The Maraca – a tool for minimizing resource conflicts in a non-periodic railway timetable

Malin Forsgren†, Martin Aronsson†, Per Kreuger†, Hans Dahlberg‡
†Swedish Institute of Computer Science (SICS), Box 1263, SE-164 29 Kista, Sweden
email: {malin,martin,piak}@sics.se
‡Trafikverket, SE-172 90 Sundbyberg, Sweden
email: hans.dahlberg@trafikverket.se

Abstract
While mathematical optimization and operations research receive growing attention in the railway sector, computerized timetabling tools that actually make significant use of optimization remain relatively rare. SICS has developed a prototype tool for non-periodic timetabling that minimizes resource conflicts, enabling the user to focus on the strategic decisions. The prototype is called *the Maraca* and has been used and evaluated during the railway timetabling construction phase at the Swedish Transport Administration between April and September 2010.

Keywords
Timetabling, conflict minimization, case study

1 Introduction

Mathematical optimization and operations research have found their way to an increasing number of application areas in railways in recent decades. For timetable generation, however, there seems to be a particularly large step to turn the research developments into actual and usable tools.

In a review of models and algorithms for railway traffic scheduling and dispatching from 2006, the author states that only 4% of the reviewed approaches had actually been implemented [15]. Although aimed at research papers presented between 1973 and 2005 and not specifically at models and algorithms used in available timetabling and dispatching tools on the market, this review and previously published overviews nevertheless indicate that the number of cases where optimization have actually been used for timetable generation remain relatively modest [11].

When it comes to timetables, one of the main reasons for the lack of practical examples of hands-on use of optimization tools is that it's very difficult to formulate a meaningful cost function rendering the best – or at least a good – timetable. The difficulty lies in the fact that there is no clear-cut definition of 'good' in mathematical terms.

This dilemma is well-known to everyone formulating and modelling real-world problems for optimization. For the same reason, an optimal result in the mathematical sense is not necessarily what the end user needs. Getting a good enough answer quickly is often what matters in industrial applications. Waiting for a slightly better solution or to prove optimality does not provide any added value unless the data used is exact and the cost function

fully represents and covers what we want to optimize. The latter is very rarely the case.

This paper briefly presents a case study at Trafikverket, the Swedish Transport Administration, of *the Maraca*, a prototype tool for timetabling developed at SICS. The purpose of the tool is to aid the timetable constructor by minimizing the resource conflicts in his or her draft timetable. The tool took its name after a percussion instrument by suggestion from an employee at Trafikverket when we had used the allegory of shaking, or rattling, the trains into their proper places in the timetable to describe what our tool did.

Generating a timetable with *the Maraca* is an iterative – and interactive – process. While scalability is important, we know that many 'soft' constraints are not in the model at all. It is therefore safe to assume that the user would rather abort a computation that takes a lot of time than wait for it to finish since the 'optimal' solution found would most likely not be the final one anyway.

A brief discussion about related work can be found in Section 2. The timetabling problem is defined in Section 3, while the underlying model is outlined in Section 4. The tool itself and its capabilities are described in Section 5. We report practical results in Section 6, and end with some conclusions and ideas for improving the tool in Section 7.

2 Related Work

The transportation needs in a country depend on geography, how the country is populated, where industries are located, and many more factors. No two countries are alike in these respects, which is one of the reasons why railway timetabling practices can differ quite significantly from country to country. This is naturally reflected also in the research on the topic.

The Swedish situation differs considerably from for instance the Dutch. In The Netherlands, the whole basic timetable structure exists before the operators request capacity, which is not at all the case in Sweden. The current basic Dutch timetable was made from scratch with the help of OR techniques and has been in operation since December 2006. It replaced a timetable structure that had been in use since 1970 [9].

Researchers in The Netherlands and Switzerland, two countries of comparable size (approximately 41,000 square km) with among the world's most heavily utilized railway networks [9], are faced with interesting timetabling problem challenges. Their respective countries are however much more densely populated than Sweden and the prerequisites for timetabling are accordingly essentially different in Sweden: while Swiss and Dutch timetables can be described by modelling an hour of typical traffic as a Periodic Event Scheduling Problem (PESP) [14], the shortest period of time that makes sense for a Swedish timetable is a whole day, making the PESP approach much less suitable here.

Despite national differences, the basic problem structure is the same, and for all countries with dense traffic, finding a valid timetable for even parts of the country is a very complex task regardless of the size of the country. The complexity is tackled by simplifying the network and its traffic, in combination with either dividing the network into different regions and solving them separately [5, 13], using clever heuristics [6], or employing a combination of all of these.

For an extensive survey of related work, Lusby et al have written an excellent and recent report about models and methods for railway track allocation, of which the timetabling problem is an important subproblem [12]. Especially the chapter about single track railway networks contains approaches that are similar to our own.

3 The Railway Timetabling Problem

The railway timetabling problem that we base this paper on requires a definition of the conflict concept.

Definition 3.1 If two trains are scheduled in a way such that they both need the same resource (track or other entity of the network) at the same time, they are said to be in conflict. The number of seconds they are in conflict is called the number of *conflict seconds*.

The need for a common resource might only be partially overlapping in time, but as soon as there is an overlap of any size, this is considered as a conflict in our model. To compare how serious conflicts are relative to each other, we specify it in the number of *conflict seconds*.

Problem 3.1 Solving the railway timetabling problem for a specific region means scheduling trains in that region in such a way that they can traverse the tracks in a conflict-free manner while also respecting all other constraints that have been imposed upon them.

