Proceedings of the 1992 IEEE
International Conference on Robotics and Automation
Nice, France - May 1992

Temporal Planning for Transportation Planning and Scheduling

Robert E. Frederking
Center for Machine Translation
Carnegie Mellon University
Pittsburgh, PA 15213
U.S.A.

Nicola Muscettola
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
U.S.A.

Abstract

The traditional view of transportation as a flow problem, to be solved by linear programming techniques, is seriously oversimplified when the domain includes inter-movement dependencies, dynamic changes in system parameters and the need for movement scheduling. In this paper we report preliminary work toward the creation of an integrated solution to these problems within the HSTS temporal planning framework. The reference problem is a simplified domain that displays some of the main characteristics of the complete transportation problem. The paper describes: how HSTS has been extended to represent aggregate resource capacity; how the simplified domain can be modeled; a constraint-directed planner that solves transportation problems in the simplified domain.

1 Introduction

Many problems in transportation can be represented as flow problems, and can be optimally solved using efficient linear programming techniques [4] [3]. But in some cases this approach is seriously oversimplified. If the problem includes dependencies between different operations, planning is necessary. If the system parameters change dynamically, the assumptions on which flow models are based become false, as in the case when the capacity of transportation facilities can change during the interval being analyzed. Finally, if detailed schedules are to be produced, answers in terms of bulk quantities do not suffice.

These problems require approaches that combine capabilities traditionally associated with planning and with scheduling, and that do not require their parameters to remain constant. Historically, temporal planners [5] [2] have dealt with combining general operators to achieve a set of goals over time but have poorly attended to issues related to the optimization of resource usage. On the other hand, schedulers [11] [10] have been concerned with allocating times and resources to operations in fixed process plans, ignoring questions of goal-oriented problem solving. The HSTS temporal planning framework [9] is an attempt to combine the capabilities of the two approaches. HSTS has been previously used for planning and scheduling the observations for the Hubble Space Telescope. HSTS emphasizes the description of the problem domain as a dynamical system organized through the use of **state variables**, i.e. persistent properties of objects in the domain. It also allows the development of opportunistic planners, where constraint posting and temporal inferences are not restricted to predefined directions on the time horizon (as in simulation and temporal projection) but the focus of problem solving can concentrate on the most congested areas of the time line.

In this paper we describe preliminary work done in the CORTES project[6], applying HSTS to a transportation planning and scheduling domain. First, we describe in more detail the transportation problems that we are addressing. We then describe the fundamental characteristics of HSTS and we concentrate on the representation of multiple capacity resources. We continue with a more detailed description of the transportation planning problem that we have initially addressed in HSTS and of its solution. Finally we describe future directions for our research.

2 The transportation problem

We are interested in addressing large-scale, complex transportation planning and scheduling problems, such as are found in disaster relief operations or other large-scale, international responses to emergency situations. For example, the transportation aspects of military operational plans (or OPLANs) must be feasible, given the allocated transportation resources [7].

0-8186-2720-4/92 $3.00

If not, they must be reworked, or have more resources allocated to them. OPLANs are very large, involving the movement of tens of thousands of individual units, which vary immensely in size and composition, from a single person or piece of cargo to an entire division. However, OPLANs do not explicitly represent justifications for precedence constraints due to the structure of the domain and are therefore difficult to modify or adapt to other situations. To concentrate on the representation of domain structure in a transportation schedule, we addressed the 'bare base' deployment scenario used at the Armed Forces Staff College (AFSC) to train joint planning officers. The goal is to turn a bare runway into a fully functioning air base. Our analysis revealed two facts:

- The domain requires the ability to represent and reason about aggregate capacity resources.
- This domain consists primarily of a moderate number (order of 10) dependency cycles, each centered around a different support function, such as air traffic control, aircraft refueling, personnel or cargo unloading, etc. The arrival of support units increases the possible arrival rate of additional support units.

We isolated one of the dependency cycles, the refueling capacity/throughput loop, as an initial 'atomic' domain. The **base refueling capacity** gives an indication of how many planes can be refueled in parallel at the base. It can be increased by bringing more **refueling units** to the base. The arrival of a unit permanently increases refueling capacity, which in turn affects the rate at which **planes** can arrive, since they use some amount of refueling capacity immediately after moving. This increases also the rate at which additional units can be brought in. The representation and solution of this problem is an important step toward a solution of the bare base scenario.

