Routing Trains Through Railway Junctions: A New Set Packing Approach

Richard Lusby[1], Jesper Larsen[2], David Ryan[1], Matthias Ehrgott[1,3]

[1]Department of Engineering Science,
The University of Auckland,
New Zealand,
{r.lusby ,d.ryan, m.ehrgott}@auckland.ac.nz

[2]Informatics and Mathematical Modelling,
Technical University of Denmark,
Denmark,
jla@imm.dtu.dk

[3] Laboratoire d'Informatique de Nantes Atlantique,
Université de Nantes,
France,
matthias.ehrgott@univ-nantes.fr

IMM TECHNICAL REPORT 2006-21

Abstract

The problem of routing trains through railway junctions is an integral part of railway operations. Large junctions are highly interconnected networks of track where multiple railway lines meet, intersect, and split. The number of possible routings makes this a very complicated problem. Here we show how the problem can be formulated as a set packing model. To exploit the structure of the problem we present a solution procedure which entails solving the dual of this formulation through the dynamic addition of violated cuts (primal variables). A discussion of the variable (train path) generation phase, as well as an efficient pricing routine in which these variables are represented by tree structures is also included. We illustrate the proposed methodology on an example junction with encouraging results. The decision support system currently being developed will enable planners to solve strategic, tactical, and operational level variants of the problem.

1 Introduction

The railway industry is rich in problems that can be modelled and solved using Operations Research techniques. In this day and age, arguably the most important of these are the ones that concern the effective allocation and utilization of available resources. One such problem, the focus of this paper, entails allocating the track capacity of a junction to a timetabled set of trains to ensure quality routings are obtained whilst adhering to a variety of operational constraints.

The diverse range of problems facing railway companies are typically categorized according to the required planning horizon. This leads to the customary three stage approach with problems being defined as strategic, tactical, or operational in nature. The problem of routing trains through railway junctions arises at each of the levels. A short description of each variant is given next.

Problems at the strategic level are characterized by lengthy time horizons and typically involve resource acquisition. Viewed in this context the problem considered here appears in the form of capacity assessment. Railway management often face the task of deciding between a number of possible investment alternatives concerning proposed infrastructure modifications to junctions. Perhaps the most influential factor in making the final decision is capacity. Railway management are very interested in knowing, with precision, what level of rail traffic the modified infrastructure would cater for. This effectively involves determining the maximum number of trains that could be routed through the junction within a given time horizon. Solutions to this problem also indicate the impact of implementing various operating policies within the proposed junction – another important aspect to consider when deciding between possible investment projects. The construction or modification of infrastructure usually involves high capital investment and has long lasting ramifications. It is hence essential that one has proper tools to assist in this process.

Tactical level problems focus more on allocating resources on an infrastructure which is assumed to be fixed. Such problems normally have a mid-term planning horizon. On this level the problem considered here answers the question of timetable feasibility. In most countries it is not uncommon for the railway system to be divided into two main areas – those responsible for the infrastructure, and those responsible for the rolling stock (train operating companies). The train operating companies each submit a preferred timetable, and the infrastructure managers determine if there exists a conflict free routing for all the trains in the amalgamated timetable. There is no clear objective at this stage, however, one typically aims to schedule the maximum number of trains while taking into account the preferences of the trains for the routes. This variant of the problem is not dissimilar to the capacity assessment approach. The only noticeable difference being that at the strategic level one is primarily concerned with finding a saturated timetable. A saturated timetable is one that contains the maximum number of trains for a given timetable period.

Operational problems are defined to be those that occur on a day-to-day operational basis when pre-determined operating policies need to be adjusted due to unforeseen disturbances. The dynamic environment in which these problems occur necessitates almost real time resolution. Due to train routes through a junction being highly interdependent, the assignment of trains to routes found at the tactical level is susceptible to disruption. The impact of late train arrival, track maintenance, or even accidents will propagate through the timetable with varying degrees of severity and quite possibly result in the predetermined operating policy becoming infeasible. The variant of the problem occurring here simply involves rescheduling the trains in such a way so as to return to the original

schedule with minimal required disruption.

Despite its apparent potential benefits, the application of Operations Research techniques to the problem of routing trains through railway junctions has been surprisingly limited with manual approaches still widely employed. The purpose of this paper is to present a set packing model applicable to all instances of the problem outlined above. We demonstrate how the train routing problem can be formulated as a set packing model, and discuss its flexibility from a modelling perspective. While being perhaps the most logical formulation of the problem, its dimensions do impose some restrictions from a computational point of view. However, we show that the dual of this formulation does possess a number of nice properties that one can exploit in solving the problem, and present a solution procedure which entails solving the dual through the dynamic addition of violated cuts (primal variables). The primal variables are train paths and are constructed using a generator which takes into account the dynamics of the trains. An efficient pricing routine in which the primal variables are represented by several tree structures is also described. We illustrate the proposed methodology with the aid of an example junction.

This paper is organized as follows. Section two gives a more detailed description of the problem and introduces some definitions. In Section three we provide a summary of earlier work and highlight the inherent limitations of these approaches. Section four describes the set packing model and its dual as well as our path generator, while Section five details the solution approach and also the primal variable tree structures. We conclude with an example in Section six, and future research directions are given in Section seven.

