Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

RALF BORNDÖRFER THOMAS SCHLECHTE STEFFEN WEIDER

Railway Track Allocation by Rapid Branching

Supported by the German Federal Ministry of Economics and Technology (BMWi), grant 19M7015B.

ZIB-Report 10-22 (August 2010)

Railway Track Allocation by Rapid Branching*

Ralf Borndörfer** Thomas Schlechte** Steffen Weider**

August 23, 2010

Abstract

The *track allocation problem*, also known as train routing problem or train timetabling problem, is to find a conflict-free set of train routes of maximum value in a railway network. Although it can be modeled as a standard path packing problem, instances of sizes relevant for real-world railway applications could not be solved up to now. We propose a rapid branching column generation approach that integrates the solution of the LP relaxation of a path coupling formulation of the problem with a special rounding heuristic. The approach is based on and exploits special properties of the bundle method for the approximate solution of convex piecewise linear functions. Computational results for difficult instances of the benchmark library TTPLIB are reported.

1 Introduction

Routing a maximum number of trains in a conflict-free way through a track network is one of the basic scheduling problems for a railway company. This *optimal track allocation problem*, also known as train routing problem or train timetabling problem, has received growing attention in the operations research literature, see [8, 2, 11, 6, 17] for some recent references. A branch on the study of advanced models that incorporate, e.g., additional robustness aspects, has already been started, see, e.g., [12]. However, the problem remains that up to now the basic problem can hardly be solved even for small instances. Corridors or single stations mark or are quickly beyond the limits of the current solution technology, such that network optimization problems can not be addressed.

Finding a good track allocation model is a key prerequisite for progress towards the solution of large-scale track allocation problems. The authors of [4] proposed a novel path coupling formulation based on train path and track configuration variables. The model provides a strong LP bound, is amenable to standard column generation techniques, and therefore suited for large-scale

*This research was funded by the German Federal Ministry of Economics and Technology (BMWi), project *Trassenbörse*, grant 19M7015B.

**Zuse-Institute Berlin (ZIB), Takustr. 7, 14195 Berlin, Germany, Email {borndoerfer, schlechte, weider}@zib.de

computation. Indeed, it was shown that LP relaxations of large-scale track allocation problems involving hundreds of potential trains could be solved to proven or near optimality in this way. However, similar results for integer solutions could not be provided at that time.

This topic is addressed in this paper. Extending the work in [4], we present a sophisticated solution approach that is able to compute high-quality integer solutions for large-scale railway track allocation problems. Our algorithm is an adaptation of the rapid branching method introduced in [3] (see also the thesis [20]) for integrated vehicle and duty scheduling in public transport. The method solves a Lagrangean relaxation of the track allocation problem as a basis for a branch-and-generate procedure that is guided by approximate LP solutions computed by the bundle method. This successful second application provides evidence that rapid branching is a general solution method for large-scale path packing and covering problems.

The paper is structured as follows. Section 2 recapitulates the track allocation problem and the path configuration model. Section 3 discusses the solution of an associated Lagrangean relaxation by the bundle method. In Section 4 we adapt the rapid branching heuristic to deal with track allocation (maximization) problems. Section 5 reports computational results. We demonstrate that rapid branching can be used to produce high quality solutions for large-scale track allocation problems.

2 The Track Allocation Problem

We briefly recall in this section a formal description of the *track allocation problem*; more details can be found in the articles [5, 8, 2]. Consider an acyclic digraph $D = (V, A)$ that represents a time-expanded railway network. Its nodes represent arrival and departure events of trains at a set S of stations at discrete times $T \subseteq \mathbb{Z}$, its arcs model activities of running a train over a track, parking, or turning around. Let I be a set of requests to route trains through D. More precisely, train $i \in I$ can be routed on a path through some suitably defined subdigraph $D_i = (V_i, A_i) \subseteq D$ from a starting point $s_i \in V_i$ to a terminal point $t_i \in V_i$. Denote by P_i the set of all routes for train $i \in I$, and by $P = \bigcup_{i \in I} P_i$ the set of all train routes (taking the disjoint union).

