Coordination of Scheduling and Allocation Agents (extended abstract)

Thomas Sjöland, Per Kreuger, Emil Åström, Per Danielsson
{alf,piak,emil,pd}@sics.se

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

September 4, 1998

Abstract. This paper describes work in progress. We are designing a *distributed model for planning and scheduling of goods transports* in a railway company. Our model emphasises the use of an agent for the coordination of distinct subproblems occurring in railway scheduling such as e.g. the allocation of track resources to transports, the allocation of vehicles to transports, and the allocation of staff to perform the transportation tasks. The intention is that by establishing formal connections between the constraint spaces of agents handling different subproblems, a coordinating agent can be designed which is capable of guiding the subsystems towards a common solution. We thus obtain a coherent view on heterogeneous spaces and focus on the essential parameters of a problem. We outline the railway scheduling problem and the rostering problem and discuss techniques which are relevant in a distributed planning support system.

1 Coordinating for Scheduling and Allocation

We focus here on coordination of railway transport systems. Railway scheduling is normally treated with various techniques from operations research [17]. Recently real world scheduling and allocation problems have been successfully modeled and solved with constraint programming techniques [38, 58]. Since some of these problems are NP-hard in themselves and interact in complex ways, problem decomposition is necessary to handle real world problem sizes.

2 Agents, Concurrency and Distribution

By subdividing a planning system into possibly physically separated subproblem solvers and specifying protocols for the interactions between the different parts of the system a distributed software model is achieved. Such models are nowadays often named *agent models*, especially if the different parts have a certain degree of independence. One reason to use this design principle could be to clarify the communication between different actors (internal and external) and different organisational units. Another is to reduce the inherent complexity of producing a train plan.

Parallel or concurrent search and distributed solving of constraint satisfaction problems might speed up the finding of the solution compared to solving the problems one at a time or in a centralised manner[15, 68].

We will investigate techniques to partition large complex problems into smaller and less composite ones where communication between subproblem solvers is limited to e.g. information about contributions to a globally defined cost.

2.1 Negotiation Protocols

We are interested in finding interfacing principles to combine several spaces to allow planning agents, to have a uniform view of a heterogeneous set of spaces, while allowing efficiently controllable and reliably traceable interactions. We wish to relate constraint-based descriptions of the spaces with different levels of detail and formally relate the different levels to each other.

The agent modeling approach discussed above necessitates definition of *negotiation protocols*. Each planning agent produces a local plan. After the local plans have been produced, the computational agents start negotiating with each other. The negotiation is conducted with respect to a globally defined cost function covering the entire planning problem. After having exchanged tasks the computational agents can produce an improved plan. This process is iterated until the negotiation process halts or is interrupted. The resulting total plan, given through the combined local plans, should converge in this process towards an optimum. Typically the items exchanged in the protocol contains some form of plan abstractions (or description) for the relevant subplans representing limitations, cost estimates etc.

2.2 Support for Agent Models in Distributed Oz/mOzArt

One goal of our project is to develop base technology in the form of software infrastructure enabling a uniform access model of the information in a set of spaces. From the perspective of the user (or a control program) the different spaces should appear as a single one. The distribution is to be transparent since the access model abstracts away from the full details of the network and differences in data representation. Furthermore the access model offers a common information model and a uniform language for access and modification of the spaces.

The platform we consider is the mOzArt platform [30] for distributed programming in Oz [64, 65, 66]. By providing a communication layer for sharing of distributed objects the mOzArt platform will give a unique support for the design of models such as the one discussed above. We will base the design of this layer on the emerging standard FIPA for agent interaction [23].

3 The Railway Scheduling Problem

A railway network is a graph where the nodes are places where significant operations can occur. The operations are e.g. meeting between trains using the same part of the net in different directions, overtaking of slower trains by faster ones, waiting for safe security distances to be achieved, unloading of cargo, reassignment of engines, cars and staff etc. It is common that the nodes are classified in different categories, e.g. only a subset of the nodes allow reassignment of engines, some nodes allow engines and cars to be parked, some don't. There is also internal state in the nodes, which can be abstracted away, such as information about exactly which tracks are connected and how they can be used. The arcs are either directed or undirected, perhaps the same arc may be undirected at certain time intervals and directed otherwise, e.g. when the net is repaired. There are various constraints that must be fulfilled. For the purpose of this paper we are concerned only with constraints expressing allowed departure and arrival times, and allowed waiting times between arrival and departure and constraints that express the use of a resource.

The *scheduling problem* can be viewed as an allocation of trips to time slots on tracks such that safety distances are upheld when two trips use a track simultaneously and so that the constraints on the use of the nodes are upheld. We refer to [38] or [39] for more details of the finite domain constraint model used in our prototype scheduler.

Coordination of the planning of track resources and the allocation of engines to trains is our main concern. We will however in the following give some more details on our modeling of the sub problems.