2 Problem Definition

A *junction* is a highly interconnected network of track where multiple railway lines meet, intersect, and split. Quite typically it includes a station, although this is not assumed in the definition. The network of track comprising the junction can be divided into a number of *track sections*. These are essentially segments of track on which for safety reasons there can be at most one train at any given time. Some examples include switches, crossings and platform track sections. For completeness, and for the purpose of remaining consistent with the previous literature, we define the perimeter of the junction to consist of a number of *entering points* and *leaving points*. These are rather self-explanatory and just indicate the points at which it is possible for trains to enter and leave the junction. Furthermore, they also identify the scope of the infrastructure concerned. A *route* through the junction is thus a sequence of track sections connecting an entering point to a leaving point. The route may or may not involve stopping at an available platform. Depending on the number of such points as well as the number of switches in the junction, there may be a significant number of routes possible. We distinguish this from a *train path* which refers to a possible traversal of a given route in time. Figure 1 below illustrates the typical characteristics of a junction as well as a possible route from entering point A to leaving point E.

A formal definition of the problem we are addressing is as follows. Given the detailed track layout of the junction as well as a proposed timetable, i.e. the respective arrival and departure times for a set of trains, what is the maximum number of trains that can be assigned a route through the junction? This is what is referred to as the feasibility problem. Other objectives are of course possible, however, this one has been chosen for expository purposes. This simple objective function does illustrate the subtle difference between the strategic and tactical level variants of the problem. If the solution to this problem is less

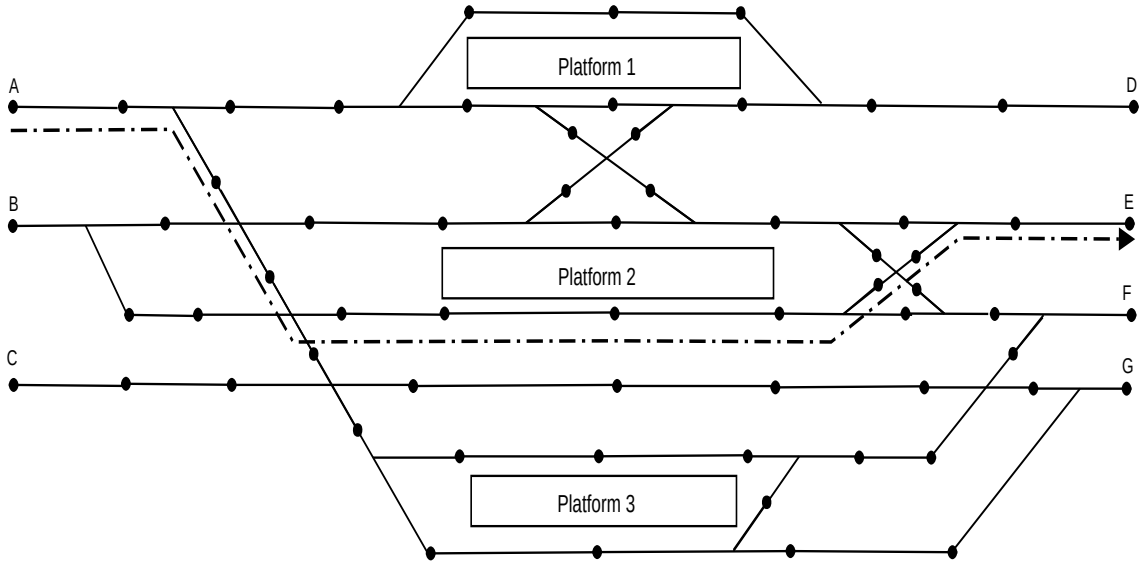

Figure 1: A possible junction showing a possible route. Dots show the boundaries of track sections.

than the cardinality of the train set, the proposed timetable is infeasible, and a saturated solution (for this timetable only) is given by the solution to the optimization problem. On the other hand, if the solution to this problem is equal to the cardinality of the train set (i.e. a conflict free routing has been found for all trains), the proposed timetable may or may not be saturated. One could look at introducing saturating trains in such a situation. Irrespective of which variant of the problem we are concerned with, the chosen routes must satisfy a number of operational constraints.

Probably the most important requirement for the selected routes is that no two trains share any part of the junction infrastructure at the same time. Routes that do not satisfy this criterion are said to be in conflict, and obviously cannot be assigned simultaneously. To prevent trains from getting too close to one another within a junction railway companies tend to implement one of two (or possibly both) systems. The *route locking and sectional release system* enforced by the Dutch railway companies (among others) stipulates that trains must lock a sequence of track sections prior to using them. For instance, when a train arrives at an entering point to the junction it must lock all the track sections it is going to use in reaching its designated platform, and then prior to departure, lock all the track sections it is going to use in leaving the junction. For the route shown in Figure 1 this would mean that the train would lock the first six track sections of its route (known as its *inbound route*) on arrival at entering point A. After waiting for some minimum amount of dwell time at the platform to allow passengers to board and alight, it would similarly claim the remaining five track sections (*outbound route*) before proceeding to leaving point E. Trains successively release each of the locked track sections after traversing them. Such a system allows trains to proceed without interruption within a junction as no track section can be simultaneously locked by two or more trains. Some buffer time is usually incorporated into the release time of the individual track sections to build some robustness into the route. The *block signalling* system implemented by the French and German railway companies is similar. Essentially the railway network is divided into a number of blocks delimited by signals, where each block may contain one or more track

sections. On entry to a block, all track sections are simultaneously locked. These are only released when the tail of the train has exited the block and some additional clearing time has elapsed. Any realistic model needs to be able to accurately incorporate these features.

Zwaneveld et al. (1996) detail a number of other constraints pertaining to customer considerations as well as train connections. For instance, it is often beneficial to have all trains travelling in a similar direction leave from platforms that are close to each other in proximity. It is also necessary to account for trains that must be coupled and decoupled. Furthermore one needs to ensure that trains satisfy the speed limits imposed on each of the track sections. There may of course be other constraints not listed here that refer to particular operating policies implemented by individual railway companies.

