SICS Technical Report
T2001:14

ISRN : SICS-T--2001/14-SE
ISSN : 1100-3154

Sequence dependent task extensions for trip scheduling

by

Per Kreuger, Mats Carlsson, Thomas Sjöland and Emil Åström

May 2001

Swedish Institute of Computer Science
Box 1263, SE-164 29 Kista, SWEDEN

Sequence dependent task extensions for trip scheduling

Per Kreuger, Mats Carlsson, Thomas Sjöland, Emil Åström
Swedish Institute of Computer Science (SICS),
Box 1263, SE-164 29 Kista, Sweden

May 2001

SICS Technical Report T2001:14
ISSN 1100-3154
ISRN: SICS-T–2001/14-SE

Abstract

A constraint model for scheduling train trips on a network of tracks used in both directions, using a headway abstraction is described. We argue that a generalisation of a straightforward job-shop scheduling formulation using sequence dependent task extensions can decrease the required resolution of network representation and hence problem size. A geometric interpretation of the model of the constraints that can be used to visualise schedules is presented. Preliminary ideas on search heuristics are presented with performance results and a set of examples.

Keywords: Constraint modeling, scheduling, sequence dependent durations, train scheduling, global constraints, scheduler implementation.

A note on publication of this material This report constitute a revised version of a short paper originally published as a workshop paper [KCO+97]in 1997. This version of this material was produced in 1998 but was never published. This report describes the original model for trip scheduling on single track rail networks and suggests an implementation of the model in terms of a non-overlap condition with setup times. At the time this material was produced its implementation was not complete and the report does not contain any evaluation of the suggested implementation. Since then the model *has* been implemented and used in a large scale implementation of a decision support system for planing of rail services. This later work has been published as [AHKL00, KAHL00, AK01, KAL01]. The reason to publish this material now is to document the original approach and create a reference for later work based on the same or similar approaches.

1 Introduction

To schedule train trips on a network of tracks is a problem that resembles but also differs in certain ways from other typical scheduling tasks. This paper describes a constraint model and solver for scheduling trains on a network of mixed double and single tracks (i.e. tracks used in both directions) using the abstract notion of headway to approximate some of the intricacies of the real network structure.

The simple case where the network consists only of double tracks can be modelled as a job-shop scheduling (JSS) problem [Bak74], [CP89], [BLP95] viewing train trips as jobs and tracks as resources. Each train trip traversing a track represents a task where the traversal duration is usually taken as the task extension.

Since in such a model each track traversal requires exclusive use of the track resource it is necessary to keep traversal durations and hence individual tracks short to achieve reasonable schedules. While this model corresponds closely to the real network with individual track segments delimited by signals and sensors, it results in very large scheduling problems (in tervdms of the number of tasks).

In order to generate tactical plans it is often sufficient to abstract from this model by collapsing individual track segments into aggregate tracks connecting nodes consisting of meeting points, splits and merges in the network. In this case, instead of taking the traversal duration as the task extension, we will use instead the notion of headway which models a minimum (time) distance between two individual trips traversing an aggregate track consisting of several individual segments.

Note that headway does not correspond directly to the task duration in a job shop scheduling problem since the actual interval between the departure times for any two trips departing on a track depends on the difference in speed between the trips. This follows from the fact that the headway has to be preserved at both end-points of the traversal and means that in general (i.e. if all speeds are different) we have to consider for each task as many time extensions as there are other tasks to be scheduled on the track resource.

Thus the trip scheduling problem can be viewed as a job shop scheduling problem with release times and either sequence dependent durations [BRRS88] or sequence dependent setup times with fixed durations [LM72], [WW77], [AB93], [JD95], [Sim96]. We will in what follows ignore the distinction between such models and simply refer to this sequence dependent time extension as the *task extension.*

Plans based on the headway abstraction are realisable in a more detailed model as long as the headway exceeds the longest traversal duration for an individual segment in the aggregate track resource but, of course, some schedules in the more detailed model will not have a corresponding schedule in the abstract one.

A fully operational system would have to take into account also several levels of warnings that the signal system issues before actually stopping a running

train. The determination of a practically useful headway for each track and type of trip will have to take also these parameters into account.

One further complication arises when part of the network consists of tracks used in both directions (single tracks) as is the case in a large part of the Swedish rail network. In the model outlined above, this case is handled by considering the time extension of a trip traversing a track in relation to any other traversing the track in the opposite direction to be the full traversal duration.

Such a model and the constraints used to ensure its realizability will be described in this paper. In this model (and its implementation) we do not consider the problem of allocating transportation needs to individual trips, nor deciding what route to take between two locations in the network. Neither do we consider the (very important) problem of assigning vehicles to individual trips and the impact this problem will have on the scheduling problem. We are currently working on extending our model in this direction and hope to present this work in a sequel to this paper.

We do not focus on the problem of generating optimal plans with respect to some simple cost function such as total time to execute the plan (makespan) but in our implementation the problem specification includes upper bounds on makespan as well as on accumulated waiting durations so it is possible to manually explore the space of feasible plans by manipulating of these parameters.

An optimizing system should use a cost function that is motivated by the total economics of the transportation system rather than on simplistic technical measures such as makespan. The study of such functions represents future work.

