JOB-SHOP PROBLEMS WITH OBJECTIVES APPROPRIATE FOR TRAIN SCHEDULING IN A SINGLE-TRACK RAILWAY

Omid Gholami
*Islamic Azad university, Mahmudabad Branch, Mahmudabad, Iran*
gholami@iaumah.ac.ir

Yuri N. Sotskov
*United Institute of Informatics Problems, National Academy of Sciences of Belarus, Minsk, Belarus*
sotskov@newman.bas-net.by

Frank Werner
*Faculty of Mathematics, Otto-von-Guericke-University, Magdeburg, Germany*
frank.werner@ovgu.de

Keywords: Job-shop, Train scheduling, Regular criteria, Simulation.

Abstract: A train scheduling problem in a single-track railway is studied using a mixed graph model for a job-shop with appropriate criteria. There are several performance evaluations for a train schedule. Optimizing a train schedule subtends minimizing total tardiness of the trains, minimizing the sum of train transit times, minimizing the makespan for a train schedule, etc. Since the corresponding job-shop problems with the above three criteria are NP-hard, several heuristic algorithms have been developed using different priorities based on release times of jobs, job due-dates and job completion times. Simulation on a computer was used for evaluating the quality and efficiency of the heuristic algorithms developed for the appropriate job-shop problems. Release times, due-dates and completion times of the jobs have been used as input parameters (priorities) in the computer simulation to see effect of them on a quality of schedules with different objective functions. The efficiency of the developed heuristics was demonstrated via simulation on a set of randomly generated instances of small and medium sizes. Computational results showed that one heuristic algorithm outperformed other algorithms tested for two of three objective functions under consideration.

1 INTRODUCTION

The paper is addressed the problem of generating an efficient schedule of passenger and freight trains in a single-track railway. We use terminology from (Lusby et al., 2011) for train scheduling and that from (Tanaev et al., 1994) for machine scheduling.

In the world, a railway traffic is increasing from year to year. The employment of railroads grows both for passenger and for freight transportation. When the density of train moving is increasing, a schedule of trains becomes more difficult for generating and control. During the last decades, a lot of new algorithms and software have been developed and published in the OR literature and in special literature in order to produce a better tool for generating an accurate and reliable train schedules.

In this paper, it is shown how one can find a train schedule which is close to optimal one with three objective functions for a single-track railway.

An obvious way to achieve a proper train schedule is going through job-shop scheduling (Burdett and Kozan, 2010; Mascis and Pacciarelli, 2002; Szpigel, 1973), although job-shop problems are fairly complicated since they belong to class of NP-hard problems (Brucker et al., 1997; Brucker et al., 2007; Tanaev et al., 1994). In order to achieve a practical size of a job-shop problem, which can be solved within a reasonable time, we propose and test several heuristic algorithms for three objective functions which are appropriate for train scheduling.

In Sections 3–6, we consider a railway network provided that a pair of sequential stations can be connected by at most one single-track (a railroad section). In particular, this is the case for the most railway systems in countries of the Middle East.

2 LITERATURE REVIEW

In (Zhou and Zhong, 2007), a resource-constrained project scheduling was used for a single-track timetabling problem. Both the track segments and stations were modelled as limited resources. A
branch-and-bound algorithm has been developed in order to obtain a feasible train timetable with a guaranteed level of the optimality. A lower bound based on Lagrangian relaxation was used to relax segment and station capacity constraints. A lower bound was used to estimate the least train delay. An upper bound was constructed via a beam search heuristic. In (Cai and Goh, 1994), a heuristic algorithm was developed for the train scheduling in a single-track railway with the assumption that all trains moving in the same direction must have the same speed and terminating siding. A greedy heuristic was proposed basing on a local optimality criterion in the event of a potential crossing conflict. The authors claim that a suboptimal solution is obtained rather quickly. Two illustrative examples with 12 trains and 20 trains were presented.

Paper (Liu and Kozan, 2011) is devoted to train scheduling problems when prioritized trains and non-prioritized trains are simultaneously traversed in a single-track railway. No-wait conditions arise because the prioritized trains (e.g., an express passenger train has a higher priority) should traverse continuously without interruptions. Non-prioritized trains (e.g., a freight train) are allowed to either enter the next section immediately (if it is free) or to remain in a section until the next section on the routing becomes available. A generic algorithm has been developed to construct a feasible train timetable in terms of the given train order. The proposed algorithm comprises several recursively used procedures to guarantee the feasibility by satisfying the no-waiting, a deadlock-free condition, and a conflict-free constraint.