3 Previous Work

As we mentioned earlier, the problem of routing trains through junctions has received relatively limited attention in the literature. Zwaneveld et al. (1996), in the development of DONS (Designer of Network Schedules) software for the Dutch railway company NS, can be regarded as the first attempt to solve this particular problem via Operations Research techniques. Aimed at the strategic level, the problem is formulated as a node packing model, and solved using a branch-and-cut procedure. It was argued that in order to deal with safety requirements imposed by the Dutch railway company, as well as the need to calculate the exact claim and release times of the track sections within any given route, one should adopt a modelling approach in which the train routes are explicitly considered. This particular line of reasoning naturally gives rise to a node packing formulation. By representing each possible train route as a node, and connecting all pairs of routes in conflict, with an edge, the node packing structure is evident. Zwaneveld et al. (1996) describe two types of conflict; nodes representing routes for the same train or any two nodes representing routes for two different trains that cannot be assigned simultaneously due to some form of conflict. The solution approach outlined solves the problem to optimality, and computational results for relatively small instances are given.

This approach is particularly effective in that it can accurately model any level of train route compatibility. As train routes are explicitly dealt with, one can just add the required edge to prevent the simultaneous selection of any two nodes regardless of the reason. However, the model does have some severe limitations from both a computational, as well as a practical point of view. Perhaps the most obvious being that one must generate all train conflicts prior to solving the problem. Furthermore, the conflict graph approach is inflexible in the sense that it does not allow additional trains to be added. This is primarily because a conflict graph is only applicable to the set of routes defining it. Hence if one wished to consider additional trains, additional routes, or even alterations to the current routes, a new conflict graph would need to be obtained by identifying all new conflicts and adding the required edges and nodes. Other noticeable limitations include the weakness of the linear programming relaxation, as well as the large number of constraints and nodes in any real life example. The branch-and-cut algorithm presented relies quite heavily on preprocessing and reduction techniques to remove the superfluous nodes in order to create a problem of manageable size. Zwaneveld et al. (2001), in a follow up paper, comment that this approach was unable to solve the routing problem for two of the larger stations in the Netherlands.

Zwaneveld et al. (2001) can be considered an extension of Zwaneveld et al. (1996) as they,

while adopting the same modelling approach, attempt to remedy some of the problems inherent in the earlier work. In Zwaneveld et al. (2001) the problem is formulated as a weighted node packing model by making the reasonable assumption that trains will prefer certain routes to others. It is also claimed that the inclusion of this preference coefficient allows a hierarchical list of objectives to be considered. Zwaneveld et al. (2001) also elect to partition a train route into the components inbound, outbound and platform. A constraint is then enforced on each component. Zwaneveld et al. (1997) show that such a partitioning can lead to a reduction in the number of route variables. Zwaneveld et al. (2001) include more sophisticated preprocessing and reduction techniques in the solution process, and conclude that this approach is sufficient for solving any routing problem arising at any Dutch railway station.

The node packing approach is also the basis of the model in Delorme (2003). His PhD thesis looks at analyzing the capacity of a junction. The only noticeable difference between this model and that of Zwaneveld et al. (2001) is that Delorme (2003) considers multiple objectives. While the feasibility problem is given the highest priority, it is also argued that one should consider optimizing the number of additional (saturating) trains as well as the preferences associated with each train. These define the objectives of the model, and are lexicographic in the above order with the last two being considered of equal importance. Delorme (2003) also acknowledges that one should include a robustness measure on the obtained routing and presents a method to do so. Essentially it involves determining the overall delay resulting from the propagation of some initial delay. Finding robust train routings is also the focus of Caimi et al. (2005). They, too, model the problem via a node packing model. A local search optimization heuristic is applied to an initial feasible solution in an attempt to increase the time slot of a chosen route, the time slot being the time interval during which a train may arrive and find its designated route open.

Delorme (2003) presents two solution methods. The first is a branch-and-cut approach, again not dissimilar to the work of Zwaneveld et al. (2001). It involves extensive preprocessing using dominance testing and the addition of valid inequalities. This was tested on both randomly generated, as well as railway specific problem instances. Although the pre-processing techniques can reduce the problem size quite significantly, it was reported that this exact solution approach was quite cumbersome. A second, more promising approach, using the Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic is proposed. Results showed that the GRASP metaheuristic was able to solve the problem close to optimality in reasonable time. Delorme et al. (2004) introduce several types of GRASP. These are tested and compared on a number of routing problems arising at the Pierrefitte Gonesse junction north of Paris. Gandibleux et al. (2005) propose an Ant Colony Optimization meta heuristic as an alternative to GRASP, and validate this through a comparison on a number of randomly generated problems as well as fifteen railway instances.

Constraint programming methods combined with simulation have been proposed by Delorme et al. (2001) and Rodriguez (2002). The former focuses on the strategic level while the latter addresses the operational. The constraint programming models essentially involve determining the occupation time of a given track section by a train using simulated values for the train's run time over a track section and the clearing time. Unlike the node packing models, this approach can be regarded as implicitly dealing with routes. Delorme et al. (2001) compare this approach with an adaptation of GRASP by applying each to a number of railway instances. The variant of GRASP was applied to a node

packing formulation. Conflicting conclusions are drawn as both have positive aspects. The constraint programming approach is useful in obtaining a good scheduling along a route, but has limitations in the number of routes it can consider. The node packing approach on the other hand can consider many routes, but as routes are explicitly modelled, cannot alter the occupation times of track sections within a route. Delorme et al. (2001) suggest that future research should be directed toward a hybrid model.

Carey and Carville (2003) are concerned with the related problem of platforming trains at stations. Their aim is to emulate existing manual methods, and hence present a simple greedy heuristic which sequentially routes trains. Trains are considered one at a time and assigned the best platform at that time.