The problem that we do consider can be concisely stated as:

- Schedule a set of train trips over a fixed network of predetermined routes where trains travel through a network with mixed double and single tracks connecting nodes where trains can meet and overtake
- Maintain reasonable bounds on waiting and total times

The rest of the paper is organised as follows: section 2 contains an introductory note illustrating a geometric interpretation of our model. This is followed by a short description of the network representation used (section 3) and how the problem specifications are built and maintained in the form of a data structure (object representation) we call plans (section 4). Then we describe the constraint model used and formulate constraints stating realizability (soundness) criteria on the generated schedules (section 5). This is followed by a discussion of the enumeration strategy used and some possibilities for improvement in this area (section 6).

We then briefly consider some performance issues (section 7) and conclude the paper with some notes on limitations in the model and possible lines of future work (section 8), conclusions (section 9), and a set of examples (section 10).

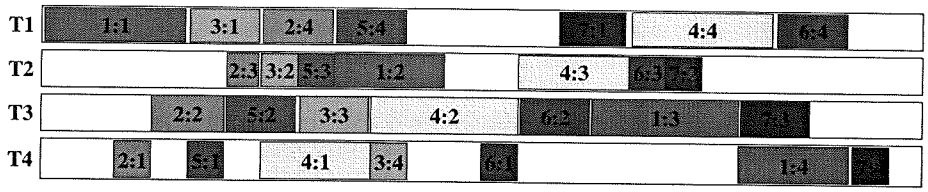

Figure 1: Naive Job shop scheduling model

2 A geometric interpretation

The following diagrams attempt to a give a geometrical intuition behind a sequence dependent task extension scheduling problem. We use an adaptation of a type of diagram traditionally used in the railway industry to represent train schedules. Formally it is a straightforward generalisation of a Gantt chart where geographical distance on a given route is plotted on the y axis of the diagram while using the x axis to represent time. Individual trips are represented as diagonal lines in these diagrams where (positive or negative) angle from the x axis represents speed. Conflicting schedules can be seen in this kind of diagram as crossing lines or lines running too close together.

We have adapted this kind of diagram to the sequence dependent task extension case by representing the resource requirement as a rhomboid surface defined by the diagonal mentioned earlier and a "signal shadow" extending positively in the dimension.

From the perspective of trips traversing the track in the opposite directions the part of the rhombus extending past the arrival time of the traversal is irrelevant. In this case it is often more intuitive to consider the truncated surface resulting from ignoring time points past the arrival time. Conflicting schedules appear as overlapping surfaces. For further examples of these kind of diagrams see figures 3 &4 of section 5.1 as well as the example schedules of section10 .

We are not aware of any published material on this kind of geometrical representations but in the Swedish rail transport industry [Mal98] a variant of this kind of diagram is referred to as "Train sequence diagrams".

Below we compare a simple diagram of this form with the Gantt chart of a straight-forward job shop scheduling formulation of a similar problem for the single track case.

In the JSS formulation (fig. 1) there is only one task extension associated with each start time. This duration, the traversal duration, is illustrated as the width of each task rectangle. Rectangles on a single track must not overlap and track traversals belonging to a single trip follow in sequence.

In the second diagram (fig. 2) the width of a cross section of a rhombus represents the headway while its total width represents the traversal duration. In the example below we have chosen to give slow trains a long headway and quick trains a short headway. In practice the choice of headway depends on both the type of trip and properties of the track as outlined above.

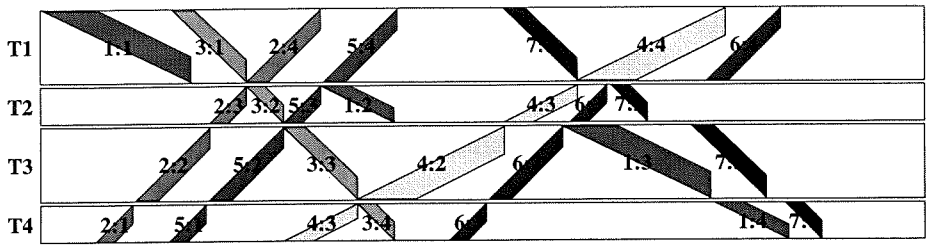

Figure 2: Sequence dependent task extention formulation

It is easy to see that for comparable total times the number of trains that can be scheduled with the above rather simplistic JSS formulation on an identical network is significantly lower than the sequence dependent task extension formulation for the single track case. As noted above it is possible to achieve a schedule similar to the second one with the JSS formulation using a network with a larger number of (shorter) tracks but then the size of the scheduling problem increases accordingly.

If job-shop scheduling can be seen as the problem of packing the rectangles (with fixed and identical extension in the dimension) of the regular Gantt diagram under the precedence constraints imposed by the jobs, the sequence depending task extension trip scheduling problem can be seen as the corresponding rhombi/truncated-rhombi packing problem.

3 Network representation

A railroad network is represented as three interrelated sets:

- *Locations*: Meeting places, end stations and depots
- *Tracks*: directed or undirected
- *Routes*: predetermined sequences of track traversals

A location is characterised by a unique identifier, its geographical coordinates, the set of tracks adjoined to it and the set of routes starting, ending at it and passing through it. Meeting places contain in addition resource and capacity limitations on the location e.g. maximum number of trains simultaneously waiting at it. It also encodes upper and lower limits on waiting durations for trains passing through it.

A track is represented by a unique identifier, the locations it connects, a length and the maximum speed at which different types of trains may traverse it. Tracks may be traversed in both or in only one direction.

