A comparison of neighborhoods for the blocking job-shop problem with total tardiness minimization

Julia Lange
Otto-von-Guericke-University Magdeburg, Germany
julia.lange@ovgu.de

Keywords: job-shop scheduling, blocking, total tardiness, heuristics.

1 Introduction

The job-shop scheduling problem is one of the well-studied issues in scheduling research. The integration of blocking constraints is motivated by real-world applications like the scheduling of trains in a network and the production of large items. It refers to the absence of buffers in the planning system, so that a job blocks a machine until its subsequent machine is idle. As a customer-oriented optimization criterion, the minimization of the total tardiness of all jobs with regard to given due dates is considered.

Different mathematical programming formulations of the blocking job-shop problem (BJSP) with total tardiness minimization are tested and discussed in Lange and Werner (2017). The results provide an indication to the necessity of heuristic methods for the BJSP. In line with this idea, several authors present heuristic approaches to tackle related types of job-shop scheduling problems. Minimizing the makespan, the BJSP is solved by a genetic algorithm in Brizuela et al. (2001) and by a tabu search heuristic in Gröflin and Klinkert (2009). Different neighborhoods adapted to a total weighted tardiness objective are presented by Kuhpfahl and Bierwirth (2016) for the job-shop problem without blocking constraints. Furthermore, Bürgy (2017) applies a graph-based tabu search approach to the BJSP considering different regular optimization criteria. Based on a generalized graph formulation for the BJSP, a hybrid branch-and-bound-method is applied to a train scheduling problem with a tardiness-based objective in D’Ariano et al. (2007).

In contrast to the techniques given in the literature, two permutation-based heuristic approaches are presented and compared in this paper. Since total tardiness minimization corresponds to a regular optimization criterion, a solution to the problem is a schedule defined by the operation sequences on the machines. Here, these sequences are given by a list or permutation of all operations. Two well-known strategies are implemented to set up a neighborhood. First, a neighbor is determined by an adjacent pairwise interchange (API) of two operations on a machine. Second, a neighbor is defined by a random leftward shift of all operations of a job in the permutation.

While these neighborhoods are successfully applied to job-shop problems without additional constraints and characteristics like connectedness can be shown easily, there are significant feasibility issues occurring in the BJSP. Since a given permutation does not necessarily correspond to a feasible schedule, complex construction and repair procedures have to be used to define feasible neighbors. Therefore, performance and characteristics of these neighborhood structures need in-depth investigation for the BJSP.

In this paper, special emphasis is given to the adjustment of the neighborhood to the optimization criterion. In line with the idea of observing a critical path for a makespan objective, neighbors are defined based on choices of interchanges and shifts made from the set of tardy jobs. General and tardiness-based neighborhoods are described and implemented in a simulated annealing (SA) metaheuristic. Computational experiments are done on train scheduling-inspired instances as well as on benchmark instances from Lawrence (1984).
Conclusions are drawn regarding the solution quality of the heuristic methods compared to the results obtained by solving the corresponding MIP formulations.

2 Problem description

A set of jobs $\mathcal{J} = \{J_i \mid i = 1, \ldots, n\}$ is given, where each job consists of a set of operations and $O_{i,j}$ denotes the $j$-th operation of job $J_i$. The technological route of a job $J_i$ is defined by the requirement of a certain machine $M_k \in \mathcal{M}$ by each operation, where $\mathcal{M}$ describes the set of machines. Additionally, release dates $r_i$ and due dates $d_i$ are given for $J_i \in \mathcal{J}$ and recirculation is allowed. Among all schedules, which are feasible with regard to technological route and blocking constraints, a schedule with minimal total tardiness is to be found. The considered BJSP is characterized by $\mathcal{J} \mid r_i, d_i, \text{block}, \text{recr} \mid \sum T_i$.

A schedule can be expressed by an operation-based representation $s^{op}$ corresponding to the permutation of operations and by a machine-based representation $s^{ma}$ corresponding to the operation sequences on the machines. Thus, every operation $O_{i,j}$ is assigned to a list index $\text{lidx}(O_{i,j}) \in \{1, 2, \ldots, n_{op}\}$ in $s^{op}$, where $n_{op}$ denotes the number of operations, and to a machine index $\text{midx}(O_{i,j}) \in \{1, 2, \ldots, R_k\}$ in $s^{ma}$, where $R_k$ denotes the number of operations on machine $M_k$. These indexes can also be referred to as positions in the permutation and on the machine, respectively.

3 Permutation-based neighborhoods for the blocking job-shop problem with total tardiness minimization

Defining neighbors by APIs is a well-known strategy in job-shop scheduling. W.l.o.g., a pair of operations will only be interchanged, if there is no idle time on the machine between these operations. In this paper, a general API neighborhood is set up by choosing the neighbor-defining API from the set of all possible pairs of operations. Furthermore, the TAPI neighborhood is described by choosing the neighbor-defining API from the set of possible pairs of operations for which the second (leftward shifted) operation belongs to a tardy job. Both neighborhoods are set up using the machine-based representation of the solution.

In order to involve some randomness in the optimization process, the TJ neighborhood is defined and operated on the operation-based representation. Here, a job is randomly chosen from the set of tardy jobs and all its operations are shifted to arbitrary positions with lower list indexes in the permutation.

