Temporal Reasoning for Mixed-Initiative Planning*

Jonathan Stillman Richard Arthur James Farley
Information Technology Laboratory
General Electric Research and Development Center
P.O. Box 8, Schenectady, N.Y. 12301
e-mail: {stillman, arthurr, farley}@crd.ge.com

Abstract

We discuss recent research involving the temporal reasoning system Tachyon, which we are developing at GE, and its application to mixed-initiative planning in the ARPA/Rome Laboratory Planning Initiative (ARPI). Tachyon is an implementation of a constraint-based model for representing and reasoning about qualitative and quantitative aspects of time. It's data model provides substantial expressiveness, fast computation over convex intervals. We have applied Tachyon in a number of areas, including scheduling, project planning, feasibility analysis, and spatio-temporal data management. In this paper we will provide an introduction to Tachyon followed by an overview of recent work undertaken to support experimentation in the air campaign planning domain.

Introduction & Motivation

Temporal reasoning problems arise in many areas, as widely ranging as physics, biology, cosmology, and psychology. In addition to these, reasoning about time is critical in numerous computer applications: databases, simulators, expert systems, and industrial scheduling and planning systems (minimizing assembly line slack time, projecting critical steps in a deployment plan to insure proper interaction between them, etc.) all need to manipulate temporal information to model the world. Although some ability to model and manipulate such information has often been built into existing software systems, it has usually been minimal, making simplifying assumptions and failing to capture the richness and complexity that accompany full integration of temporal representations. This is especially true when incompleteness of information, non-instantaneous events, complex temporal relationships, or uncertainty pervade the application domain.

Issues in temporal reasoning and representation have provided fertile ground for research for many years, and many fascinating questions remain unresolved. Our work has been driven by the desire to develop an approach to temporal reasoning that can provide effective, efficient temporal reasoning capabilities critical to such applications. The research is thus driven in two directions: to expand the understanding of the theoretical underpinnings of temporal reasoning, and to develop practical, usable software that can be applied in a number of settings. Our approach is embodied in the prototype software tool *Tachyon*[1]. We recognized Tachyon's applicability to a number of problems of both military and commercial interest, and have applied it to plan recognition tasks, where it was used to validate temporal sequencing of events as an aid in formulating plan hypotheses, to plan generation and execution monitoring, to force expansion, and to retrieval and situation refinement in a prototype spatio-temporal data management system. In this last application, we used Tachyon's constraint propagation capabilities together with partial information about interrelated events to provide temporal refinement for tasking support (Stillman 1993a; Stillman 1993b).

*This work has been supported in part by ARPA and US Air Force Rome Laboratory.

Technical Background

When one examines the ways in which temporal information occurs and interacts in planning and scheduling domains, a number of capabilities and features emerge that we felt were critical to provide in a versatile and powerful temporal reasoning system. These include the ability to:

- deal with uncertainty regarding the exact time and duration of occurrence of events[2], e.g., *X will occur sometime in the morning*, and *truck refueling takes between 15 and 40 minutes*,
- express both quantitative and qualitative constraints between events, e.g., *X is before or meets Y*, and *X ends between 10 and 15 minutes before Y starts*,

[1] A *tachyon* is a theoretical subatomic particle capable of traveling faster than the speed of light, perhaps even traveling backwards through time. The name is chosen for its relation to time, and our emphasis on performance.

[2] Although we will use the term *event*, it should be noted that one could as easily refer to an arbitrary proposition that has temporal extent.

- express parameterized qualitative constraints between events, e.g., *X is before Y by at most 6 days*,
- provide multiple granularities, e.g., seconds, hours, days, etc., and their combinations, e.g., days:hours:minutes, day:month:year,
- promote ease of use via graphical input and display capabilities,
- run as a subprocess in other applications as well as stand-alone,
- utilize techniques that will remain effective even in very large application domains,