4 Representations

We use a specification language for specifying *train orders*, a set of *trip specifications* over a rail network. The language allows the expression of more or less exact time specifications e.g. for departure and arrival times and for stops at arbitrary places in the route of a train. The language allows the specification of all the information required by the different planning components, including recurrent (e.g. daily or weekly) trips.

A rail network is represented as three interrelated sets:

- **Locations:** Meeting places, some allowing turning
- **Tracks:** directed and 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 local 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 (links) are either directed (only movements in one direction are allowed) or non-directed. More than one track can connect two nodes. In the case of three or more tracks between two nodes, the train order must express which route (and thereby which tracks) is to be used for a train.

Our network representation contains appr 330 links (tracks) and 270 nodes (meeting locations) with, for each station, an integer representing the available track resources, i.e. the maximum number of trains allowed to be simultaneously waiting at the station. We express the constraints on stations representing limits for track resources as cumulative constraints [1].

A route finally, is represented by a unique identifier and an ordered sequence of tracks (we use tracks rather than nodes since we wish to be explicit about the choice of tracks). Routes refer indirectly to a set of locations connected by its tracks. The problem of generating suitable routes for a given trip demand on a network, consisting only of locations and tracks, is not currently addressed in our work.

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

5 Problem Representation - Plans

5.1 Trips

Trips model individual trains traversing the network. A trip is represented by a unique identifier, the route it follows, an engine 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.

5.2 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 another duration we call 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 parameters are currently determined from the 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.

5.3 Plans and Schedules

We use the notion *plan* to denote a (set of) schedule(s) 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 a *trip*.

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 generated from the plan is a set of constraints over a set of finite domain variables and constants representing respectively the departure time, waiting duration, traversal duration and headway associated with each task.

5.4 Job-shop Scheduling

The simple case where the network consists only of double tracks can be modelled as a job-shop scheduling (JSS) problem [3, 10, 11, 5] viewing train trips as jobs and tracks as resources. Each train trip traversing a track represents a task where the traversal duration is 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 terms 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 only meeting points, splits and merges in the network. In this case, instead of taking the traversal duration as the task extension, we 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 [6, 8] or sequence dependent setup times with fixed durations [43, 74, 67, 37, 55]. Here we 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.

The scheduling problem that we 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 makespan

6 Rostering of Engines

Rostering is the problem of assigning resources (such as engines) to perform the train orders. The modelling in [24] is interesting background information.

We summarize some points that have arisen in a dialog with engine planning experts in SJ (the Swedish Rail):

- A *train chain* is a least unbreakable activity (mission) for an engine. A train chain can contain several train numbers but is performed by one and the same traffic unit (including the engine). The train chain defines the route on which the traffic unit is run, while the contained train numbers define departure and arrival times at turn/end points.
- An *engine roster* is a sequence of services (in this case train chains) to be performed in a particular order. Often it is also demanded that the rosters are closed circuits and that they can be performed repeatedly a number of times. The engines are planned to drive repeated (most often cyclical) routes, also called circuits.
- One critical aspect on the rosters is the connection and disconnection of cars, *turning*. The least time for this operation, *the turning time*, usually varies between 15 and 30 minutes, depending on e.g. properties of the station on which the turn occurs.
- A rostering plan will also contain a number of generated orders for empty trains which are to be added to the original train orders. Transportation of *empty trains* are planned train chains which are being performed by traffic units which only contain engines without cargo. These are used if the engine is needed at some other location than where an ordinary train chain has been finished. This is of course costly and should not be used if not necessary. A variant is to let engines be part of an already planned train chain which in

general should be preferred. In the case of engines this kind of a train chain is named a *multiple engine*. The empty trains necessary to create connected engine rosters should be also generated and scheduled as a result of the rostering planning phase. The allocation of engines should ideally (in order to minimize costs) utilize the existing set of engines in such a way that empty trips and waiting times at turns is minimized[1].
- *Type week* - the plan for a week, i.e. trains that recur regularly during a number of weeks. The rosters for engines are not currently described with type weeks.
- The cost measures for the rostering plan are e.g. accumulated waiting time for all rosters. Another demand is that the rosters should be approximately equally long so that the engines are used efficiently. This demand can also be formulated as a demand on waiting times for both the time an engine spends waiting to start its first trip and the time that passes from the end of its last trip until the time point when the last roster is finished. The cost for transfers of engines varies depending on which type of train is being handled.
  - Ordinary trains - least cost
  - Transportation of empty trains as part of an ordinary train, middle
  - Empty trains travelling alone, highest cost

The fixed cost for using an engine in a roster is also a parameter in the cost function.

In a planning phase for the network we consider there are 3900 train chains and up to 1500 possible trips of empty trains, i.e. 5400 train chains. As an estimate the planner must consider up to about 500 000 possible turnings.

The current state-of-the art planning systems are based on OR-techniques [24] and have the following performance: - 2 min, full scale engine rostering plan without considering engine types - 5 hours, plan considering engine types

This is due to a difference in complexity of the two formulations of the problem as discussed in [44].