A route finally, is represented by a unique identifier and an ordered sequence of tracks. Routes refer indirectly to a set of locations connected by its tracks. The problem of generating suitable routes for a given transport demand on a network, consisting only of locations and tracks, is not addressed in this paper.

We use the following terminology for some elementary relations: Tracks connect locations. Routes use individual tracks. Locations are served by routes.

4 Problem representation - plans

We use the notion plan to denote a data structure representing a scheduling problem in terms of a given network and a specification of a required set of train trips. This specification allows us to limit the arrival, departure and waiting durations for arbitrary locations in the route of the trip. We implemented also a specification language for cyclic trip specifications and experimented with trip connections at specified locations. These two extensions of the specification language interact in complex ways. To investigate how to combine these two aspects of a transport specification problem remains as further work.

4.1 Trips

Trips model individual trains traversing the network. A trip is represented by a unique identifier, the route it follows, a vehicle resource requirement, a location resource requirement and speed parameters used to determine traversal durations for the tracks in its route.

Trips are said to pass locations, traverse tracks and follow routes.

4.2 Tasks

A task represents the traversal of a track by an individual trip. With each task we associate a unique identifier and the departure time and waiting duration at the origin of the traversal. We associate also with each task a traversal duration and a headway.

- The traversal duration represents the minimum time distance between the departures of any two trips traversing a track in opposite directions
- The headway is the minimum duration after a trip departs from or arrives at a location in the network before a following trip running in the same direction on the same track may depart from or arrive at that location

These are currently determined from length of the track and the speed parameters of the track and the trip. In a more realistic setting the determination of suitable headways is of course a non-trivial practical problem.

4.3 Schedules

A schedule is the result of assigning values to the time points and durations associated with each task in a plan. The scheduling problem consists in finding such an assignment.

A constraint representation of the scheduling problem is extracted from the plan as a set of finite domain variables and constants representing respectively

the departure, waiting duration, traversal duration and headway associated with each task.

5 Constraint model

The scheduling problem is modelled with constraints over finite domains [vH89].

Let a pair (i, j) denote a task representing the traversal of the track i by trip j. For each task (i, j) let S_{ij} denote the departure time of trip j on track i and W_{ij} the (waiting) duration between the arrival of trip j at the origin location for its traversal of track i and its departure on i.

Let D_{ij} denote the traversal duration and H_{ij} the headway for the task . Furthermore, let R_k^j denote the k:th track used by trip j when it follows its route.

D_{ij}, H_{ij} and R_k^j are problem parameters while S_{ij} and W_{ij} are the problem variables.

In the following we will denote upper and lower bounds on a variable X by $\overline{X}$ and $\underline{X}$, respectively. The bounds are defined as:

$$\underline{X} \leq X \leq \overline{X}$$

5.1 Constraints

There are four distinct sets of constraints used in the model. The first set is used to impose bounds on the departure and waiting durations given by the problem specification itself, typically bounds for the first departure or the last arrival of a trip and lower bounds on the waiting durations. The waiting durations can also be limited from above by bounding the sum of the waiting durations associated with each trip and the sum of all waiting durations in the plan.

The second set of constraints is used for maintaining the precedence of the track traversals for each trip while the third set handles resource limitations for the locations in the network. The fourth set of constraints, finally, is used to enforce a serialisation of all the tasks associated with each track.

Informally

1. All bounds given by the problem specification must be respected
2. For each trip the durations of each track traversal must follow in the sequence defined by its route
3. For each location the number of trips simultaneously "waiting" (e.g. meeting or overtaking) at it may never exceed a fixed maximum associated with the locationij
4. For each track

- Any two trains travelling in the same direction must respect headway during the whole traversal even if speeds are different (see fig. 3)

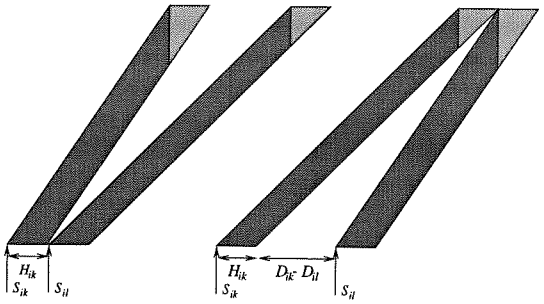

Figure 3: Two different cases of headway for trips running at different speed

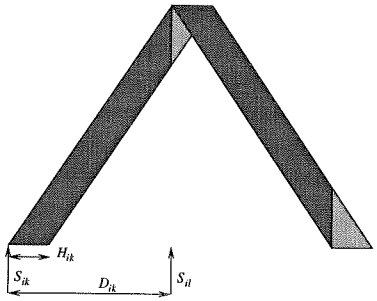

Figure 4: Exclusive use of single track by two trips

- Any two trains in opposite direction must not simultaneously use the same (single) track (see fig. 4)

We will now examine the formulation of these constraints in some detail.

5.1.1 Specified time bounds

The departure times S_{ij} are bounded by $\underline{S}_{ij}$and $\overline{S}_{ij}$ which are derived from the plan specification.

The waiting duration W_{ij} at the departure location of a track i for a trip j is in general limited from below by the minimal waiting duration associated with the location and with additional limits given by the plan specification. It can also be limited from above, both individually (e.g. by some constant or a fraction of the preceding traversal duration), by limits on the accumulated waiting duration associated with each trip and on the sum of all waiting durations in a plan.