Note that the timetabling problem as defined above often does not have a feasible solution. To find a timetable in practice, the only way to go about is to relax the constraints. Thus, the point of *the Maraca* is *not* to solve the timetabling problem, but to aid a timetable constructor in finding a timetable that is close enough to the original requirements. *This is the key to* the Maraca *and cannot be emphasized strongly enough*.

4 The Model

The model presented here has been refined for approximately five years and reflects the Swedish circumstances correctly enough to be truly useful in practical examples. It is however still evolving. Liebchen and Möhring, half serious, half joking, aptly refer to the process of constantly having to adjust the model to incorporate previously unknown restrictions as a "verbal cutting-plane algorithm" [11].

While it is possible to parameterize different cases that will arise and have the user choose the appropriate set of parameters prior to any computations, it is of course not realistic to attempt to make an exhaustive list of all cases that can possibly emerge in the railway timetabling context. The situation is too complex.

There seems to be no way around the verbal cutting-plane algorithm mentioned above. A practical tool has to allow the user to add exceptions to the model on the fly relatively easily. We proceed to give an example taken from the real-world case that this paper is based on.

Example: The Station Model The stations in our model were initially simply locations that could host a certain number of trains at the same time. However, in a more realistic model we cannot assume the existence of dynamic loops at all stations (allowing for trains to pass simultaneously without having to stop), so it soon became clear that this did not adequately capture the behavior of the trains at the stations. Stations were therefore categorized according to how much time is required between the arrivals of two trains at the location.

Recently, while sitting next to a timetable constructor during the timetabling process, we discovered that the timetable constructor at one instance chose to plan in a way that blocked the traffic in one direction with a train for the duration of its stop and turnaround time, even when there was an alternative that would use much less capacity.

At first glance the decision seemed counterintuitive, but of course the timetable constructor had a valid reason. The station was next to a school, and due to previous incidents at this station he didn't want any children to have to cross the tracks while walking between the station and the school.

We didn't yet have any means of mimicking this behavior in the model, but because it consumed so much capacity we could not turn a blind eye to it. Without going into detail, the normal equations posted to regulate trains at this station had to be complemented to make sure that the solution would not contain a take-over in the concerned direction before this train would be out of the way.

It is obvious that exceptions like the one in the example above will always arise. And if the tool lacks any smooth means of letting the user incorporate them immediately as they arise, the user will likely not accept the solution that the tool provides him or her with.

4.1 Choice of Abstraction Level

Inherent in the concept of a model is that it is a simplification of the reality it is intended to capture. Certain aspects of the railway network and the supposed traffic in it are simplified and abstracted in the planning process. What is significant and what is not, is actually only a question of where to draw the line. The heart of the matter is what level of detail we actually need for the purpose at hand.

The detail level chosen for a model can of course always be challenged. The downside of any abstraction is that it does not merely hide the details – it lacks any knowledge of them. This is in fact also the merit of an abstraction. Stripping the problem of unnecessary detail might make a computationally intractable problem tractable, and it will provide an overview that might be invalueable.

The tool we have implemented that has been used for the experiments presented in this paper was always intended to be used in connection to TrainPlan, the timetabling tool used at Trafikverket today. Therefore, the general level of detail is deliberately chosen to be close to what TrainPlan requires. This concerns primarily the geography, which is more or less imported straight from TrainPlan to our tool. Since TrainPlan is, at Trafikverket, used mainly as a visualization tool, it does not itself need any knowledge about station layouts and how trains may and may not interact there. The timetable constructor alone is responsible for creating a conflict-free timetable.

Sometimes we come to a point where our simplification is too coarse to be useful because the abstraction fails to encapsulate what we need to know in order to be able to model the real world to the detail level that is called for. In the example at the beginning of this section, the need for complementing the model for a special case would not have come up if we had already had the station layout and the movements of the trains at the station accurately modelled. In this case, the abstraction we use is good enough for nearly all stations, and since it is possible to use a more detailed model locally for the special case only when the need for it has been identified, it is clearly better to do so than to clutter up the whole model with too many details.

4.2 Mathematical Formulation

The model consists mainly of a combination of precedence and resource constraints capturing most of the conditions on rail traffic relevant to timetable generation. The most important precedence constraints are those ordering the track traversals and dwell times at stations on the path of a train moving through the geography. Resource constraints include station capacity constraints as described in [2] and a type of non-overlap condition on track sections which will be described in some detail below.

The model differs significantly from models in common use for Periodic Event Scheduling Problems (PESP) since the kind of problems we consider are periodic only on a much larger time scale than normally considered and sometimes also contain aperiodic events. Our model is more similar to general scheduling models such as those for job shop problems.

There are a number of additional constraints in the system, such as overlap conditions on stops for pairs of trains necessary for passenger or goods transfers and certain additional separations between incoming trains for some stations.

Here we will concentrate on the model for the track resources which is a variant of that formulated in [1]. The main addition compared with previous versions is that we provide a mechanism to relax each constraint individually. This makes it possible to associate a cost with the amount of breakage of each condition, which has turned out to be very useful in the context of a support system for semi-manually constructing a large-scale timetable.

We do not model each individual track section down to the level of the automatic train control and signaling elements but divide the tracks into longer stretches connecting points and stations in accordance with Trafikverket's geography files from TrainPlan. These tracks are frequently used in both directions which has to be handled differently than when all trains on a section move in the same direction.