Let $s(v) \in S$ be the station associated with departure or arrival event $v \in V$, $t(v)$ the time, and $J = \{s(u)s(v) : (u, v) \in A\}$ the set of all railway tracks. An arc $(u, v) \in A$ *blocks* the underlying track $s(u)s(v)$ for the time interval $[t(u), t(v)[$, and two arcs $a, b \in A$ are *in conflict* if their respective blocking time intervals overlap. Two train routes $p, q \in P$ are in conflict if any of their arcs are in conflict. A *track allocation* or timetable is a set of conflict-free train routes, at most one for each request set. Given arc weights w_a, $a \in A$, the weight of route $p \in P$ is $w_p = \sum_{a \in p} w_a$, and the weight of a track

allocation $X \subseteq P$ is $w(X) = \sum_{p \in X} w_p$. The *track allocation problem* is to find a conflict-free track allocation of maximum weight.

The track allocation problem can be modeled as a multi-commodity flow problem with additional packing constraints, see [8, 2, 11]. This model is computationally difficult. We consider in this article an alternative formulation as a *path coupling problem* based on 'track configurations' as proposed by the authors of [4]. A valid configuration is a set of arcs on some track $j \in J$ that are mutually not in conflict. Denote by Q_j the set of configurations for track $j \in J$, and by $Q = \bigcup_{j \in J} Q_j$ the set of all configurations. Introducing 0/1-variables x_p, $p \in P$, and y_q, $q \in Q$, for train paths and track configurations, the track allocation problem can be stated as the following integer program:

$$
\begin{aligned}
\text{(PCP)} \quad \max \quad & \sum_{p \in P} w_p x_p & & & & \text{(i)} \\
\text{s.t.} \quad & \sum_{p \in P_i} x_p & & \leq 1, & \forall\, i \in I \quad & \text{(ii)} \\
& & \sum_{q \in Q_j} y_q & \leq 1, & \forall\, j \in J \quad & \text{(iii)} \\
& \sum_{a \in p \in P} x_p - & \sum_{a \in q \in Q} y_q & \leq 0, & \forall\, a \in A \quad & \text{(iv)} \\
& & x_p, y_q & \geq 0, & \forall\, p \in P,\ q \in Q \quad & \text{(v)} \\
& & x_p, y_q & \in \{0,1\}, & \forall\, p \in P,\ q \in Q. \quad & \text{(vi)}
\end{aligned}
$$

The objective PCP (i) maximizes the weight of the track allocation. Constraints (ii) state that a train can run on at most one route, constraints (iii) allow at most one configuration for each track. Inequalities (iv) link train routes and track configurations to guarantee a conflict-free allocation, (v) and (vi) are the non-negativity and integrality constraints. Note that the upper bounds $x_p \leq 1$, $p \in P$, and $y_q \leq 1$, $q \in Q$, are redundant.

Introducing appropriately defined matrices $A \in \mathbb{Q}^{I \times P}$, $B \in \mathbb{Q}^{J \times Q}$, $C \in \mathbb{Q}^{I \times A}$, $D \in \mathbb{Q}^{J \times A}$, and a weight vector $w \in \mathbb{Q}^P$, program (PCP) can be stated in matrix form as follows:

$$
\text{(PCP)} \ \max \ w^T x,\ Ax = \mathbf{1},\ By = \mathbf{1},\ Cx - Dy \leq 0,\ (x, y) \in \{0,1\}^{P \times Q}.
$$

The authors of [4] have shown that train path and track configuration variables can be priced by solving shortest path problems in suitably defined acyclic digraphs, such that the LP relaxation of program (PCP) can be solved in polynomial time.

3 A Bundle Approach

The PCP consists of a train routing and a track configuration sub-model that are linked by coupling constraints. The sub-models are easy, but time consuming to solve using a column generation procedure based on acyclic

shortest path computations, the coupling constraints are simple but numerous. This combinatorial structure can be exploited using a Lagrangean relaxation approach in which, of course, precision and speed of convergence are critical issues. It turns out that the bundle method fits perfectly with such a scheme.