Note that all waiting durations (except the first one for each trip) can be derived from the departure times and traversal durations once these have been determined. A non-singleton domain for the waiting duration can be thought of as slack.

The search heuristics outlined in section 6is very sensitive to large domains for the waiting duration variables and it is essential to impose as strong limits

as possible on them to achieve reasonable performance.

$\underline{W}_{ij}$ and $\overline{W}_{ij}$ are given by the plan specification. For each trip j let $\overline{\overline{W}}_j$ denote the specified limit on the accumulated waiting duration for the trip and W a limit on the total accumulated waiting time for all trips.

Then, for each trip j the following relation between the sum of waiting times for j and its bound must hold:

$$\sum_k W_{R_k^i j} \leq \overline{\overline{W}}_j$$

and finally for the sum of the waiting times for the whole plan and its bound:

$$\sum_{(i,j)} W_{ij} \leq W$$

5.1.2 Trip constraints

For each trip j, the tasks $\left(R_k^j, j\right)$ belonging to it must be sequentialised. This is achieved by requiring that the following relation holds:

$$\forall j \forall (k < |R^j|)(S_{R_{k+1}^j j} = S_{R_k^j j} + D_{R_k^j j} + W_{R_k^j j})$$

5.1.3 Location constraints

Each location in the network has a maximum number of track resources that can be used simultaneously. We make sure that the number of tracks used at each location at any time point does not exceed this limit by posting a cumulative constraint [AB93]. A cumulative constraint expressing the following relation is required to hold for each location:

$$\forall i \forall t(|\{j | S_{ij} - W_{ij} \leq t \leq S_{ij}\}| \leq C_{ij})$$

where $|T|$ denotes the cardinality of the set T and C_{ij} denotes the (track) resource limitation at the location from where the trip j departs on track i.

5.1.4 Track constraints

Recall that we want to ensure for each track that is traversed by at least two trips that:

1. For each pair of trips traversing the track in opposite directions, the intervals during which the traversals take place will not overlap
2. For each pair of trips traversing the track in the same direction, headway is respected at both departure and arrival locations

Define a (disjointness) relation $\diamond$ on pairs (S, D) of time points and durations as follows:

$$(S_i, D_i) \diamond (S_j, D_j) \Leftrightarrow (S_i + D_i \leq S_j) \vee (S_j + D_j \leq S_i)$$

Then for each track i and each pair of trips k and l traversing it one of the following two relations must hold:

1. If trips k and l traverse i in *opposite* directions

$$(S_{ik}, D_{ik}) \diamond (S_{il}, D_{il})$$

2. If k and l traverse i in the *same* direction

$$(S_{ik}, \max(H_{ik}, H_{ik} + D_{ik} - D_{il})) \diamond (S_{il}, \max(H_{il}, H_{il} + D_{il} - D_{ik}))$$

The max expressions above are used to analyse the cases when speeds are different. These correspond to the left and right parts of fig. 3 above. Note how the relative speed, i.e the traversal time difference comes into play here and potentially introduces a unique task extension for each pair of trips traversing a track in the same direction. Note also that the task extension is bounded from below by the headway and from above by the traversal duration since this can safely be assumed to always be larger than the headway.

5.2 A note on local vs. global constraints

The above formulations directly translate to an implementation using constraint propagators for additions and equalities and for the disjointness relation . These are local constraints, i.e. in this case constraints involving only the problem variables for a single task.

In many cases it is for efficiency reasons necessary to take a more global perspective to be able to implement more sophisticated reasoning about the properties of a specific problem. It is e.g. possible to reason about which tasks from a selected subset of the tasks associated with a track that can and cannot precede the others, e.g. using edge-finding techniques [CP89], [AC91], [CP94] and task-intervals [CL94].

This kind of reasoning is generally realised by so called global constraints [BC94], [PL95] and represent key technology in the achievements of constraint programming on large scheduling problems.

We are not aware of any published material or any system available today that implements global reasoning on scheduling problems with sequence dependent task extensions and would argue that this could be a useful feature of a constraint programming system.

The edge-finding algorithms that have appeared in the literature reason about tasks (i, j) that have variable release times S_{ij} and fixed non-zero durations. A straightforward generalisation to the sequence dependent task extension case and hence to job shop problems with sequence dependent setup times

is to reason about the lower bounds of variables D_{ij} representing the sequence dependent task extension.

The lower bound translates in the trip scheduling case to the headway and in the sequence dependent setup time setting to the task duration plus the minimal setup time between (i, j) and any other task (i, k) to be scheduled on the same resource (machine).

We propose to further generalise the edge-finding to take into account the effective task extension, i.e. the bounds imposed on D_{ij} by considering which tasks (i, k) that can in fact be a successor of (i, j) on the resource i. As the set of possible successors of a task gets narrowed down during search, the lower bound of the effective duration can be adjusted up and the edge-finder can draw stronger conclusions.

A constraint abstraction based on these ideas is being developed.

6 Search strategies

Since the model uses disjunctions in the disjointness constraints for the tracks, to enumerate the departure times generally involves search. We have investigated two distinct methods to do this.

- The most straightforward approach is to enumerate the departure times themselves in some order, rely on propagation to further restrict the domains of remaining departure times and to detect failure in the case of an inconsistent instantiation.
- An alternative approach is to enumerate variables representing the serialisation choices between the tasks.

6.1 Enumeration of departure times

The performance figures given in section 7 below were achieved with the fairly straightforward search heuristic to first choose one of the most constrained departure time variables and then bisecting its domain trying first to constrain the variable to lower half of the original domain.