B. Szpigel (Szpigel, 1973) was the first who identified the similarities between a job-shop problem and a train scheduling in a single-track railway. The former was solved in (Szpigel, 1973) using a branch-and-bound algorithm, the initial linear programming excluding order constraints. Branching was required if the current solution contains trains which are in a conflict (i.e., when trains turn out to be located on the same railroad section at the same time). The objective was to minimize a weighted sum of train transit times. The computational results for 5 single-track sections and 10 trains has been reported. The same problem was considered in (Carey and Lockwood, 1995) via binary mixed integer programming similarly to that considered in (Jovanovic and Harker, 1991). Temporal constraints were identical to those used in (Szpigel, 1973). The objective was to minimize the deviation from the ideal arrival times and the departure times for the trains to be scheduled.

In (Mladenovic and Cangalovic, 2007), a job-shop problem was used as a way to solve the train scheduling problem where a route was interpreted as follows: The route is a sequence of facilities the train must cross from the origin to a destination. Assuming that the train trips are jobs to be scheduled, which require the elements of infrastructure as restricted resources, it was done by the mapping of the initial problem into a special case of a job-shop problem. In order to solve the job-shop problem, a constraint programming approach has been developed. A support to fast finding a good schedule was offered by an original separation and a bound-and-search heuristic. To improve the time performance, a surrogate objective function was used which had a smaller domain than the actual objective function.

In (Dorfman and Medanic, 2004), a discrete-event model was used to schedule the traffic on a railway network. This model was computationally efficient and generates near optimal schedules with respect to a number of time-of-travel-related criteria. In (Burdet and Kozan, 2010), the train scheduling was interpreted in terms of a job-shop problem with parallel machines. A disjunctive graph model was used in several algorithms with a makespan objective. It was demonstrated that solutions with a good quality may be obtained within a reasonable CPU-time.

3 PROBLEMS SETTINGS AND TESTING

One of the main problems in the management of a railway network is a train scheduling (timetabling) problem, in which it is necessary to determine a schedule (timetable) for a set of given trains that does not violate railway constraints. This problem has to be solved at a tactical level of the railway planning process (Lusby et al., 2011). For the case of a single-track railway, a train scheduling may be interpreted as the following job-shop problems.

There are $n$ jobs $J = \{J_1, J_2, \ldots, J_n\}$ to be processed on $m$ different machines $M = \{M_1, M_2, \ldots, M_m\}$. Time $p_{ij} > 0$ needed for processing an operation $O_{ij}$ of a job $J_i \in J$ on a machine $M_j \in M$ is known. Operation preemptions are not allowed, and machines routes $O' = (O_{i1}, O_{i2}, \ldots, O_{im})$ for jobs $J_i \in J$ may be different for different jobs. A job $J_i \in J$ is available for processing from time-point $r_i \geq 0$. Time-point $d_i$ defines a due-date for completion job $J_i$. A machine $M_k \in M$ can process a job $J_i \in J$ at most once. So, any two operations $O_{ij}$ and $O_{ik}$, $j \neq k$, of the same job $J_i \in J$ have to be processed by different machines of set $M_i$, i.e., inequality $n_i \leq m$ holds (such a problem is called a classical job-shop).

One objective is to find a schedule minimizing the sum $\sum_{i=1}^{n} T_i$ of the tardiness $T_i = \max\{0, C_i - d_i\}$
for jobs $J_i \in J$. Hereafter, $C_i$ denotes the completion time of a job $J_i \in J$. According to the three-field notations $\alpha|\beta|\gamma$ used for machine-scheduling problems, the above job-shop problem is denoted as $J|r|\Sigma T_i$. If $\gamma = \Sigma_{i=1}^{n} C_i$, then this problem is denoted as $J|r|\Sigma C_i$. If $\gamma = \max_{i=1}^{n} C_i$, then it is denoted as $J|r|\max C_i$.

Problems $J|r|\Sigma T_i$, $J|r|\Sigma C_i$ and $J|r|\max C_i$ arise in train scheduling for a single-track railway: To determine the best train schedule among those which do not violate single-track capacities. For passenger trains, criteria $\Sigma T_i$ and $\Sigma C_i$ are more important than $C_{max}$, while for freight trains, criteria $C_{max}$ and $\Sigma C_i$ are more important than $\Sigma T_i$.