6.1 Set Partitioning

The problem to determine the rosters can be formulated as set partitioning with additional constraint, e.g. regarding time [61]. The set of train orders should be divided in as many disjunct subsets as there are engines. Further conditions on each subset of trains is that it should constitute a connected roster where turnings are only allowed on a subset of all the meeting locations (the turning locations) and the order between trains is implied by the roster.

How the rostering problem can be formulated as a constraint problem is discussed e.g. by Simonis [61]. We have implemented a rostering algorithm for engines in the same spirit which handles a medium size problem. This corresponds to the state-of-the-art in this area[58].

[1] In strategic planning it is an important goal to minimize the number of engines needed to perform a typical set of plans.

It should be noted that the "set-partitioning" abstraction in itself doesn't give any support for the generation of suitable empty trains or multiple trains. How these shall be generated with the above sketched approach is not yet determined. Simonis [61] solves the problem (empty trains). Note that in the model an empty train can be added anywhere in a given roster and that we in order to find an optimal rostering plan need a cost function which considers both the total waiting time and costs of empty trains.

6.2 The Approach of the TUFF-project to Engine Rostering

In order to investigate the possibilities to allocate engine and track resources at the same time we specified and implemented an engine rostering algorithm using constraints. This algorithm allocates a determined or minimal number of engines to a train order so that all trips can in principle be performed within the given time interval and the trips are ordered in a connected sequence for each engine.

A rostering plan constrains the earlier train order by limiting the possible arrival and departure times for end stations significantly. Our current approach is to in a first phase specialize train orders to orders of rosters containing train orders with more precise demands on departure times and fixed traversal times. To each train we associate a predecessor or successor so that the set of trains is divided into disjunct sequences (rosters). With each train is also associated a waiting time before the next train in the roster departs. As a part of the problem we should also determine which trains start and stop each roster and conditions imposed on the locations at which this happens.

The result of a rostering plan is then in a second phase used to sharpen the input for the track allocation algorithm. The train planner verifies whether the rostering plan can be instantiated to a track allocation plan. If this is not successful, the initiative returns to the rostering planner which generates a new proposal. In any case an economic evaluation of the generated track allocation plan and overly costly parts of the plan are identified. For the purpose of eliminating these a bidding protocol between the two planners might be useful.

6.3 Representation of Rosters

Represent a roster as an ordered sequence of turnings. A turning is defined by a turning location V, an arriving train A with an interval for the allowed arrival time points, a departing train B with an interval for departure time and a variable Vt for the turning time (identical with the waiting time at the end station for A) the value of which is given by the difference between departure time and arrival time. The turning time variable Vt is limited downwards by a least turning time (defined by the station and the engine type) and upwards by requirements given by the turning location, individual rosters and the whole plan.

To determine B for all turnings given A and a set of constraints on Vt we call the engine allocation problem. B is to be selected among the trains that are scheduled for departures after A's arrival time at V. This set is only partly determined by a train order since the order between arrivals and departures at a meeting location is partly the result of the work performed by the track allocation planner.

What the rostering planner needs to do is thereby to partly determine (partially order) departures and arrivals at the turning locations, i.e. to possibly specialize the train order. Given such a specialized and extended train order a rostering plan with a reasonable waiting time can often be generated. If not, the train order is relaxed to make this possible.

We are now facing two problems:

– To generate a partial order of arrivals and departures for end stations without generating a complete schedule.
– Given such a partial order, to compute a rostering plan with reasonable limits on the total waiting time, roster lengths and empty trains.

An approach to generate (enumerate) the rostering plan could be to breadth-first (in parallel) at the end of each roster add as successor the train whose earliest departure time from the turning location is closest after its earliest arrival time. Limits on waiting times and time from the last arrival in each roster to the last in the whole plan can be assigned so that the length of the rosters (in time) is limited downwards.

7 Rostering of Staff

The rostering of staff concerns assigning staff to the train orders in such a way that all train orders are manned and that various regulations conerning e.g. vacation, working hours etc. are maintained. Lacking an appropriate modelling of the particularities of this problem, we consider for now this problem as just another rostering problem, orthogonal to the engine rostering.

8 Iterative Planning

Our software for planning of track allocations and engine rosters can be understood as a support system for a manager responsible of the planning task. A fundamental property is that planning steps can be iterated since the input and output data has the same format. This allows a successive planning where new orders are added to those that have already been given resources, even though this approach excludes the finding of optimal plans/schedules.

9 Modelling Train Planning with an Agent Model

We wish to define a suitable agent model expressing a division of the problem solving in sub problems solved (partly) independently.

The rostering planner for engines, together with the track allocation planner and their interfaces form a minimal agent model and the procedure for interaction between these is a rather trivial negotiation protocol.

Other approaches that might prove relevant concern e.g. auction models for allocation of track capacity [35].