For two opposing trains on a single track section, the time separation between their time of entry to the section is required to be at least equal to the traversal time of the train first entering it. The traversal time on the track sections are picked from a database to reflect the actual movement of the train based on whether it starts and/or ends in a full stop, again according to the model in TrainPlan. At stations, there is a common restriction that trains must arrive with some minimum time separation, typically either one, two or three minutes depending on the station, complying with common practice and abiding to the regulations at Trafikverket [3].

We can argue that even more time than the traversal time of the first train has to elapse before a second (opposing) train may enter a section since we have to take the clearing time and the setting-up of new routes into account – among other things (see e.g. [10]). However, this assumes a more detailed model than currently implemented at Trafikverket in TrainPlan. Any extra time judged to be necessary by the timetable constructor can be added manually, and any such added time also carries over to our system in the optimization step.

When two trains move in the same direction the condition becomes more involved (see [8]) but it boils down to a headway that depends on both the length of the signal blocks on the section and the relative speed of the trains in question. This headway is constant and is below captured in the same way as in the case of opposing trains. The only difference is that the task duration d_i in the first case is the traversal time of the first train to enter the section, and in the second case a headway calculated from properties of the track section and the difference of the traversal times of the trains on the section.

4.3 Relaxable Scheduling Constraints

First, let us express the simplest possible non-overlap constraint for a task ri representing the train i traversing the track resource r: either the end time (the start time s_{ri} plus the duration d_{ri}) of task ri is less than or equal to the start time s_{rj} of task rj: $s_{ri} + d_{ri} - s_{rj} \leq 0$, or the same is true for task rj in relation to task ri: $s_{ri} - s_{rj} - d_{rj} \geq 0$. We reflect this disjunction in a boolean variable p_{rij}:

$$s_{ri} + d_{ri} - s_{rj} - M\,(1 - p_{rij}) \leq 0, \tag{1}$$

$$s_{ri} - s_{rj} - d_{rj} + M\,p_{rij} \geq 0 \tag{2}$$

where M is any constant large enough to dominate the equation in which it occurs. This is, of course, a standard formulation that occurs everywhere in the literature (see e.g. [4, 7]) but how do we proceed if we want to *count* and relax individual constraints of this type and e.g. optimize for the least number of such broken conditions?

In the case where we *do* want to allow an overlap we need an additional boolean that cancels the effect of the above equations. We want to do this in a way so that whenever this variable takes the value 0, our equations will be equivalent to the ones above, and cancel them out otherwise:

$$s_{ri} + d_{ri} - s_{rj} - M_1\,(1 - p_{rij}) - M_2\,w_{rij} \leq 0, \tag{3}$$

$$s_{ri} - s_{rj} - d_{rj} + M_1\,p_{rij} + M_2\,w_{rij} \geq 0. \tag{4}$$

In this formulation the booleans p_{rij} express the ordering of the trains on the track and w_{rij} if the condition may be broken or not.

By setting M_2 to 1 and relaxing the binary requirement on w_{rij} we may instead, in the search for solutions, use it in the cost function to measure (and penalize) the total amount of time during which the resource is over-allocated.

4.4 Compact Representation of Very Large Schedules

Trains typically depart regularly over several different time scales. E.g. one train may depart on exactly the same time five times a week and at another time the remaining two. That week may be repeated for several months, after which we have an anomalous week in connection with a holiday after which the original week pattern is again repeated a number of times until there is a seasonal change which initiates another weekly pattern, and so on.

This type of pattern can be used to represent very large scheduling problems compactly. The idea here is taken directly from TrainPlan. It is similar to what Caimi refers to as the projection to a periodic instance [5]. The timetable for a full year is collapsed into a single day, but associated with each pattern i, in turn associated with a specific train, is a set of days $\mathbf{t}_i$ for which the pattern is active. Thus the same train may occur several times in different variants, all appearing in the collapsed single day, but with different day patterns. Each pattern is represented only once.

All resource constraints are filtered with respect to the day occurrences so that an identified conflict between two trains i and j is recognized only if $\mathbf{t}_i \cap \mathbf{t}_j \neq \emptyset$ also holds. This is a key to the scalability of the model to the long-term scheduling problem posed by the real industrial case, where all trains for a whole year may need to be scheduled at once.

4.5 Cutting the Solution Space

This section provides a motivation for the way we radically reduce the size of the original timetabling problem – a problem that is computationally intractable – to something for which a solution can actually be found within a reasonable time frame.

Swedish Law and Directive from the EU
According to Swedish law, The Railways Act (2004:519), the infrastructure manager has to treat requests for capacity and allocate capacity to applicants in a non-discriminatory way that ensures competitive neutrality. This is in line with a Directive from the European Union (Directive 2001/14/EC). The same Directive also imposes requirements on the coordination process:

- "During the scheduling process . . . , when the infrastructure manager encounters conflicts between different requests he shall attempt, through coordination of the requests, to ensure the best possible matching of all requirements." (Art 21 point 1)
- "When a situation requiring coordination arises, the infrastructure manager shall have the right, within reasonable limits, to propose infrastructure capacity that differs from that which was requested." (Art 21 point 2)

Whenever two or more railway undertakings request capacity in a way that cannot be satisfied simultaneously, the task of Trafikverket is to resolve the conflict while remaining neutral. Trafikverket first talks to the operators in question. If they do not voluntarily change their requests so that all of them can be accomodated even after Trafikverket hosts meetings where all the parties attend and discuss the situation, Trafikverket formally declares the infrastructure congested [17].

Conflict Resolution Tools Available
Since last year, once the infrastructure has been declared congested, a newly developed mechanism is triggered, which in the end determines who gets access to the infrastructure and in what way. This mechanism is transparent to the railway undertakings, and is as such neutral in the sense that no operator will gain commercial advantage as a result of human-made, subjective decisions at Trafikverket.

