Optimization Methods for Large-scale Production Scheduling Problems

Roland Braune, Stefan Wagner, Michael Affenzeller

Upper Austria University of Applied Sciences
Hauptstrasse 117
4232 Hagenberg
{roland,stefan,michael}@heuristiclab.com

Abstract. In this paper we present a computational study of optimization methods for production scheduling problems which can be described by a job shop model. Contrary to most existing publications in this field our research focuses on the performance of these methods with respect to large-scale problem instances. The examined methods rely on a graph model as a solution representation and have originally been designed for problems of small size. We apply them to a set of semi-randomly generated problem instances whose properties have been transferred from common (smaller) benchmarks. The experiments are based on tardiness minimization and the results are evaluated in relation to a priority rule based heuristic.

1 Introduction

A common way of describing production scheduling problems in an abstract form is to reduce them to a job shop model [5]. This model is widely discussed in literature and has been subject to intensive methodic research over the past decades. In a job shop production jobs are subdivided into operations. Each such operation is assigned a particular machine on the shop floor on which it has to be processed. The order in which a job's operations are executed is predetermined and can be different for each job. This is also referred to as the *technological order* of operations or as the *precedence constraints* of the associated scheduling problem. Each machine can process at most one operation at a time (*capacity constraint*) and there is no preemption allowed which means that once the execution of an operation has started, it cannot be interrupted until it is finished.

We denote the set of all jobs by $J = \{j \mid 1 \leq j \leq n\}$ and the set of all machines by $M = \{k \mid 1 \leq k \leq m\}$. It is assumed that each job has to be processed on every machine, hence $O = \{o_{jk} \mid 1 \leq j \leq n, 1 \leq k \leq m\}$ constitutes the set of all operations. Operation o_{jk} refers to the operation of job j which is to be executed on machine M_k. Each operation o_{jk} requires p_{jk} time units to execute, which we call the operation's processing time. A *feasible schedule* S is a set of starting times s_{jk} for each operation which satisfies both, the precedence and the capacity constraints. The problem of finding a schedule which is as good

as possible with regard to a given objective function is considered a combinatorial optimization problem.

Given a particular schedule S, the completion time of each job j is denoted by C_j. The minimization of the *makespan* $C_{max} = \max(C_1, \ldots, C_n)$ is the most common objective in the area of job shop scheduling. However, in our contribution we use a different objective function, namely the total weighted tardiness. In this context, each job is assigned a due date d_j and a weight w_j. The tardiness T_j of a job j is defined as $T = \max(0, C_j - d_j)$. The optimization problem consists in minimizing the expression $\sum_{j=1}^{n} w_j T_j$. In the 3-field notation of Graham et al. [7] this problem is denoted by $Jm||\sum_{j=1}^{n} w_j T_j$.

The considered problem is $\mathcal{NP}$-hard since it is a generalization of the single-machine problem $1||\sum_{j=1}^{n} w_j T_j$ which is known to be strongly $\mathcal{NP}$-hard [9].

Literature on $Jm||\sum_{j=1}^{n} w_j T_j$ is quite sparse. Priority rule based approaches are described in [15] and [2]. Singer and Pinedo propose a Branch&Bound method [14] and a shifting bottleneck procedure [12]. In [8] Kreipl describes an Iterated Local Search method for tardiness job shops. Furthermore, Genetic Algorithms (e.g. [11]) and Tabu Search (e.g. [3]) have also been applied to this kind of problem.

In this contribution, we focus on a comparison of selected methods for tardiness minimization in job shops. Since most of the research in this field concentrates on relatively small problems (up to 400 operations), we investigate how these methods perform when applied to much larger instances (up to 4000 operations) as they display the situation of a real-world scenario much closer.

In Section 2 we introduce the graph model for job shop problems. The methods which are subject to our investigations are outlined in Section 3. Section 4 describes in detail how the benchmark problem set was generated followed by the discussion of the experimental results (cf. Section 5). Conclusions and outlook on further research are given in the final Section 6.