In a job-shop approach to train scheduling, trains and railroad sections are synonymous with jobs $J_i \in J$ and machines $M_j \in M$, respectively. An operation $O_{ij}$ is regarded as a movement of a train $J_i \in J$ across a railroad section $M_j \in M$. A positive number $p_{ij}$ denotes the time required for train $J_i \in J$ to travel through railroad section $M_j \in M$. A non-negative number $r_i$ denotes the departure time of train $J_i \in J$, which is given in the official train timetable. A positive number $d_i$ denotes the official arrival time of train $J_i \in J$ (a due-date for the desired completion time $C_i$ of a job $J_i \in J$) to the terminal station in the route $O$.

Problems $J|r|\Sigma T_i$, $J|r|\Sigma C_i$ and $J|r|\max C_i$ are complicated in computational sense since their special cases belong to the class of NP-hard problems (Tanaev et al., 1994). In order to achieve a practical size of a classical job-shop problem, which can be solved heuristically within a reasonable time, we coded a shifting bottleneck algorithm, which was originated in (Adams et al., 1988) for a job-shop problem $J|r|\max C_i$. However, testing the program realizing a shifting bottleneck algorithm for the problem $J|r|\Sigma T_i$ showed unsatisfactory large CPU-time when number $n$ of trains was large (Sotskov and Gholami, 2012). The simulation showed that this algorithm can handle 125 operations (e.g., 25 trains on 5 railroad sections) within half an hour of CPU-time. For larger job-shop problems, the CPU-time grows very quickly.

In Sections 4 and 5, we develop heuristic algorithms, which run faster than shifting bottleneck algorithm providing a quality of the objective functions (Section 6) which is close to the quality of schedules constructed by the shifting bottleneck algorithm.

4 MIXED GRAPH MODEL

Problems $J|r|\Sigma T_i$, $J|r|\Sigma C_i$ and $J|r|\max C_i$ described in Section 3 can be formulated using a mixed graph model $G = (Q, C, D)$ (Tanaev et al., 1994) or a disjunctive graph model (Sussmann, 1972).

Let $Q$ denote the set of operations $O_{ij}$, $J_i \in J$, $j \in \{1, 2, \ldots, n_i\}$, to be executed by machines $M$ and a dummy operation $O_{00}$ associated with the beginning of a schedule and a dummy operation $O_{n_i+1}$ associated with the completion of the jobs $J_i \in J$.

Two operations $O_{ij}$ and $O_{lk}$, which have to be executed by the same machine $M_j \in M$, cannot be simultaneously processed by this machine. This restriction is presented by edge $[O_{ij}, O_{lk}] \in D$.

Two consecutive operations $O_{ij}$ and $O_{i,j+1}$ of the same job $J_i \in J$ have are connected by arc $(O_{ij}, O_{i,j+1}) \in C$, where $1 \leq j \leq n_i - 1$. The arc $(O_{ij}, O_{i,j+1})$ means that operation $O_{i,j+1}$ has to be started after completion operation $O_{ij}$.

The processing time $p_{ij}$ is prescribed to the arc $(O_{ij}, O_{i,j+1}) \in C$, and two processing times $p_{ij}$ and $p_{ik}$ are prescribed to the edge $[O_{ij}, O_{lk}] \in D$.

For a dummy operation $O_{00} \in Q$, the arc $(O_{00}, O_{lk})$ with the weight $r_k$ is included into the set $C$ for each job $J_i \in J$. For the dummy operation $O_{n_i+1}$, the arc $(O_{00}, O_{n_i+1})$ with the weight $p_{in}$ is included into the set $C$.

Problems $J|r|\Sigma T_i$, $J|r|\Sigma C_i$ and $J|r|\max C_i$ are modelled by a mixed graph $G = (Q, C, D)$. The due-dates $d_i$ are used in calculating the objective function $\gamma = \Sigma_{i=1}^{n} T_i$ for a schedule constructed.

Since operation preemptions are not allowed, a schedule on a mixed graph $G = (Q, C, D)$ may be defined as a sequence of the starting times $s_i = (s_{00} = 0, s_1, \ldots, s_{n_i-1}, s_{n_i+1}, \ldots, s_{n_i}, \ldots, s_{n_i+1})$, $s_i$, $s_j$, $s_{ji}$ of all the operations $Q$ such that the conjunctive constraint

$$s_i^h - s_i^l \geq p_{ij}$$

has to be satisfied for each arc $(O_{ij}, O_{lk}) \in C$, and the disjunctive constraint

$$s_i^h - s_j^i \geq p_{ij} \text{ or } s_j^i - s_i^h \geq p_{lk}$$

has to be satisfied for each edge $[O_{ij}, O_{lk}] \in D$.

