A Case Study on Earliness/Tardiness Scheduling by Constraint Programming

Jan Kelbel, Zdeněk Hanzálek

Department of Control Engineering, Faculty of Electrical Engineering
Czech Technical University in Prague
kelbelj@fel.cvut.cz, hanzalek@fel.cvut.cz

Abstract. The aim of this paper is to solve a problem on scheduling with earliness and tardiness costs using constraint programming approach, and to compare the results with the ones from the original timed automata approach. The case study, a production of lacquers, includes some real life features like operating hours or changeover times. Considering the earliness and tardiness costs of the case study, the problem was converted to the problem with objective of minimizing the total waiting time. Then a search procedure was applied.

Keywords: scheduling, constraint programming, earliness and tardiness costs

1 Introduction

This paper deals with an application of constraint-based scheduling on a scheduling problem with earliness and tardiness costs. The case study, a production of lacquers, is originally from project AMETIST [1], where it was successfully solved using timed automata approach [2]. The primary motivation of this work is to compare the application of constraint programming with the timed automata solution (that was better than a MIP approach [1]).

The problem can be classified as the resource-constrained project scheduling problem, in structure similar to the job shop scheduling problem, with distinct due dates and release dates, and with job dependent earliness and tardiness costs. The scheduling problem with earliness and tardiness costs (E/T) and given due dates is NP-hard already in one-machine version [3]. There are OR methods specialized to solve some restricted E/T problems, e.g. [4]. More close to our problem is the E/T job shop scheduling problem in [5], where hybrid CP/MIP approach was used. In contrast, pure CP techniques are described in this paper.

Our method to solve the case study results from its feature that earliness costs are almost 50 times smaller than tardiness costs. The problem was converted to the problem with objective of minimizing the total waiting time subject to release times and due dates. Then a variation of time-directed labeling procedure was applied.

The paper [2] introduces three versions of the lacquer production problem. Two of them, a *basic case study* and an *extended case study*, are in focus of our

work. A *stochastic case study*, which deals with unavailability of resources due to breakdown or maintenance, is described in detail in [6].

The paper is organized as follows. The scheduling problem is described in Sect. 2, while Sect. 3 deals with the some aspects of constraint model and describes our approach to the search procedure. Results of experiments are presented in Sect. 4.

As a constraint programming environment we chose ILOG Solver and Scheduler via ILOG OPL Studio.

2 The Problem Statement

The production of a lacquer is described by a recipe. The recipe defines processing steps, called tasks according to the scheduling theory, to be performed. The definition includes processing times of the tasks, resources required for each task, and precedence constraints. Some of the precedences have delay constraints, i.e. maximal and minimal delays (time lags) between tasks are specified.

The case study contains 29 jobs, i.e. orders of lacquer to be produced. The job is specified by the recipe, quantity, release date (earliest start time) and due date. Processing times of tasks are dependent on the quantity of a lacquer. Three different recipes are included in the problem – for lacquer types uni, metallic and bronze.

The tasks of one recipe form a sequence, where each task requires one resource. Further, there is a special resource G_1 or G_2 (mixing vessels) that is needed for nearly the whole time of production of the job. In order to avoid multiprocessor tasks, a shadow task is created to model processing on the special resource. The recipes are depicted in Fig. 1.

Case Studies

The paper introduces three instances of the two case studies. The goal of the *basic case study* (denoted as BF) is to find any feasible schedule using simplified model of the production. Each task of the recipe has processing time that is the same for all orders, and all resources are available continuously.

The *extended case study* (EO) is the cost optimization problem. Objective function introduced in [2] is a combination of total weighted earliness (a cost for storage of orders that are finished too early), total weighted tardiness (a penalty payment for delayed orders), and changeover cost. Due to the complexity of this objective function evaluation, a simpler objective function of total earliness of all jobs while tardy jobs are not allowed was introduced:

$$F = \sum_{j \in \mathcal{J}} d_j - C_j, \tag{1}$$

where $\mathcal{J}$ is set of jobs, d_j is due date and C_j is completion time of job j, and $d_j \geq C_j \quad \forall j \in \mathcal{J}$. The value of the production total cost, which is needed for comparison with other approaches, is computed subsequently from the results