9.1 Alternative Agent Models

In general it is possible to make a division into agents in several alternative dimensions. The choice of division is controlled partly by performance demands on the planning and partly by organizational demands on the architecture of the system (e.g. distributed use). In the case where the purpose is to considerably increase the performance by running the planning on several computers it is required that the problem can be divided in such a way that parallel computation is enabled.

In an agent model where a plan manager is responsible for the total plan cost optimization, the different agents could be made more independent and the generated plan could possibly be the result of an iterative process where the agents estimate the costs that occur in the frame of their respective areas of responsibilities and generate a set of internal bids (plan suggestions) converging towards a globally optimal cost.

9.2 Possible Problem Division

Some different approaches to subdivision of the planning/scheduling problem that might be useful are listed:

- **Regional** Regional planners can be agents that schedule the local plans on a regional level while a national planner takes care of the common plan for the long distance trains[2].
- **Train types** Divide the system based on the different types of trains.
- **Customer** Divide the system based on the different responsibilities for the service salesmen, i.e. different groups of customers.
- **Type week** Divide the planning in different time intervals with e.g. a planning agent per type week. Since this partitioning gives largely independent partial problems, at least for the scheduling problem, we might have here an interesting possibility for parallelization.
- **Organizational** Different planners are used for scheduling, engine rostering and staff rostering. The agent architecture could be separated based on these different subsystems.

A table indicates some different subdivisions in the timetabling problem for the swedish railroad.

[2] This method is currently implemented for the track allocation planner in an early prototype form.

Partial domains	tracks	engines	staff
X2000	X	X	X
Intercity	X	X	X
Interregional	X	X	X
Goods-national	X	X	X
Goods-regional	X	X	X
...	X	X	X

9.3 An Agent Model Based on Organizational Considerations

Figure 1 shows the agent model we currently consider.

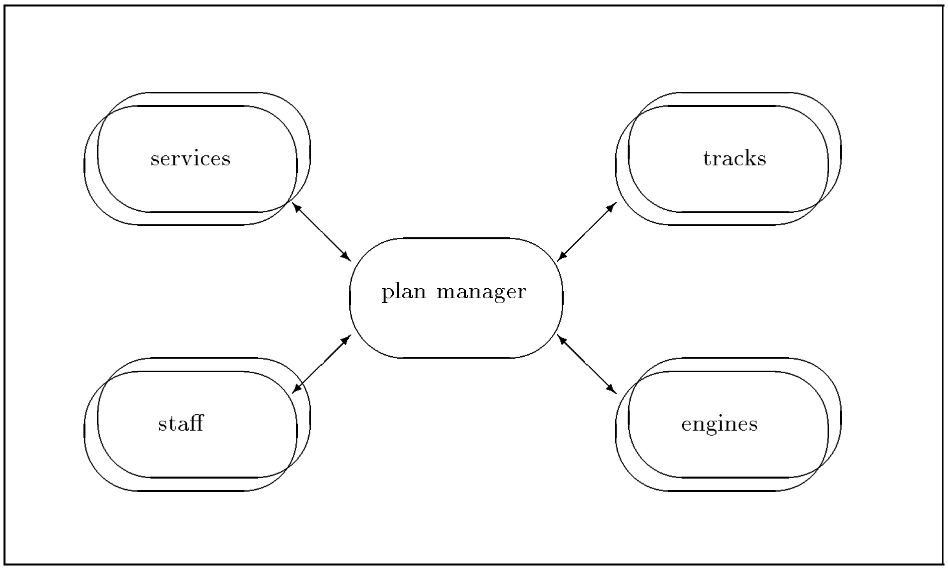

Fig. 1. Agent model based on organization.

The agent model divides the system as follows:

- The *service producers* generate train orders and negotiate contracts with customers
- The *track allocation planners* supply track resources (generate time table)
- The *engine planners* supply engines (generate engine rosters)
- The *staff planners* supply staff (generate staff rosters)
- The *plan manager* coordinates the negotiations and maintain the total plan

The different types of agents (except the plan manager) might exist in several *clones* each responsible for some sub-problem of the given type.

9.4 Protocols and Interactions

In order to make good suggestions concerning the coordination problem we need to understand:

- The underlying problem domains
- How (partial) solutions in the respective domains affect costs and opportunities in the others
- The "protocols" that are currently used in the manual process
- How the suggested plans are iterated today and how they are expected to look in five years
- What the software infrastructure will look in five years

9.5 Formalising the Protocols

Some of the work items remaining to formulate are

- Abstract descriptions of the different problem domains which are detailed enough to allow reasoning about proposed partial plans
- Representations of specifications and partial solutions in the respective domains
- Mechanisms to generate specialized specifications given abstract descriptions of the planning situation for the different domains
- Relevant and computationally reasonable cost functions for the respective domains and globally for the coordination agent

10 Sharing vs. Abstraction of Constraint Spaces

Solving constraints in a distributed fashion and coordinating separate solvers are related but different problems [68]. In distributed constraint solving the problem is essentially to distribute a constraint space over several computers, while coordinated solvers may or may not use distribution to solve separate problems. In the case of coordination there is a need to relate the constraint formulations used in the different solvers, thus enabling a coordinating agent to guide the sub solvers. Bidding strategies [25] similar to contract nets [63] should be expressed with explicit protocols where constraints or abstract descriptions thereof are communicated. The abstraction of constraint expressions from different spaces must guarantee that important information is not lost.