We will not go into great detail about this process, only highlight the parts that are relevant to our own research. Observing the Directive, and the Railways Act (Chapter 6, §3), the motivation for the mechanism is that the overall traffic solution with the greater benefit to society as a whole should prevail [17].

- "The priority criteria shall take account of the importance of a service to society, relative to any other service which will consequently be excluded." (Art 22 point 4)

At Trafikverket, the choice has been made to appreciate the contribution to society from a certain planned traffic by comparing it to the original requests made by the railway undertakings. Any deviation from their expressed wishes reduces the calculated worth of the traffic. Different transports belong to different priority classes (clearly defined and motivated before the process starts), and how much a deviation of a given magnitude affects the calculation depends on how important the transport in question is in a more general sense.

According to the Priority Criteria Model at Trafikverket, a train loses its commercial value altogether if its final path deviates too much from the original request [16]. Trains with no commercial value can be removed without affecting the value of the traffic to society, meaning that in theory a conflict-free solution to any problem is guaranteed to exist – even if it means that not all trains can be scheduled.

When resolving conflicts in a congested part of the infrastructure, Trafikverket will use the new tool to compare different timetable suggestions to each other. Ultimately this means that the different suggestions will be ranked by their importance and by how much they deviate from the operators' original requests.

Using priority criteria is not the final answer to the question "what is a good timetable", and should thus not be used in that way. They are only meant to constitute a good-enough and neutral (transparent) way of breaking ties when the negotiations have stalled. But they are bound to affect the whole process since it is indeed possible for an operator to adamantly refuse to accept any deviation from their original intent in case a train of theirs needs access to a particular part of the infrastructure at the same time as someone else's, and ultimately have the conflict settled by applying the priority criteria.

Introducing Domains

In an ideal world, we would want to first try to find a timetable entirely within the bounds given by the railway undertakings, and if no conflict-free solution can be found, we would relax the railway undertakings' requirements with a large-enough slack so that an optimized solution to the problem from the society's perspective can be guaranteed. Unfortunately, allowing the variables involved to take on any values while steering to find a solution matching the requirements in the best possible way via a clever formulation of the objective function works in theory, but not in practice.

First of all, the solution space for the problem sizes we want to consider would be enormous, meaning that computation times to solve the problems would be intolerable. Second, how to weigh a particular deviation from one requirement against another even on the conceptual level is not obvious at all, let alone how to evaluate the aggregated effect of several deviations. The Priority Criteria Model attempts to do the latter. As a tie-breaker it is acceptable, but we believe that applying priority criteria alone is not the way to go to obtain a good timetable solution as long as the model for honoring the operators' requests is not 'perfect' in all respects.

This dilemma motivates the idea behind our tool. The point is that we don't define what a good timetable is, only what a valid solution is (in terms of being free from resource conflicts). Having seen closely how a timetable constructor works, we appreciate that there will always be exceptions and softer constraints that need to be taken into consideration to get an acceptable solution, but they cannot all be included in the general model.

Instead of attempting a monolithic approach, we advocate the "human cutting-plane algorithm", involving an iterative process in which the timetable constructor takes all strategic decisions. The purpose of the tool is to only 'wash away' as much of the conflicts as possible from a draft timetable, aiming to make the least possible change to it in the process. The latter increases the chance that the strategic decisions made by the timetable constructor are kept intact. We believe that the user needs to recognize his or her own thinking in the solution that is produced by the tool, or else he/she will hesitate to use it, or at least not be as helped by it as would otherwise be the case.

Our approach manifests itself by the use of relatively small domains, or time slots, for

events. They are defined by providing tolerable time slots for the start and end times for a train (or the times for when the train enters and exits the various regions), and possibly for some of the intermediate arrivals or departures, and letting all other time slots be derived by time slot propagation. The problem that is built basically describes what a conflict-free timetable is, but allows resource conflicts to occur. The objective function is formulated to primarily minimize such conflicts. A secondary goal is to minimize the deviation from the given (draft) timetable.

5 The Prototype Tool: The Maraca

The tool takes an already existing schedule and optimizes it with respect to resource conflicts. The data for the trains in the existing schedule might be taken directly from the operators and reflect their ideas of when their trains should run, but the intended procedure is to base the computations on data representing a draft timetable at some stage in the planning phase.

To enable the test of *the Maraca* at Trafikverket, their version of TrainPlan was very recently extended to include the capability of importing XML (exporting the timetable as XML had already been implemented shortly prior to this recent extension). The import functionality is currently very slow, and the interface of our prototype tool is still not intended to be used by anyone else but us (the researchers), but all in all, we can say that recent developments has made real-world tests at Trafikverket possible.

5.1 Basic Conditions

The paths for the trains through the railway system are not subject to be changed by the tool. The minimum traversal times for the trains on the track sections are fetched from a database. Consideration is taken to the train type and if the movement starts and/or ends in a full stop. On top of the minimum traversal times, any number of seconds can potentially be added by the tool as long as all constraints are met (see Section 5.2). The same principle applies to the durations of dwell times.

5.2 Options and Parameters

The main options and parameters that are at the user's disposal are listed below. In addition, the tool can be 'tweaked' to handle other cases as they become relevant.

Variably Applied Slack Applying *slack* means allowing a particular event to deviate time-wise from its given time in the existing, underlying schedule. The tool allows the user to set two global parameters to control by how much train trips and dwell times generally can be prolonged. The same parameters can also be set individually per train.