Tachyon's Data Model

A graph-based *temporal constraint network* paradigm served many of our needs well, thus providing a good starting point for the Tachyon data model. Other researchers have also explored the use of constraint networks in temporal domains. Those most relevant to our pursuits include (Valdés-Pérez 1986), (VanBeek & Cohen 1990), and (Dechter, Meiri, & Pearl 1991). Temporal constraint networks are a specialization of general constraint networks, formulated by (Montanari 1974). A constraint network is simply a graph in which *nodes* correspond to *variables* and *edges* constrain the values the associated variables can be assigned. The *constraints* express relations between the variables. Assigning unique domain values to the variables is an *instantiation*. An instantiation satisfies a constraint if the variable assignments do not violate the constraint. A graph instantiation is *consistent* if it satisfies all the constraints of the network. A temporal constraint network is a constraint network in which variables correspond to events, and constraints represent the relationships between pairs of events.

Temporal constraint networks typically use quantitative values to express the allowable relationships between events. The other prominent representational paradigm for time is qualitative, as exemplified by James Allen's interval calculus (Allen 1983). Table 1 shows a listing of the qualitative temporal relations that make up Allen's calculus. One can see that these relationships form a natural set, providing adequate expressiveness for many qualitative temporal reasoning problems.

Tachyon's model, like that of (Dechter, Meiri, & Pearl 1991), is quantitative in nature, i.e., point-based rather than interval-based, but assumes a more complex notion of event, and provides capabilities for both quantitative and qualitative constraints to be expressed and manipulated.

More formally, a Tachyon network is a directed graph $G = (V, E)$, in which each element of V is an ordered 6-tuple $(eps, lps, epf, lpf, d_{min}, d_{max})$, where *eps* is *earliest possible start*, *lps* is *latest possible start*, *epf* is *earliest possible finish*, *lpf* is *latest possible finish*, d_{min} is *minimum duration*, and d_{max} is *maximum duration*, and in which each element of E is an ordered pair of elements of V. Furthermore, elements of E are labeled with *constraints*. Tachyon edge constraints are expressed internally by an 8-tuple, the semantics of which are described in Table 2.

The 8-tuple constraint between node 1 and node 2:
Minimum time between Start 1 and Start 2
Maximum time between Start 1 and Start 2
Minimum time between Start 1 and Finish 2
Maximum time between Start 1 and Finish 2
Minimum time between Finish 1 and Start 2
Maximum time between Finish 1 and Start 2
Minimum time between Finish 1 and Finish 2
Maximum time between Finish 1 and Finish 2

Table 2: Constraint 8-tuple

In describing many temporal reasoning problems one needs to specify constraints for which constraint propagation is intractable in this paradigm. Such constraints are called *nonconvex*. Unfortunately, introducing them may greatly increase the complexity of processing the network (Arthur & Stillman, 1992; Stillman, Arthur, & Deitsch 1994). Nonconvex constraints are needed to reason about multiple tasks that require a single resource (e.g., vehicle/tool/machine) where there are no precedence constraints between the competing tasks. Such situations arise frequently in planning and scheduling domains. While we do not claim to provide a general solution to this problem (this is highly unlikely as the problem of determining whether a solution exists is NP-complete) one key purpose of our developing a temporal reasoning environment is to serve as a test-bed for evaluating new methods of coping with nonconvex constraints.

Qualitative constraints are expressed by introducing two concepts, *epsilon*, the smallest distance possible, and *infinity*, the largest. Convex disjunctions of Allen's interval constraints are thus allowed trivially, and nonconvex are allowed through disjunctions of 8-tuples. Tachyon's model also allows for parameterization of the constraints **before**, **overlaps**, **overlapped by**, and **after**. Each of these is given the ability to take on two parameters, representing the minimum and maximum distance to which they refer. One must exercise care in introducing such parameterized qualitative relationships, however, as they can introduce intractability. Otherwise convex disjunctions may lose convexity when parameters are added.