11 Abstraction for Reducing Complexity

Constraint solving is often limited by a prohibitively large search space, e.g. by the enumeration of many redundant variants.

The use of meta-operations such as constraint relaxation, redundant modeling [42] etc. could be utilized to improve the search behaviors of a solver. Another approach is to use abstraction to speed up search problems. This has been investigated in AI for general search problems [34].

11.1 Abstract Constraint Solving for Scheduling

Abstraction of constraint models can be used also for efficiency. A particular technical idea that comes naturally in our application is that of *Abstract Constraint Solving*. By constructing a suitable abstract constraint problem we get a constraint problem over a smaller domain than that of the immediate (concrete) formulation of the constraint problem.

By defining the abstraction so that a safeness relation (for non-solutions) is upheld we guarantee that a solution in the abstract domain corresponds to a concrete solution.

This methodology should generate feasible solutions faster than by solving the concrete problems directly, although optimal solutions that would be found with a more detailed larger search space might be lost in the process.

11.2 The Abstraction of Time in the Scheduling Problem

Time points in the constraints are represented as finite domain variables in our scheduler [38]. This means that the scheduling problem is discretized. Since different parts of the network might have different loads there is a possibility of generating a large number of redundant solutions with this approach. This is a problem in itself and for reaching an optimal solution various propagation methods and search heuristics might be applied. We are however primarily interested in feasible solutions since the makespan of the schedule could be controlled by a global parameter of the maximum time allowed for the total solution.

Our first suggestion for abstraction is fairly straight forward: Use fewer timepoints in the intervals described by the finite domain variables.

A second orthogonal suggestion is this: use a simpler network such that only those nodes where significant operations such as overturns, meetings and reallocation of engines or staff are allowed, are taken into account. A later phase can generate times for stops at intermediate nodes.

11.3 Soundness of the Abstraction

In order for an abstract constraint problem to be a solution a (negative) safeness criterion must hold between the set of solutions described by the concrete constraint C and its concretization g, and abstraction a.

$$s \notin C \rightarrow s \notin g(a(C))$$

This is understood as a statement that any tuple s that is *not* a solution to the concrete problem C is also not a solution to the concretisation of the abstract problem $g(a(C))$.

The equivalent logical reformulation

$$s \in g(a(C)) \rightarrow s \in C$$

gives the immediate property that we want.

In the case of using fewer timepoints this obviously holds as long as the timepoints occurring in the abstract problem are also in the concrete formulation.

In the case of using only a subset of the nodes the problem becomes a bit more subtle. Consider first a concrete example:

$$a \longrightarrow b \longrightarrow c \longrightarrow d \longrightarrow e$$

where the trip is associated with departure and arrival times for each of the nodes in such a way that the constraints hold.

Consider now an abstracted formulation of the constraint for the transportation between a and e:

$$a \longrightarrow c \longrightarrow e$$

We need to guarantee that the traversal time parameter for the arc between nodes a and c in the abstract network cover the longest possible time between a and c in the concrete network. This is of course doable [3] but might prohibit the finding of a schedule if the abstraction is applied on parts of the network where there is a lot of traffic.

11.4 Abstraction of Rosters in the Allocation Problem

For the set partitioning problem of allocating train orders to engines it could be useful to abstract away the explicit reasoning about time-points and rather use only the partial order between train orders implied by the geometry of the net.

Consider a partially ordered set of train orders $O=\{x_0 \xrightarrow{*} y_0, x_n \xrightarrow{*} y_n\}$ between nodes.

Partition the train orders over a set of engines $V = \{E_1 \ldots E_m\}$ in such a way that the partial order over O is guaranteed.

Here the important information is the order, rather than the exact timepoints of the turnings connecting the train orders into a roster.

The partial order can of course be expressed in terms of constraints over departure and arrival times, as indicated above, but it might be profitable to consider a different representation for the rostering problem solver, the result of which is then translated into constraints for the time points in the coordination protocol. It is desired that such translation of representations can be expressed as safe abstractions (and concretizations).

11.5 Abstraction Hierarchy

The safeness relation gives as expected a hierarchy of constraint problems (and domains) with increasingly less detail, and thus likely faster execution but possibly poorer plans the higher you reach in the hierarchy. By switching attention dynamically between different representation of the problem based on some

[3] Here we disregard the complication that particular constraints on locations dismissed in the abstraction, such as cumulative constraints on the number of co-located trains at such locations at any time point, might be hard to maintain.

heuristics it might be possible to keep a suitable quality of the plans while speeding up the solution process. This technique is known as widening/narrowing in abstract interpretation [18].