A Lagrangean dual of model PCP arises from a Lagrangean relaxation of the coupling constraints PCP (iv) and a relaxation of the integrality constraints PCP (vi) and (vii):

$$\text{(LD)} \quad \min_{\lambda \geq \mathbf{0}} \left[\max_{\substack{Ax=\mathbf{1},\\ x\in[0,1]^P}} (w^\mathsf{T} - \lambda^\mathsf{T} C)x + \max_{\substack{By=\mathbf{1},\\ y\in[0,1]^Q}} (\lambda^\mathsf{T} D)y \right].$$

LD is equivalent to the dual of the LP relaxation of PCP, and hence provides upper bounds for PCP. Introducing functions

$$\begin{aligned} f_P &: \mathbb{R}^A \to \mathbb{R}, \quad \lambda \mapsto \max(w^\mathsf{T} - \lambda^\mathsf{T} C)x,\ Ax = \mathbf{1},\ x \in [0,1]^P \\ f_Q &: \mathbb{R}^A \to \mathbb{R}, \quad \lambda \mapsto \max(\lambda^\mathsf{T} D)y,\ By = \mathbf{1},\ y \in [0,1]^Q \\ f_{P,Q} &:= f_P + f_Q, \end{aligned}$$

LD can be stated more shortly as follows:

$$\text{(LD)} \quad \min_{\lambda \geq \mathbf{0}} f_{P,Q}(\lambda) = \min_{\lambda \geq \mathbf{0}} \left[f_P(\lambda) + f_Q(\lambda) \right].$$

The functions f_P and f_Q are convex and piecewise linear. Their sum $f_{P,Q}$ is therefore a decomposable, convex, and piecewise linear function; $f_{P,Q}$ is, in particular, non-smooth. This is precisely the setting for an application of the *proximal bundle method* (PBM) to a maximization problem, see [14, 15, 13, 3, 20] for details.

When applied to LD, the PBM constructs cutting plane models of the functions f_P and f_Q in terms of subgradient bundles J_P^i and J_Q^i that are used to produce two sequences of iterates $\lambda^i, \mu^i \in \mathbb{R}^A$, $i = 0, 1, \ldots$. The points μ^i are called *stability centers*; they converge to a solution of LD. The points λ^i are trial points calculated by solving a quadratic program over a trust region around the current stability center, whose size is controlled by some positive weight u:

$$(QP_{P,Q}^i) \qquad \lambda^{i+1} := \operatorname*{argmin}_{\lambda} f_{P,Q}(\lambda) - \tfrac{u}{2} \left\| \mu^i - \lambda \right\|^2. \tag{1}$$

A function evaluation at a trial points results either in a shift of the stability center, or in an improvement of the cutting plane model. A key point is that the high-dimensional quadratic program $(QP_{P,Q}^i)$ (the dimension is equal to the number of coupling constraints) has a dual whose dimension coincides with the number subgradients in the current bundle. The method converges for a bundle size of two, typical sizes in practice are around 10 or 15. This dimension reduction makes the problem computationally tractable.

Another key point is that the PBM produces a sequence not only of approximate dual, but also of approximate primal solutions, that converge, in contrast to, e.g., subgradient methods or the volume method, both to optimal LP solutions:

- The series (μ^i) converges to an optimal solution of LD, i.e., an optimal dual solution of the LP relaxation of PCP.
- The series $(x_P^i(\lambda^i), y_Q^i(\lambda^i))$ defined as

$$(x_P^i(\lambda^i), y_Q^i(\lambda^i)) = \left(\sum_{\lambda_j \in J_P^i} \alpha_{P,j}^i x_P(\lambda_j), \sum_{\lambda_j \in J_Q^i} \alpha_{Q,j}^i y_Q(\lambda_j) \right)$$

  converges to an optimal primal solution of the LP relaxation of PCP.

Here, $\alpha_{Q,j}^i$ are optimal solutions of the dual of the quadratic program $(QP_{P,Q}^i)$, and $x_P(\lambda_j) = \text{argmax}_{x \in [0,1]^P} f_P(\lambda_j)$ and $y_Q(\lambda_j) = \text{argmax}_{y \in [0,1]^Q} f_Q(\lambda_j)$ are optimal primal solutions of f_P and f_Q. Note that in our application, determining x_P and y_Q amounts to computing optimal train paths and track configurations; this can be done by acyclic shortest path calculations. The primal approximation is useful to guide branching decisions, see next section.