New stops A new stop is a non-commercial train stop not present in the underlying timetable. The user can decide per location whether new stops can be introduced at that location or not.

Strict improvement Two trains that are originally in conflict for *x* seconds will be in conflict for *x* seconds or less in the solution provided by the tool if this option is turned on.

Preserve train precedences on links For each link (track section) and any two trains scheduled to traverse the link in a non-interfering way, one of the trains clearly precedes the other. A conflict occurs when the link usages of two trains overlap temporally in a disallowed way. Note that there is no meaningful precedence relation in a conflict situation. Minimizing the resource conflicts with fixed train precedences preserves a vital characteristic of the draft timetable and is an option in *the Maraca*.

The motivation for the "strict improvement" option is that it is often relevant to consider a conflict to be more serious the longer the two trains need the same resource simultaneously. To prevent the risk that, by minimizing the number of conflict seconds, 'less harmful' conflicts are worsened, we can choose to allow only strict improvements during the resource conflict minimization. In that way, no individual conflict is allowed to become worse.

5.3 Optimizing

As soon as more than one day of traffic is optimized, the weight of a resource conflict in the cost function is directly proportional to the number of days during which the conflict appears. If the option is used for which conflicts are also judged by how serious they are (see Section 5.2), this is also reflected in the cost function.

Assume that the days on which each train $i \in J$ departs, within the considered period (e.g. a year), is represented by a set of numbered days $\mathbf{t}_i$. For each pair of trains i and j in J, let $k_{rij} = |\mathbf{t}_i \cap \mathbf{t}_j|$ represent the number of conflicts between i and j should their usage of a resource $r \in R$ overlap in time. Let w_{rij} (as in Section 4.3) represent either (as a boolean) the fact that two trains i and j need resource r at the same time (the overlap may be partial), or (as a continous variable) the amount of time that they overlap. Then

$$\sum_{r \in R, ij \in J} k_{rij} w_{rij}$$

represents the cost of all the remaining conflicts in a schedule. The semantic reading of this is that the more frequent and/or serious a particular conflict is, the larger its cost, properly giving its solution higher priority during the minimization.

For any solution optimal in this sense, there may be many others with the same cost that differ only in the exact times that individual trains depart and arrive. In most cases, we would like the produced timetable to be optimal in terms of conflicts but deviate from the original proposal as little as possible. To achieve this we introduce, for each departure, a type of slack variable that measures the distance from the original proposal. We introduce into the cost function a sum of all such slacks and scale them with a factor of $\frac{1}{1000}$. In practice this almost always means that the solutions we produce are optimal in terms of the number or extent of conflicts but are otherwise as close as possible to the original proposal.

This mechanism provides an essential usability advantage since the produced results are supposed to be inspected and adjusted manually. Another feature that the user can optionally use is to require the solutions produced to never increase the extent of any remaining conflict (see Section 5.2). This may give less opportunities for improvements in other parts of the

Table 1: Minimizing the number of conflicts (Case 1)

	Original problem		Solution statistics		
Scenario	# trains	# confl	# confl	int. gap [%]	CPU time [s]
MAY 3	293	4281	2485	0.00	1.23
MAY 7	485	10759	4574	0.00	2.79
MAY 17	496	12268	5151	0.00	3.57
MAY 21	637	25570	13034	0.01	11.89
MAY 27	779	49129	22425	0.49	* = *timeout*
JUN 4	846	59016	28163	2.57	*
JUN 14	869	45737	17557	6.53	*
JUN 16	883	45635	17194	8.69	*
JUN 21	921	47086	20172	9.30	*
JUN 24	981	49369	20751	14.24	*

timetable but is appreciated by the manual planners since such conflicts may e.g. have been manually assessed to be small enough to be insignificant.

6 Results

The tests, below referred to as Case 1 and Case 2, have been performed to check both the validity and the scalability of the model for its intended use: to aid the timetable constructor while he or she is constructing the timetable for a specified geographical region, from now on referred to as *the constructor's region*. Sweden is divided into 16 such regions.

Our original intention was to simulate the effect of *the Maraca* being in actual use at Trafikverket this year (2010) by regularly receiving data from a timetable constructor during the timetabling process, feeding the data into our program, washing away conflicts, and last but not least feed the solution back into TrainPlan at Trafikverket and have the constructor inspect it.

We had to revise this plan since the XML import function unfortunately was delayed until the timetabling process was finished. The tests we report here have been carried out by saving data snapshots (XML) during the timetabling process, optimizing on them, and only later, when the XML import function had been implemented, discussing the results with the timetable constructor. We tried to discuss the results earlier by displaying the generated timetable solutions in our own GUI, but it soon became obvious that it was too time-consuming to evaluate the solutions that way, so we decided to wait.

The results of the test runs for Case 1 and 2 are summarized in Tables 1, 2 and 3. Tables 1 and 3 show the number of remaining conflicts (Case 1) or conflict seconds (Case 2) and corresponding CPU time (or integrality gap in case the optimization timed out) when the different scenarios were solved with CPLEX 12.1 on a ThinkPad T60, with Intel processor Core(TM)2 Duo, 2 GHz, under Ubuntu Linux. Table 2 shows the number of constraints, variables, and binaries for the original formulation and after the pre-solve phase in CPLEX for Case 1.

