

EFFICIENT HEURISTIC OPTIMIZATION IN SINGLE MACHINE SCHEDULING

Roland Braune
Michael Affenzeller
Stefan Wagner

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

KEYWORDS

Production Scheduling, Single Machine, Heuristic Optimization, Neighborhood Search, Tabu Search, Genetic Algorithms

ABSTRACT

In this paper, we present a comparison of heuristic optimization algorithms for single machine scheduling problems with and without arbitrary job release times. The main focus of our analysis is the efficiency of the examined methods, i.e. how they perform under strictly limited computation time. Furthermore, we study the effects of arbitrary release times on problem difficulty and required computation time. Experimental results are presented for benchmark problem instances of reasonable size.

INTRODUCTION

The *Single Machine Total Weighted Tardiness Problem* (see e.g. Pinedo, 2002) can be stated as follows: A set of n jobs has to be scheduled on a single machine. The machine can only process one job at a time and the execution of a job cannot be interrupted. Each job j becomes available at time zero, requires a predefined processing time p_j , has a positive weight w_j and a due date d_j . A schedule is constructed by sequencing the jobs in a certain order such that the completion time C_j of each job can be computed. If the completion time exceeds a job's due date, the objective function gets increased by a tardiness penalty $w_j T_j$, where $T_j = \max\{0, C_j - d_j\}$. The optimization goal is to find a processing order which minimizes the value of the sum

$$\sum_{j=1}^n w_j T_j$$

The problem can be written in short form using the three-field representation (Graham et al., 1979): $1||\sum_{j=1}^n w_j T_j$.

By introducing arbitrary release dates r_j for jobs, the problem becomes more general. The starting time of a job not only depends on the completion time of its predecessor in the schedule sequence, but also on its release date. This problem ($1|r_j|\sum_{j=1}^n w_j T_j$) is closer to the real world, but it has not attracted that much attention until now as it was the case for $1||\sum_{j=1}^n w_j T_j$.

Of course, scheduling in industrial environments is not restricted to one machine since the majority of products are manufactured in multiple stages on different machines. However, single machine scheduling provides a basis from which the extension to more complex problems can be made. This is especially the case when considering bottleneck-based scheduling, where a multiple-machine scheduling problem is decomposed into one machine problems which are solved one at a time starting with the bottleneck machine. In such a context, efficient algorithms for single machine scheduling problems are of great importance, in particular when dealing with large-scale problem instances.

PROBLEM COMPLEXITY AND SOLUTION APPROACHES

The problem $1||\sum_{j=1}^n w_j T_j$ is known to be strongly \mathcal{NP} -hard (Lawler, 1977). As far as single machine problems with unequal release times are concerned, only the \mathcal{NP} -hardness of the total tardiness problem has been shown directly (Rinnooy Kan, 1976). However, since $1||\sum_{j=1}^n w_j T_j$ and $1|r_j|\sum_{j=1}^n T_j$ are subproblems of $1|r_j|\sum_{j=1}^n w_j T_j$, the latter can therefore be concluded to be also strongly \mathcal{NP} -hard.

Several enumerative methods have been presented for $1||\sum_{j=1}^n w_j T_j$, such as Branch&Bound algorithms (e.g. Potts and Van Wassenhove, 1985).

Although these exact approaches have been constantly improved, they are not able to solve problem instances with more than 40 jobs until now. The only existing Branch&Bound algorithm for $1|r_j|\sum_{j=1}^n w_j T_j$ is the one introduced by (Akturk and Ozdemir, 2000), which has been applied to problems up to 20 jobs. This indicates that for large, industrial-sized scheduling problems only heuristic optimization techniques are to be considered.