Clearly such a strategy does not lend itself well to optimisation. The search tree for the departure times themselves is simply too large. However given reasonable size restrictions on the domains for the waiting duration variables it is a quick way to find in reasonable solution in most of the (rather underconstrained) cases we considered so far. Unfortunately search explodes when the bounds place us close to optimal total time which effectively excludes traditional branch and bound search for optimal solutions.

6.2 Enumeration of variables representing serialisation choices

We have been experimenting with serialising the tasks for each track by representing the order between every pair of tasks on a track by a boolean variable

and enumerating these (quadratically many) boolean variables before actually assigning values to the departure times.

This approach has the attractive property that it identifies all solutions that do not reorder the tasks on the track resource, thus reducing the search space significantly. This makes it more realistic to consider branch and bound search for better solutions than the initial one.

Another approach may be to use permutation of task orders in the spirit of [Zho97]. This involves using a vector of FD variables representing the order of the individual tasks for each resource and using a global difference constraint [Rég94], [Lec96] to ensure that the vector contains a valid order. This gives us a linear number of decision variables while the domain of each variable is linear in the size of the problem. We have so far not explored this strategy in full detail but we expect that this representation will reduce the memory required to represent the problem (in comparison with the boolean variable approach) and that the difference constraints will improve propagation during search.

Furthermore it might be more natural with approaches like these to formulate more intricate search strategies such as those found in the literature on local search [PS82], [KGV83], [Glo89], [Gu92], [SK92], [Wal97]. To formulate a good strategy for the trip scheduling domain using such ideas is of course not trivial and represents future work.

We have so far not been able to reproduce the performance results given with the relatively naive strategy outlined in section 6.1. This is probably related to the fact that we have quite weak propagation on the track resources (see section 5.2 above). We believe however that with stronger propagation some approach based the ideas outlined here will eventually prove to be superior.

Choosing to represent potential schedules by a vector of decision variables simplifies rescheduling and incremental updates of existing schedules. Investigations of these aspects of constraint programming is currently under way.

7 Performance

The performance of the system is affected by a number of properties of the problem.

First the problem size is obviously a key factor although it is not clear how the size should be measured. The number of trips for a given network is a measure that comes naturally to mind, but the number of tasks varies with the lengths of the routes for the trips. In addition the length of the routes depends on the granularity of the network. The number of tasks is therefore a better measure.

In this section the number of tasks for each problem is plotted against the computation time (recall that a task is a traversal of a track). We consider three separate sets of randomly generated problems, each distributing a varying number of trips over a set of predefined routes. The maximum number of tasks in the experiments reported here is in the 2 500 to 3 000 range, which corresponds to about 200 trips. The number of variables for this kind of problem is between

9 500 and 13 000 in our experiments and the number of constraints is in the 17 500 to 30 000 range. The experiments are done on a 248 MHz Sun Ultra Enterprise with 1 GB of memory.

From figures 5 and 6 (and also 7 to a certain extent) it can be seen that the computation times grow approximately quadratically with the number of tasks.

Figures 5 through 7 differ in how constrained the departure times are. In figure 5 they are totally free while figures 6 and 7 show more constrained problems. In figure 6 the day of the week is specified, e.g. "Tuesday" (note that the planning horizon is one week). In figure 7 the departure times, in addition to the day, has a time interval in which the departure should take place. The size of this interval ranges from 0 (in which case the departure time is already instantiated) to 8 hours.

It is apparent from the figures that the more constrained the departure times are, the faster the calculations can be done. Note also that in real-world problems the departure times are very seldom unconstrained since customers often require the departure and arrival times of the transports to be within tight intervals.

Not all problems in figure 7 can be solved since some of them are over-constrained. It is noteworthy however that the detection of failures is relatively fast. In fact, these inconsistencies (with the exception of one) are detected without any search at all. This level of propagation is possible since we do not allow any waiting at locations in these experiments.

Waiting times at locations is an important factor for the performance of the system. If no waiting is allowed then the arrival times can immediately be calculated and potential inconsistencies can be found quickly. If we allow waiting then search is required to decide arrival times and propagation is thus delayed. The search space is also much larger than before.

All figures in this section show a distribution of the calculation times (i.e. the time is not necessarily exactly the same for two problems with the same number of tasks). The reason for this is that the distribution of the trips over the tracks is an important factor. E.g. a problem where many trips traverses the same tracks is more difficult than if they traversed different ones. This is especially apparent in figure 7 since the more constrained departure times make these problems more sensitive to variance in the number of tasks per track.

8 Future work

Since there is currently a lot of time and memory spent in search, search behaviour has to be analysed in more detail. Both domain dependent heuristics and general search procedures such as intelligent backtracking techniques [XSS92], limited discrepancy search [HG95] and, as already mentioned in section 6.2, local search techniques should be investigated.