4 The Set Packing Model and its Dual

Given a set $S = \{1, 2, .., m\}$ of m elements as well as a set $X = \{X_1, X_2, .., X_n\}$ of n subsets of the elements of S, a packing P is a subset of X such that $X_j \cap X_k = \emptyset$, $X_j \neq X_k \in P$. Quite typically each subset of X has a value associated with it, and one aims at finding a packing with maximum total value. The general formulation of the set packing problem is given below.

$$\begin{array}{lrcl} \text{Maximize} & p^T x & & \\ \text{subject to} & Ax & \leq & e \\ & x & \in & \{0,1\}^n, \end{array} \tag{1}$$

where p_j is the value associated with subset X_j. The column Ae_j of the A matrix corresponds to subset X_j, with a_{ij} equal to one if element i is contained in subset j, and zero otherwise. The binary decision variable x_j takes the value one if subset X_j is chosen, and is zero otherwise.

The problem of routing trains through railway junctions is a natural application of the set packing model. Recall that to provide a conflict free routing through the junction for a set of trains, one must ensure that at most one train is locking any track section at any given time. To capture both the spatial and temporal components inherent in such a restriction, one needs to identify a constraint for each track section in a sequence of uniform time intervals. This is analogous to taking a snapshot of the junction in each time interval. The time interval used is obtained through a discretization of the timetable period. This is consistent with what is done in planning in practice although the duration of the time interval is railway company dependent. A typical value might be about 15 seconds. This modelling approach allows one to represent a path through the junction for a particular train as a column of zeros and ones. A one in a particular row indicates that the train is claiming that particular track section at that particular time, while a zero indicates otherwise. By identifying S as a set of m time period track section pairs, X as the set of subsets of time period track section pairs, and Ae_j the column of the A matrix representing the corresponding path for subset X_j, the set packing structure of (1) is obvious.

Since each train has a number of possible routes (each with many paths) through the junction (including of course the *null path* which corresponds to the train not being routed at all), to ensure that we select one and only one of a train's possible paths we include

a generalised upper bound constraint for each train. Our complete model is given below. We will refer to this model as our primal problem.

$$\begin{array}{lll} \text{Maximize} & p^T x & \\ \text{subject to} & \begin{bmatrix} T_1 & T_2 & \dots & T_t \\ R_1 & R_2 & \dots & R_t \end{bmatrix} x & \begin{matrix} = & e \\ \leq & e \end{matrix} \\ & x & \in \quad \{0,1\}^n, \end{array} \tag{2}$$

where t is the number of trains. Both p and x consist of t smaller vectors which reflects the ordering of the variables. The entries of p indicate the benefit received in assigning the corresponding train path. The constraint matrix consists of submatrices T_i and R_i ($i = 1, 2, .., t$). T_i is a $(t \times n_i)$ matrix of the form

$$T_i = e_i e^T \text{ with } e_i \text{ the } i\text{th unit vector and } e^T = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix},$$

and R_i is an $(m \times n_i)$ matrix of the form

$$R_i = (r_{l,k}) = \begin{cases} 1 \text{ if the } k\text{th path for train } i \text{ uses time interval track section pair } l \\ 0 \text{ otherwise.} \end{cases}$$

Our decision variables are given by the binary variables

$$x_{ij} = \begin{cases} 1 \text{ if train } i \text{ is assigned path } j \in P_i \\ 0 \text{ otherwise.} \end{cases}$$

P_i is the set of paths for train i (indexed 1 to n_i). There are n variables in total (n=$\sum_{i=1}^{t} n_i$). The dual vectors corresponding to the GUB constraints and the time period track section constraints are given by π_1 and π_2, respectively.

Unlike the node packing models proposed by Zwaneveld et al. (1996) and Zwaneveld et al. (2001), this particular approach deals implicitly with the train routes as it attempts to identify and resolve conflicts between trains during the solution procedure rather than having to generate them all a priori and explicitly represent the infeasiblities. This being the case, we can easily consider additional trains, additional routes, and delayed trains. Identifying and resolving conflicts is embedded in the optimization and there would be a limited amount of additional effort required to include such extra variables. Providing that they represent feasible paths through the junction, they are nothing more than extra columns in the model.

The above model is very similar in structure to a well known optimization model known as the crew rostering model developed by Ryan (1992). Indeed, one could regard a train as being a crew member and the sequence of time period track section pairs it claims as its sequence of duties. Thus the columns defining routes for trains are analogous to the columns defining lines of work for crew members. From a modelling perspective the only noticeable difference between the two models appears in the right hand side. In the crew rostering model this is not restricted to being of packing nature. In terms of the dimensions of the two models there is a major difference. The crew rostering model is characterized by a small row dimension with many variables. It is not uncommon for crew members to have hundreds of thousands of possible lines of work, however, there are only relatively

few constraints corresponding to the duties that must be allocated. The set packing model above, on the other hand, is characterized by a constraint matrix with many constraints and relatively few variables. Any reasonably sized junction, and reasonable timetable period will lead to a problem with thousands of time period track section constraints. In contrast, trains only have a limited number of sensible routes through the junction.

To determine the best solution approach to adopt, one should consider what properties an optimal solution to the primal problem (LP) would have. We would expect there to be, comparatively speaking, few variables. We also believe that there would be a significant number of inactive constraints: It seems highly unlikely that all track sections will be locked in all time intervals. The dimensions of the primal problem naturally suggests one should consider the dual formulation. In the dual setting at optimality these observations would be mirrored in the form of very few active constraints, and most of the dual variables being non-basic at value zero. The dual formulation has a similar dimension to the crew rostering model. Experience tells us that such models can be solved efficiently. The solution approach that we have adopted, which is explained in detail in Section 5, hence focuses on the dual and attempts to exploit its small row dimension as much as possible. For completeness, the dual formulation of (2) is given below.

$$\begin{aligned} \text{Minimize} \quad & e^T\pi_1 + e^T\pi_2 \\ \text{subject to} \quad & A^T \begin{pmatrix} \pi_1 \\ \pi_2 \end{pmatrix} \geq p \\ & \pi_2 \geq 0, \end{aligned} \tag{3}$$

where all notation is as previously defined.