12 Conclusion

We have specified (loosely) the problems of scheduling and allocation in railway planning and outlined some considerations relevant to the design of an model with a coordination agent. We also discussed a methodology called *abstract constraint solving* and indicated its possible use in the scheduling and rostering problems.

The next few months our project will focus on a study of the planning-domains and their corresponding (OR-based) planning tools, possibly comparing the current approaches with constraint based tools. This will enable the design of a coordinating agent which is relatively independent of the nature of the underlying planner algorithms.

13 Acknowledgements

The research reported in this paper is a part of a study of the train planning domain currently done in cooperation with the Swedish State Railways (SJ). One of our motivations behind this cooperation is the opportunity to evaluate the feasibility of constraint and constraint logic programming [32] 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. The cooperation with the mOzArt team at SICS headed by Seif Haridi and the colleagues at DFKI is invaluable. We would also like to acknowledge Jan E. Olsson who contributed the initial idea to investigate this domain and initiated the contacts with our industrial partners.

References

1. Aggoun, A., Beldiceanu, N. "Extending CHIP in order to Solve Complex Scheduling and Placement Problems". In *Mathematical Computer Modelling* 17(7): pp. 57-73, Pergamon Press Ltd. 1993.
2. Applegate, D., Cook, W. "A Computational Study of the Job-Shop Scheduling Problem". In *ORSA Journal of computing*, 3(2): 149-156, 1991.
3. Baker, K. R. *Introduction to sequencing and scheduling* Wiley & Sons, 1974.
4. N. Beldiceanu, N, Contejean, E. "Introducing Global Constraints in CHIP". In *Mathematical Computer Modelling* 20(12): 97–123, Pergamon Press Ltd. 1994.
5. Baptiste, P., Pape, C. L. "A Theoretical and Experimental Comparison of Constraint Propagation Techniques for Disjunctive Scheduling". In the *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* Montreal, Quebec, pp. 400-606, 1995.

6. Bianco, L., Ricciardelli, S., Rinaldi, G., Sassano, A. "Scheduling tasks with sequence-dependent processing times". In *Naval Research Logistics* 35, pp. 177-184, 1988.
7. Brännlund U., Lindberg P.O., Nõu A., Nilsson J.E. "Railway Timetabling Using Lagrangian Relaxation", November 1996
8. Brucker P., Thiele O. "A branch & bound method for the general-shop problem with sequence dependent setup-times". In *OR Spektrum*, 18:1450-161, 1996.
9. Bussieck M.R., Winter T., Zimmermann U.T. "Discrete Optimization in Public Rail Transport". In *Mathematical Programming* 79 (1997) 415-444
10. Carlier, J., Pinson, E. "An Algorithm for Solving the Job-Shop Scheduling Problem". In *Management Science* 35(2): 164-176, 1989.
11. Carlier, J., Pinson, E. "A Practical Use of Jackson's Preemptive Schedule for Solving the Job-Shop Problem". In *Annals of Operations Research*, 26: 269-287, 1990.
12. Carlier, J., Pinson, E. "Adjustments of Heads and Tails for the Job-Shop Scheduling Problem". In the *European Journal of Operational Research*, 78:146-161, 1994.
13. Carlsson, M., Eriksson, H., Kreuger, P., Åström, E. *TACIT Project deliverable D2.2: Modelling of Ovako Steel Application*, (SICS), 1997.
14. Caseau, Y, Laburthe, F. "Improved CLP Scheduling with task intervals". In Van Hentenryck, P. (ed.) *Logic Programming, Proceedings of the Eleventh International Conference on Logic Programming.* Santa Margherita Ligure, Italy, MIT Press 1994. `http://www.ens.fr/ laburthe/papers.html#iclp94`
15. Censor Y., Zenios, A. S. *Parallel Optimization. Theory, Algorithms and Applications.* Oxford University Press. ISBN 0-19-510062-X
16. Christodoulou N., Stefanitsis E., Kaltsas E., Assimakopoulos V. "A Constraint Logic Programming Approach to the Vehicle-Fleet Scheduling Problem". In *Proc. of Practical Applications of Prolog PAP'94*
17. Cordeau J.-F., Toth, P. and Vigo D. *A Survey of Optimization Models for Train Routing and Scheduling.* February 23, 1998.
18. Cousot, P. "Draft Program Analysis Position Statement: The Abstract Interpretation Perspective". In *ACM Workshop on Strategic Directions in Computing Research*, MIT Laboratory for Computer Science, Cambridge, Mass. USA June 14-15, 1966
19. Crabtree I.B. "Resource Scheduling - comparing simulated annealing with constraint programming". In *Practical Applications of Constraint Technology PACT'95*
20. Dalfiume A., Lamma E., Mello P., Milano M. "A Constraint Logic Programming Application to a Distributed Train Scheduling Problem". In *Practical Applications of Prolog PAP'95*
21. David J.-M., Chew T.-L. "Constraint-based applications in production planning; examples from the automotive industry". In *Practical Applications of Constraint Technology PACT 95*
22. Davis R., Smith R.G. "Negotiation as a Metaphor for Distributed Problem Solving". In *Journal of Artificial Intelligence* 20 (1983) 63–109
23. Dickinson I. *Agent Standards.* Agent Technology Group. Hewlett-Packard Laboratories, Bristol, UK.
24. Drott J., Hasselberg E., Kohl N., Kremer M. *A Planning System for Locomotive Scheduling.* SJ Stab Tågplanering and Carmen Systems AB.