4 Rapid Branching

We propose in this section a branch-and-generate (BANG) approach (i.e., a branch-and-price algorithm with partial branching, see [18]) for the construction of high-quality integer solutions of the PCP.

The main idea of this *rapid branching heuristic* is that a fix of a single variable to zero or one has almost no effect on the value of the LP relaxation of a problem such as the PCP, see [16]. The authors of [3], see also the thesis [20], proposed in the context of integrated vehicle and duty scheduling a heuristic that tries to overcome this problem by a combination of cost perturbation to "make the LP more integer", partial pricing to generate variables that are needed to complete an integer solution down in the tree, a selective branching scheme to fix large sets of variables, and an associated backtracking mechanism to correct wrong decisions. Our setting is of obvious similarity, and it will turn out that rapid branching can indeed be successfully applied to solve large-scale track allocation problems.

We use the following notation. Recall the PCP

$$\text{(PCP)} \quad \max_{0 \le x,y \le \mathbf{1}} \; w^T x, \; Ax = \mathbf{1}, \; By = \mathbf{1}, \; Cx - Dy \le 0, \; (x,y) \in \{0,1\}^{P \times Q}.$$

Let $l, u \in \{0,1\}^{P \times Q}$, $l \le u$, be vectors of bounds that model fixings of variables to 0 and 1. Denote by $L := \{j \in P \times Q : u_j = 0\}$ and $U := \{j \in P \times Q : l_j =$

1} the set of variables fixed to 0 and 1, respectively, and by

$$(\mathrm{PCP})(l,u) \quad \max_{l\le x,y\le u} \; w^T x,\; Ax = \mathbf{1},\; By = \mathbf{1},\; Cx - Dy \le 0,\; (x,y) \in \{0,1\}^{P\times Q}$$

the IP derived from PCP by such fixings. Denote further by $N \subseteq P \times Q$ some set of variables which have, at some point in time, already been generated by a column generation algorithm for the solution of PCP. Let RPCP and $\mathrm{RPCP}(l,u)$ be the restrictions of the respective IPs to the variables in N (we assume that $L, U \subseteq N$ holds at any time when such a program is considered, i.e., variables that have not yet been generated are not fixed). Finally, denote by MLP, $\mathrm{MLP}(w,l,u)$, RMLP, and $\mathrm{RMLP}(w,l,u)$ the LP relaxations of the integer programs under consideration; MLP and $\mathrm{MLP}(w,l,u)$ are called *master LPs*, RMLP and $\mathrm{RMLP}(w,l,u)$ *restricted master LPs* (the objective w is included in the notation for $\mathrm{MLP}(w,l,u)$ and $\mathrm{RMLP}(w,l,u)$ for reasons that will become clear in the following Section 4.1).

Rapid branching tries to compute a solution of PCP by means of a search tree with nodes $\mathrm{PCP}(l,u)$. Starting from the root $\mathrm{PCP} = \mathrm{PCP}(0,\mathbf{1})$, nodes are spawned by additional variable fixes using a strategy that we call *perturbation branching*. The tree is depth-first searched, i.e., rapid branching is a plunging (or diving) heuristic. The nodes are analyzed heuristically using restricted master LPs $\mathrm{RMLP}(w,l,u)$. The generation of additional columns and node pruning are guided by so-called *target values* as in the branch-and-generate method. To escape unfavorable branches, a special *backtracking mechanism* is used that performs a kind of partial binary search on variable fixings. The idea of the method is as follows: we try to make rapid progress towards a feasible integer solution by fixing large numbers of variables by perturbation branching (Section 4.1) in each iteration, repairing infeasibilities or deteriorations of the objective by regeneration of columns if possible and by controlled backtracking otherwise (Section 4.2).