4.1 The Integer Properties of the Primal

The strength of the LP relaxation is an important factor for any integer program. Let us consider the nature of the LP optimal solution of our primal model. The generalised set packing model includes significant structure and has strong integer characteristics. The constraint matrix is partitioned into train blocks, as indicated by the T_i and R_i submatrices. The T_i submatrices can be identified as the GUB constraints, while the R_i submatrices show the contribution of particular train paths (of the same train) to the set packing constraints. Each column of R_i pertains to a possible path through the junction for the train it represents.

We know that fractions in the LP relaxation of the set packing model will be due to the presence of odd order two cycles (odd order submatrices with row and column sums equal to two) in the constraint matrix (see Berge, 1972). The addition of a GUB constraint for each train, however, removes the possibility for such fractions within each train block as it makes each a perfect matrix. Perfect matrices permit odd order two cycles to exist, but ensure potential fractions are infeasible through the addition of a constraint which is not part of the odd order two cycle (see Padberg, 1973). Thus, fractions will only be due to the presence of odd order two cycles across multiple blocks. In other words, fractions will be the direct result of two different trains competing for the same track section in the same time period. While we expect fractions to occur, the cause should be relatively easy to identify given the nature of the problem.

5 Solution Approach

The solution approach we propose consists of two main steps. The first step involves finding all possible train paths (variable generation), while the second entails solving the problem. A detailed description of each is given next.

5.1 Variable Generation

In our modelling we assume that all train routes are generated a priori, but not train paths. Modelling train paths essentially involves determining the distribution of zeros and ones in a particular column. In other words, indicating when and at what time a train will lock particular track sections during its passage through the junction. This depends on two key factors. The safety system enforced by the rail company stipulates how many additional track sections will need to be locked, while the dynamics of the trains (the acceleration and deceleration capabilities) will determine the exact running time of a train on a given track section.

In modelling train dynamics we make several assumptions. Whenever a train enters a new track section in its route, it is limited to doing the following:

- Proceeding with constant velocity,
- accelerating at a constraint rate,
- decelerating at a constant rate.

This approach of constant acceleration and deceleration rates is consistent with previous work on modelling train dynamics; Zwaneveld et al. (1996) and Lu et al. (2004) both adopt a similar approach. We believe it to be a realistic approach in that it generates only practically feasible paths. Extra restrictions are also included to ensure a train will not accelerate immediately after decelerating and vica versa. This is to ensure only smooth train paths are considered.

We assume that for each of a train's possible paths, the entering track section, entering time, and entering speed of the train is known. We also make the assumption that a train will not be accelerating/decelerating on entry to a junction. This defines the initial point of the train path, and is represented as a time space node which stores this information. Similarly, a time window in which the train may depart the junction, as well as the leaving track section are also known. In addition to this each train is assigned a maximum permitted amount of time it can be in the junction. The first time space node is then extended based on the dynamics of the train given the possibilities outlined above. A new time space node will be generated each time a train enters a new track section on its route. This process continues until all possible train paths have been enumerated. Train paths will terminate as feasible if they leave the junction in the designated time window. However, they will terminate as infeasible if this is not the case, or if the duration of the train run through the station exceeds the maximal tolerance. A diagram illustrating this process is given below.

The figure above represents the possible ways a train may traverse the first three track sections of its route. The number of the track section it is entering is given on the node. The successor nodes are generated based on the dynamics of the train. The arrows emanating from nodes correspond to the different possible kinematic options available.

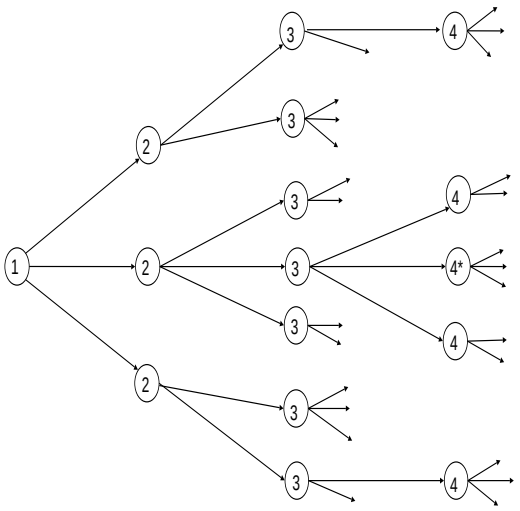

Figure 2: Modelling Train Paths.

Each track section is assumed to have both a minimum and maximum permitted velocity. This could be an infrastructure restriction (i.e. speed limit), or the maximum permitted speed of the train. As a result of this, one needs to be careful when extending nodes. To ensure feasible extensions are guaranteed a backwards preprocessing step is used to adjust the maximum and minimum velocities on a given track section. This prevents a train from reaching a situation in which it will be unable to comply with the speed requirements of the subsequent track section.

As can be seen from the diagram, the first three track sections can be traversed via a number of different acceleration and deceleration patterns. The fact that we consider only constant rates of acceleration and deceleration, and that different track sections often have different lengths, means it is extremely unlikely that duplicate time space nodes will be generated via two different traversal patterns. On the rare occasion that this does occur, removing one of the labels through domination would also be unlikely. This is primarily because each time space node is defined by an entering time, entering track section, entering speed and entering acceleration. All of which would need to be equal for domination to occur. This means that the modelling described above in fact builds a tree structure. For each route of each train there will be a different tree. On traversing any tree one would obtain all possible ways to traverse the corresponding route.