The model lacks limitations on vehicle and staff resources. These are serious limitations in practice and essential for this kind of technique to be useful in solving real-life problems. There is as far as we can see nothing in our model

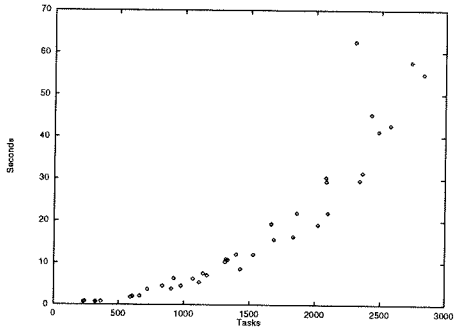

Figure 5: Free departure times

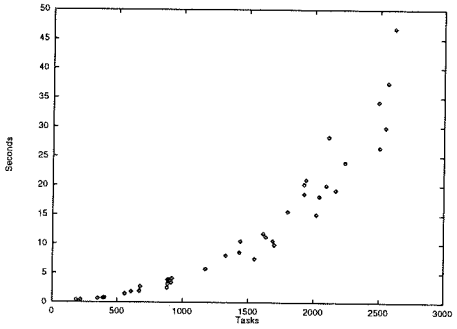

Figure 6: Departure times specified by day of week only

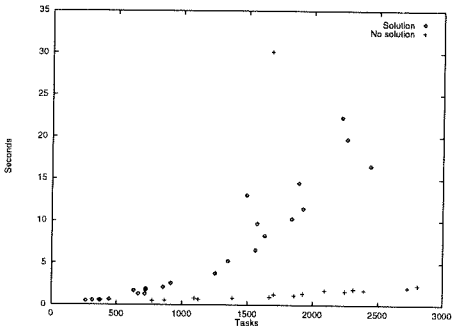

Figure 7: Departure times specified by day of week and a departure time intervals varying in size from 0 to 8 hours

that would not naturally extend to a more complex model considering also these aspects. We address the issues of comibing a scheduling problem with a vehicle routing problem in a current project.

In practice it will probably also be both possible and necessary to subdivide the full problem into several smaller and relatively independent subproblems. For larger realistic problems such as producing schematic schedules several months in advance problem subdivision is most likely the only possible approach. The standard way to do this is to subdivide the problem into that of producing several weekly and daily schedules and to do detailed scheduling for separate geographic regions given non-local constraints on e.g. interregional trains. This aspect is also addressed in the current project.

9 Conclusions

We have described a constraint model for trip scheduling and a corresponding solver on a number of randomly generated problems on a relatively realistic networks. With a relatively straightforward solver we have been able to attack problems of non-trivial size. Furthermore we have suggested some initial ideas on how to generalise edge-finding to sequence dependent task extensions and outlined some directions for future improvements with respect to search.

The abstractions used in the model not only give (in our view) a good understanding of the kind of reasoning required to solve this class of problems, but it also gives a substantially smaller amount of variables (and thereby search) needed to perform this scheduling task. We have indicated the relation of the model to the well studied job shop problems with sequence dependent setup times.

10 Sample schedules

The following diagrams illustrate some sample schedules generated by our implementation. The implementation was done in the Oz programming language [Smo95], [ST+95] using the embedded Tcl/Tk interface for the GUI parts. For the interpretation of the diagrams see section 2 above. Note that these examples were generated without any bound on the number of trips simultaneously meeting of overtaking at the locations.

10.1 Trade-off between total time and accumulated waiting duration

The trade-off between total time and accumulated waiting duration is nicely illustrated by the three schedules given in figures 8-10.

In the first (fig. 8) we have set a very tight limit on the waiting duration for all trains at the intermediate location (Södertälje). So tight, in fact, as to exclude any meetings at this location. In the second schedule (fig. 9) we loosen

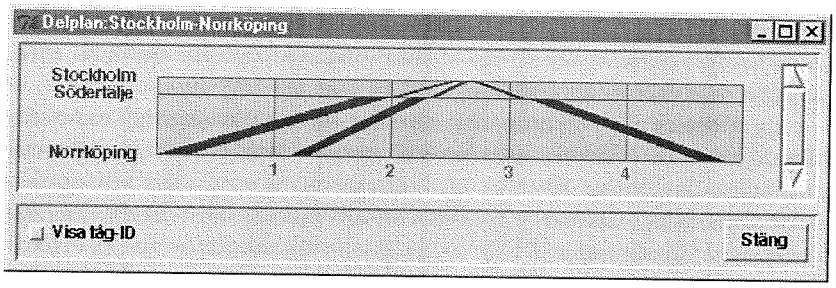

Figure 8: No waiting allowed — Long makespan

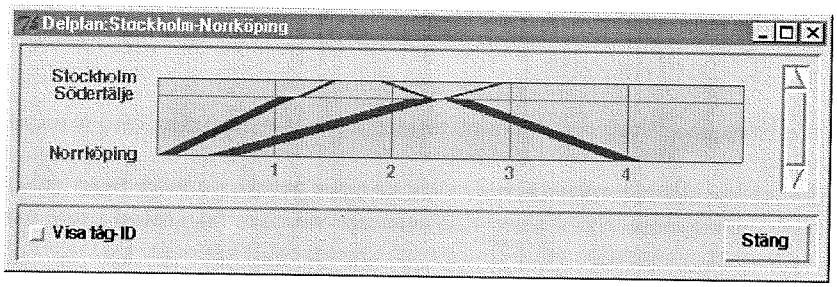

Figure 9: Some waiting allowed — Shorter makespan

this bound slightly to allow a maximum of two trains to meet at this location. In the last (fig. 10) finally we allow an arbitrarily long waiting duration thus allowing all three trains to meet at this point. Note how the total time decreases throughout this process.

Note also that headway is respected even where trains travel at different speeds. The steeper the angle the quicker the train.