Propagation of Constraints

Tachyon uses a modification of the Bellman-Ford single source shortest-path algorithm to propagate temporal information and tighten the bounds of variables in the graph. This is sufficient for graphs that consist solely of convex constraints, and differs from Dechter *et. al.*, who use the Floyd-Warshall all pairs shortest

before	←x→ ←y→	after	←x→ ←y→
meets	←x⊣ ⊢y→	metby	⊢x→ ←y⊣
overlaps	←x→ ←y→	overlappedby	←x→ ←y→
starts	⊢x→ ⊢y→	startedby	⊢x→ ⊢y→
during	←x→ ←y→	contains	←x→ ←y→
finishes	←x⊣ ←y⊣	finishedby	←x⊣ ←y⊣
equals	⊢x⊣ ⊢y⊣		

Table 1: **Allen Relations** The relation x · (e.g., before)- y is illustrated by the relative positions of the intervals in this table. Simultaneous starts and finishes are indicated by vertical end-brackets (⊢), while the angular end-brackets (←) indicate otherwise. Line length represents relative duration of the events.

path algorithm. Descriptions of these can be found in most algorithms textbooks, e.g., (Cormen, Leiserson, & Rivest 1990). Both algorithms have $O(n^3)$ time complexity, where n is the number of nodes. In the testing we have done, the Bellman-Ford algorithm provided a substantial performance increase over the Floyd-Warshall algorithm, especially when the corresponding graphs are fairly sparse.

As mentioned above, the need to specify nonconvex constraints between events arises frequently in practice. We are currently exploring the applicability of several graph-based decomposition techniques to bringing down the cost of searching for a feasible solution to problems in which there is a nontrivial amount of nonconvexity present after simple heuristics, e.g., path consistency, have been applied. Even these, techniques, however, do not guarantee efficient solution in large networks of constraints.

We are also exploring the performance of several competing heuristic approaches as they can be applied to temporal reasoning in mixed initiative planning. Our focus is on Tabu Search and Genetic Algorithms: these two techniques have received a great deal of attention recently, and have been identified repeatedly as holding great promise for application to practical applications. Both of these approaches to solving related combinatorial optimization problems heuristically (in particular scheduling problems) have already demonstrated promising preliminary results. The reader is referred to (Reeves 1993) for a detailed overview of these and some closely related techniques.

We are also exploring ways to provide a natural representation for defining coarse constraints and a mechanism for using them in temporal reasoning. In underconstrained problems, this feature would be used to select the most desirable answer by developing a preference among the available multiple solutions . Similarly, in overconstrained situations, the partial ordering induced by this feature will be used to "relax" one or more soft constraints and find an acceptable solution to such restrictive problems. We foresee that the use of this system will yield better solutions, reflecting the user's preference criteria, will improve the generation of solutions to difficult non-convex problems, and will produce intuitive (and therefore explainable) relaxation methods to obtain reasonable solutions in overconstrained problems. A particular benefit of this approach is the exploitation of the decision maker indifference or his/her lack of predictive knowledge.

Implementation and Interface(s)

Tachyon is implemented in C^{++} on a Sun workstation under UNIX, with graphics using the X-Windows-based C^{++} class library InterViews. It has been implemented so that it can be used in a number of ways. In some applications, such as our abductive reasoning system *Patti*$^{++}$, where it was used to validate temporal sequencing of events as an aid in formulating plan hypotheses, and in a recent integration experiment with SRI's SOCAP system, it is most convenient to use Tachyon as a "batch" process, in which I/O is performed using inter-process communication or text files. In many others, direct interaction with users necessitates a graphical user interface (GUI). In still others, Tachyon acts in a hybrid mode, in which it is a "server" to another application, and can switch between a GUI-based interaction and embedded service to the client application. This has been facilitated considerably by our recent addition of automatic layout capabilities to Tachyon. Finally, we have integrated data structures that provide an efficient query interface.

A picture of the Tachyon interface with a simple network is shown in Figure 1.