Using the above weighted mixed graph $G = (Q, C, D)$, to define a sequence of the starting times $s_i$, one has to replace each edge $[O_{ij}, O_{lk}] \in D$ by either the arc $(O_{ij}, O_{lk})$ with the weight $p_{ij}$ or the arc $(O_{lk}, O_{ij})$ with the weight $p_{lk}$ respecting to the disjunctive constraint (2) in such a way that no circuit arises in the obtained digraph.

As a result, the set of edges $D$ will be substituted by a chosen set $D'$, the mixed graph $G = (Q, C, D')$ will be transformed to a circuit-free digraph $G_i = (Q, C \cup D', \varnothing)$, and an operation sequence for each machine of the set $M$ will be determined.

Since the cardinality of the set $Q$ is equal to $|Q| = 1 + \sum_{i=1}^{n} (n_i + 1)$, using a critical path method in $O(n^2)$ time, one can build a unique semiactive schedule defined by the weighted digraph $G_i$. A schedule is called semiactive if no operation $O_{ij}$, $J_i \in J$,
$j \in \{1,2,\ldots,n\}$, can start earlier without delaying the processing of some operation from the set $Q$ or (and) without altering the processing sequence of the operations on any of the machines $M$.

The main complexity of the problem $J|r_i|\gamma$ with regular criterion $\gamma$ is to find an optimal circuit-free digraph $G = (O, C \cup D, \varnothing)$ generated by the mixed graph $G = (O, C, D)$. In the other words, it is necessary to find such a set of arcs $D$ for substituting the set of edges $D$ in the mixed graph $G$ that the objective function $\gamma$ has the minimal value among all other circuit-free digraphs generated by the mixed graph $G$ via replacing each edge $(O_{ij},O_{lk}) \in D$ either by arc $(O_{ij},O_{lk}) \in D$ or by arc $(O_{lk},O_{ij}) \in D$.

5. **HEURISTIC ALGORITHMS**

We developed three types of heuristic algorithms named as Ordinal, Max-PT and Min-PT types. We used release times, completion times and due-dates as priorities in ordering jobs $J_i \in J$ for processing on the same machine of set $M$. As a result, we obtained nine heuristic algorithms of three types with three priority rules.

5.1 Ordinal-algorithm

Ordinal-algorithm makes a sequence of operations $O_{ij}$ on different machines of set $M$ in the order as they are requested for processing jobs $J_i \in J$.

In the first iteration, Ordinal-algorithm finds the first job (i.e., operation $O_{i1}$) of a job $J_i \in J$ for machine $M_i \in M$ processing operation $O_{i1}$. Then depending on which priority rule is used, an Ordinal-algorithm computes either release time or completion time or due-date as a priority of operation $O_{i1}$.

For definiteness, let Ordinal-algorithm use a release time of operation $O_{i1}$ as its priority. Then algorithm compares release time $r_{i1}$ of operation $O_{i1}$ with release times of all the operations $O_{jk}$ of other jobs $J_j \in J$, $i \neq j$, on the same machine $M_i \in M$ processing operation $O_{jk}$. If release time $r_{i1}$ is smaller than release time of operations of other jobs on the same machine $M_i \in M$, then arc starting from operation $O_{i1}$ and ending in operation $O_{jk}$ has to be added to digraph $(Q, C, \varnothing)$. Otherwise, a symmetric arc $(O_{jk},O_{i1})$ has to be added to digraph $(Q, C, \varnothing)$.

Release time $r_{i1}$ means an earliest start time of operation $O_{i1}$ which can be computed due to the following recursion: $r_{i1} = \max\{r_{i1} + p_{i1}\}$, where maximum is taken for all operations $O_{ij} \in O$ preceding operations $O_{i1}$ in the digraph already constructed, the release time of source operation $O_{00}$ being equal to zero.

The above procedure is continued for the second job request (iteration 2), then for the third job request (iteration 3) and so on until all machines requests have been satisfied. We called this version of the algorithm as Ordinal-SRT (Shortest Release Time).

Other two versions of Ordinal-algorithm based on either completion time priority or due-date priority are called Ordinal-SCT (Shortest Completion Time) and Ordinal-SDD (Shortest Due-Date), respectively.

5.2 MaxPT-algorithm

MaxPT-algorithm (Maximum Processing Time) tends to first schedule jobs that need more processing time on all machines $M_i \in M$.

