Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 50-1 (2017) 12490–12495

An Efficient Algorithm for Scheduling a Flexible Job Shop with Blocking and No-Wait Constraints ⋆

A. Aschauer, F. Roetzer, A. Steinboeck, A. Kugi

Automation and Control Institute, Technische Universität Wien, Gusshausstraße 27-29, 1040 Vienna, Austria ({aschauer, roetzer, steinboeck, kugi}@acin.tuwien.ac.at)

Abstract: Optimal scheduling in industrial processes is crucial to ensure highest throughput rates and low costs. This paper presents the implementation of a scheduling algorithm in a hot rolling mill, which features several reheating furnaces and which is characterized by bidirectional material flow, blocking, and no-wait constraints. The scheduling problem is solved by a decomposition into a timetabling algorithm and a sequence optimization procedure. For the timetabling task, where the sequence of products is assumed to be fixed, a new recursive algorithm to generate a non-delay feasible schedule is developed. The sequence optimization procedure searches for the optimum product sequence and makes heavy use of the timetabling algorithm. A competitive starting sequence is generated by a construction heuristic and iteratively improved by a tabu search algorithm.

Keywords: scheduling, timetabling, flexible job shop, parallel machines, blocking, no-wait constraints, construction heuristic, tabu search, hot rolling mill, metal processing

1. INTRODUCTION

Hot rolling is a main cost driver in the production of metal plates. Here, the reheating is highly time and energy consuming. If larger series of equal products are produced, it is straightforward to schedule the reheating and rolling process. However, large series are becoming rare. The increasing diversification of product portfolios, individualization of products, lean inventory, and just-in-time production add to the complexity of the scheduling task. This is why an optimal scheduling algorithm is required. It should ensure best possible utilization of the available production facilities and highest throughput rates. In the considered plant, mainly molybdenum products are processed. This valuable metal requires an accurate scheduling algorithm to ensure that the material is rolled at the right time and with the right temperature.

1.1 Problem Description

Figure 1 shows an outline of the considered plant. The manipulator M conveys the raw material (slabs) from the charging station to one of the four furnaces F1-F4 for reheating. After reheating, the manipulator moves the product to the roller table with the reversing mill stand R, where it is rolled in several rolling passes. Many products require intermediate reheating steps or final annealing in one of the furnaces. For this purpose, the products have to be handled by the manipulator again. If furnaces are operated at the same temperature, they can be considered as identical. For every product, a production plan specifies

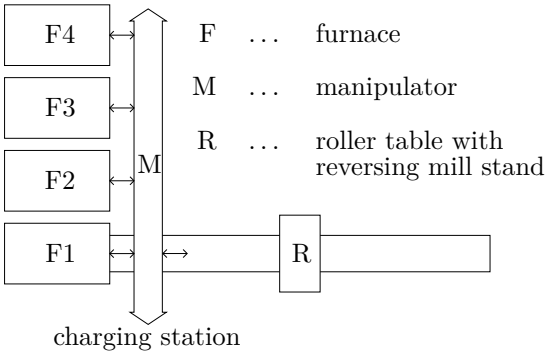

Fig. 1. Outline of the hot rolling mill.

the sequence of heating and rolling steps, the heating temperatures, and the process times or at least their lower and upper bounds.

Henceforth, a product is called a job and the single processing and manipulation steps are called tasks. For every job, the predefined sequence of tasks must be adhered to. Moreover, waiting times between tasks are not allowed due to the absence of space for the products to wait (no buffer) and the product quality would suffer (temperature decrease due to cooling and oxidation of the product surface). The processing time for heating tasks has a lower and an upper bound. If a heating task is scheduled to be longer than the lower bound time, the respective furnace is blocked longer than required, which may reduce the throughput rate. Manipulation and rolling tasks have a fixed duration, which should not be changed to avoid wrong rolling temperatures. The furnaces, the manipulator, and the roller table with the reversing mill stand are called machines. The furnaces are operated at constant temperatures, which are specified in the production plan. Furnaces with identical temperatures are called parallel machines. Each machine can only handle one task at a time

⋆ Great thanks are addressed to the industrial research partner Plansee SE supporting this work. Moreover, financial support from the EU project Power Semiconductor and Electronics Manufacturing 4.0 (SemI40), under grant agreement No 692466, is gratefully acknowledged. The project is co-funded by grants from Austria, Germany, Italy, France, Portugal, and - Electronic Component Systems for European Leadership Joint Undertaking (ECSEL JU).

10.1016/j.ifacol.2017.08.2056

and each job can only be processed by one machine at a time. The objective of the desired scheduling algorithm is the minimization of the makespan (total processing time) of a given number of jobs. In this minimization procedure, all constraints have to be satisfied.

1.2 Literature Review

The above problem is known in the literature as flexible job shop scheduling problem (FJSP) with blocking and no-wait constraints. The complexity of the task considered in this paper originates in the existence of parallel machines. For the classical job shop scheduling problem (JSP) with unlimited buffers between consecutive tasks, there exists a rich literature, e. g., (Pinedo, 2016; Błażewicz et al., 2001). A common way to represent this scheduling problem is a disjunctive graph. Pinedo (2016) described an exact branch-and-bound algorithm for a JSP with two machines. For larger problems, he recommends the so-called shifting bottleneck heuristic (Pinedo, 2016). There are also some studies on FJSPs, where certain tasks can be performed by several machines. Brucker and Schlie (1991) were among the first who addressed this problem. The complexity of a FJSP roots in the fact that additionally an assignment of tasks to machines is necessary. Important works have been made by Paulli (1995) and Dauzère-Pérès and Paulli (1997). Paulli (1995) presented a hierarchical approach which first assigns tasks to machines and second solves the classical JSP. Starting from this initial solution, this procedure is iteratively repeated. That is, tasks are reassigned and the classical JSP is solved again. Dauzère-Pérès and Paulli (1997) described an integrated approach of assignment and scheduling based on the help of an extended disjunctive graph.