5.2 Primal Variables as Tree Structures

The representation of the train routes as tree structures is advantageous from a computational point of view. It allows one to price the paths (primal variables) contained in the tree very efficiently. Using Figure 2 as an example once more, the arcs indicate the length of time the train spends on the track section associated with the node it emanates from, hence telling us which primal constraints all train paths containing this arc would cover. We can easily store an accumulated dual value at each node by summing the appropriate dual variables along the arcs leading to it. Through assigning each node an accumulated dual value one has effectively priced out a component of the reduced cost for all train paths using this arc. For example, all subsequent nodes of 4* would have the accumulated dual variables to that point in common. This is much more effective than pricing individual

variables as one only ever needs to back track to the first arc of difference on pricing a different train path; equivalent to a pre-order walk of the tree. If the accumulated dual of a leaf node (which represents a complete path) is favourable, the variable it represents is easily obtained by tracing back through each predecessor node.

5.3 Solution Method

To solve the set packing model above we propose an approach which attempts to exploit the small row dimension of the dual formulation as much as possible. Essentially the method involves dynamically updating the dual problem through the addition of primal entering variables as violated dual constraints.

We begin with an initial dual problem which has just one constraint for each train. This is analogous to having a restricted master problem in the primal setting. We elect to assign each train its null path. If we allow only one path for each train to be in this initial problem, it can be shown that the dual problem is unbounded if there is a conflict among the chosen paths. Without including more paths in the initial dual problem, or starting from a suboptimal conflict free routing, we believe the assignment of trains to null paths is a valid starting point.

The solution to the dual problem gives us the dual variables for the primal set packing LP. This solution vector can easily be used to price out the tree structures for favourable train paths. A primal variable (dual cut) pool is also set up (initially null) which will store a subset of the paths. This primal variable pool is priced prior to the trees.

In our pricing of the tree structures we are effectively implementing a form of partial pricing. We only ever call the pricing routine for a particular train and a particular route, and thus initially focus on one tree. If a favourable train path is found in this tree it is returned immediately to the optimization in the form of a violated dual constraint. The reduced cost of the primal entering variable is the extent to which the corresponding dual constraint is violated. Hence, on appending the constraint to the dual problem, an artificial variable with an initial value equal to the reduced cost is included to ensure we have a starting basis for the next iteration. The dual problem is then reoptimized to obtain updated dual variables, and the process repeats itself. In effect, we equate an iteration of the primal simplex with the addition of a dual constraint. If, on the other hand, a favourable train path is not found in the tree called for, another tree (for a different train or route) is examined. This continues until either a primal entering variable has been found, or we have priced all trees. In the latter case, we would declare optimality with the optimal solution to the primal problem being the dual vector of the optimal solution to the dual problem.

In an effort to maintain a small dual basis throughout the solve an inactive constraint removal routine is also implemented. If at a particular solution to the dual an inactive constraint appears, it is removed. Inactive constraints in the dual indicate that the corresponding primal variable is at value zero and can be removed. In the next section we will demonstrate how this dynamic approach of adding and removing constraints is effective in keeping the dual basis small.

6 Example Junction

To demonstrate our model and solution approach we test it on the junction given in Figure 3. While this junction is fictional and purely for expository purposes, it has been created

based on the Pierrefitte-Gonesse junction which is used in the test cases of Delorme et al. (2001), Delorme et al. (2004), Delorme (2003), Gandibleux et al. (2005), and Rodriguez (2002). The purpose of this was to simulate a real life situation as much as possible. The test junction consists of 50 track sections. Each of which has an upper and lower speed limit and is either unidirectional (indicated by an arrow) or bidirectional. The different directions entering/leaving the junction have been arbitrarily labelled A,B,C, and D.

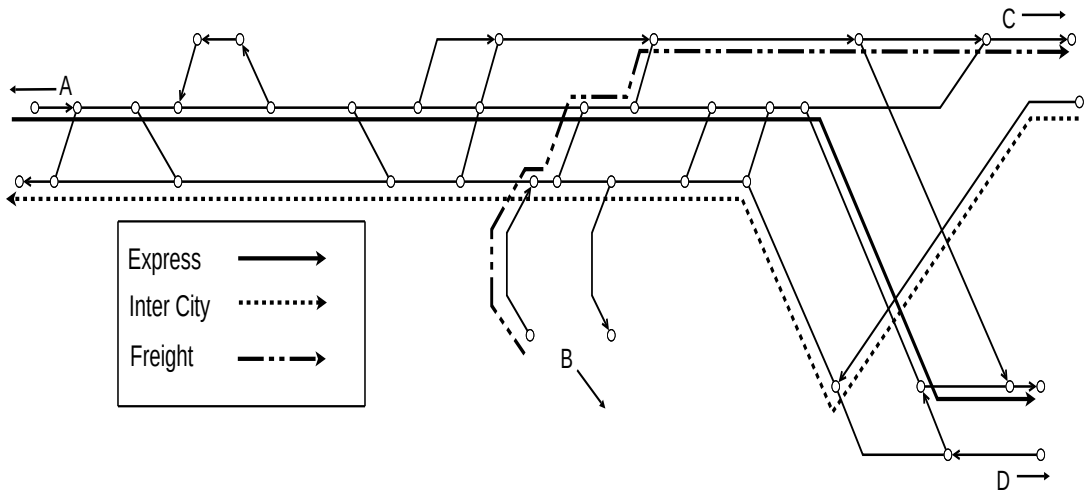

Figure 3: Test Junction.