A review and comparison of such methods in the context of $1||\sum_{j=1}^n w_j T_j$ is provided by (Crauwels et al., 1998). Neighborhood-based approaches, in particular Tabu Search, as well as Genetic Algorithms turned out to be effective methods for weighted tardiness optimization. During the past few years, several enhanced variants of these methods have been developed (Congram et al., 2002; Avci et al., 2003; Bilge et al., 2006). Anyway, the applied techniques are very sophisticated and tailored to the specific properties of $1||\sum_{j=1}^n w_j T_j$. On the other hand, literature on heuristic approaches to $1|r_j|\sum_{j=1}^n w_j T_j$ is very sparse. As a consequence, we decided to rely on existing methods for $1||\sum_{j=1}^n w_j T_j$ and to adapt them in order to handle arbitrary release times. We further modified the methods for maximum efficiency at a reasonable level of solution quality.

In the following sections we describe three different methods for the two single machine problems, two neighborhood based ones - a descent method and a Tabu Search algorithm - and a Genetic Algorithm.

NEIGHBORHOOD-BASED SEARCH

Neighborhood search, which is often also referred to as local search, is a widely used paradigm in heuristic optimization. It describes a whole class of methods sharing the same basic principle but using different mechanisms to control the search.

Starting from an initial solution, which is constructed randomly or heuristically, a neighborhood or local search algorithm traverses a sequence of steps in the solution space of a given problem. In each step, it performs a transition from a current solution s to a new solution s' , which is located in the "neighborhood" of s . The term "neighbor" means that the characteristics of s and s' differ only slightly from each other. The actual decision which of all possible neighboring solutions to visit next and the termination criterion depend on the search guidance of the respective method.

The central issue in neighborhood search is the definition of the neighborhood itself. In the scope of this paper, we use two different neighborhood mod-

els, assuming that a solution is represented by a permutation (sequence) of jobs:

- The *Non-Adjacent Pairwise Interchange* (NAPI) neighborhood:
Given a sequence (permutation) of jobs of length n , a neighbor is generated by interchanging two arbitrary jobs within the sequence.
- The *Adjacent Pairwise Interchange* (API) neighborhood:
The API neighborhood is a subset of the NAPI neighborhood. It only considers interchanges of directly adjacent jobs within a sequence.

In the context of machine scheduling, the initial solution is usually generated using a simple *priority dispatch rule* (PDR) (Morton and Pentico, 1993) heuristic. For our tests we used the Apparent Tardiness Cost (ATC) rule which determines a job's priority index π_j as follows:

$$\pi_j = (w_j/p_j)e^{\frac{-\max(0, d_j - t - p_j)}{k\bar{p}}}$$

where k is a constant (set at 2 for our experiments), t denotes the earliest starting time and \bar{p} the average processing time of all considered unscheduled jobs.

In the following, we introduce two different neighborhood search methods which we modified for application to $1||\sum_{j=1}^n w_j T_j$ and $1|r_j|\sum_{j=1}^n w_j T_j$.

Descent

A descent algorithm, often also referred to as a "hill-climber", is the simplest form of neighborhood search and can be considered as a greedy algorithm. It accepts a new solution only if it improves the objective function value. The algorithm terminates if no improving solution can be found in some state.

For our tests, we use a modified hill-climber which also allows neutral transitions with respect to the solution quality. The termination criterion is satisfied if neither an improving nor an equivalent solution can be found or if a maximum number of neutral transitions has been reached. The NAPI neighborhood model is applied in the following manner: We limit the neighborhood size and randomly determine the pairs of positions to be swapped.

Tabu Search

Tabu Search (TS) (Glover and Laguna, 1997) is a neighborhood search method designed to explore the solution space beyond local optima. Similar to the simple descent algorithm, TS performs a walk through the solution space by iteratively switching

over to a new neighboring solution. In the context of Tabu Search, each such transition is called a *move*. TS always performs the move leading to the *best* solution in the current neighborhood, which may also be worse than the current one. In order to prevent cycling and becoming trapped in local optima, TS uses a short-term memory, called *tabu-list*. The tabu-list records attributes of recently visited solutions. Moves which lead back to such a solution are set *tabu*, hence forbidden, for a certain time period. The size of the tabu-list, called the *tabu-tenure*, determines how long such moves are forbidden. Tabu Search algorithms are usually enhanced by *diversification* and *intensification* strategies in order to explore unvisited regions of the solution space or to intensify the search in promising, already examined areas.