The literature on FJSPs with blocking and no-wait constraints is rather scarce. Hall and Sriskandarajah (1996) and Allahverdi (2016) gave surveys of scheduling methods with no-wait constraints. Most works in this field consider flow shop scheduling problems. Nevertheless, some works on JSPs are mentioned. Many successful approaches handle the JSP with no-wait constraints by decomposing the problem into a timetabling and a sequencing part. Schuster and Framinan (2003) achieved excellent results with a non-delay timetabling algorithm and a tabu search algorithm for sequence optimization. Framinan and Schuster (2006) could further improve these results by a combined non-delay and inverse timetabling approach. Samarghandi et al. (2013) presented a study about different combinations of timetabling and sequencing algorithms. Raaymakers and Hoogeveen (2000) examined a FJSP with blocking and no-wait constraints. They decomposed the problem into timetabling, machine assignment, and sequencing. Mascis and Pacciarelli (2002) investigated the JSP with blocking and no-wait constraints by generalized disjunctive graph methods. They pointed out that these methods entail a trade off between feasibility and quality of the obtained solution.

1.3 Content

Feasibility is indispensable for the problem considered in this paper. Therefore, the well established decomposition into timetabling and sequence optimization is used in this work. The contribution of this paper is the simultaneous handling of no-wait and blocking constraints, where the blocking time can also be limited. A recursive timetabling algorithm is designed so that jobs are scheduled with minimal finishing time without violating any constraints by all of the jobs' tasks. Moreover, to the best knowledge of the authors, the assignment to parallel machines is done in a new way. In Section 2, the recursive timetabling algorithm is presented. Section 3 focuses on sequence optimization. A starting sequence is generated by a construction heuristic and further improved by a tabu search algorithm. In Section 4, scheduling results for data from an industrial production environment are given. Conclusions are drawn in Section 5.

t in s	$c_m(t)$
0	0
120	1
240	2
360	1
600	2
⋮	⋮
∞	0

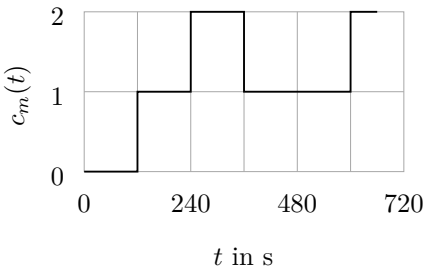

Fig. 2. Capacity utilization of a machine type m with $c_{m,max} = 2$.

2. TIMETABLING

In the course of timetabling, a given sequence of jobs is scheduled one by one so that all constraints are satisfied and the finishing time of each job is minimized. The total number of jobs is N_J. Each job $j \in \{1, \ldots, N_J\}$ is associated with $N_{T,j}$ tasks. The timetabling algorithm determines the starting times $t_{j,n}$ and the durations $d_{j,n}$, $\forall j$, $n = 1, \ldots, N_{T,j}$ of all tasks of all jobs. The durations $d_{j,n}$ of the tasks are bounded by the lower bounds $\underline{d}_{j,n}$ and by the upper bounds $\overline{d}_{j,n}$ which are specified by the production plan. For the manipulating and rolling tasks, these bounds are identical, i. e., $\underline{d}_{j,n} = \overline{d}_{j,n}$. The sequence of the jobs is stored in the so-called permutation vector $\mathbf{p} = (p_1, \ldots, p_i, \ldots, p_{N_J})$, $p_i \in \{1, \ldots, N_J\}$ with $p_i \neq p_j$ for $i \neq j$.

The proposed algorithm also assigns tasks to machines. To avoid an initial assignment of tasks to one of the parallel machines, which could finally lead to suboptimal solutions, so-called machine types m are defined. These machine types have a maximum capacity $c_{m,max}$ according to the number of parallel machines. The capacity utilization $c_m(t)$ over time is stored in a table of values for every machine type, see Fig. 2. Free time slots occur whenever $c_m(t) < c_{m,max}$ holds true. When the timetabling of all N_J jobs is finished, an assignment of the tasks to machines can be easily made.

Timetabling is done one by one for each job j. The tasks $n = 1, \ldots, N_{T,j}$ of each job j must satisfy the following constraints:

- No waiting times between consecutive tasks: $t_{j,n} + d_{j,n} = t_{j,n+1}$ for $n = 1, \ldots, N_{T,j} - 1$
- Valid durations of the tasks according to the lower and upper bounds: $\underline{d}_{j,n} \leq d_{j,n} \leq \overline{d}_{j,n}$, $\forall n$
- The maximum number of simultaneous tasks at the machine types: $c_m(t) \leq c_{m,max}$, $\forall m$

Therefore, the recursive function FINDSCHEDULE() which is summarized in Algorithm 1 is developed. The function tries to schedule the tasks according to the strategy *As early & short as possible, as late & long as necessary.*