2 Graph Model

A job shop problem can be mapped to a disjunctive graph $G = (N, C, D)$ [13]. N denotes the node set, which contains one node for each operation o_{jk}, one source node U and n sink nodes $V_1, \ldots, V_n$ (one for each job). C is the set of directed (conjunctive) arcs and represents the technological order of operations whithin each job. The disjunctive arc set D consists of pairs of directed arcs between each two operations on the same machine. Each node is assigned a weight which corresponds to the operation's processing time. Figure 1 shows a disjunctive graph for a problem involving 3 jobs and 3 machines (3×3).

A schedule can be obtained by selecting one of two possible arcs between each pair of operations on the same machine. The result, which is also referred to as a (complete) *selection*, represents a directed graph. The schedule is feasible if the corresponding graph is acyclic.

Given a complete selection, one can compute the longest path $L(U, V_j)$ from the source to sink node V_j. The length of the longest path equals the completion

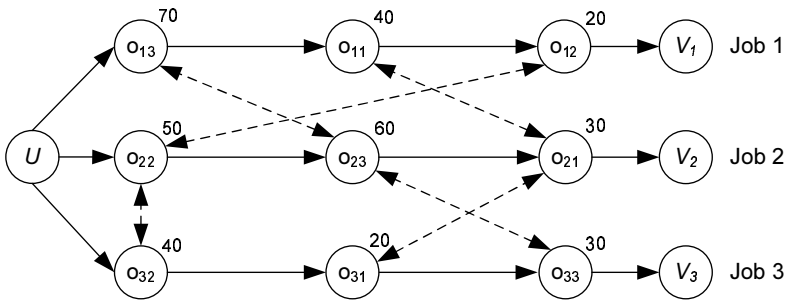

Fig. 1. Disjunctive Graph for a 3×3 Problem

time C_j of job j. A path $L(U, V_j)$ is called *critical* if it leads to a completion time $C_j > d_j$ and thus to job tardiness.

3 Methodology

Exact solution methods in the area of job shop scheduling are limited to small problems up to 400 operations. Heuristic approaches on the other hand have the advantage of being applicable to much larger instances. In our contribution, we therefore selected two techniques which appeared quite promising for such problem sizes: The Shifting Bottleneck Procedure by Pinedo and Singer [12] in a slightly modified form and the Iterated Local Search algorithm by Kreipl [8]. Both methods are based on the graph model described in Section 2 and their most important characteristics are outlined in the following two sections.

3.1 The (Modified) Shifting Bottleneck Procedure

The Shifting Bottleneck Procedure (SBP) is a problem decomposition approach originally proposed for the minimum makespan job shop [1]. It splits the original problem into m single machine problems and sequences them (separately) one after the other in bottleneck order. An outline of the basic principle is given in Algorithm 1, where $M_0 \subseteq M$ denotes the set of already scheduled machines.

We basically implemented the Shifting Bottleneck Procedure for tardiness scheduling as described in [12] except for the subproblem optimization. Pinedo and Singer propose a partial enumeration heuristic for the solution of the single machine problems. However, due to excessive running times this method is impractical for large single machine instances involving 50 or more operations. Especially when using enhanced control structures such as reoptimization or backtracking the efficient optimization of subproblems is of great importance. Even metaheuristics such as Tabu Search or Genetic Algorithms turned out to be too time consuming in this special context. Therefore we applied a simple first improvement local search algorithm based on principles described in [4] with

Algorithm 1 Shifting Bottleneck Procedure - Main Flow

```
Initialize disjunctive graph G
Set M_0 ← ∅
while M_0 ≠ M do
  for all k ∈ M \ M_0 do
    Formulate single machine (sub)problem
    Solve single machine problem
  end for
  Select bottleneck machine k' ∈ M \ M_0
  Insert directed arcs for machine k' into G
  Set M_0 ← M_0 ∪ {k'}
  Reoptimize scheduled machines (optional)
end while
```

modifications concerning the handling of delayed precedence constraints and the specific objective function.