4.1 Perturbation Branching

The idea of *perturbation branching* is to solve a series of MLPs with objectives $w^i, i = 0,1,2,\ldots$ that are perturbed in such a way that the associated LP solutions x^i are likely to become more and more integral. In this way, we hope to construct an almost integer solution at little cost. The perturbation is done by increasing the utility of variables with LP values close to one according to the formula:

$$\begin{array}{lll} w_j^0 & := w_j, & j \in N \\ w_j^{i+1} & := w_j^i + w_j \alpha x_j^2, & j \in N, \quad i = 0,1,2,\ldots. \end{array}$$

The idea behind this quadratic perturbation is that variables with values close to 1 are driven towards 1. The progress of this procedure is measured in terms

of the potential function

$$v(x^i) := w^\mathsf{T} x + \delta |B(x^i)|,$$

where ϵ and δ are parameters for measuring near-integrality and the relative importance of near-integrality (we use $\epsilon = 0.1$ and $\delta = 1$), and $B(x^i) := \{j \in N : x^i_j > 1 - \epsilon\}$ is the set of variables that are set or almost set to one. The perturbation is continued as long as the potential function increases; if the potential does not increase for some time, a spacer step is taken in an attempt to continue. On termination, the variables in the set $B(x^i)$ associated with the highest potential are fixed to one. If no variables at all are fixed, we choose a single candidate by *strong branching*, see [1]. Objective perturbation has also been used in [19] for the solution of large-scale set partitioning problems, and, e.g., in [9] in the context of general mixed integer programming.

Algorithm 1: Perturbation Branching.

Data: RMLP(w, l, u), integrality tolerance $\epsilon \in [0, 0.5)$, integrality weight $\delta > 0$, perturbation factor $\alpha > 0$, bonus weight $M > 0$, spacer step interval k_s, iteration limit $k_{\max}$

Result: set of variables B^* that can be fixed to one

init $i \leftarrow k \leftarrow 0$; $w^0 \leftarrow w$; $B^* \leftarrow \emptyset$; $v^* \leftarrow \infty$;
while $k < k_{\max}$ **do** `/* maximum number of iterations not reached */`
  compute $x^i \leftarrow \operatorname{argmax} \text{RMLP}(w^i, l, u)$;
  set $B^i \leftarrow \{j : x^i_j \geq 1 - \epsilon,\ l_j = 0\}$;
  set $v(x^i) \leftarrow w^\mathsf{T} x^i + \delta |B^i|$;
  if x^i is integer **then**
    set $B^* \leftarrow B^i$; `/* candidates found */`
    break;
  else
    if $k \equiv 0 \mod k_s$ and $k > 0$ **then**
      set $j^* \leftarrow \operatorname{argmax}_{l_j = 0} x^i_j$;
      set $w^i_j \leftarrow M$;
      set $B^* \leftarrow B^i \cup \{j^*\}$; `/* spacer step */`
    else
      if $v(x^i) > v^*$ **then**
        set $B^* \leftarrow B^i$; $v^* \leftarrow v(x^i)$; $k \leftarrow -1$; `/* progress */`
      end
      set $w^{i+1}_j \leftarrow w^i_j + \alpha w_j (x^i_j)^2 \quad \forall j$; `/* perturb */`
    end
  end
  set $i \leftarrow i + 1$; $k \leftarrow k + 1$;
end
if $B^* = \emptyset$ **then**
  set $B^* \leftarrow \{j^*\} \leftarrow$ strongBranching() ; `/* strong branching */`
end
return B^*;

Algorithm 1 gives a pseudocode listing of the complete perturbation branch-

ing procedure. The main work is in solving the perturbed reduced master LP (line 3), generating new variables if necessary. Fixing candidates are determined (line 4) and the potential is evaluated (line 5). If the potential increases (lines 15–17), the perturbation is continued (line 18). If no progress was made for k_s steps (line 10), the objective is heavily perturbed by a spacer step in an attempt to continue (lines 10–13). However, even this perturbation does not guarantee that any variable will get a value above $1-\epsilon$, if $\epsilon < 1/2$. If this happens and the iteration limit is reached, a single variable is fixed by strong branching (line 24).