3 Representing plans in HSTS

Transportation problems require to be able to deal with dependencies involving state and resource capacity (e.g., a unit that requires a plane to move from A to B can be allocated space only on a plane that is also moving from A to B). This can be done by using the HSTS planning and scheduling framework [9]. The two main components of the framework are a **domain description language**, for modeling the structure and dynamics of the physical system at multiple levels of abstraction, and a **temporal data base**, for representing possible evolutions of the state of the system over time.

In this section we describe the basic primitives provided by HSTS and the extensions needed to represent aggregate resource capacity.

3.1 Representing state

An HSTS model is subdivided into **state variables**, each of which can assume one and only one value in any instant of time. A value has the form $R(x_1, x_2, \ldots, x_n)$. For example, a plane *?p* has a location, represented by state variable $Loc(?p)$, that can assume value $MOVE(?p, ?u, ?src, ?dst)$ representing the fact that $?p$ is in transporting unit *?u* from location *?src* to location *?dst*. HSTS is interval based, i.e., if a value occurs on a state variable, it persists for a continuous non-zero time interval. A value can occur under conditions specified through a **duration specification** and a **compatibility specification**.

Figure 1 shows a hypothetical value descriptor. The **duration** is expressed as a range constraint, $[d, D]$, with d and D representing respectively a lower bound and an upper bound function. The rest of the descriptor specifies the **compatibilities** that have to be satisfied. A compatibility specification is an AND/OR graph connecting several elementary compatibilities. Each compatibility is composed of a temporal relation and the specification of a segment of behavior on a state variable. For example the compatibility $[met\_by\ \langle \nu, Loc(?p), AT(?src)\rangle]$ associated to $\langle Loc(?p), MOVE(?p, ?u, ?src, ?dst)\rangle$ in Figure 1 specifies that in every legal behavior, the value $MOVE$ must occur immediately after the value AT on $Loc(?p)$. The symbol ν is one of two different kind of segments of evolution of a state variable: ν, constraining a single value, as in the example above, and σ, for sequence compatibilities. A sequence that can be substituted by an unspecified number of values occurring on the same state variable, all of which must satisfy a constraint associated with the sequence. We will see examples of sequences when we will discuss aggregate capacity state variables.

The temporal relations used in HSTS are equivalent to those described in [1] but augmented with metric temporal distences; for example, $T_1\, before([d, D])T_2$ indicates that the start of token T_2 follows the end of token T_2 with a delay falling within the range $[d, D]$.

Behaviors can be constructed within the HSTS Temporal Behavior Data Base. The unit of description of temporal behavior is the **token**, a quadruple $\langle sv, type, st, et\rangle$, where sv is one of the state variables in the system model, $type$ is a subset of the state vari-

$\langle Loc(?p), MOVE(?p, ?u, ?src, ?dst)\rangle$
duration: $[dur(?src, ?dst), Dur(?src, ?dst)]$
compatibilities:
AND ($[met\_by\ \langle \nu, Loc(?p), AT(?src)\rangle]$
$[meets\ \langle \nu, Loc(?p), REFUEL(?dst)\rangle]$
$[eql\ \langle \nu, Loc(?u), MOVE(?p, ?u, ?src, ?dst)\rangle]$)

Figure 1: HSTS value descriptor

able's possible values, and *st* and *et* are the token's start and end times respectively. Tokens represent an uninterrupted segment of evolution of a state variable. During the planning process a token can be refined by being split into any number of component tokens; however, a token that has been designated to represent the occurrence of a value cannot be further split. A token that can be split is referred to as a **plan constraint**; one that cannot be split is referred to as a **plan value**. The TDB also allows the representation of **token sequences** which implement the occurrence of a sequence specification. Tokens and token sequences are connected by a network of constraints: **temporal constraints**, relating the start and end times of each token, and **type constraints**, referring to the type of each token. Temporal and type constraints derive either from the expansion of compatibilities and durations extracted from the model of the system, from requirements directly imposed by the user and therefore constituting the problem to be solved, or from refinement decisions taken during the problem solving process where one of multiple alternatives needs to be explored.

3.2 Representing Aggregate Resource Capacity

At the base of the HSTS representation philosophy is the assumption that it is possible to identify each state variable into which a system model is decomposed and that each state variable can assume one of a handful of symbolic values. However, this basic mechanism of representation can become very cumbersome. For example, to reason on the allocation of available space on a plane to materials, we would have to subdivide space on the plane into "unit of space" state variables, with values 'free' or 'used', subdivide also the materials into units of space, and allocate capacity each unit of material space to a unit of plane space. Although this might be necessary for a detailed map of the allocation of plane space, it is overly detailed for cases when we need only an aggregated characterization of the use of space.