3.2 The Iterated Local Search Algorithm

Iterated Local Search (ILS) or Large Step Optimization methods in the job shop scheduling domain have been extensively studied first in [10]. The main idea behind this kind of local search algorithms is the alternation of intensification and diversification phases. In the intensification or small step phase usually a simple neighborhood search (descent) algorithm gets applied in order to find the nearest local minimum. The diversification or large step phase on the other hand is responsible for exploring the solution space in order to find new promising regions.

For our experiments we implemented the ILS algorithm for the total weighted tardiness job shop problem proposed by Kreipl [8]. The method uses a neighborhood structure which is based on the graph model as described in Section 2. Neighboring solutions are basically obtained by interchanging operations which are on at least one critical path in the directed graph corresponding to the current solution. In order to compact the schedule after the interchange, two further related operations may be interchanged.

In order to escape from local minima, the large step phase needs to accept neighboring solutions which are worse than the current solution. This process is controlled by a Metropolis algorithm with fixed temperature. The number of iterations is determined depending on how close the previous small step phase got to the best solution reached so far.

4 Problem Setup

Benchmark problems for tardiness job shops are rare. For this reason, it is of common practice to modify well-known makespan related benchmark problems by adding due dates and weights to them. Pinedo and Singer [12] used instances

from the OR-Library[1] and adapted them in the following way: Given a *due date tightness factor* f, the due dates d_j are computed as $d_j = r_j + \lfloor f \cdot \sum_{k=1}^{m} p_{jk} \rfloor$, where $r_j = 0$ is assumed for all jobs. For 20% of all jobs they choose $w_j = 4$, 60% are assigned $w_j = 2$ and the remaining 20% are set to $w_j = 1$.

Since Pinedo and Singer used only 10×10 instances, and the OR-library problems are generally of limited size, we had to create our own benchmarks. Additionally, our aim was to focus on congested job shops involving much more jobs than machines (rectangular job shops). For our experiments we applied four different problem configurations: $\{50, 100\}$ jobs on 10 machines and $\{100, 200\}$ jobs on 20 machines.

Similar to most of the OR-Library instances, the operation processing times were sampled from a uniform distribution. However, as for the machine sequences (technological orders) a different approach has been chosen: The machine sequences, often also referred to as *job routings*, were created according to predefined template distributions. In real world manufacturing scenarios, job routings are also not completely different from each other: Some machines are always passed through in the same order, other machines are dedicated entry or exit points, etc. Although most of the OR-Library instances have job routings sampled from a uniform distribution, the resulting actual machine positions within the routings are not evenly distributed due to the small number of samples. What looks like a drawback at first sight, is exactly what we want: A certain degree of similarity between job routings.

For our experiments we extracted positional information from the original OR-Library instances and transferred it to our newly generated large-scale instances. This course of action has the following advantage: The resulting problems are "similar" to the original problems. The transfer of machine positions preserves some important properties, especially with regard to problem difficulty in the context of the Shifting Bottleneck Procedure, which permits a better interpretation of results.

5 Experimental Results

The computational experiments were carried out using randomly generated problems of size 50×10, 100×10, 100×20 and 200×20. The job routings are based on 5 different benchmark instances taken from the OR-library according to Section 4: `abz5`, `ft10`, `la19`, `orb03` and `orb10`. The experiments were coded in C# in the HeuristicLab optimization environment [16] and run on a 3 GHz Pentium 4 PC with 1 GB RAM.