25. Fischer K., Müller J.P., Pischel M., Schier D. *A Model For Cooperative Transportation Scheduling* German Center for Artificial Intelligence (DFKI), Saarbrücken, Tyskland. (1995)
26. Fromherz M., Saraswat V. "Model-Based Computing: Constricting Constraint-Based Software for Electro-Mechanical Systems (extended abstract)". In *Practical Applications of Constraint Technology PACT 95*
27. Glover, F. "Tabu search - Part I" In *ORSA Journal of Computing*, 1:190-206, 1989.
28. Gu, J. "Efficient local search for very large-scale satisfiability problems" In *Sigart Bulletin* vol. 3 #1, 8-12, 1992.
29. Hammer M., Champy J. *Reengineering the Corporation* HarperBusiness, 1993
30. Haridi, S., Van Roy, P., Brand, P., Schulte, C. "Programming Languages for Distributed Applications". Invited paper in *New Generation Computing*, Vol. 16, No.3, pp 223-261, 1998. Ohmsa Ltd. and Springer Verlag., Tokyo.
31. Harvey, W. D., Ginsberg, M. L. "Limited Discrepancy Search". In *Proc. of the Intntl. Joint Conf. on Artificial Intelligence*, pp 607-613,1995.
32. Van Hentenryck, P. *Constraint Satisfaction in Logic Programming. Programming Logic Series.* The MIT Press, Cambridge, MA, 1989.
33. Herold A. "Experiences in Constraint Programming from the CHIC User Group". In *Practical Applications of Constraint Technology PACT 95*
34. Holte, R.C., Mkadmi, T., Zimmer, R.M., MacDonald, A.J. "Speeding up problem solving by abstraction: a graph oriented approach". In *Artificial Intelligence* 85 (1996) 321-261.
35. Isacson G., Nilsson J.-E. "An Experimental Comparison of First- and Second-Price Auctions - The Case of Track Capacity Allocation". In *Fifth International Conference on Competition and Ownership in Land Passenger Transport*, Leeds, England, maj 1997.
36. Joborn M. *Empty Freight Car Distribution at Swedish Railways - Analysis and Optimization Modeling.* Lic. Thesis Division of Optimization, Dept. of Mathematics, Linköping University, 1995.
37. Jordan, C., Drex, A. L. "A comparison of constraint and mixed-integer programming solvers for batch sequencing with sequence-dependent setups" In *ORSA Journal on Computing* 7: 160-165, 1995.
38. Kreuger P., Carlsson M., Olsson J., Sjöland T., Åström E. "The TUFF Train Scheduler". In proc. *Workshop on Constraint Environments* of International Logic Programming Symposium 1997, NY, USA. G. Puebla (coord.)
39. Kreuger P., Carlsson M., Olsson J., Sjöland T., Åström E. "The TUFF Train Scheduler - Two Trip Scheduling on Single track Networks". In *Workshop on Constraint Applications with Constraints* of International Conference on Constraint Programming 1997, Linz, Austria. A. Davenport (coord.)
40. Kirkpatrick, S., Gelatt, C., Vecchi M. "Optimization by Simulated Annealing". In *Science*, 220:671-680, 1983.
41. Leconte, M. "A bounds-based reduction scheme for constraints of difference". In *Constraint-96, Second International Workshop on Constraint-based Reasoning*, Key West, Florida, 1996.
42. Cheng, A B.M.W., Lee, J.H.M. and Wu, J.C.K. "Speeding Up Constraint Propagation By Redundant Modeling". In *Second International Conference on Principles and Practice of Constraint Programming*, pages 91–103, Springer-Verlag, LNCS 1118, Cambridge, Massachusetts, USA. August, 1996. Available in `http://www.cse.cuhk.edu.hk/ isl/dPub2.html#Li`.