In the first step, MaxPT-algorithm calculates a sum of the processing times (a total processing time) of all operations $O_{ij}$, $j \in \{1,2,\ldots,n\}$, for each job $J_i \in J$. Before scheduling the maximum sum of processing time of a job $J_i \in J$ is equal to the length of a critical path in the digraph $(Q, C, \varnothing)$. MaxPT-algorithm sorts jobs $J$ in non-increasing order of their total processing times and selects a job with the maximum total processing time.

MaxPT-algorithm, starts to process the first request (operation $O_{i1}$) of the job $J_i$ with maximum total processing time, then the second request of the same job and so on until the last request of job $J_i$.

At each operation, MaxPT-algorithm computes one of three priorities: either release time, or completion time or due-date depending of which of three versions of a MaxPT-algorithm it is. Algorithm compares the chosen priority with those of all the operations of other jobs on the same machine. Then either arc $(O_{i1},O_{jk})$ or arc $(O_{jk},O_{i1})$ is added to digraph $(Q, C, \varnothing)$ depending on the larger priority of job $J_i$ and job $J_j$. The added arc defines order for processing jobs $J_i$ and $J_j$.

Then MaxPT-algorithm realizes the same procedure for the others jobs that are sorted by non-increasing of the sums of their processing times. We called this version of algorithm that uses release time as priority by MaxPTRT-algorithm (Maximum Processing Time, Release Time).

MaxPTCT-algorithm (Maximum Processing Time and Completion Time) is another version that compares job completion times as priorities and MaxPTDD-algorithm (Maximum Processing Time, Due-Date) compares due-dates as priorities.
5.3 MinPT-algorithm

MinPT-algorithm (Minimum Processing Time) basically is similar as its counterpart, MaxPT-algorithm, but contrary tries to first schedule jobs $J_i \in J$ that need less total processing time on all machines. MinPT-algorithm sorts jobs in non-decreasing order of their total processing times and then schedule jobs on each machine $M_k \in M$ with non-decreasing of corresponding priorities.

Three versions of MinPT-algorithm are MinPTRT-algorithm (Minimum Processing Time, Release Time), MinPTCT-algorithm (Minimum Processing Time and Completion Time) and MinPTDD-algorithm (Minimum Processing Time, Due-Date).

6 COMPUTATIONAL RESULTS

Ordinal-algorithm, Max-PT algorithm, Min-PT algorithms have been coded in Borland Delphi. For simulating, a laptop computer with the following specification was used: Intel, coreTM 2 Duo, CPU T6400, 2.00 GHz and 2GB Internal Memory, Windows 7, Ultimate 32 bit. We were interested to investigate experimentally an effect of choosing different algorithm and priorities on three objective functions. We randomly generated job-shop problems with size $n \times m$, where $n = m \in \{5, 6, \ldots, 12\}$ to see effect of these algorithms on different objective functions for job-shop problems.

First we compared makespan objective function for randomly generated problems $J | r_i | C_{\text{max}}$ with different size $n \times m$, where $n = m$. We compared the makespan values obtained by all developed algorithms for instances with the same input data. The results are presented in Fig. 1, which shows that quality of a schedule obtained by algorithms generally depends on the input data, but both OrdinalSCT-algorithm and OrdinalSRT-algorithm slightly outperform other algorithms for the makespan criterion.

In the next experiments, nine algorithms were used to solve heuristically the job-shop problems with objective function $\sum C_i$. Figure 2 shows that OrdinalSCT-algorithm provides schedules with the best quality among other algorithms tested.

In the last experiments, we evaluated objective function $\sum T_i$. Figure 3 shows that Ordinal-SCT again provides schedules with the best quality among other algorithms tested.

In Table 1, CPU-time taken by nine algorithms to solve heuristically different job-shop problems are given. For all randomly generated problems with size $n = 5 = m$ and $n = 6 = m$, the CPU-time was less than 1 s.

7 CONCLUSION

A problem of finding an optimal train schedule has several criteria. We consider three of them and developed nine heuristic algorithms to solve heuristical the corresponding job-shop problems. We made different variants of heuristic algorithms and compare
Our simulation shows that Ordinal-SCT generates the best schedule to minimize the schedule makespan, the sum of job completion times and the sum of job tardiness. Our simulation showed that using more complicated algorithms (like a shifting bottleneck) for solving the same problems needed more CPU-time with slightly improvement of the objective function values. As a future work we recommend to compare other parameters and new objective functions appropriate for a train scheduling. Note that OrdinalSCT-algorithm may be easily generated for the weighted objective functions $\sum w_i T_i$ and $\sum w_i C_i$ allowing a scheduler to take into account train priorities.

**REFERENCES**