HSTS can represent aggregated capacity as an **aggregate state variable**. The value of an aggregate state variable at a given time is a summary of the value of a corresponding set of atomic state variables at the same instant of time. In the transportation planning domain, the use of cargo or parking space or the generation or use of refueling capacity by a unit or plane at a base falls into this category.

A set of atomic state variables constitutes the conceptual base on which the aggregation is built. In our discussion, they are **atomic resources** that can be used by one and only one **operation** at a time. An operation OP_i is the value assumed by the *state* state variable of a **job** $?j$, $St(?j)$, while $?j$ is undergoing the specified operation. If $?j$ is not undergoing any operation, the value of $St(?j)$ is *IDLE*. An atomic resource $?r$ has a single atomic state variable, $St(?r)$, with possible values *OPER* (processing some operation) and *IDLE*.

The occurrence of OP_i and of *OPER* is regulated by the following bidirectional compatibility:

$$\langle \nu, St(?j), OP_i\rangle\, eql\, \langle \nu, St(?r), OPER\rangle$$

If the atomic resources in a pool $?r\_p$ are perfectly substitutable, they can be aggregated into a single aggregate state variable, the **aggregate processing capacity** of the pool, $Cap(?r\_p)$. At any instant of time, the aggregate state variable will assume a single value that will summarize the distribution of values over its component state variables at that time. $Cap(?r\_p)$ gives the number of resources in the pool that hold each of the values *OPER* and *IDLE*; its values are represented as follows:

$$\{\langle OPER, n_1\rangle, \langle IDLE, n_2\rangle\}$$

indicating that n_1 atomic resources in $?r\_p$ are in an *OPER* state and n_2 are in an *IDLE* state. The number of resources in $?r\_p$ at that instant of time is $n_1 + n_2$. In general, a value for an aggregate state variable is a list of such entries $\langle value, counter\rangle$.

Compatibility constraints on values of aggregate state variables specify one or more atomic values and, for each value, the number of atomic resources affected. For example, assuming that OP_i requires c_i atomic resources, we will have:

$$\langle St(?j), OP_i\rangle \rightarrow [eql\ \langle \sigma, Cap(?r\_p), \{\langle OPER, INC(+c_i)\rangle, \langle IDLE, INC(-c_i)\rangle\}\rangle]$$

This means that whenever OP_i occurs, a sequence of values must be found on $Cap(?r\_p)$, and the start

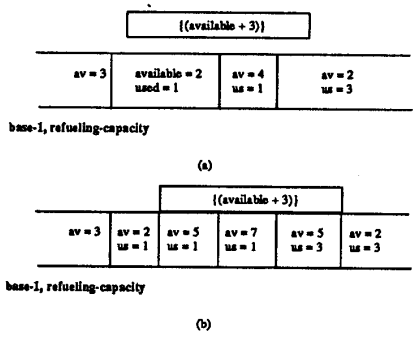

Figure 2: Posting a sequence constraint on an aggregate capacity state variable

and end times of the sequence must coincide with the start and end of OP_i, as indicated by the temporal relation *eql*. The type specification describes the local effect of the compatibility on each of the values in the sequence, i.e, the number of atomic resources that are *OPER* is incremented by c_i, while the number of those that are *IDLE* is decremented by c_i.

At time τ, the actual value of an aggregate state variable can be computed once the set of constraints that contain τ is known. If we suppose we have n_{opr} entries of type $\langle OPER, INC(c_i)\rangle$ and n_{idle} entries of type $\langle IDLE, INC(c_j)\rangle$, the value $\{\langle OPER, n_1\rangle, \langle IDLE, n_2\rangle\}$ at time τ satisfies the relations:

$$n_1 = \sum_{i=1}^{n_{opr}} c_i \qquad n_2 = \sum_{j=1}^{n_{idle}} c_j$$

where c_i and c_j can be both positive (creation) or negative (consumption).

During the planning process, the evolution of an aggregate state variable is represented in the temporal data base by a sequence of plan constraints determined by the imposition of a set of sequence constraints (Figure 2). Note that the temporal extension of each aggregate state variable's value is not fixed. This is an important difference from other scheduling systems, where the times must be fixed if the values of aggregate capacities are to be fixed [11] [10].