For the single machine scheduling problems we employ a TS algorithm which is mainly based on (Crauwels et al., 1998). However, we extended the base algorithm by a diversification strategy which works as follows: After a predefined number of neutral moves, we penalize moves based on their objective function's frequency of occurrence:

$$f'(m) = f(m) + p * f(m) * H(f(m))$$

where $f(m)$ denotes the objective function value after performing the move and thus the move evaluation, $f'(m)$ the penalized move evaluation, p a penalty factor and $H(x)$ the number of times the given value has been encountered during the current diversification phase. The diversification phase lasts for a given duration after which the collected frequency information is discarded.

GENETIC ALGORITHMS

A Genetic Algorithm (GA) (Holland, 1975) is an optimization method which is inspired by nature, precisely speaking, by the process of biological evolution. Its three main mechanisms, *selection*, *recombination (crossover)* and *mutation* are applied iteratively to a set of solutions, called the *population*. During each iteration step, also called generation, a new population of solutions is generated by applying a *crossover operator*. This operator takes two solutions and produces one or more new solutions by recombining their elements in a certain predefined way. With a given probability, the resulting solutions may be subject to a subsequent mutation step, which slightly modifies solution components in a random manner. The probability of a solution to be chosen for crossover depends on its *fitness value* which is strongly related to the solution's quality with respect to the objective function: A higher fitness value yields a higher selection probability.

The Genetic Algorithm we use for our experiments can be described as a Standard Genetic Algorithm which has been modified in order to fit better into the problem environment. The most important aspects concerning our Genetic Algorithm are summarized in the following:

- *Solution encoding*
We adopt a permutation based representation for the sake of compatibility with the neighborhood search methods. Additionally, it is the most natural encoding for sequencing and scheduling problems.
- *Crossover and mutation*
We use the Order Crossover (OX) as defined by Syswerda (Syswerda, 1991) and an insert-based mutation operator.
- *Initialization*
The initial population is basically initialized at random. However, we additionally insert one solution generated by the ATC heuristic. By this, we possibly introduce relevant building blocks for high quality solutions into the population and it is further guaranteed that the final result is never worse than the ATC solution.
- *Hybridization*
In order to further improve solution quality, we hybridize our Genetic Algorithm with local search. A strict best-improvement descent method based on the API neighborhood is used to reoptimize solutions after the crossover.

EXPERIMENTS

In this section, we present computational results for the two types of single machine scheduling problems considered in this paper. Since efficiency is our main concern, we impose strict time limits for all optimization runs and analyze the resulting solution quality. All algorithms have been implemented in C# (.NET 2.0) using the HeuristicLab framework (Wagner and Affenzeller, 2005) and are executed on a 2.6 GHz Pentium-4 with 1024 MB RAM.

Equal (Zero) Release Times

Our experiments regarding $1 || \sum_{j=1}^n w_j T_j$ are based on a set of well known benchmark problems taken from the OR-library (Beasley, 1990). Unfortunately, the OR-library instances are limited to 100 jobs. However, 200-job problems generated according to the same scheme have been used in (Avci et al., 2003). The authors kindly provided us with problem data and best found solutions.

Table 1: Parameters for the neighborhood methods

	$n = 100$		$n = 200$	
	DN _{1/5}	TS	DN _{1/5}	TS
Neighborhood Size	$4n, 2n$	$4n$	$4n, 2n$	$4n$
Tabu Tenure	-	60	-	120
<i>Diversification:</i>				
Max neutral moves	-	2n	-	2n
Duration	-	n/2	-	n/2

Table 2: Parameters for the Genetic Algorithm

	n (100 or 200)
Population Size	Linear Rank
Selection	OX
Crossover	5 %
Mutation Rate	
<i>Hybridization:</i>	
% improved by local search	25 %

Tables 3 and 4 summarize the experimental results. The Δ -values represent the relative percentage deviation from the best known solutions. The last column displays the number of optimal solutions found by each method. All tests were run with a time limit of 4 seconds for $n = 100$ and 10 seconds for $n = 200$ and the results are averaged over 10 independent runs. The descent method (DN) was also run in a multi-start version (DN₅), which means that the minimum out of 5 very short runs (4/5 seconds each) is considered as the result of a the whole 4 second run. The parameters for each method are summarized in Table 1 and 2.