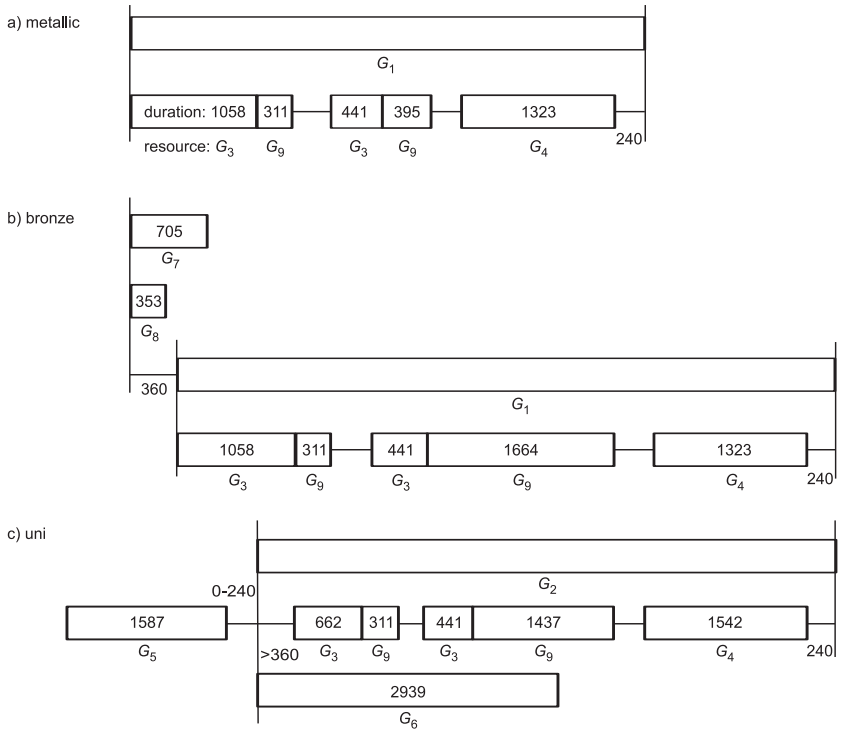

Fig. 1. The lacquer recipes.

of the optimization. Next, some resources have more exact model of behaviour concerning changeover times and costs. Operating hours of resources have an exact model with breaks. Tasks cannot be scheduled during breaks, and since some tasks have processing time longer than the available time between breaks, these task are breakable, i.e. allowed to be interrupted by breaks.

The third case study instance is the *extended case study* with *performance factors* (EOP), where the performance factors model the unavailability of resources due to breakdown or maintenance [2]. The performance factor is assigned to each resource, and is used to extend the processing time of a task requiring the resource.

Resources

The resources are grouped according to their types. Some groups contain more than one machine, i.e. there are more machines of that type. Inside the group, the machines are parallel identical resources in terms of the scheduling theory [7], or in terms of constraint-based scheduling [8], the groups are discrete resources with capacity greater than one.

Groups with parallel identical resources are:

- Mixing vessel metal $G_1 = \{R_{11}, R_{12}, R_{13}\}$ is used in production of metallic and bronze lacquers.
- Mixing vessel uni $G_2 = \{R_{21}, R_{22}\}$ is used for production of uni lacquers.
- Dose spinner $G_3 = \{R_{31}, R_{32}\}$ is used in all recipes.
- Filling station $G_4 = \{R_{41}, R_{42}\}$ is used in all recipes. These are resources with changeover time and cost – cleaning is needed when two successive jobs are of different lacquer type.

Dedicated resources are:

- Disperser $G_5 = \{R_5\}$ is used in uni lacquer recipe.
- Dispersing line $G_6 = \{R_6\}$ is used in uni lacquer recipe.
- Bronze mixer $G_7 = \{R_7\}$ is used in bronze lacquer recipe.
- Bronze dose spinner $G_8 = \{R_8\}$ is used in bronze lacquer recipe.

Finally, there is unrestricted resource laboratory G_9.

3 Modelling the Problem

With naive application of constraint programming, only solutions for smaller instances of the problem were found. So the state space of the problem was reduced, and a search heuristic was applied, as it is described in following paragraphs.

Constraints in the model

Heuristic constraint called *non-overtaking* [2] is used in order to reduce the search space of the problem. It ensures that the job started earlier will be finished earlier. We used even more constraining version of this heuristic. Tasks of a job with earlier due date are constrained to start earlier. Jobs of the same lacquer recipe of the type r are indexed with $i \in \{1, \ldots, n_r\}$ according to the increasing due date. $\mathcal{T}_r$ is the set of indices of tasks in recipe of the type r, and $S_{i,k}$ represents starting time of the task k of the job i. Then the constraints are declared as follows:

$$S_{i,k} \leq S_{i+1,k} \ , \quad \forall i \in \{1, \ldots, n_r - 1\} \text{ and } \forall k \in \mathcal{T}_r.$$

Applying this heuristic on the model with constant processing times of tasks will remove symmetry from the search space. However, this heuristic was used also in the *extended case study* with task processing times dependent on the quantity of lacquer, leading to loss of some solutions.

Concerning the filtering algorithms on resources, the Edge-Finding algorithm [8] was used on unary and also on discrete resources. Due to the changeover times, the discrete resource G_4 was modelled as a set of alternative unary resources.

Search Procedure

The search procedures that are available in OPL studio are written to minimize C_{max}. For example, the *SetTimes* search heuristic [9, 10] builds the schedule from earliest starting dates, and also default labeling procedure starts from the beginning of the domains. The objective of our problem is to schedule tasks as late as possible, but still not to exceed due date. The idea of our search procedure is to try to build the schedule backwards. At first, we try to place tasks as near as possible to the due dates.

One alternative of implementation of this approach is to create a new search procedure. The other one is that the time axis can be reversed and standard search procedures used, i.e. to convert the problem to the problem with objective of minimizing the total waiting time subject to release times and due dates.