4.2 Binary Search Branching

The fixing candidate sets B^* produced by the perturbation branching algorithm are used to define nodes in a branch-and-generate search tree by imposing bounds $x_i = 1$ for all $i \in B^*$. This typically fixes many variables to one, which is what we wanted to achieve. However, sometimes too much is fixed and some of the fixings turn out to be disadvantageous. In such a case we must backtrack. We propose to do this in a binary search manner by successively undoing half of the fixes until either the fixings work well or only a single fix is left. This procedure is called *binary search branching*.

Here are the details. Let B^* be a set of potential variable fixes and $K = |B^*|$. Order the variables in B^* by some criterion as $i_1, i_2, \ldots, i_K$ and define sets

$$B^*_k := \{i_1, \ldots, i_k\}, \quad k = 1, \ldots, K.$$

Consider search tree nodes defined by fixing

$$x_j = l_j = 1, \quad j \in B^*_k, \quad k = K, \lceil K/2 \rceil, \lceil K/4 \rceil, \ldots, 2, 1.$$

These nodes are examined in the above order. Namely, we first try to fix all variables in B^*_K to one, since this raises hopes for maximal progress. If this branch comes out worse than expected, it is pruned, and we backtrack to examine $B^*_{\lceil K/2 \rceil}$ and so on until possibly B^*_1 is reached. In this situation, the single fix is applied imperatively. The resulting search tree is a path with some pruned branches, i.e., binary search branching is a plunging heuristic. In our implementation, we order the variables by increasing reduced cost of the restricted root LP, i.e., we unfix half of the variables of smallest reduced cost. This sorting is inspired by the scoring technique of [7]. The decision whether a branch is pruned or not is done by means of a *target value* as introduced in [18]. Such a target value is a guess about the development of the LP bound if a set of fixes is applied; we use a linear function of the integer infeasibility. If the LP bound stays below the target value, the branch develops according to our expectations, if not, the branch "looks worse than expected" and we backtrack.

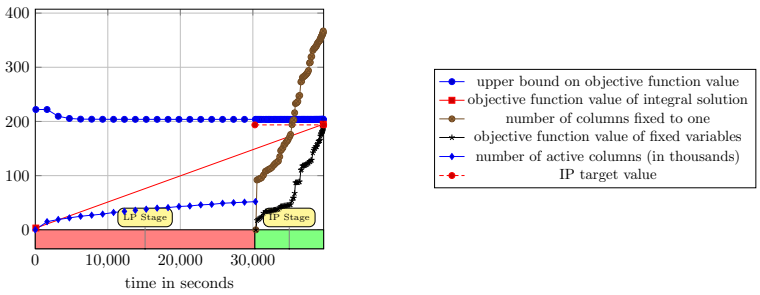

Figure 1: Solving a track allocation problem with `TS-OPT`; dual (LP) and primal (IP) stage.

5 Computational Results

We test our approach on a selection of three large instances that are freely available from the benchmark library TTPLIB, see [10]. They are associated with a macroscopic railway network model of the area spanned by the cities of Hannover, Kassel, and Fulda in Germany. This HaKaFu network consists of 37 stations and 120 tracks (HAKAFU_SIMPLE_37_120_6), giving rise to 4320 different headway times for 6 standard train types. The test instances differ with respect to requests for trains, i.e., by traffic demand, and we remark that simple greedy or rounding procedures fail to construct satisfactory solutions for them.

Table 1 gives some statistics on the number of requested trains ($|I|$), the number of tracks ($|J|$), the number of coupling arcs ($|A|$), and the total sizes of the train routing and the track configuration digraphs ($|V_I|, |A_I|, |V_J|, |A_J|$) associated with the test instances. The coupling arcs are those arcs that correspond to train movements along a track; they are in one-to-one correspondence with the coupling constraints (PCP) (iv). The remaining arcs corresponding to pull-ins and pull-outs, and to movements and parkings in stations do not give rise to conflicts (the instances do not involve station capacities) and do therefore not give rise to coupling constraints.