This graphical editor allows loading and saving of temporal data files that are compatible with the batch mode version. Thus, one could use Tachyon to test consistency of a network, fine tune it, or test it in *"What if..."* scenarios as desired, then save the

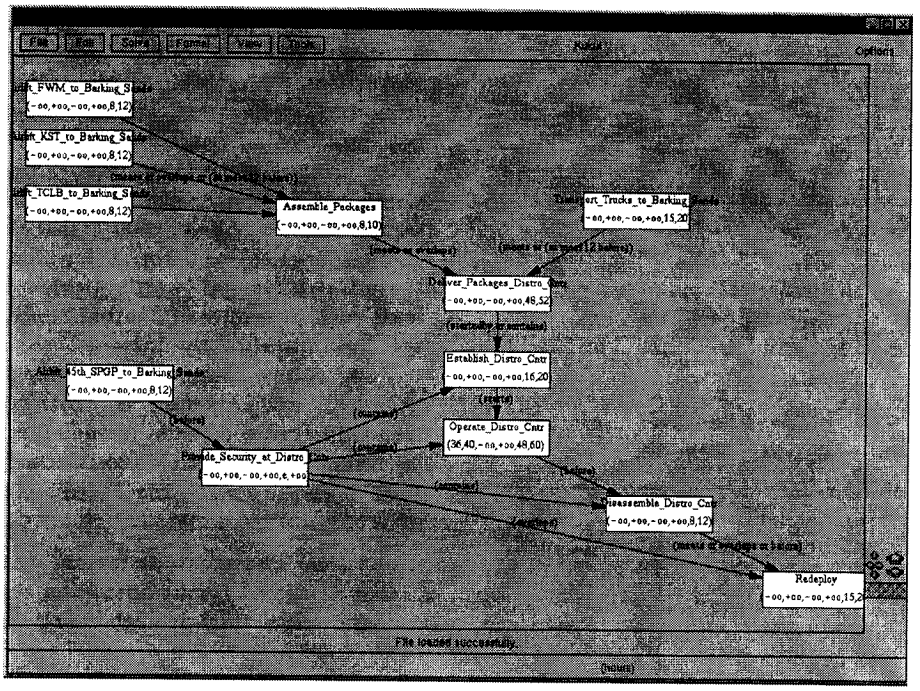

Figure 1: The Tachyon temporal constraint network editor, shown with a simple planning example.

network in a file the batch-mode system can use. The graphical interface itself is a CAD-like direct-manipulation editor for the graphical representation of the underlying network.

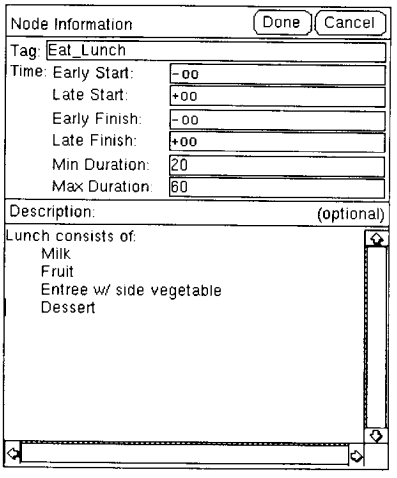

Figure 2: Node Information menu

Nodes and edges are tied to dialog boxes that allow modification and entry of the associated values in the data base. Examples of these are shown in Figures 2 and 3. A scheme that enforces convexity on the constraint is used when selecting Allen relations. The canvas on which the graph appears can be panned and sized as needed. Nodes and edges can be added, moved, deleted, etc. as desired.

Tachyon also provides a timeline view of events, somewhat akin to a Gantt chart. An example of this is shown in Figure 4. This view is complicated by the fact that there is much that may be unknown about when a given event may be occurring. Each large rectangle corresponds to an event's 6-tuple, and is appropriately positioned on the time line. Multiple events are arranged vertically. The length of an event's rectangle corresponds to the possible length of the event. The left side corresponds to the earliest possible start time, while the right side corresponds to the latest possible finish time. The interval of uncertainty for the start time is represented by a green band in the top left corner. Similarly, a red band in the bottom right corner indicates the interval of potential finish times. The length of the two black bands across the center of the block show the minimum and maximum durations for the event. Infinities in all cases are drawn to the left and/or right side of the screen, as appropriate.