43. Lockett, A. G., Muhlemann, A. P. "A scheduling problem involving sequence dependent changeover times". In *Operations Research* 20: 895-902, 1972.
44. Löbel A. *Optimal Vehicle Scheduling in Public Transit.* Dept. of Mathematics, Dr. Thesis, Fachbereich 3, Mathematik, Technische Universität, Berlin, 1997. Published by Shaker Verlag
45. Martin C. *Logistics and Supply Chain Management* Financial Times Pitman Publishing, 1992
46. Müller M., Popov K., Schulte C., Würtz J. *Constraint Programming in Oz.* DFKI, Saarbrücken, Tyskland, 1995
47. Müller M., Würtz J. *Finite Domain Programming in Oz.* DFKI, Saarbrücken, Tyskland, 1995.
48. Olsson J., Krueger P., Eriksson H., Sjöland T., and Åström E. *TUFF förstudie - Tågplanering För Framtiden.* KTM Project Report 96.
49. Olve N.-G., Roy J., Wetter M. *Balanced Scorecard*, Liber Ekonomi, 1997.
50. Puget, J.-F., Leconte, M. "Beyond the Glass Box: Constraints as Objects". In Lloyd, J. (ed.) *Proceedings of International Logic Programming Symposium (ILPS95)* pp. 513-527, Portland 1995.
51. Papadimitriou, C. H., Steiglitz K. *Combinatorial Optimization: Algorithms and Complexity* Prentice-Hall, Englewood Cliffs, NJ, 1982.
52. Régin, J. C. "A filtering algorithm for constraints of difference in CSPs" In *AAAI-94, Proceedings of the Twelfth National Conference on Artificial Intelligence.* 1994.
53. Risberg, T. "Simuleringsteknik på Spåret". In *Modern Datateknik* 1966:3, s29–33
54. Schulte C. *Open Programming in Oz.* DFKI , Saarbrücken, Tyskland, 1996.
55. Simonis, H. *Modelling Machine Set-up Time in CHIP* Technical Report COSY/TR, 1995.
56. Personal communication with Christer Malm at the Swedish State Railways (SJ), February 1998.
57. Selman, B., Kautz, H. "An empirical study of Greedy Local Search for Satisfiability Problems" In *Proc. AAAI-92*, pp. 440-446, 1992.
58. Simonis H. "A Problem Classification Scheme for Finite Domain Constraint Solving". In *CP'96 applications workshop*, COSYTEC SA, Orsay, France.
59. Simonis H. *Calculating Lower Bounds on a Resource Scheduling Problem.* COSYTEC SA, Orsay, France.
60. Simonis H., and Cornelissens T. *Modelling Producer/Consumer Constraints.* COSYTEC SA, Orsay, France and Beyers & Partners, Brasschaat, Belgium.
61. Simonis H. *The Use of Exclusion Constraints to Handle Location Continuity Conditions.* COSYTEC SA, Orsay, France.
62. Simonis H. *Modeling Machine Set-up Time in CHIP.* COSYTEC SA, Orsay, France.
63. Smith R.G. "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver". In *Readings In Artificial Intelligence*, pp357–366. (1980)
64. Smolka G. *A Survey of Oz.* DFKI , Saarbrücken, Tyskland, 1995
65. Smolka G. "The Oz Programming Model" In van Leeuwen, J, ed. *Computer Science Today.* Lecture Notes in Computer Science volume 1000: 324-343. Springer Verlag, Berlin, 1995.
66. Smolka, G., Treinen, R. eds. *DFKI Oz Documentation Series.* Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany 1995.

67. So, K. T. "Some heuristics for scheduling jobs on parallel machines with setups" In *Management Science* 36: 467-475, 1990.
68. Solotorevsky, G. and Gudes, E. "Algorithms for Solving Distributed Constraint Satisfaction Problems". In *AIPS-96* 191-198, 1996.
69. Wallace M. "Introducing the Practical Applications of Constraints". In *Practical Applications of Constraint Technology PACT 95*
70. Walser, J. "Solving Linear Pseudo-Boolean Constraint Problems with Local Search" In *Proc. AAAI-97, 1997.*
71. Van Marcke K., Tubbax B. *SKAI: A Knowledge Based Environment for Scheduling Traction Equipment and Personnel.* KT Memo 94-2
72. Van Marcke K. *Knowledge Based Scheduling with the SKAI Environment.* KT Memo 94-01
73. Van Marcke K. *Knowledge Based Scheduling: A Case Study.* KT Memo 93-3, Knowledge Technologies n.v.
74. White, C. H., Wilson, R. C. "Sequence dependent set-up times and job sequencing" *International Journal of Production Research* 15, pp. 191-202, 1977.
75. Würtz J. "Oz Scheduler: A workbench for Scheduling Problems". In *IEEE International conference on Tools with Artificial Intelligence (ICTAI'96)*, 1996.
76. Xiong, Y., Sadeh, N., Sycara, K. "Intelligent Backtracking Techniques for Job-Shop Scheduling" In the *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 1992.
77. Zhou, J. "A Permutation-Based Approach for Solving the Job-Shop Problem", In *Constraints*, 2:185-213, 1997.

This article was processed using the LaTeX macro package with LLNCS style