Consistency of the state of a temporal data base can be checked by temporarily assuming that no more sequence constraints will be posted and, therefore, the plan constraints can be safely substituted with plan values that can be computed by applying constraints like those for n_1 and n_2, above. The data base will be inconsistent when an aggregate value contains a counter whose value is negative. Notice however that, in the case where the physical system allows the generation of capacity (as for aggregate processing capacity), partial inconsistency can be resolved without backtracking by posting additional compatibilities providing the missing capacity.

4 Transportation Planning within HSTS

4.1 Representing the atomic domain

The state variables in this domain are the refueling and throughput properties of three types of objects: **units**, **planes**, and **bases**.

Each unit has two associated state variables, its location *Loc* and its state *St*. A unit's *Loc* can have the values *AT* and *MOVE*. These correspond to the unit being stationed at some base (e.g., home or destination) or being in transit. A unit's *St* can have the values *NOT_OPER* or *OPER*. These indicate whether it is capable of providing refueling capacity. When *OPER*, it adds enough capacity to refuel one additional plane. Each plane has one state variable, *Loc*, which can have the values *IDLE*, *MOVE*, and *REFUEL*.[1]. A base has one aggregate state variable, its refueling capacity *R_C*, containing distributions of two values, *AVAIL* and *USED*, indicating the total amount of available and used refueling capacity at any time. The principal compatibilities describing this problem are:

- A unit becomes *OPER* δt units of time after a *MOVE*[2]:

  $\langle Loc(?u), MOVE(?p, ?u, ?src, ?b)\rangle \rightarrow$
  $[bf([\delta t, \delta t])\ \langle \nu, St(?u), OPER(?u, ?b)\rangle]$

- The *MOVE* of a unit is concurrent with the *MOVE* of a plane:

  $\langle Loc(?u), MOVE(?p, ?u, ?src, ?b)\rangle \rightarrow$
  $[eql\ \langle \nu, Loc(?p), MOVE(?p, ?u, ?src, ?b)\rangle]$

- The *MOVE* of a plane is immediately followed by *REFUEL*:

[1]For this abstract model, representing refueling state and location separately would have introduced irrelevant complications.

[2]This is necessary to prevent a degenerate problem, where each unit brought in adds enough capacity to handle its own plane, thus immediately allowing an arbitrary number of units to be brought in.

$\langle Loc(?p), MOVE(?p, ?u, ?src, ?b)\rangle \rightarrow$
$[meets\ \langle \nu, Loc(?p), REFUEL(?p, ?b)\rangle]$

- The unit increases $R\_C(?b)$ while it is *OPER*:

$\langle St(?u), OPER(?u, ?b)\rangle \rightarrow$
$[eql\ \langle \sigma, R\_C(?b), \{\langle AVAIL, INC(+1)\rangle\}]$

- The *REFUEL* of the plane creates a demand on $R\_C(?b)$:

$\langle Loc(?p), REFUEL(?p, ?b)\rangle \rightarrow [eql\ \langle \sigma, R\_C(?b),$
$\{\langle USED, INC(+1)\rangle, \langle AVAIL, INC(-1)\rangle\}\}]$

4.2 The atomic domain planner

The HSTS model of a domain creates an implicit space of legal sets of state variable value sequences, within which any partial (or complete) solution to a problem must lie. However, in order to describe any specific partial solution, a particular set of legal choices must be made. Many such sets of choices will result in inconsistent sets of compatibilities, not corresponding to any possible system behaviors. Finding a consistent set of choices (i.e., planning) can still be very difficult. Within the HSTS least-commitment framework, the final solution is a representation of a range of behaviors that can be directly simulated, all guaranteed legal.

In the case of transportation planning and scheduling, one must select actual units to supply required support, and select actual ranges of arrival times for these units. This selection is ultimately based on the needs of some set of units whose operation at the destination directly fulfills external (top-level) goals. In our domain, the top-level goal is represented as a request for a large amount of *USED* refueling capacity during some future interval. The posting of the request creates an interval of time in which the *AVAIL* capacity is negative.

The planning process begins with an HSTS **fetch** for intervals where the base refueling capacity is below zero, locating the top-level problem. The planner then finds which types of values provide the type of capacity needed, and which state variables can have these values. It selects enough instances of these variables to satisfy the demand, creates the appropriate value tokens for them, and constrains these tokens to occur over the required interval. This solves the top-level problem.