For comparison purposes, we added results achieved by the *Problem Space Genetic Algorithm* (PSGA), one of the best available approaches for single machine scheduling problems, as reported in (Avci et al., 2003). It has to be remarked that these results were obtained under different conditions (no time restrictions).

According to Table 3, the Genetic Algorithm produces the most stable and robust results for $n = 100$. The TS algorithm produces a high Δ_{max} value, although its average percentage deviation is still relatively low. The GA cannot maintain its result quality for the 200-job problems but still obtains the lowest Δ_{max} value. The TS algorithm produces the lowest mean deviation. However, it had severe problems with one of the problem instances yielding a Δ_{max} of 10.05 %. In general, the GA seems to produce the most stable overall result quality. Its population-based exploration of the solution space is obviously less prone to be trapped in local optima as is the case for the neighborhood-based methods. It should also be remarked that the multi-start hill-climber (DN₅) shows a still acceptable level of solution quality on average, although it is rather unstable.

Table 3: Results for the 100-job instances

Method	Δ_{avg}	Δ_{median}	Δ_{max}	Opt.
DN ₁	2.28 %	0.21 %	220.74 %	27
DN ₅	0.38 %	0.07 %	16.06 %	41
GA	0.04 %	0.00 %	0.61 %	64
TS	0.13 %	0.00 %	4.01 %	64
PSGA	0.02 %	-	0.30 %	83

Table 4: Results for the 200-job instances

Method	Δ_{avg}	Δ_{median}	Δ_{max}	Opt.
DN ₁	0.67 %	0.14 %	25.49 %	23
DN ₅	0.49 %	0.06 %	16.99 %	28
GA	0.31 %	0.06 %	4.02 %	33
TS	0.22 %	0.01 %	10.05 %	30
PSGA	0.07 %	-	0.59 %	48

Arbitrary Release Times

Since there are no publicly available benchmark instances for $1|r_j|\sum_{j=1}^n w_j T_j$, we generated a set of problems according to the scheme reported in (Akturk and Ozdemir, 2000): Processing times p_j and weights w_j were randomly sampled from the uniform distributions $[1,10]$ and $[1,100]$. The release times r_j follow a uniform distribution between 0 and $\alpha \sum_{j=1}^n p_j$, where $\alpha \in \{0.0, 0.5, 1.0, 1.5\}$. Due dates were computed using a randomly determined slack time $d_j - (r_j + p_j)$ ranging from 0 to $\beta \sum_{j=1}^n p_j$, where $\beta \in \{0.05, 0.25, 0.5\}$. Using all possible combinations of processing time and weight ranges and all possible pairs of values for α and β , we obtained 48 different problem instances for each problem size.

We cannot directly compare our results to existing best known solutions, hence we report the improvement (in percent) over solutions generated by the ATC rule. Tables 5 and 6 show the average improvement for each pair of α and β . Due to our first experiments, unequal release times add a considerable degree of difficulty, hence we relaxed the time limits to 5 seconds for $n = 100$ and 15 seconds for $n = 200$. The experimental setup is basically the same as for the equal release time problems. The parameters for the GA were not changed since we did not observe quality improvements. Only the neighborhood sizes for the descent and TS algorithms were doubled.

It can be observed that the highest improvement ratios are obtained for the problems with scattered release times and loose due dates. However, only minor improvements are possible for instances with $\alpha = 0$ which correspond to the weighted tardiness problem with equal release dates ($1||\sum_{j=1}^n w_j T_j$). Tabu Search shows the best overall behaviour for both $n = 100$ and $n = 200$, whereas the Genetic Algorithm cannot keep up with the neighborhood-based methods for $\alpha > 0$. Obviously the structure of the solution space becomes different under arbitrary