The instance we consider consists of nine trains. All trains enter the junction within 645 seconds of each other and the timetable period is 1140 seconds. The trains are one of three types; namely Express, Inter City, or Freight. The difference in train category is reflected in the acceleration and deceleration capabilities of each. The values used are consistent with those of Danish trains; again in order to reflect a real life situation. In this particular case trains have between one and three possible routes (an example of each is given in the diagram). The train data is summarized in Table 1. The entrance and exit times are given in elapsed seconds from the start of the timetabling horizon. The relevant direction is bracketed.

Table 1: Train Data

Train	# Routes	Entering Time	Latest Exit Time	Type
1	2	0 (B)	800 (C)	Freight
2	3	80 (A)	660 (C)	Inter City
3	3	120 (D)	600 (A)	Express
4	2	140 (A)	760 (D)	Express
5	1	145 (C)	1140 (B)	Freight
6	3	160 (B)	1140 (C)	Freight
7	1	320 (C)	1140 (A)	Inter City
8	2	620 (A)	1140 (C)	Express
9	2	645 (D)	1140 (A)	Express

As we explained in Section 5, the first step in the solution process is to generate the necessary tree structures. For this problem 19 such trees are required. The largest tree

contains 492,907 nodes representing 262,142 possible paths, while the smallest consists of 73 nodes representing 37 paths. In total, the 19 tree structures contain approximately 729,000 possible paths. In reality we believe it is unlikely that there would be so many possible ways to traverse the routes. However, by considering trees that are bigger than we would expect, we can gain an insight into the time required to initialise trees in general. For the problem at hand all 19 tree structures are initialized in less than three seconds.

The pricing of the trees is also very quick. To price all the paths in a particular tree involves a pre-order traversal of the tree and one would expect this to be on par with, if not faster than, the time required to generate the tree itself. Indeed this is the case. Small trees can be priced in fractions of a second, while the larger ones may take slightly longer. The largest tree in the problem considered here takes no more than half a second to price all 262,142 variables. The traversal code used to price the trees has been stress tested on a tree containing 1.7 million nodes. The 1 million variables that this tree represented took between two and three seconds to price. We believe this to be very encouraging as trees are not likely to ever contain this many possible paths. Table 2 below provides some statistics for the trees. S refers to the smallest tree, L refers to the largest tree, and #Paths (T) gives the total number of paths for each train.

Table 2: Tree Data

Train	#Nodes (S)	#Paths (S)	# Nodes (L)	# Paths (L)	# Paths (T)
1	1,362	790	5402	3,315	4,105
2	17,473	7,888	432,732	158,428	265,140
3	22,491	11,291	56,782	27,647	62,282
4	14,234	6,347	492,907	262,142	268,489
5	73	37	-	-	37
6	2,287	1298	6,714	4,052	8,765
7	20,875	12,098	-	-	12,098
8	7,286	2,425	82,251	18,386	20,811
9	13,800	6,865	192,585	83,083	89,948

With our approach the problem took less than five seconds to solve. This includes the time involved in generating all paths. The results were obtained on a Pentium 4, 3GHz computer with 750MB of RAM. We used FORTRAN and C++ for the programming. The following statistics are interesting to note. During the solve 44 constraints (train paths) were added to the dual, 34 were removed, and there were never more than 23 constraints in the dual basis. The dual problem never contained more than 5 paths for the same train. The small number of active constraints at any solution to the dual indicates that the number of primal variables at positive value is small. The primal problem has some 3,808 constraints for this example, and hence one would expect a highly degenerate solution if solving the primal problem. We have demonstrated here that by solving the dual we can maintain a very small basis throughout the solve and in doing so restrict our focus to a smaller set of primal variables.

The fractional solution obtained by the solver indicated that it is possible to achieve an objective of nine (equivalent to routing all trains) by fractionally assigning paths to trains. Integer solutions found by the dual during the execution of the solver show that a lower bound of 7 on the number of trains that can be assigned a path can be obtained. From the optimal fractional solution we can easily identify where a conflict occurs, and which

trains are involved. Table 3 below provides information regarding the clashes present in the optimal solution to the LP relaxation.

Table 3: Clash Table

	Clash 1	Clash 2	Clash 3	Clash 4
Trains involved	6&8	6&8	4&6	4&9
Time Period	56	67	38	45
Track Section	19	21	10	35

Although no specific branch-and-bound routine is included in the software at this stage, to ascertain whether or not an integer solution routing all trains exists, a form of primal constraint branching was implemented. A conflict was chosen arbitrarily from the optimal fractional solution and a decision as to which train would use the particular time period track section was made (again arbitrarily, although in reality one of the two trains may have a higher priority than the other). As a result of this certain variables for certain trains were no longer feasible. The problem was then resolved with a smaller set of variables. Three such branches were required to show there did exist an integer solution which routed all trains. This indicates that this formulation of the problem naturally lends itself to the constraint branching framework. Future research will look at better techniques to identify the branch to impose.

If the objective value is ever less than the number of trains being considered, we can confidently state that the proposed timetable is infeasible and the only way of achieving feasibility would entail changing the timetable, or considering alternative routes for trains.

7 Conclusions and Discussion

In this paper we have shown that the problem of routing trains through railway junctions can be formulated as a set packing problem and solved efficiently via a procedure which dynamically updates the corresponding dual problem through the addition of violated cuts as well as the removal of redundant constraints. The example in Section 6 demonstrates how one can maintain a dual basis throughout the solve by adopting this approach. This is, of course, extremely beneficial from a computational point of view.