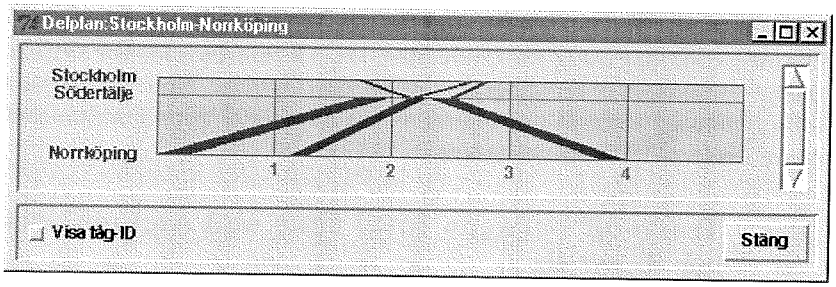

Figure 10: No limit on waiting time — Optimal makespan

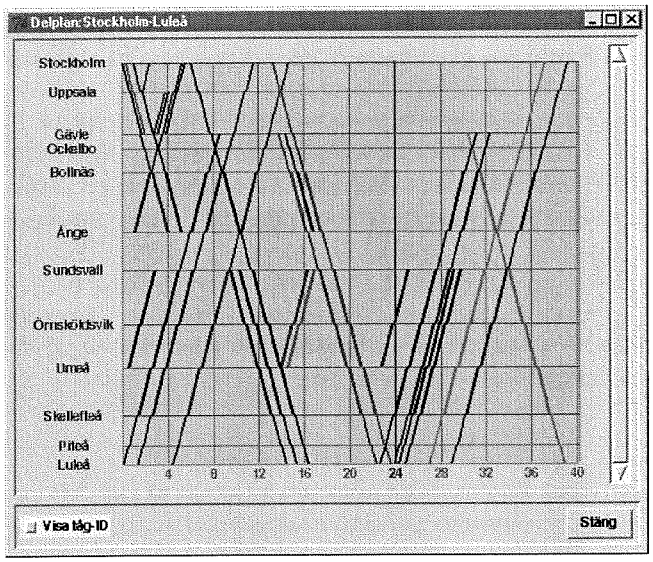

Figure 11: Short waiting durations

10.2 Some larger schedules

The following examples illustrate some larger schedules and how bounds on the waiting durations influences the sparseness of the schedules. The following diagrams represent identical fragments (all traversals of the tracks of one particular route) of 2 different schedules. The first (fig. 11) with low tolerance on waiting durations, the second (fig. 12) with a high one. Note how the total time is decreased with about 25% by bounding waiting durations to allow approximately three trips to meet at each location.

Acknowledgements

The results reported in this paper is largely one of the results of a study of the train planning domain done in cooperation with the Swedish State Railways (SJ). One of our motivations behind this cooperation was the opportunity to evaluate the feasibility of constraint and constraint logic programming [vH89] techniques for various problems in their domain.

We would like to thank Jolanta Drott and her group at the Swedish State Railways (SJ) for presenting us with the problem and for good cooperation within the TUFF series of projects.

We would also like to acknowledge Jan E. Olsson, the leader of the Complex Operations laboratory at SICS who contributed the initial idea to investigate this domain, initiated the contacts with our industrial partners and helped throughout the TUFF and TUFF II projects to maintain good relations with them.

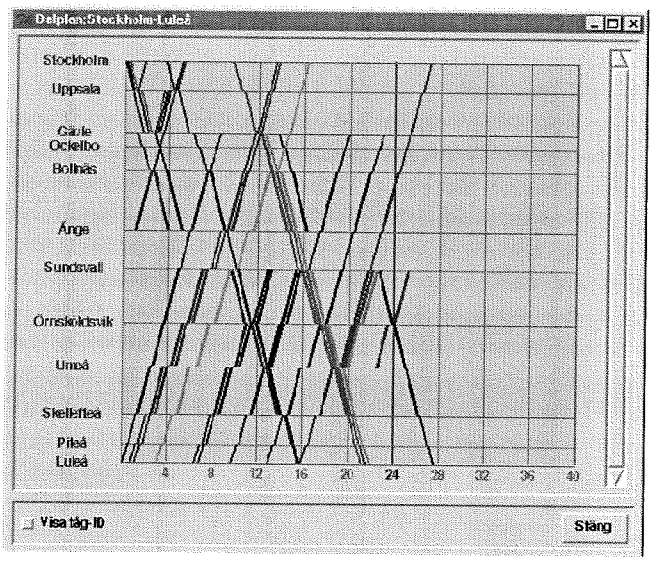

Figure 12: Longer wationg durations allowed

References

[AB93] A. Aggoun and N. Beldiceanu, *Extending CHIP in order to solve complex scheduling and placement problems*, Mathematical Computer Modeling **17** (1993), no. 7, 57–73.

[AC91] D. Applegate and W. Cook, *A computational study of the job-shop scheduling problem*, ORSA Journal of Computing **3** (1991), no. 2, 149–156.

[AHKL00] Martin Aronsson, Per Holmberg, Per Kreuger, and Simon Lindblom, *ACOOR rapport 1, TUFF systemöversikt och arkitektur*, SICS Technical Report T2000:06, Swedish Institute of Computer Science, 2000.

[AK01] M. Aronsson and P. Kreuger, *A constraint model for a cyclic time personnel routing and sceduling problem*, Tech. report, Swedish Institute of Computer Science, 2001.