Because *SetTimes* sometimes missed all solutions , we used a search heuristic, which is a simple version of time-directed labeling procedure [11]:

1. sort final tasks of jobs by latest possible completion time in non-increasing order
2. for each task state two alternatives in the search tree: assign the latest possible completion time as the end time of the task, or decrease the latest possible completion time of the task.

Finally, the search heuristic was modified to be used with reversed time axis model, which was a chosen alternative.

The search heuristic defines the shape of the search tree. It was explored using two search strategies – Depth First Search (DFS) and Limited Discrepancy Search (LDS, denoted as Slice-Based Search in OPL).

4 Experimental Results

The three case study instances introduced in Sect. 2 were the subjects of the experiments. The problem instance stated in [2] can be simplified without influence on the result. Task requiring unrestricted resource laboratory can be replaced by a delay constraint, and some tasks can be put together. Finally, we end up with 139 tasks (of which 110 are breakable) in 29 jobs.

Time for the whole schedule is 9 weeks in 1 minute resolution (90720 minutes), but the time available for each job is about 2 weeks from release date to due date, that makes the search space smaller. Moreover, due dates of the jobs are almost evenly distributed from 3^{rd} to 9^{th} week.

The results are presented in Tab. 1, where column headers DFS-NO and LDS-NO determine the used search strategy with the *non-overtaking* constraint in the model, while LDS column corresponds to the model without the *non-overtaking* constraint. The "timed automata" column corresponds to results from [2].

The constraint programming experiments were run on Intel P4 3GHz CPU with 1GB of RAM, using OPL Studio 3.6 under Windows XP, timed automata experiments were run on Intel P4 Xeon 2.6GHz [2].

Table 1. Results for test instances

	DFS-NO		LDS-NO		LDS		timed automata	
	cost	CPU	cost	CPU	cost	CPU	cost	CPU
BF	–	0.09 s	–	0.09 s	–	–	–	0.45 s
EO	455,758	1374 s	455,758	18.4 s	457,523	181 s	*	*
EOP	1,234,959	1026 s	1,186,851	12.7 s	920,240	220 s	≈ 2,100,000	≈ 600 s

– . . . value not applicable
* . . . value not available

Comparing the results in columns DFS-NO and LDS-NO, wee can see big speed up in computation, i.e. the Limited Discrepancy Search helped our search procedure a lot.

Concerning the impact of *non-overtaking* constraint, it leads to more or less significantly better results, but at the high expense of CPU time consumed by optimization.

5 Conclusion and Future Work

We showed a solution of a case study on resource-constrained scheduling problem with earliness/tardiness costs. With constraint programming approach, we found a solution of the EOP problem that is notably better than the solution found with timed automata. However, we tested our method only on three instances of the problem. There is also an expectation, that the 29 job instances are maybe ones of the largest solvable in reasonable time. The plan for the future is at first to test the method on randomly generated data and on modified standard job shop benchmarks. Then our method could be improved, possibly by using approaches to solve similar problems.

References

1. AMETIST: European Community Project IST-2001-35304 (Advanced Methods for Timed Systems). http://ametist.cs.utwente.nl/ (2002)
2. Behrmann, G., Brinksma, E., Hendriks, M., Mader, A.: Production Scheduling by Reachability Analysis - A Case Study. In: Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), published by IEEE Computer Society Press, Los Alamitos, California (2005)
3. Baker, K.R., Scudder, G.D.: Sequencing with earliness and tardiness penalties: A review. Operations Research **38**(1) (1990) 22–36
4. Sourd, F., Kedad-Sidhoum, S.: The one machine scheduling with earliness and tardiness penalties. Journal of Scheduling **6**(6) (2003) 533–549
5. Beck, J.C., Refalo, P.: A Hybrid Approach to Scheduling with Earliness and Tardiness Costs. Annals of Operations Research **118**(1–4) (2003) 49–71
6. Bohnenkamp, H.C., Hermanns, H., Klaren, R., Mader, A., Usenko, Y.S.: Synthesis and stochastic assessment of schedules for lacquer production. In: 1st Int. Conf. on

Quantitative Evaluation of Systems (QEST), published by IEEE Computer Society Press, Los Alamitos, California (2004)
7. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Werglarz, J.: Scheduling Computer and Manufacturing Processes. Second edn. Springer-Verlag, Berlin (2001)
8. Baptiste, P., Le Pape, C., Nuijten, W.: Constraint-Based scheduling: Applying Constraint Programming to Scheduling Problems. Kluwer Academic Publishers (2001)
9. Le Pape, C., Couronne, P., Vergamini, D., Gosselin, V.: Time-versus-Capacity Compromises in Project Scheduling. In: Proceedings of the Thirteenth Workshop of the U.K. Planning Special Interest Group. (1994)
10. ILOG S.A.: ILOG OPL Studio 3.5 Language Manual. (2001)
11. van Hentenryck, P., Perron, L., Puget, J.F.: Search and strategies in OPL. ACM Transactions on Computational Logic **1**(2) (2000) 285–320