We have also demonstrated that by representing the train routes as tree structures we can price a significant number of train paths very efficiently. Initializing the tree structures also requires little time. While the trees generated for the example in Section 6 are much bigger than we would expect in general, it is very encouraging to see that even these can be priced extremely quickly. We believe the modelling technique and solution approach described herein to be very promising.

We have identified a number of directions for future work on this project. Firstly, as was mentioned in Section 6, a branch-and-bound routine is not yet included in our software. This is a vital component of any solver for integer programs. We intend to implement primal constraint branching with the added complexity that it must be carried out in the dual setting. The branching routine we are hoping to implement will determine the maximum number of trains that can be routed and, in the event of an infeasible timetable, adjust routes by the minimum amount of delay required to avoid conflicts and thus route all trains. A branch is enforced by removing any variable in the current basis that does not honour the branch, as well as preventing banned variables from being priced through

the inclusion of some mechanism in the pricing routine. Removing banned variables from a primal basis equates to making active constraints in the dual inactive. A possible mechanism we are considering to prevent banned variables from being priced is to generate a new tree structure at each node in the branch-and-bound tree in which all paths honour the branch. This will ensure we are only ever dealing with permissible paths for trains.

The size of the tree structures will become very large if there are many track sections in the routes. This is because trains do have a number of possible ways to traverse a given track section. If a track section is quite short, then there will only be a small difference in the time space nodes generated for each of the possible traversals of the track section. A method which aggregates a series of similar track sections into one will be beneficial in that it will reduce the size of the tree structures as well as the number of possible paths.

In a similar way one can aggregate paths. It is quite likely that a number of the time space nodes generated will have similar characteristics, which in turn means a number of the paths generated will be similar. Evidence of this can be seen in the tree structures of Section 6, where we believe it is extremely unlikely a train will have realistically 260,000 different ways to traverse a particular route. Hence a method which collapses similar time space nodes into one single node will also be successful in reducing the number of paths possible. Both track section aggregation and path aggregation will be beneficial from a computational point of view.

A pre-analysis step which involves searching the tree structures looking for possible conflicts will enable us to predict the likelihood of fractions being in the optimal solution to the LP relaxation. As discussed above, fractions will only occur as a result of two different trains competing for the same track section in the same time interval. If a pre-analysis step showed that there were relatively few conflicts we would expect the number of active constraints with fractional dual variables at optimality to be quite small. On the other hand, if there were a significant number of conflicts found in the pre-analysis step, it would indicate that there is quite a high possibility of a fractional solution. Such information prior to solving the problem would be useful.

In our modelling we have assumed that trains do not have preferences for routes, nor that there is a priority list amongst the trains. This is not realistic in practice. We plan to include weights on train routes to indicate the respective preference of the trains for each of its routes. This is so that favourable paths for trains are chosen. A priority list for the trains would also be useful in the event of clashes occurring.

Finally, we would like to extend this approach to similar railway problems. For example, the problem of dispatching trains along a single stretch of track connected by multiple stations is one problem in the railway industry that has received significant attention. The software we are currently developing could easily be extended to solve this problem.

References

C. Berge. Balanced matrices. *Mathematical Programming*, 2(1):19 – 31, 1972.

G. Caimi, D. Burkolter, and T. Herrmann. Finding delay-tolerant train routings through stations. In H. Fleuren, editor, *Operations Research Proceedings 2004: Selected Papers of the Annual International Conference of the German Operations Research Society (GOR) Jointly Organized with the Netherlands Society*, pages 136–143. Springer Verlag, 2005.

M. Carey and M. Carville. Scheduling and platforming trains at busy complex stations. *Transportation Research, Part A*, 37A(3):195 – 224, 2003.

X. Delorme. *Modélisation et résolution de problèmes liés à l'exploitation d'infrastructures ferroviaires*. PhD thesis, Université de Valenciennes et du Hainaut Cambrésis, 2003.

X. Delorme, X. Gandibleux, and J. Rodriguez. GRASP for set packing problems. *European Journal of Operational Research*, 153(3):564 – 580, 2004.

X. Delorme, J. Rodriguez, and X. Gandibleux. Heuristics for railway infrastructure saturation. *Electronic Notes in Theoretical Computer Science*, 50(1):39 – 53, 2001.

X. Gandibleux, J. Jorge, S.Angibaud, X. Delorme, and J. Rodriguez. An ant colony optimization inspired algorithm for the set packing problem with application to railway infrastructure. In *Proceedings of the Sixth Metaheuristics International Conference (MIC2005)*, pages 390–396, 2005.

Q. Lu, M. Dessouky, and R.C. Leachman. Modeling train movements through complex rail networks. *ACM Transactions on Modeling and Computer Simulation*, 14(1):48 – 75, 2004.

M.W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5(1):199 – 215, 1973.

J. Rodriguez. Constraint programming for real-time train circulation management in railway nodes, October 2002. Presentation from the 3rd AMORE Research Seminar, The Netherlands.

D.M. Ryan. The solution of massive generalized set partitioning problems in aircrew rostering. *Journal of the Operational Research Society*, 43(5):459 – 467, 1992.

P.J. Zwaneveld, L.G. Kroon, and H.E. Romeijn. Routing trains through railway stations: complexity issues. *European Journal of Operational Research*, 98:485 – 498, 1997.

P.J. Zwaneveld, L.G. Kroon, H.E. Romeijn, M. Salomon, S. Dauzere-Peres, S.P.M. van Hoesel, and H.W. Ambergen. Routing trains through railway stations: Model formulation and algorithms. *Transportation Science*, 30(3):181 – 194, 1996.

P.J. Zwaneveld, L.G. Kroon, and S.P.M. van Hoesel. Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128: 14 – 33, 2001.