Then, for each of these state variables, its token's compatibilities are **implemented**, that is, constraints between the token and other values are enforced, to guarantee that this is a legal behavior. Single-unit compatibilities are done first, followed by those that affect other units. This ordering is important in general, since local constraints may limit the choices available to more global ones. This corresponds to developing process plans for individual jobs before scheduling their operations. Since dependencies between different units of the same type are expressed through aggregate variables, this ordering is equivalent to saying that compatibilities that do not affect aggregate variables are done first. The set of local compatibilities in this domain are simply the first three listed in the previous subsection.

Next, the compatibility for the effect of plane refueling on the aggregate capacity is implemented. This requires the choice of a particular time interval for plane refueling, relative to the intervals of different levels of aggregate capacity. This is done in one of three modes: planes are allocated times as late as possible, as early as possible, or at user-selected times. This variability demonstrates the complete flexibility of the order of decisions in 'simulated time' (time in the modeled domain). This flexibility allows for **opportunistic** decision-making, where decisions are made in the most efficient order, not in any pre-determined temporal order. Other temporal planners generally cannot make decisions this flexibly when working at the most detailed level, as in [5].

Finally, the effect of the unit becoming operational on the aggregate capacity is implemented. In both of these last two steps, some search may be needed. The interval initially chosen may not produce a legal configuration, due to the simple mechanism for picking an interval and a limitation of the current aggregate variable mechanism. Currently, once the contribution from a state variable to an aggregate variable is calculated, its relative position in the aggregate cannot be changed without backtracking. This violates the least-commitment principle, and leads to problems: some intervals on the aggregate variable have zero length. If our simple interval selection rule selects a zero-length interval for a non-zero length event, an inconsistency results. Currently the easiest way to handle this is to implement, detect the inconsistency, and backtrack. Very limited search is needed, since non-zero intervals are much more frequent. Fixing this failure of least-commitment is high on our research agenda.

When these steps have been carried out for the necessary number of variables, a complete and consistent behavior has been described that fulfills the top-level goals.

5 Conclusion

Temporal planning methodologies can be applied to solve transportation planning problems that are beyond the scope of traditional linear programming techniques. In our work we have addressed one such problem and identified a fundamental type of dependency among its entities. We have then demonstrated that problems involving this kind of dependency can be solved within the HSTS temporal planning and scheduling framework. To solve the full bare base deployment scenario, we need to extend our current problem solver to incorporate heuristic knowledge in order to select the most appropriate units and time intervals for values, and carry out local search if necessary.

In order to deal with real-world scale problems, it will be necessary to develop further problem aggregation and abstraction techniques. One promising direction concentrates on taking advantage of the temporal flexibility of the HSTS framework by combining least-commitment constraint posting methodologies with probabilistic estimates of resource usage [8]: the goal is to avoid spelling out unnecessary details whenever possible while insuring high quality possible executions of the temporal plan.

References

[1] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

[2] J.F. Allen and J.A. Koomen. Planning using a temporal world model. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 741–747, 1983.

[3] L. Bodin, B. Golden, A. Assad, and M. Ball. Special issue on the routing and scheduling of vehicles and crews. *Computers and Operations Research*, 10(2), 1983.

[4] S.P. Bradley, A.C. Hax, and T.L. Magnanti. *Applied Mathematical Programming.* Addison-Wesley Publishing Co., 1977.

[5] T. Dean, R.J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel time, and resources. *Computational Intelligence*, 4:381–398, 1988.

[6] M.S. Fox and K. Sycara. Overview of cortes: A constraint based approach to production planning, scheduling and control. In *Proceedings of the Fourth International Conference on Expert Systems in Production and Operations Management.* ESPOM-90, 1990.

[7] S.H. Hanes, editor. *The Joint Staff Officer's Guide.* U.S. Government Printing Office, 1988. Publication 1, Armed Forces Staff College.

[8] N. Muscettola and S.F. Smith. A probabilistic framework for resource-constrained multi-agent planning. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 1063–1066. Morgan Kaufmann, 1987.

[9] N. Muscettola, S.F. Smith, A. Cesta, and D. D'Aloisi. Coordinating space telescope operations ina an integrated planning and scheduling architecture. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1369–1376, 1991.

[10] N. Sadeh. *Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling.* PhD thesis, Schhol of Computer Science, Carnegie Mellon University, March 1991.

[11] S.F. Smith, P.S. Ow, J.Y. Potvin, N. Muscettola, and D. Matthys. An integrated framework for generating and revising factory schedules. *Journal of the Operational Research Society*, 41(6):539–552, 1990.