The parameters for the Shifting Bottleneck Procedure are summarized in Table 1. It has to be remarked that the SB procedure was not able to produce reasonable results for instances greater than 100×10. Furthermore, the backtracking control structure could only be applied to the 50×10 instance. In the 100-job case the computational effort needed to solve the subproblem instance

[1] `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`

Table 1. Parameter Settings for the Shifting Bottleneck Procedure

	50×10	100×10
Subproblem solution method	FI Descent	FI Descent
Subproblem ATC parameter	2	4
Reoptimization	yes	yes (2x)
Backtracking	yes	no
Backtracking aperture size	2	-

Table 2. Due Date Tightness Factors

Size	f_{tight}	f_{loose}
50×10	4	4.5
100×10	7	8
100×20	4.5	5
200×20	8	9

Table 3. Algorithm Running Times

Size	**SB (avg.)**	**ILS (fixed)**
50×10	338 sec.	300 sec.
100×10	90 sec.	600 sec.
100×20	-	1200 sec.
200×20	-	2400 sec.

was too high to allow backtracking in reasonable time. Therefore the 100-job instances were run with reoptimization only.

The Iterated Local Search algorithm has been applied to all instances under the same parameter settings. Modifications concerning the number of iterations in the diversification phase did not yield consistently better results, not even for the large instances. Neither did the extension of time limits.

Besides the methods described in Section 3, we applied a priority rule based heuristic as a comparison baseline. The heuristic is based on a *non-delay* scheduling algorithm [6] and it uses the *Apparent Tardiness Cost* (ATC) rule [15] for conflict set resolution.

Table 4 shows the computational results for our benchmark set. The two optimization algorithms have been applied to the problems under tight and loose due dates respectively (cf. Table 2). The due date tightness factors f have been determined experimentally such as to fit for all five instances.

Concerning the running times (cf. Table 3) we first measured the average running time for the SB procedure applied to the 50×10 instances. We then used this value as a base time limit for the ILS algorithm. Anyway, the SB running time for the 100×10 problems is significantly lower. This is due to the absence of the backtracking control structure as explained above.

It can be observed that the Shifting Bottleneck Procedure does not show a constant performance for the tackled problems. Especially the results for the loose due date problems are not consistent, since in many cases they were worse than the ones obtained for tight due dates. As for the 100×10 instances, which were run without backtracking, no improvements at all could be gained over the rule based solution.

The ILS algorithm on the other hand achieves a considerably good solution quality, even for the tight due date case. For some instances the tardiness could even be minimized to zero, which corresponds to a 100 % improvement. However, as for the 200×20 instances with tight due dates, the algorithm seems to reach its limits. Improvements are constantly below 10 % and could not be

Table 4. Computational Results - Relative Percentage Deviation from Baseline

		Tight Due Dates		Loose Due Dates	
Base problem	**Size**	**SB**	**ILS**	**SB**	**ILS**
abz5	50 × 10	21.91 %	-26.69 %	-14.06 %	-55.82 %
	100 × 10	51.11 %	-21.02 %	78.35 %	-68.66 %
	100 × 20	-	-27,39 %	-	-100,00 %
	200 × 20	-	-6,54 %	-	-36,47 %
la19	50 × 10	33.14 %	-28.69 %	62.26 %	-53.71 %
	100 × 10	65.09 %	-13.19 %	85.98 %	-25.08 %
	100 × 20	-	-18,99 %	-	-100,00 %
	200 × 20	-	-4,77 %	-	-62,86 %
ft10	50 × 10	-1.31 %	-39.02 %	-24.95 %	-55.22 %
	100 × 10	70.65 %	-20.99 %	91.36 %	-41.25 %
	100 × 20	-	-22,44 %	-	-34,05 %
	200 × 20	-	-5,08 %	-	-8,42 %
orb10	50 × 10	1.49 %	-37.57 %	-29.67 %	-60.35 %
	100 × 10	42.96 %	-21.31 %	118.56 %	-52.47 %
	100 × 20	-	-22,51 %	-	-63,20 %
	200 × 20	-	-6,01 %	-	-22,14 %
orb03	50 × 10	54.49 %	-21.63 %	63.54 %	-44.02 %
	100 × 10	214.28 %	-18.19 %	396.09 %	-27.62 %
	100 × 20	-	-12,03 %	-	-29,60 %
	200 × 20	-	-3,77 %	-	-8,95 %

significantly increased by parameter tuning nor by providing more computation time. We conjecture that only drastic changes to the algorithm such as different neighborhood structures may improve the performance for this kind of problems.