Table 5: Results for the 100-job instances with unequal release times

α	β	ATC	DN		GA		TS	
		$\sum_{j=1}^n w_j T_j$	$\sum_{j=1}^n w_j T_j$	Improv. %	$\sum_{j=1}^n w_j T_j$	Improv. %	$\sum_{j=1}^n w_j T_j$	Improv. %
0.0	0.05	1686694	1683650	0.18 %	1682922	0.22 %	1682924	0.22 %
0.0	0.25	1448031	1433587	1.00 %	1433105	1.03 %	1433097	1.03 %
0.0	0.50	626065	617563	1.36 %	616795	1.48 %	616740	1.49 %
0.5	0.05	575083	572244	0.49 %	572459	0.46 %	557661	3.03 %
0.5	0.25	378631	287428	24.09 %	350793	7.35 %	274170	27.59 %
0.5	0.50	66112	57906	12.41 %	60628	8.30 %	56968	13.83 %
1.0	0.05	41629	38564	7.36 %	39841	4.29 %	31479	24.38 %
1.0	0.25	1982	645	67.45 %	906	54.31 %	657	66.83 %
1.0	0.50	59	0	100.00 %	0	100.00 %	0	100.00 %
1.5	0.05	12539	9488	24.33 %	10075	19.65 %	9359	25.36 %
1.5	0.25	854	3	99.71 %	3	99.71 %	3	99.71 %
1.5	0.50	280	255	8.77 %	0	100.00 %	0	100.00 %
Total		403163	391778	2.82 %	397294	1.46 %	388588	3.62 %

Table 6: Results for the 200-job instances with unequal release times

α	β	ATC	DN		GA		TS	
		$\sum_{j=1}^n w_j T_j$	$\sum_{j=1}^n w_j T_j$	Improv. %	$\sum_{j=1}^n w_j T_j$	Improv. %	$\sum_{j=1}^n w_j T_j$	Improv. %
0.0	0.05	6843055	6836874	0.09 %	6835082	0.12 %	6835106	0.12 %
0.0	0.25	5139569	5128546	0.21 %	5126121	0.26 %	5125537	0.27 %
0.0	0.50	2816649	2789396	0.97 %	2789613	0.96 %	2787576	1.03 %
0.5	0.05	2414790	2403980	0.45 %	2410618	0.17 %	2387096	1.15 %
0.5	0.25	1209013	940952	22.17 %	1195804	1.09 %	932230	22.89 %
0.5	0.50	408558	216130	47.10 %	402520	1.48 %	216617	46.98 %
1.0	0.05	28725	22019	23.35 %	28199	1.83 %	21223	26.12 %
1.0	0.25	75	0	100.00 %	16	79.34 %	0	100.00 %
1.0	0.50	640	0	100.00 %	27	95.83 %	0	100.00 %
1.5	0.05	7721	1262	83.66 %	2993	61.23 %	1475	80.90 %
1.5	0.25	622	30	95.18 %	43	93.14 %	30	95.18 %
1.5	0.50	91	0	100.00 %	0	100.00 %	0	100.00 %
Total		1572459	1528266	2.81 %	1565920	0.42 %	1525574	2.98 %

trary release dates. The only small advantage of the TS algorithm over the hill climber emphasizes this assumption.

CONCLUSION AND OUTLOOK

We presented efficient heuristic optimization methods for single machine scheduling problems with and without release times and validated them against common benchmark instances and self-generated problem suites. Despite the strictly limited computation time, the algorithms were able to produce high quality results, especially for $1 || \sum_{j=1}^n w_j T_j$. However, we observed that unequal release times strongly contribute to problem difficulty and obviously change the structure of the solution space. It can be concluded that experience concerning approaches, which perform well for $1 || \sum_{j=1}^n w_j T_j$, may not directly be transferred to $1 |r_j| \sum_{j=1}^n w_j T_j$. Anyway, the achieved results for unequal release times are satisfactory and represent a significant improvement compared to priority rule based heuristics.

Future research will be directed towards further analysis of the specific properties of $1 |r_j| \sum_{j=1}^n w_j T_j$ and the topology of its solution space. Based on the good performance of neighborhood search methods for this kind of problem, we plan to examine further methods in this context, e.g. iterated local search algorithms, with respect to their applicability and efficiency.