Table 2: Problem sizes for the different scenarios (Case 1)

	Original formulation			After preprocessing		
Scenario	# constr	# variables	# binaries	# constr	# variables	# binaries
MAY 3	26975	29725	435	2158	2219	191
MAY 7	45058	46517	1256	3946	3793	395
MAY 17	46941	48091	1460	4217	4006	472
MAY 21	64228	62928	2646	5391	5086	743
MAY 27	86781	80730	5968	9194	8359	1922
JUN 4	104453	92697	9529	12766	10985	3008
JUN 14	111316	96607	10999	15391	12846	3681
JUN 16	116617	99491	12497	17181	13898	4296
JUN 21	119849	101519	12162	16261	13345	3707
JUN 24	127151	106464	12913	17161	14099	3840

6.1 Real-World Data

Ten data snapshots were made. The problems based on those data sets were built with *the Maraca* for different number of trains at various stages of the timetabling process in Sweden in 2010. Every scenario was named after the date during which the snapshot was made.

Every snapshot includes more trains than the previous one and reflects the timetable constructor's reality: the active trains are those that originate in his/her region plus the ones that have been passed on to him/her from a constructor who works with a neighboring region. The whole process resembles a relay race, and every train that has a path that goes through more than one region is a baton that is passed from constructor to constructor. At the same time, every constructor has knowledge about what trains will eventually have to be scheduled in his/her region, and can sometimes guess their respective entrance times before they are passed on to him/her. Making room in advance for certain trains is sometimes even absolutely essential in order to be able to schedule them at all when the time comes.

6.2 Parameter Settings

All scenarios during the first set of tests (Case 1) had the same parameter settings, both for *the Maraca*, and for CPLEX when run on the LP file output by *the Maraca*. CPLEX was run with the option "set emphasize mip 4", "set mip cut all 2", and timeout 900 seconds (15 minutes). For Case 2, timeout was set to 900 seconds, but otherwise the default settings for CPLEX were used. We set CPLEX to optimize for 15 minutes since we believe that this is a reasonable time for a user to wait for a response from the system.

The constructor's region had one part containing double track and one containing single track. The trains on the double track section were given less possibilities to move around compared with the trains on the single track section. The motivation for this was that the constructor was mainly focusing on the double track section because he deemed the traffic there much harder to schedule, so we didn't want the solution from *the Maraca* to deviate much from his own strategic thinking.

To be specific, all take-overs on the double track section were fixed to specific locations (the ones the constructor had chosen for them), but their exact event times were allowed to change slightly in case this led to an overall reduction of the number of conflicts in the timetable. On the single track section, the traversal times of the trains could be prolonged with 600 seconds in general. Setting slack sizes individually per train is possible in *the Maraca*, and some trains were indeed given a bigger slack size than this.

New stops could be introduced for the trains on the single track section, meaning that *the Maraca* could change the locations for meetings and take-overs for the trains there.

6.3 Case 1

Looking at both Table 1 and 2, we can see that the number of constraints, variables and binaries naturally increase with the number of trains, as do the CPU time to solve the problem, or the integrality gap for the cases when CPLEX has interrupted the computation before optimality had been proven.

For all data sets in Case 1, we have provided CPLEX with an initial solution corresponding to the draft timetable. The many tests made prior to the real-world test that we report about here have shown that this usually cuts the CPU time significantly when minimizing the number of conflicts. Although not reported in the table, the number of conflicts were reduced to half compared with the original already after less than two minutes also for the two most congested cases (JUNE 21 and JUNE 24). Also not presented in the table, the building of the problem never took more than three minutes.

In general, we observe that CPLEX does a really good job in the pre-solve phase to reduce the number of constraints and variables. This also indicates that our problem formulation is probably not very efficient.

The integrality gap after 15 minutes is fairly large – almost 15% – for the most complex case (JUNE 24). While it would of course be even better if CPLEX terminated with an optimal solution within the allotted time, JUNE 24 is a scenario that well reflects the complexity of the toughest real-world cases that *the Maraca* is intended to be applied to. As the tests have shown, letting *the Maraca* work on a problem for up to 15 minutes always resulted in a timetable that was most significantly improved compared with the draft when it came to actual resource conflicts.

6.4 Case 2

For a second set of test runs, we minimized the number of conflict seconds instead of the number of conflicts. The parameters and settings were essentially identical to those used in Case 1, with three important differences: (1) default settings were used for CPLEX, (2) we did not provide CPLEX with an initial solution, and (3) the number of conflict seconds were not allowed to be reduced at the expense of making other conflicts worse (see Section 5.2 and the option “Strict improvement”).

Table 3 clearly shows that the number of conflict seconds in the original problem need not be proportional to the number of trains at all. Moreover, much more obvious here than in the case of minimizing the number of conflicts, the capability of *the Maraca* to reduce resource conflicts depends on a combination of many things, including how many conflict seconds there are in the draft timetable, the amount of slack that has been applied, how congested the tracks are, and how close to each other the trains in the draft are.

Table 3: Minimizing the number of conflict seconds (Case 2)

	Original problem		Solution statistics		
Scenario	# trains	# confl. secs	# confl. secs	int. gap [%]	CPU time [s]
MAY 3	293	104566	5668	0.00	0.24
MAY 7	485	97655	5551	0.00	0.80
MAY 17	496	104883	8337	0.00	0.80
MAY 21	637	313625	51762	0.00	2.30
MAY 27	779	2086348	1172562	0.00	11.96
JUN 4	846	1567453	951850	0.60	*timeout*
JUN 14	869	831148	120729	0.00	70.94
JUN 16	883	817546	214469	2.75	*timeout*
JUN 21	921	905032	131927	0.00	43.71
JUN 24	981	944863	134602	0.00	59.88

The CPU time needed to minimize the number of conflict seconds in general is less than when minimizing the actual number of conflicts, and we reached the optimum in less than a minute for most scenarios.

