Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

STEFAN HEINZ* THOMAS SCHLECHTE
RÜDIGER STEPHAN** MICHAEL WINKLER

Solving steel mill slab design problems

* Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.
** Technische Universität Berlin, Institut für Mathematik, Straße des 17. Juni 136, 10623 Berlin, Germany

ZIB-Report 11-38 (September 2011)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail : bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

SOLVING STEEL MILL SLAB DESIGN PROBLEMS

STEFAN HEINZ, THOMAS SCHLECHTE, RÜDIGER STEPHAN,
AND MICHAEL WINKLER

Abstract. The steel mill slab design problem from the CSPLIB is a combinatorial optimization problem motivated by an application of the steel industry. It has been widely studied in the constraint programming community. Several methods were proposed to solve this problem. A steel mill slab library was created which contains 380 instances. A closely related binpacking problem called the multiple knapsack problem with color constraints, originated from the same industrial problem, was discussed in the integer programming community. In particular, a simple integer program for this problem has been given by Forrest et al. [4]. The aim of this paper is to bring these different studies together. Moreover, we adapt the model of [4] for the steel mill slab design problem. Using this model and a state-of-the-art integer program solver *all* instances of the steel mill slab library can be solved efficiently to optimality. We improved, thereby, the solution values of 76 instances compared to previous results [13]. Finally, we consider a recently introduced variant of the steel mill slab design problem, where within all solutions which minimize the leftover one is interested in a solution which requires a minimum number of slabs. For that variant we introduce two approaches and solve all instances of the steel mill slab library with this slightly changed objective function to optimality.

1. Introduction

The *steel mill slab design problem* is motivated by a real world application from the steel industry. Mathematically, the problem consists of a set of $n \in \mathbb{N}$ orders, each order j coming with a size $s_j \in \mathbb{N}$ and color $c_j \in C$, where C is a finite set. Furthermore, we are given a set $K := \{k_1, \ldots, k_m\} \subset \mathbb{N}$ of $m \in \mathbb{N}$ capacities. The task is to equip each used slab with one capacity and assign each order to exactly one slab with the requirements that the selected capacities are respected and that each slab only processes orders of at most two different colors. The objective is to minimize the *leftover* that is the total loss or equivalent the residual capacity. Recently, Schaus et al. [13] considered a variation of the problem by adding a second criteria to the objective. Within all solutions, which minimize the leftover, one searches for a solution with a minimal number of used slabs.
The steel mill slab design problem is problem number 38 of the CSPLIB[1]. This library provides one instance which consists of 111 orders with 88 different colors, and 20 possible capacities. We call this instance the *original instance*. Furthermore, there exists a steel mill slab library [14]. This library contains 380 instances which are grouped into 19 classes each with 20 instances. These instances have been created by changing the set of possible capacities of the original instance. This means, the orders are the same as the one of the original instance. The capacities are generated uniformly and range between 10 and 50; the 19 classes are ranging from having 2 to 20 possible capacities. For more details about the generation of these instances and the library we refer to [13].

[1] `http://www.csplib.org/`

Outline. In the following section, we give a brief overview on different approaches to solve the steel mill slab design problem and related binpacking problems. We recall among others a set packing formulation [4] to the so-called *multiple knapsack problem with color constraints* which can be perceived as a slight generalization of the steel mill slab design problem. In Section 3, we adapt this model to the steel mill slab design problem and the variant where also the number of used slabs is minimized. In Section 4 we report on our computational results. Using a state-of-the-art integer program solver, we solved *all* instances of the steel mill slab library and the original instance to optimality improving the solution value of 76 instances. Moreover, we solved all these instances for the above-mentioned modification of the problem to optimality as well.

2. Related work

In the past, several different models have been proposed to solve the steel mill slab design problem. A first set of constraint programming models was presented by Frisch et al. [5] and first computational results for a (small) subset of orders of the original instance were given by the same authors in [6]. Dawande et al. [3] presented an asymptotic polynomial time approximation scheme and two 3-approximation algorithms. Hnich et al. [9] introduced an integer programming formulation, a constraint programming formulation, and a hybrid model and solved also one instance which consists of a subset of orders of the original instance. A first optimal solution of the original instance (total loss of zero) was given by Gargani and Refalo [7] using a large neighborhood search heuristic. Van Hentenryck and Michel [15] introduced a constraint programming model which can be used to solve the original instance using a heuristic approach. Recently, Schaus et al. [13] presented a collection of different constraint-based solving techniques for this problem and introduced the steel mill slab library [14]. All previously used models and solving techniques are not capable of solving all instances of the steel mill slab library.