All our computations were performed on computers with an Intel Core 2 Extreme CPU X9650 with 3 GHz, 6 MB cache, and 8 GB of RAM. Figure 1 shows a typical run of our code `TS-OPT`. In the initial LP stage (red or dark), a global upper bound is computed by solving the Lagrangean dual using column generation and the bundle method. The bundle method converges after approximately 9 hours and pricing 50.000 variables. In the succeeding IP stage (green or light) an integer solution is constructed by the rapid branching heuristic. It can be seen that the upper bound does almost not move, i.e., the final integer solution has virtually the same objective value as the LP relaxation, and that indeed often large numbers of variables are fixed to one throughout the course of the rapid branching heuristic.

scenario	trains ($\|I\|$)	tracks ($\|J\|$)	$\|A\|$	$\|V_I\|$	$\|A_I\|$	$\|V_J\|$	$\|A_J\|$
REQ_31	1062	79	6006	11397	16493	12162	26694
REQ_32	1140	101	11187	22980	34852	22568	59037
REQ_33	570	101	5845	11490	17426	11884	31095

Tab. 1: Track allocation test instances.

5.1 Bundle Calibration

Figure 2 compares the effect of different choices for the size of the bundle (2, 5, 10, 15, 20, 25) on the solution of the root LP relaxation of our test instances. It can be seen that larger bundles lead in general to a reduction in the number of iterations to a certain limit. However, larger bundles also produce larger and more difficult quadratic programs, such that the total solution time increases after a certain point. A bundle size of 10 or 15 seems to be a good choice.

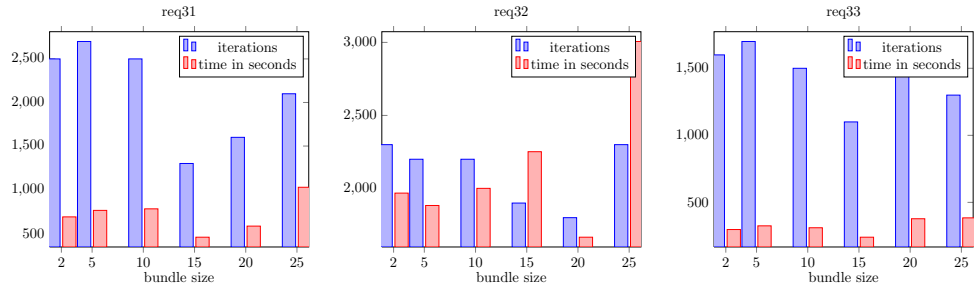

Figure 2: Testing different bundle sizes.

5.2 Rapid Branching

Tables 2 and 3 show results for solving the test instances by our code `TS-OPT`. The tables list the number of scheduled trains in the best solution found, the upper bound, the optimality gap, the total running time in CPU seconds, and the number of (rapid) branching nodes. The computations in Table 2 have been performed with an aggressive choice of the rapid branching integrality tolerance of $\epsilon = 0.4$, Table 3 shows the results for a cautious choice of $\epsilon = 0.2$. It can be seen that the aggressive choice tends to be faster, because more variables are fixed at once to explore fewer nodes, but the solution quality is lower. By choosing $\epsilon = 0.2$, high quality solutions for large-scale track allocation problems involving hundreds of train requests can be computed.

References

[1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding cuts in the TSP (a preliminary report). Technical report, Center for Discrete

scenario	$\|I\|$	trains in solution	upper bound	objective of solution	gap in %	time	branching nodes
REQ31	1062	356	464.40	457.79	1.44	45min	53
REQ32	1140	288	240.71	231.19	4.12	1h52min	56
REQ33	570	154	126.38	122.03	3.57	18min	47

Tab. 2: Solving track allocation problems by rapid branching (int. tolerance $\epsilon = 0.4$)

scenario	$\|I\|$	trains in solution	upper bound	objective of solution	gap in %	time	branching nodes
REQ31	1062	356	464.41	457.54	1.50	5h	59
REQ32	1140	298	240.71	239.61	0.46	11h	67
REQ33	570	154	126.38	122.03	3.57	1h23min	51

Tab. 3: Solving track allocation problems by rapid branching (int. tolerance $\epsilon = 0.2$)

Mathematics and Theoretical Computer Science (DIMACS), March 1995. DIMACS Technical Report 95-05.