All three neighborhoods are exemplary illustrated for an instance with 3 machines and 4 jobs in Figure 3. A feasible schedule is given, where job $J_4$ is tardy and job $J_2$ is finished on time. A neighbor in the TJ neighborhood is generated by shifting all operation of the tardy job $J_4$ to positions with lower list indexes. As an example of a TAPI neighbor, the pair $O_{4,1}$ and $O_{4,2}$ on machine $M_1$ is chosen, since operation $O_{4,2}$ belongs to the tardy job $J_4$. In the more general API neighborhood, one possible neighbor-defining API reverses the order of the operations $O_{3,2}$ and $O_{2,2}$ on machine $M_2$.

In operating these three neighborhoods, the resulting operations-based representations are infeasible with regard to blocking constraints for most of the neighbors. A complex repair procedure is applied to construct feasible neighbors while taking the neighbor-defining API as given. By doing so, necessary changes in the schedule are made to regain feasibility. Größlin and Klinkert (2009) present a connected neighborhood for the BJSP based on a job-insertion technique with an underlying disjunctive graph. The neighborhoods considered in this paper are more general and operate on a simple list structure. Since the connectivity is not yet shown, computational experiments are a good index of performance.
Fig. 1. Illustration of three different neighbors of a schedule

4 Computational Results

The computational experiments are done on randomly generated train scheduling-inspired instances (TS) as well as on Lawrence instances (LA) adapted for the BJSP. The release dates \( r_i \) of the jobs are generated so that jobs are forced to overlap in time and the due dates are determined by \( d_i = \delta \cdot \sum p_{i,j} \) with a tight due date factor of \( \delta = 1.2 \). The size of the instances is denoted by \((m, n)\), where \( m \) corresponds to the number of machines and \( n \) indicates the number of jobs. There are five different instances for each instance size.

A simulated annealing (SA) is used to solve the given problems, where the TJ neighborhood is applied with a probability of 0.1 and the API and TAPI neighborhoods are complementary applied with a probability of 0.9, respectively. A geometric cooling scheme \( t_{i+1} = k \cdot t_i \) with \( k \in \{0.99, 0.995, 0.999\} \). The initial and terminal temperature, \( t_0 \) and \( T \), are chosen in accordance to the range of the objective function values with \((t_0, T) \in \{(20, 10), (200, 50), (1000, 100)\})\). The total number of iterations done by the SA ranges dependent on the instance size between 11000 and 64000.

The best out of the five runs for each instance (w.r.t. the objective function value) is compared to the results obtained by solving a MIP formulation for the BJSP with IBM ILOG CPLEX 12.6.1. as given in Lange and Werner (2017). The computational experiments involving the MIP solver, the SA with tardiness-based neighborhoods (TJ, TAPI) and the SA mainly relying on the general neighborhood (TJ, API) are summarized in Table 1. For each instance size, the number of instances solved to optimality (opt) by the MIP solver and the number of instances, for which a feasible solution is found, are given. Below, the number of instances for which the variants of the SA obtained the optimal solution or improved the best known feasible solution (opt/im) is stated. Additionally, the number of instances for which the SA approaches reached a solution within a 10% gap compared to the best known solution found by the MIP solver is given.

The comparison of the performance of the neighborhoods is done firstly based on the total number of instances solved with (near-)optimal solutions and secondly regarding the number of instances solved to optimality or improvement for each instance size. The superior setting is emphasized in bold face. It can be observed that the SA with the API neighborhood obtains better or equivalent solutions for eight of the ten given instance sizes. This indicates that due to the tardiness-based optimization criterion and a high number of interdependencies caused by blocking constraints the idea of an adaption to the objective

and potential. Furthermore, it is not clear whether a restriction of the API neighborhood based on the optimization criterion is reasonable for the minimization of total tardiness.
Table 1. Computational results of applying an SA to the BJSP

<table>
<thead>
<tr>
<th>instance type</th>
<th>MIP</th>
<th>SA - (TJ, TAPI)</th>
<th>SA - (TJ, API)</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt</td>
<td>5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>feasible</td>
<td>-</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>&lt; 10%</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>

function does not improve the API neighborhood for the BJSP. Statistical issues, which are not presented in detail here, show that the larger number of possible neighbors causes a higher deviation in the objective function values of the best solutions found but leads to better results on average.

5 Conclusion

In this paper, general and tardiness-based neighborhood structures for the BJSP are embedded in a SA and tested with regard to their performance compared to MIP solving techniques. Computational experiments are done on train-scheduling inspired and benchmark instances. The results give evidence to the fact that an adaption of the API neighborhood to the objective by exclusively choosing leftward interchanged operations of tardy jobs does not improve the solution quality. Significant feasibility issues involved by the blocking constraints seem to necessitate a larger number of possible neighbors to obtain (near-)optimal solutions with higher frequency.

References

Lange, J., F. Werner, 2017, 'Approaches to modeling train scheduling problems as job-shop problems with blocking constraints', Journal of Scheduling, published online, DOI 10.1007/s10951-017-0526-0.
Oddi, A., R. Rasconi, A. Cesta and S. Smith, 2012, 'Iterative improvement algorithms for the blocking job shop', in ICAPS.