The optimization is however displaying a more stochastic behavior for Case 2 than for the first case. For two scenarios, optimality could not be proven within the allotted time, but there is no clear-cut reason as to why not. For JUN 4, when the same CPLEX settings as those used for Case 1 were used (see Section 6.2), and an initial solution was fed to CPLEX, CPLEX terminated after 676 seconds with the same number of conflict seconds as reported in Table 3. However, in general, using the initial solution corresponding to the draft timetable showed to be a bad idea for Case 2. A natural next step is of course to investigate the underlying reasons for this behavior.

6.5 The Constructor's Evaluation

As mentioned in the beginning of this section (Section 6), the tests of *the Maraca* on real-world data could not be evaluated during the actual timetabling process like we had originally hoped and planned for. Only after the process had been completed and the Train Plan of 2011 had been published (in September 2010), did we get the chance to ask a constructor to look closely at the solutions produced by our tool.

To be able to get a feel for the differences between the final solution (manually made by him) and the solution given by *the Maraca*, the two timetables were read into TrainPlan in different so called layers. Figure 1 shows the original draft timetable for a section of double track for 90 minutes in light grey lines. The suggested solution from *the Maraca* is visible as black lines, and only where the solution from *the Maraca* differs from the draft timetable.

From the discussions about the differences between the two solutions, we were able to draw three conclusions.

1. A lot of the trains were identical in the two solutions.

2. The constructor could easily spot which of the train runs that he thought were valid,

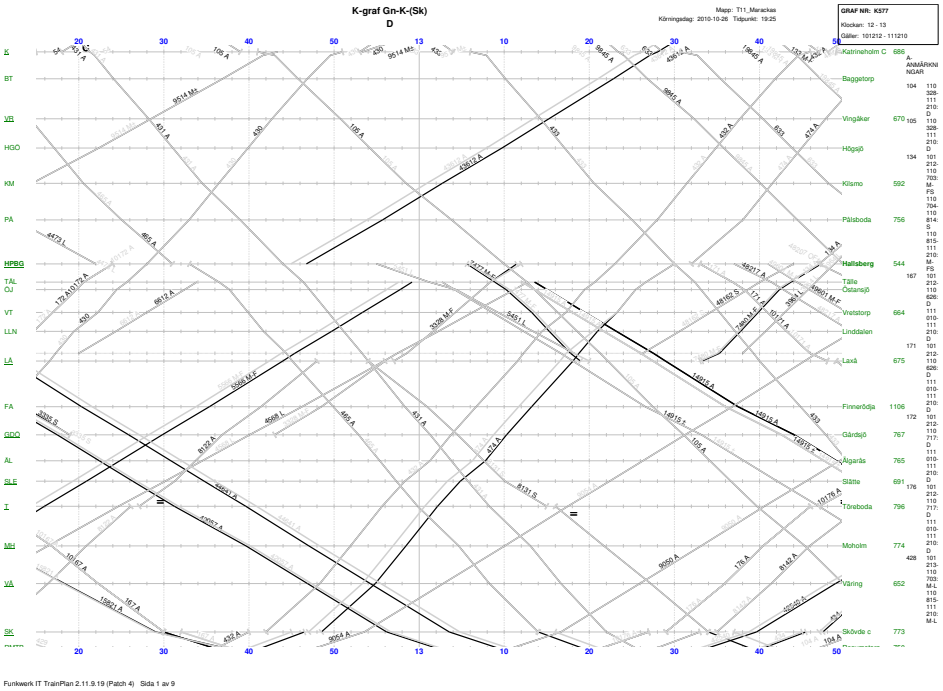

Figure 1: Screenshot from TrainPlan with the constructor's solution in grey and the solution from *the Maraca* in black (where it differs from the constructor's).

and which ones he would have rejected immediately (and why).

3. The solution from *the Maraca* for the double track section was judged to be of better quality than the solution generated for the single track section.

The first point was expected especially on the double track section since the locations for the meetings were fixed there, but also for the single track section since the slack sizes were moderate, and the objective function is formulated to minimize the deviation from the draft.

The second point is a crucial factor for the success of a tool like this. While it of course is partly dependent on the individual and his or her knowhow, this observation emphasizes the importance of using a visualization tool that the constructor is well familiar with. The constructor is in the end the only judge of what in the solution from *the Maraca* he or she can use, and what he or she needs to reject. If the constructor cannot see and evaluate the solution fast by inspection, it will not be sensible to use the tool in an iterative process.

Last but not least, the constructor's verdict was that he would benefit right away from the solution provided by *the Maraca* on the double track section, where the tool was not allowed to change the locations for the meetings and take-overs, only their event times. The resource conflict minimization on this section apparently did not violate his own ideas of what works and what does not work, whereas he on the other hand did identify some problems with the solution on the single track section where the tool had had greater freedom to solve conflicts.

Note that the indended, iterative use, was not mimicked at all in this evaluation process. Many of the shortcomings of the first – and only – solution produced by *the Maraca* in this case could probably have been remedied by making manual adjustments and then running the optimization again. With this fact in mind, the constructor's evaluation was indeed encouraging, and our belief is that the prototype is definitely ready to be taken to the next level.

7 Conclusions

The two case studies that were carried out indicate that Trafikverket would benefit from introducing mathematical optimization in the timetabling process. The most important observations made during the study are summarized in this section.

