with common occurrences in practice is addressed. An algorithmic approach is developed in order to solve it with the objective of minimizing the makespan. The algorithmic approach involves generating an initial feasible schedule by a dense schedule dispatching heuristic, followed by implementing a tabu search (TS) algorithm in the improvement phase.

The performance of the introduced solution approach is evaluated by comparing its results to results found in the literature for 100 benchmark instances which were solved using GA by Matta [2]. Competitive results were obtained and the best makespan computed through the TS approach is found to be better than, or at least equal to, the best solutions for 88 out of the 100 benchmark instances. The proposed TS was capable to produce provably optimal solutions for 40 instances, and the deviation of average makespan of 100 instances from the lower bound is found to be 5.96%. On the other hand, 31 problems were solved to optimality using GA, and the GA resulted in 11.36% deviation of average makespan of the 100 problems from the lower bound.

From the conducted experiments, it is evident that the proposed TS algorithm outperforms existing GA in scheduling large shops, i.e. with large number of operations, in terms of the number of optimal solutions reached and average deviation from the solutions' lower bounds. Furthermore, the computational time by the proposed TS is found to be less than that of GA in 96 cases, and the average computational time of TS is less than 60% of the GA's average computational time.

The developed TS approach is apparently not restricted to the special proportionate case of the multiprocessor open shop scheduling problem. The neighborhood search function used is applicable to the general case in which job processing times on process-

ing centers is job-dependent and are allowed to vary among jobs. It is also applicable to the more general case in which the processing speed may vary among the parallel machines in each center, relaxing the restriction of having identical machines.

This research emphasizes the success of TS utilizing the neighborhood search function that is based on dividing the critical path into blocks which are then used to define efficient moves. This methodology has proven to be very successful for many scheduling problem configurations including the job shop, the flow shop and their flexible or multiprocessor variants, as well as open shop. Future extensions can build on the results obtained in this research by developing a hybrid approach that combines random search population-based techniques such as genetic algorithm and memetic algorithm with the single-solution-based tabu search approach developed here.

References

1. Matta, M.: An empirical and theoretical study of outpatient scheduling problems employing simulation and genetic algorithm methodologies. Ph.D. thesis, Duke University (2004)
2. Matta, M.E.: A genetic algorithm for the proportionate multiprocessor open shop. Computers & Operations Research **36**(9), 2601–2618 (2009). DOI 10.1016/j.cor.2008.11.009
3. Matta, M.E., Elmaghraby, S.E.: Polynomial time algorithms for two special classes of the proportionate multiprocessor open shop. European Journal of Operational Research **201**(3), 720–728 (2010). DOI 10.1016/j.ejor.2009.03.048
4. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. SpringerLink : Bücher. Springer (2012)
5. Naderi, B., Fatemi Ghomi, S., Aminnayeri, M., Zandieh, M.: A study on open shop scheduling to minimise total tardiness. International Journal of Production Research **49**(15), 4657–4678 (2011). DOI 10.1080/00207543.2010.497174
6. Gonzalez, T., Sahni, S.: Open Shop Scheduling to Minimize Finish Time. Journal of the Association for Computing Machinery **23**(4), 665–679 (1976)
7. Shmoys, D.B., Stein, C., Wein, N.J.: Improved approximation algorithms for shop scheduling problems. SIAM Journal on Computing **23**(3), 617–632 (1994)
8. Chen, B., Strusevich, V.A.: Approximation Algorithms for Three-Machine Open Shop Scheduling. ORSA Journal on Computing **5**(3), 321–326 (1993)
9. Fang, H.l., Ross, P., Corne, D.: A Promising Hybrid GA / Heuristic Approach for Open-Shop Scheduling Problems. In: 11th European Conference on Artificial Intelligence, pp. 590–594 (1994)
10. Alcaide, D., Sicilia, J., Vig, D.: A Tabu Search Algorithm for the Open Shop Problem. Sociedad de Estadistica e Investigacion Operativa **5**(2), 283–296 (1997)
11. Liaw, C.f.: A tabu search algorithm for the open shop scheduling problem. Computers & Operations Research **26**, 109–126 (1999)
12. Liaw, C.F.: Applying simulated annealing to the open shop scheduling problem. IIE Transactions **31**(5), 457–465 (1999). DOI 10.1080/07408179908969848
13. Liaw, C.F.: A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research **124**(1), 28–42 (2000). DOI 10.1016/S0377-2217(99)00168-X
14. Prins, C.: Competitive genetic algorithms for the open-shop scheduling problem. Mathematical Methods of Operations Research (ZOR) **52**(3), 389–411 (2000). DOI 10.1007/s001860000090
15. Blum, C.: Beam-ACO–hybridizing ant colony optimization with beam search: an application to open shop scheduling. Computers & Operations Research **32**(6), 1565–1591 (2005). DOI 10.1016/j.cor.2003.11.018
16. Sha, D., Hsu, C.Y.: A new particle swarm optimization for the open shop scheduling problem. Computers & Operations Research **35**(10), 3243–3261 (2008). DOI 10.1016/j.cor.2007.02.019
17. Schuurman, P., Woeginger, G.J.: Approximation algorithms for the multiprocessor open shop scheduling problem. Operations Research Letters **24**, 157–163 (1999)

18. Sevastianov, S., Woeginger, G.: Linear time approximation scheme for the multiprocessor open shop problem. Discrete Applied Mathematics **114**(1-3), 273–288 (2001). DOI 10.1016/S0166-218X(00)00375-9
19. Sevastianov, S.V., Woeginger, G.J.: Makespan minimization in open shops : A polynomial time approximation scheme. Mathematical Programming **82**, 191–198 (1998)
20. Naderi, B., Fatemi Ghomi, S., Aminnayeri, M., Zandieh, M.: Scheduling open shops with parallel machines to minimize total completion time. Journal of Computational and Applied Mathematics **235**(5), 1275–1287 (2011). DOI 10.1016/j.cam.2010.08.013
21. Glover, F.: Tabu search: Part I. ORSA Journal on Computing **1**, 190–206 (1989)
22. Glover, F.: Tabu search: Part II. ORSA Journal on Computing **2**, 4–32 (1990)
23. Roy, B., Sussmann, B.: Les problemes dordonnancement avec constraints disjunctives. Tech. Rep. 9, SEMA, Paris (1964)
24. Abdelmaguid, T.F.: Permutation-induced acyclic networks for the job shop scheduling problem. Applied Mathematical Modelling **33**(3), 1560–1572 (2009). DOI 10.1016/j.apm.2008.02.004
25. Grabowski, J., Nowicki, E., Zdrzałka, S.: A block approach for single-machine scheduling with release dates and due dates. European Journal of Operational Research **26**(2), 278 – 285 (1986). DOI 10.1016/0377-2217(86)90191-8
26. Nowicki, E., Smutnicki, C.: A Fast Taboo Search Algorithm for the Job Shop Problem. Management Science **42**(6), 797–813 (1996)
27. Laarhoven, P.J.M.v., Aarts, E.H.L., Lenstra, J.K.: Job shop scheduling by simulated annealing. Operations Research **40**(1), pp. 113–125 (1992)
28. Glover, F.: Future paths for integer programming and links to artificial intelligence. Computers & Operations Research **13**(5), 533 – 549 (1986). DOI 10.1016/0305-0548(86)90048-1