Support for Mixed-Initiative Planning

Specifying and monitoring the interaction of events is a critical aspect of military planning and execution, and affects these functions in several ways. For example, once primary objectives are chosen, the sequence of activities that must be undertaken to achieve these objectives is often complex and full of interdependencies, e.g., if one goal is met 2 days late, what is the effect on the rest of the mission? Another area with which we have had recent experience involves the phasing of

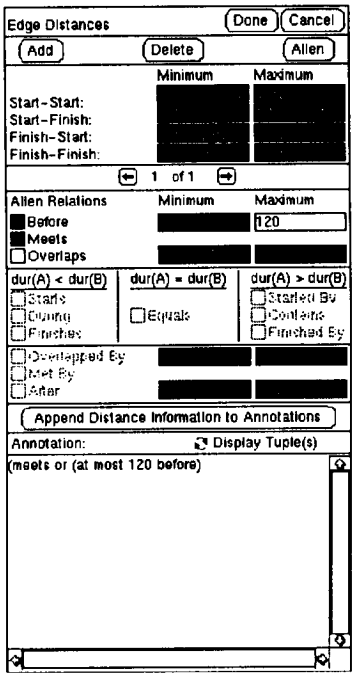

Figure 3: Edge Information menu

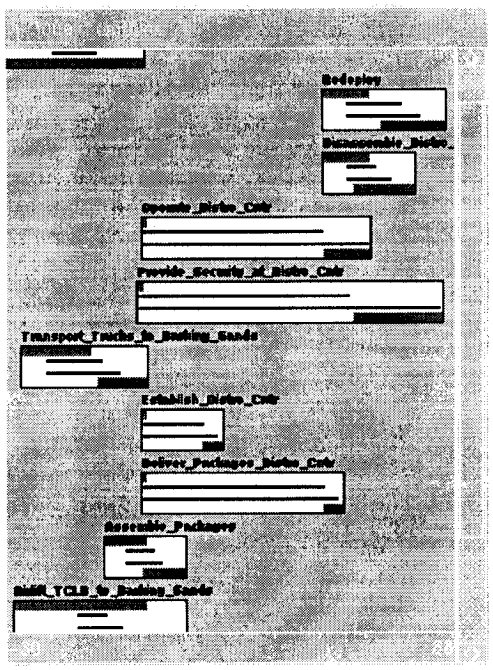

Figure 4: A time line view of a simple example.

forces to meet support and sustainment requirements: in current practice temporal constraints both between and within forces are considered in initial deployment schedule development, yet these important aspects are not captured in current systems. These systems are quite limited in their ability to represent and reason about the temporal aspects of a plan. When modification of a plan is necessary, there is no way (short of user experience) to determine which changes introduce temporal infeasibility, manifested by logistical problems such as failure of combat support to arrive within the time specified by doctrine.

Fast, robust temporal reasoning systems with the power of expressiveness to capture the complex relationships that may exist in such domains could provide significant improvements to the current mode of operation. It is our goal to use Tachyon to provide a number of such improvements over current systems, including:

Capture: The ability to formally capture and later refine temporal constraints even early in the planning cycle, when they are inherently inexact and subject to change,

Development Course of action (COA) development and adaptation,

Analysis Temporal feasibility analysis in mixed-initiative systems, throughout the planning and execution phases,

Communication: Tools for communicating such constraints throughout the planning cycle and throughout the levels of command,

Versatility and Reuse: Abstract systems of temporal constraints can capture a great deal of information that can be used as templates for building specific instances.

Thus, we envision Tachyon supporting the user as he builds a plan (either manually or in a mixed-initiative mode with plan development tools), going through a process of successive refinement, starting from a fairly rough set of requirements. Even at the earliest stage, there are temporal dependencies that must be observed, and carried through the levels of planning and replanning. We plan to use Tachyon to capture these (some may be built from doctrinal examples), then to track and refine them throughout the planning process, notifying the user when things become temporally infeasible, and provide him with tools for recovering from points of infeasibility.