Generating a timetable with *the Maraca* is an iterative – and interactive – process. While scalability is important, we know that many 'soft' constraints are not in the model at all. It is therefore safe to assume that once a significant improvement has been found, the user would rather abort a computation that takes a lot of time than wait for it to finish since the 'optimal' solution found would most likely not be the final one anyway. The tests show that the tool's scalability is adequate for real-sized problems as long as we are mainly interested in an improvement rather than interested in proving optimality.

The actual minimization of resource conflicts is not the only required step in *the Maraca* that takes time. Exporting XML from the existing timetabling visualization tool, building the problem, and then reading the result back to the visualization tool are three other time-consuming tasks. All of these have to work in order for *the Maraca* – or any other similar tool – to be truly useful in practice.

There is an important conceptual difference between concentrating on minimizing the number of resource conflicts and minimizing the number of conflict seconds. This difference became obvious already by looking at and comparing the number of conflicts and con-

flict seconds in the original draft timetables as they change over time during the timetabling process: the number of conflict seconds is by far the greatest in the middle of the process, when the number of trains is high but before the constructor has had a fair chance to sort the situation out. It then drops dramatically towards the end of the construction phase. The same trend is not at all as visible when we look at the absolute number of conflicts.

One of the reasons for the reduction of conflict seconds in the draft timetables towards the end of the process is of course that some conflicts in the later stages of timetabling are so small that they are negligible, making the number of conflict seconds drop but keeping the number of remaining conflicts fairly high. In other words, in practice, the constructor clearly aims to reduce the number of conflict seconds rather than minimize the absolute number of conflicts. That alone is a reason for helping him or her to do it more efficiently with the help of a tool like *the Maraca*.

The key to the success of our tool is that the user easily can understand what is being optimized. He or she makes the decisions, and the tool performs the computations. The strategy outlined by the user is still clearly visible in the result even after a large number of conflicts have been reduced or eliminated. In this way, the timetable constructor – not the tool – decides what a good timetable is. The tool keeps track of the hard-coded facts, makes sure that any remaining conflicts are highlighted, and enables the user to focus on the main strategic decisions.

8 Acknowledgement

The project has been funded by Banverket (the Swedish Rail Administration), grant F 08-6473/AL50, and by Trafikverket (the Swedish Transport Administration) under the same grant. As of April 1, 2010, Trafikverket carries out activities and operations that were previously undertaken mainly by Banverket and Vägverket (the Swedish Road Administration).

References

[1] Martin Aronsson. Slutrapport för projektet TUFF, TågplaneUtveckling För Framtiden. SICS Technical Report T2006:07, Swedish Institute of Computer Science, 2006.

[2] Martin Aronsson, Markus Bohlin, and Per Kreuger. MILP formulations of cumulative constraints for railway scheduling - a comparative study. In Jens Clausen and Gabriele Di Stefano, editors, *ATMOS 2009 - 9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany, 2009. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Germany.

[3] Banverket. Riktlinjer för tidtabellskonstruktion på statens spåranläggningar. Föreskrift TF601, 2000.

[4] Jacek Blazewicz, Moshe Dror, and Jan Weglarz. Mathematical programming formulations for machine scheduling: A survey. *European Journal of Operational Research*, 51(3):283–300, April 1991.

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

[6] Alberto Caprara, Matteo Fischetti, and Paolo Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, September-October 2002.

[7] Ahmet B. Keha, Ketan Khowala, and John W. Fowler. Mixed integer programming formulations for single machine scheduling problems. *Comput. Ind. Eng.*, 56(1):357–367, 2009.

[8] P. Kreuger, M. Carlsson, T. Sjöland, and E. Åström. Sequence dependent task extensions for trip scheduling. Technical Report T2001:14, SICS, 2001.

[9] L.G. Kroon, Dennis Huisman, E.J.W. Abbink, P-J Fioole, M. Fischetti, G. Maroti, A. Schrijver, A. Steenbeek, and R. Ybema. The new Dutch timetable: The OR revolution. Econometric Institute Report EI 2008-19, Erasmus University Rotterdam, Econometric Institute, 2008.

[10] Alex Landex. Evaluation of railway networks with single track operation using the UIC 406 capacity method. *Networks and Spatial Economics*, 9:7–23, 2009.

[11] Christian Liebchen and Rolf H. Möhring. A case study in periodic timetabling. *Electronic Notes in Theoretical Computer Science*, 66(6):18 – 31, 2002. ATMOS 2002, Algorithmic Methods and Models for Optimization of Railways (ICALP 2002 Satellite Workshop).

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

[13] M. A. Salido, M. Abril, F. Barber, L. Ingolotti, P. Tormos, and A. Lova. Domain-dependent distributed models for railway scheduling. *Know.-Based Syst.*, 20(2):186–194, 2007.

[14] Paolo Serafini and Walter Ukovich. A mathematical model for periodic scheduling problems. *SIAM J Discr. Math.*, 2(4):550–581, November 1989.

[15] Johanna Törnquist. Computer-based decision support for railway traffic scheduling and dispatching: A review of models and algorithms. In Leo G. Kroon and Rolf H. Möhring, editors, *5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, Dagstuhl, Germany, 2006. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[16] Trafikverket. Järnvägsnätsbeskrivning 2011, appendix 4.2. Network Statement published on the web, September 2010. http://www.trafikverket.se/PageFiles/10549/bilaga_4_2_prioriteringskriterier_jnb_2011_andring_2010_02_06_anpassad.pdf.

[17] Trafikverket. Järnvägsnätsbeskrivning 2011, chapter 4. Network Statement published on the web, September 2010. http://www.trafikverket.se/PageFiles/9340/kapitel_4_2011.pdf.