Kalagnanam et al. [10] and Forrest et al. [4] studied a closely related binpacking problem called the *multiple knapsack problem with color constraints*. The problem provides another view on the same industrial application as the steel mill slab design problem. The problem input consists of m slabs, each slab j coming with a capacity $k_j \in \mathbb{R}$, and n items, each item i coming with a size $s_i \in \mathbb{R}$, a color $c_i \in \mathbb{N}$, and a specification in form of a subset of slabs indicating from which slabs this item can be manufactured. We say that an item is *valid* for a slab if the item can be manufactured from it. The goal is to find an assignment such that each slab contains valid items of at most two different colors, the capacities of the slabs are respected, and the unused capacity of the used slabs is minimized. For this problem, Kalagnanam et al. [10] presented a compact integer programming formulation, while Forrest et al. [4] gave a set packing formulation and designed a simple column generation approach. Their computational results indicate that this method is superior in practice. They solved an instance with 439 orders, 347 different colors, and 24 slabs (two having the same capacity). Due to the additional assignment restrictions, a slab has a restricted set of items which can be manufactured from it. The number of different colors of the associated valid items is at most 222. This instance is called `mkc` and is part of the MIPLIB2010 [11].

Interpreting,[2] this instance w.r.t. the steel mill slab design problem, it consists of 439 orders, 23 different slab capacities, and 347 colors. Forrest et al. [4] tried to solve an even larger instance called `mkc7`. This instance has 74 slabs, 9484 orders, and 233 colors within the context of the multiple knapsack problem with color

[2]We ignored the assignment restrictions and allowed an arbitrary number of slabs of each capacity.

constraints. This boils down to 70 different slab capacities and 642 colors w.r.t. the steel mill slab design problem. See the appendix for a more detailed description of the transformation and the particular problem instances in the context of steel mill slab design.

One main reason why these two binpacking problems are hard to solve in practice is that the used models, with the exception of the set packing formulation of Forrest et al. [4], are symmetric. In these models, orders are explicitly assigned to slabs, and therefore, symmetry naturally arises by permuting slabs. It is well known, for instance, that symmetry causes branch-and-bound algorithms to perform poorly, since the resulting problems change only marginally after branching, see Barnhart et al. [2]. In principle, one can respond to this difficulty by either adding symmetry breaking constraints to the given model or by avoiding such a symmetric model in advance. The first strategy was pursued by several authors. Van Hentenryck and Michel [15] partly broke symmetry using a customized search routine. Other symmetry breaking techniques are discussed in [6]. The set packing formulation of Forrest et al. [4], however, provides a model that avoids this kind of symmetry, which is one explanation for the performance of their column generation algorithm.

3. Integer programming formulation

Adapting the set packing formulation of Forrest et al. [4], we obtain an integer programming formulation for the steel mill slab design problem. This model does not contain the kind of symmetry mentioned in the previous section.

Let S be the set of all feasible slab designs. A slab design s is an assignment vector $\lambda_s \in \{0,1\}^n$. This vector defines which orders belong to this particular slab design s. This means, order $j \in \{1,\dots,n\}$ belongs to slab design s if $(\lambda_s)_j$ is one. A slab design is *feasible* if the total order size is not greater than the largest available capacity and if s contains orders of at most two different color classes. Each slab design s comes with an unique leftover l_s which is given by

$$l_s = \min\{k \in K \mid k \geq \sum_{j=1}^{n} (\lambda_s)_j\} - \sum_{j=1}^{n} (\lambda_s)_j.$$

Introducing for each feasible slab design $s \in S$ a binary decision variable x_s which is one if s is used and zero otherwise, we can formulate the steel mill slab design problem as an integer program:

$$\begin{aligned} \min \quad & \sum_{s \in S} l_s x_s && && (1)\\ \text{subject to} \quad & \sum_{s \in S} (\lambda_s)_j x_s = 1 && \forall j \in \{1,\dots,n\} \\ & x_s \in \{0,1\} && \forall s \in S. \end{aligned}$$

This is a set partitioning problem. The objective is to minimize the total leftover. The equalities are set partitioning constraints to ensure that for each order j exactly one slab design s is chosen. Finally, the last conditions state that all variables are binary.

In contrast to the setting of Forrest et al. [4] we consider the case that all orders must be covered. This is simply reflected by the transition from packing to partitioning constraints. As a result we focus on pure minimizing of the total leftover whereas Forrest et al. [4] additionally consider to maximize satisfied orders, i.e., they combine both goals in one objective function. In general we would propose the same solution methodology as Forrest et al. [4] to cope with such formulations. Since the number of columns/slab designs can become quite large, an integer program like the one above is usually solved with a branch-and-price [2] algorithm.