[Bak74] K. R. Baker, *Introduction to sequencing and scheduling*, Wiley & Sons, 1974.

[BC94] N. Beldiceanu and E. Contejean, *Introducing global constraints in CHIP*, Mathematical computer modeling **20** (1994), no. 12, 97–123.

[BLP95] P. Baptiste and C. Le Pape, *A theoretical and experimental comparison of constraint propagation techniques for disjunctive scheduling*, Proceedings of the fourteenth international joint conference on artificial intelligence (Montreal, Quebec), 1995, pp. 400–606.

[BRRS88] L. Bianco, S. Ricciardelli, G. Rinaldi, and A. Sassano, *Scheduling tasks with sequence-dependent processing time*, Naval Research Logistics **35** (1988), 177–184.

[CL94] Y. Caseau and F. Laburthe, *Improved CLP scheduling with task intervals*, Proceedings of the eleventh International Conference on Logic Programming (Santa Margherita Ligure, Italy) (P. van Hentenryck, ed.), vol. 78, MIT Press, 1994.

[CP89] J. Carlier and E. Pinson, *An algorithm for solving the job-shop problem*, Management Science **35** (1989), no. 2, 164–176.

[CP94] J. Carlier and E. Pinson, *Adjustments of heads and tails for the job-shop scheduling problem*, European Journal of Operational Research **78** (1994), 146–161.

[Glo89] F. Glover, *Tabu search — Part 1*, ORSA Journal of computing **1** (1989), 190–206.

[Gu92] J. Gu, *Efficient local search for very large-scale satisfiability problems*, SigArt Bulletin **3** (1992), no. 1, 8–12.

[HG95] W. D. Harvey and M. L. Ginsberg, *Limited discrepancy search*, Proceedings of the International joint conference on artificial intelligence, 1995, pp. 607–613.

[JD95] C. Jordan and A. L. Drex, *A comparison of constraint and mixed-integer programming solvers for batch sequencing with sequence-dependent setups*, ORSA Journal on Computing **7** (1995), 160–165.

[KAHL00] Per Kreuger, Martin Aronsson, Per Holmberg, and Simon Lindblom, *ACOOR rapport 2, Översikt av tekniker och metoder*, SICS Technical Report T2000:07, Swedish Institute of Computer Science, 2000.

[KAL01] P. Kreuger, M. Aronsson, and S. Lindblom, *Task structure abstraction*, Tech. report, Swedish Institute of Computer Science, 2001.

[KCO+97] P. Kreuger, M. Carlsson, J. Olsson, T. Sjöland, and E. Åström, *The tuff train scheduler - Trip scheduling on single track networks*, The proceedings of the Workshop on Industrial Constraint-Directed Scheduling held at the Third International Conference on Principles and Practice of Constraint Programming (Schloss Hagenberg, Linz, Austria), Ed. A. Davenport, 1997.

[KGV83] S. Kirkpatrick, C. Gelatt, and M. Vecchi, *Optimization by simulated annealing*, Science **220** (1983), 671–680.

[Lec96] M. Leconte, *A bounds-based reduction scheme for constraints of difference*, Proceedings of Second international workshop on constraint-based reasoning (Key West, Florida), 1996.

[LM72] A. G. Lockett and A. P. Muhlemann, *A scheduling problem involving sequence dependent changeover time*, Operations Research **20** (1972), 895–902.

[Mal98] C. Malm, *Personal communication*, Swedish State Railway, Feb 1998.

[PL95] J.-F. Puget and M. Leconte, *Beyond the glass box: constraints as objects*, Proceedings of the International Logic Programming Symposium (*ILPS95*) (Portland) (J. Lloyd, ed.), 1995, pp. 513–527.

[PS82] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: Algorithms and complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[Rég94] J.-C. Régin, *A filtering algorithm for constraints of difference in CSP's*, Proceedings of the twelfth national conference on artificial Intelligence *AAAI-94*, 1994.

[Sim96] H. Simonis, *Modeling machine set-up time in CHIP*, Proceedings of the workshop on applications at the international conference of Constraint programming CP'96, 1996.

[SK92] B. Selman and H. Kautz, *An empirical study of greedy local search for satisfiability problems*, Proceedings of the national conference on artificial Intelligence *AAAI-92*, 1992, pp. 440–446.

[Smo95] G. Smolka, *The Oz programming model*, Computer science today (J. van Leeuwen, ed.), Lecture notes in computer science, vol. 1000, Springer verlag, 1995, pp. 324–343.

[ST+95] G. Smolka, R. Treinen, et al., *DFKI Oz documentation series*, German Research Center for Artificial Intelligence (DFKI), Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany, 1995.

[vH89] P. van Hentenryck, *Constraint satisfaction in logic programming*, Programming logic series, The MIT press, Cambridge, MA, 1989.

[Wal97] J. Walser, *Solving linear pseudo-boolean constraint problems with local search*, Proceedings of the national conference on artificial Intelligence *AAAI-97*, 1997.

[WW77] C. H. White and R. C. Wilson, *Sequence dependent set-up times and job sequencing*, International journal of production research **15** (1977), 191–202.

[XSS92] Y. Xiong, N Sadeh, and K. Sycara, *Intelligent backtracking techniques for job-shop scheduling*, Proceedings of the third international conference on principles of knowledge representation and reasoning, 1992.

[Zho97] J. Zhou, *A permutation-based approach for solving the job-shop problem*, Constraints **2** (1997), 185–213.