6 Conclusions and Perspective

We have carried out a performance analysis of two different optimization methods for large-scale tardiness job shops. On the basis of benchmark problem instances which have been created in a semi-random manner for this purpose, we can draw the following main conclusions:

- The modified Shifting Bottleneck Procedure shows unstable and inconsistent behaviour with respect to solution quality
- The performance of the SB procedure seems to strongly depend on backtracking
- The SB procedure works well for small problems. However, the method turned out to be impractical for problems greater than 50 × 10, at least for the problem structure and objective function used in this paper.
- The ILS algorithm obtained significant improvements for problems up to 2000 operations - without any modifications to the basic version.
- Problems involving more than 4000 operations are considerably harder to solve for the ILS method. The neighborhood structure we used seems to be insufficient for such a huge solution space.

Our future research in this area will strongly focus on Iterated Local Search methods and their further enhancement for large-scale problem instances. In essence this will include the development and analysis of new neighborhood concepts for tardiness job shops and their effective embedding into the algorithms.

References

1. J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391–401, 1988.
2. E. J. Anderson and J. C. Nyirenda. Two new rules to minimize tardiness in a job shop. *International Journal of Production Research*, 28(12):2277–2292, 1990.
3. V. A. Armentano and C. R. Scrich. Tabu search for minimizing total tardiness in a job shop. *International Journal of Production Economics*, 63(2):131–140, 2000.
4. R. Braune, M. Affenzeller, and S. Wagner. Efficient heuristic optimization in single machine scheduling. In A. Bruzzone, A. Guasch, M. Piera, and J. Rozenblit, editors, *Proceedings of the International Mediterranean Modelling Multiconference I3M 2006*, pages 499–504. Piera, LogiSim, Barcelona, Spain, 2006.
5. S. French. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop*. Chichester, Horwood et al., 1982.
6. B. Giffler and G. L. Thompson. Algorithms for solving production scheduling problems. *Operations Research*, 8:487–503, 1960.
7. R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Operations Research*, 5:187–326, 1979.
8. S. Kreipl. A large step random walk for minimizing total weighted tardiness in a job shop. *Journal of Scheduling*, 3:125–138, 2000.
9. J. K. Lenstra, A. H. G. Rinnooy Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.
10. H. R. Lourenco. Job shop scheduling: Computational study of local search and large step optimization methods. *European Journal of Operational Research*, 83:347–364, 1995.
11. D. C. Mattfeld and C. Bierwirth. An efficient genetic algorithm for job shop scheduling with tardiness objectives. *European Journal of Operational Research*, 155(3):616–630, 2004.
12. M. Pinedo and M. Singer. A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Naval Research Logistics*, 46:1–17, 1999.
13. B. Roy and B. Sussmann. Les problèmes d'ordonnancement avec contraintes disjonctives. Note D.S. no. 9 bis, SEMA, Paris, France, Décembre 1964.
14. M. Singer and M. Pinedo. A computational study of branch and bound techniques for minimizing the total weighted tardiness in job shops. *IIE Transactions*, 30(2):109–118, 1997.
15. A. P. J. Vepsalainen and T. E. Morton. Priority rules for job shops with weighted tardiness costs. *Management Science*, 33(8):1035–1047, 1987.
16. S. Wagner and M. Affenzeller. Heuristiclab: A generic and extensible optimization environment. In *Proceedings of the 7th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA '05)*, Springer Computer Science, pages 538–541. Springer, 2005.