REFERENCES

- Akturk, M. S. and Ozdemir, D. (2000). An exact approach to minimizing total weighted tardiness with release dates. *IIE Transactions*, 32:1091–1101.
- Avcı, S., Akturk, M. S., and Storer, R. H. (2003). A problem space algorithm for single machine weighted tardiness problems. *IIE Transactions*, 35:479–486.
- Beasley, J. (1990). Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

- Bilge, U., Kurtulan, M., and Kirac, F. (2006). A tabu search algorithm for the single machine total weighted tardiness problem. *European Journal of Operational Research*. In Press.
- Congram, R. K., Potts, C. N., and Van de Velde, S. (2002). An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67.
- Crauwels, H. A. J., Potts, C. N., and Van Wassenhove, L. N. (1998). Local search heuristics for the single machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 10(3):341–350.
- Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers.
- Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Operations Research*, 5:187–326.
- Holland, J. H. (1975). *Adaption in Natural and Artificial Systems*. University of Michigan Press.
- Lawler, E. L. (1977). A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Annals of Operations Research*, 1:331–342.
- Morton, T. E. and Pentico, D. W. (1993). *Heuristic Scheduling Systems with Applications to Production Systems and Project Management*. Wiley, New York, NY.
- Pinedo, M. (2002). *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 2nd edition.
- Potts, C. N. and Van Wassenhove, L. N. (1985). A branch and bound algorithm for the total weighted tardiness problem. *Operations Research*, 33(2):363–377.
- Rinnooy Kan, A. H. G. (1976). *Machine Scheduling Problems: Classification, complexity and computations*. Nijhoff, The Hague.
- Syswerda, G. (1991). Schedule optimization using genetic algorithms. In Davis, L., editor, *Handbook of Genetic Algorithms*, pages 332–349. Van Nostrand Reinhold, New York.
- Wagner, S. and Affenzeller, M. (2005). Heurist-clab: 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.

ACKNOWLEDGEMENTS

The research results presented in this paper have been achieved in the context of a project called "Production Planning Optimization", which is supported by the Upper Austrian Mechatronics Cluster together with four local manufacturing companies.

AUTHOR BIOGRAPHIES

ROLAND BRAUNE obtained his MSc in Computer Science at the Johannes Kepler University, Linz, Austria in 2004. Starting with his diploma thesis ("Optimization of Production Planning in a Manufacturing Company"), he has been concerned with research in the field of heuristic optimization for real-world production scheduling problems. He is currently working as a research associate at the Research & Development Competence Center of the Upper Austrian University of Applied Sciences, Hagenberg, Austria. His e-mail address is: roland@heuristiclab.com and his Web-page can be found at <http://www.heuristiclab.com/contact/braune.html>.

MICHAEL AFFENZELLER has published several papers and journal articles dealing with theoretical aspects of Genetic Algorithms and Evolutionary Computation in general. In 1997 he received his MSc in Industrial Mathematics and in 2001 his PhD in Computer Science, both from the Johannes Kepler University, Linz, Austria. He is professor at the Upper Austrian University of Applied Sciences, Hagenberg, Austria and associate professor at the Institute of Formal Models and Verification at Johannes Kepler University, Linz, Austria since his habilitation in "Applied Systems Sciences with particular regard to Heuristic Optimization" in 2004. His e-mail address is: michael@heuristiclab.com and his Web-page can be found at <http://www.heuristiclab.com/contact/affenzeller.html>.

STEFAN WAGNER also received his MSc in Computer Science from Johannes Kepler University Linz, Austria in 2004 (title of his diploma thesis: "Looking Inside Genetic Algorithms"). He now holds the position of an associate professor at the Upper Austrian University of Applied Sciences, Hagenberg, Austria. His research interests include Evolutionary Computation and Heuristic Optimization, Theory and Application of Genetic Algorithms, Machine Learning and Software Development. His e-mail address is: stefan@heuristiclab.com and his Web-page can be found at <http://www.heuristiclab.com/contact/wagner.html>.