Tachyon may also be used within an appropriate planning tool for refinement of tasking orders based on adversary doctrine. The basic idea is that if one has doctrinal information about the adversary in the form of temporal models, together with partial information about what the adversary has been doing, one can use Tachyon to help determine when they might engage in the next action expected based on adversary doctrine. This can be used to conclude when to respond, but to

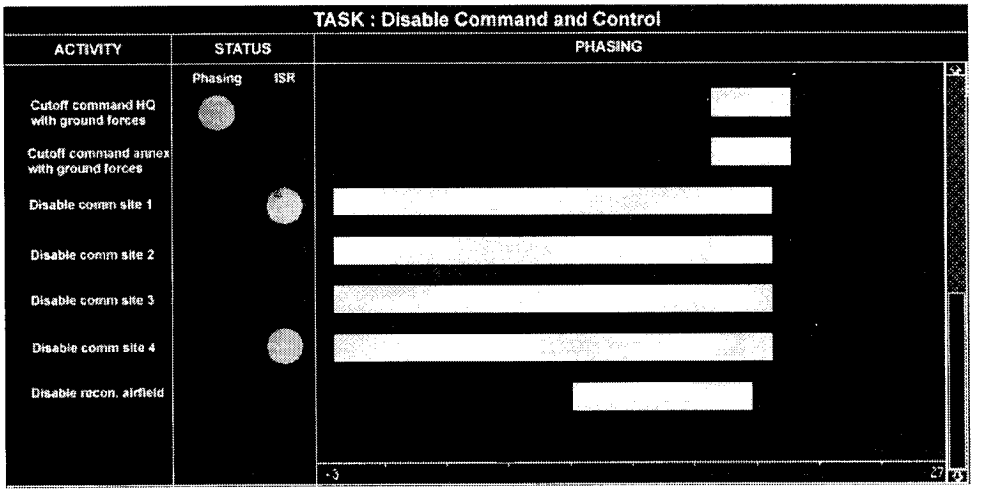

Figure 5: A prototype graphical view of temporal information maintained in Tachyon as it might be presented to a human planner responsible for monitoring feasibility of plans along multiple dimensions, and reacting when necessary, perhaps by initiating a dialogue with Tachyon or a full planning system.

date we've mainly used it to generate tasking requests for surveillance.

Benefits, Challenges

It has been our experience (gained via informal user interaction and workshops) that domain experts and operational users recognize the great potential benefits of such capabilities, including:

- Lower plan development time and costs,
- Better, more robust plans,
- Greater ability to perform multiple contingency evaluation,
- Ease of use, providing a shorter learning curve

Both technical challenges and "cultural" obstacles exist. Since current systems cannot exploit the complex temporal inter-relationships that exist among plan elements, there are no mechanisms in place for capturing these formally. Additionally, it is clear that if a proposed system places a new burden at any point in the process it will not be well-received by users. Finally, a proposed system must be integrated easily into either existing or emerging plan development environments. We have undertaken several enhancements to Tachyon to facilitate this transition, and discuss these below.

Extensions to Tachyon

We have enhanced Tachyon recently in several ways to facilitate its use in large mixed initiative planning systems. Of particular interest are ways of dealing with the human aspect of complexity of large systems of constraints and with identifying and recovering from inconsistencies.

Hierarchy We have recently incorporated into Tachyon the ability to represent temporal constraint networks hierarchically. Hierarchical representation provides users with the ability to manage large systems of constraints in a modular way, concentrating on the appropriate level of detail for the task at hand. Additionally, this enables modular editing of networks, thus allowing one to postpone instantiation of a hierarchical node representing a subsystem until more detailed information is available. For example, a hierarchical node representing the configuration and response capabilities of one firefighting company can be swapped for another, which might have fewer vehicles with which to respond, but a quicker response time.

Adding this capability to Tachyon has had an impact on both the underlying conceptual model and the graphic interface. In our approach, we allow the user to create hierarchical representations dynamically to reflect a particular view of the data. In a scheduling domain, for example, one might want to create one view based on resources and a completely different view based on various stages of execution or on product, all while maintaining the underlying semantics and constraints. From a graphical point of view, the network is presented as a graph, although each node may now represent a complex constraint network, and each edge may represent the fact that some constraint exists between the subgraphs represented by the joined nodes. It is worth noting that at this point hierarchical currently have only annotational value, to visually cue the presence of an underlying constraint between the graph outside the Hierarchical node and some node within. Multiple constraints between leaf nodes (nodes with no children) may be represented by a single hierarchical edge.