[2] Ralf Borndörfer, Martin Grötschel, Sascha Lukac, Kay Mitusch, Thomas Schlechte, Sören Schultz, and Andreas Tanner. An auctioning approach to railway slot allocation. *Competition and Regulation in Network Industries*, 1(2):163–196, 2006. ZIB Report 05-45 at http://opus.kobv.de/zib/volltexte/2005/878/.

[3] Ralf Borndörfer, Andreas Löbel, and Steffen Weider. A bundle method for integrated multi-depot vehicle and duty scheduling in public transit. In Mark Hickman, Pitu Mirchandani, and Stefan Voß, editors, *Computer-aided Systems in Public Transport*, volume 600 of *Lecture Notes in Economics and Mathematical Systems*, pages 3–24. Springer-Verlag, 2008. ZIB Report 04-14 at http://opus.kobv.de/zib/volltexte/2004/790/.

[4] Ralf Borndörfer and Thomas Schlechte. Models for railway track allocation. In Christian Liebchen, Ravindra K. Ahuja, and Juan A. Mesa, editors, *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2007. http://drops.dagstuhl.de/opus/volltexte/2007/1170.

[5] U. Brännlund, P.O. Lindberg, A. Nou, and J.-E. Nilsson. Railway timetabling using Lagrangian relaxation. *Transportation Science*, 32(4):358–369, 1998.

[6] Gabrio Caimi. *Algorithmic decision support for train scheduling in a large and highly utilised railway network*. PhD thesis, ETH Zurich, 2009.

[7] Alberto Caprara, Matteo Fischetti, and Paolo Toth. Algorithms for the set covering problem. *Annals of Operations Research*, 98:2000, 1998.

[8] Alberto Caprara, Michele Monaci, Paolo Toth, and Pier Luigi Guida. A Lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Appl. Math.*, 154(5):738–753, 2006.

[9] Jonathan Eckstein and Mikhail Nediak. Pivot, cut, and dive: a heuristic for 0-1 mixed integer programming. *J. Heuristics*, 13(5):471–503, 2007.

[10] Berkan Erol, Marc Klemenz, Thomas Schlechte, Sören Schultz, and Andreas Tanner. TTPlib 2008 - A library for train timetabling problems. In A. Tomii, J. Allan, E. Arias, C.A. Brebbia, C. Goodman, A.F. Rumsey, and G. Sciutto, editors, *Computers in Railways XI*. WIT Press, 2008.

[11] Frank Fischer, Christoph Helmberg, Jürgen Janßen, and Boris Krostitz. Towards solving very large scale train timetabling problems by Lagrangian relaxation. In Matteo Fischetti and Peter Widmayer, editors, *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany, 2008. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany.

[12] Matteo Fischetti, Domenico Salvagnin, and Arrigo Zanette. Fast approaches to improve the robustness of a railway timetable. *Transportation Science*, 43(3):321–335, 2009.

[13] C. Helmberg. *Semidefinite Programming for Combinatorial Optimization.* Habilitation Thesis, Technische Universität Berlin, October 2000.

[14] K. C. Kiwiel. Proximal bundle methods. *Mathematical Programming*, 46(123):105–122, 1990.

[15] K. C. Kiwiel. Approximation in proximal bundle methods and decomposition of convex programs. *Journal of Optimization Theory and applications*, 84(3):529–548, 1995.

[16] M.E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Oper. Res.*, 53(6):1007–1023, 2005.

[17] Richard Lusby, Jesper Larsen, Matthias Ehrgott, and David Ryan. Railway track allocation: models and methods. *OR Spectrum*, December 2009.

[18] R. Subramanian, R.P. Sheff, J.D. Quillinan, D.S. Wiper, and R.E. Marsten. Coldstart: Fleet assignment at delta air lines. *Interfaces*, 24(1):104–120, 1994.

[19] D. Wedelin. An algorithm for a large scale 0-1 integer programming with application to airline crew scheduling. *Annals of Operations Research*, 57:283–301, 1995.

[20] Steffen Weider. *Integration of Vehicle and Duty Scheduling in Public Transport*. PhD thesis, TU Berlin, 2007. http://opus.kobv.de/tuberlin/volltexte/2007/1624/.