A user may choose to expand or collapse a hierar-

chical substructure at any time. Graphically, this will replace the hierarchical node with the subnetwork it represents (or vice versa). In many cases, expansion of a hierarchical node necessitates laying out the graph again.

Propagation of constraints in hierarchical networks differs modestly from the non-hierarchical case. One must decide how to summarize in a hierarchical node the constraints implied by the contained subnetwork. Because of the level of flexibility we have provided to users, constraint propagation operates on the expanded graph, effectively ignoring hierarchical structure. (We have experimented with expressive restrictions that permit hierarchical propagation, and may include these in later versions of the software.) Some issues also arise regarding summarization of diagnosis of inconsistency.

Inconsistency Diagnosis Despite one's best intentions, things don't always proceed smoothly, according to plan. In some cases, a drastic change of course is required when things go wrong, but in many cases, temporal inconsistency introduced by deviations from plan can be overcome though relaxation of constraints or by substituting resources. This need may arise when the originally stated constraints become so tight that they can no longer be met, or when a newly introduced constraint is logically inconsistent with others. Tracking these down can be difficult and frustrating, even for the experienced user, and might be enough to keep novice users (who are even more likely to make logical mistakes) from using the system.

We have recently added two capabilities to Tachyon to facilitate diagnosis. The first is rather simple, and involves allowing the user to select any subset of the network and propagate the constraints local to that subset, ignoring all others. Despite its simplicity, this technique is sometimes quite effective at tracking down inconsistencies. The second technique involves finding minimal inconsistent cycles in the underlying temporal constraint system, and reporting this information to the user for action.

The particular approach used by Tachyon to detect negative cycles in the graph is derived from an algorithm proposed by (Ishima, Tsukiyama, & Shinoda 1990). The algorithm begins with an acyclic subgraph, then iterates through the remaining weighted edges, checking to see if its addition to the sub-graph produces a negative cycle. If so, then the nodes and edges in the path are noted as an inconsistency, and the algorithm continues checking the remainder of the edges. This approach to detecting inconsistencies has several advantages over others for this application:

- Since we begin with an acyclic graph and iteratively add only those edges which do not cause negative cycles, we are ensured of finding only true negative cycles.
- In the best case, the algorithm finds all negative cycles, and in the worst case, if all negative cycles share a common edge, only one of them will be detected.
- The algorithm can be interrupted at any point in the iterative process, allowing the user to address a problem detected early in the process without waiting for a complete analysis of the network.
- The inconsistency is detected within a low-level representation of the network, providing useful information about the root cause of the inconsistency, and the potential for automated diagnosis.

Future Directions, Conclusions

We have described a model for temporal reasoning and its implementation. In addition to the applications mentioned above, we are currently experimenting with use of Tachyon in a number of application areas, including project planning, equipment delivery/deployment, other applications to scheduling in manufacturing, and temporal consistency checks for knowledge bases.

Tachyon is a prototype tool, and as such we are constantly modifying it. Some of the enhancements that are either planned or in progress include:

- an agent-based architecture, together with a supporting expanded query capability,
- expanded resource handling capability,
- more sophisticated diagnostic support, including improved visualization of inconsistencies, within both graphical and timeline views of the network, as well as (eventually) automated diagnosis and explanation of inconsistencies,
- constraining hierarchical edges,
- new heuristic propagation algorithms, to handle relaxation of constraints when exact algorithms are impractical,
- preference specification and propagation, i.e., the ability to specify a preference over the feasible interval for events.

Tachyon's distribution is restricted, and subject to government approval. Those interested in acquiring a copy of Tachyon may contact the authors.

References

Allen, J.F., 1981. "An Interval-Based Representation of Temporal Knowledge," Proceedings of IJCAI-7, pp. 741-747.

Allen, J.F., 1983. "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, **26.11**, pp. 832-843.

Arthur, R., and Stillman, J., 1992. "Tachyon: A model and environment for temporal reasoning," GE CRD Technical Report.

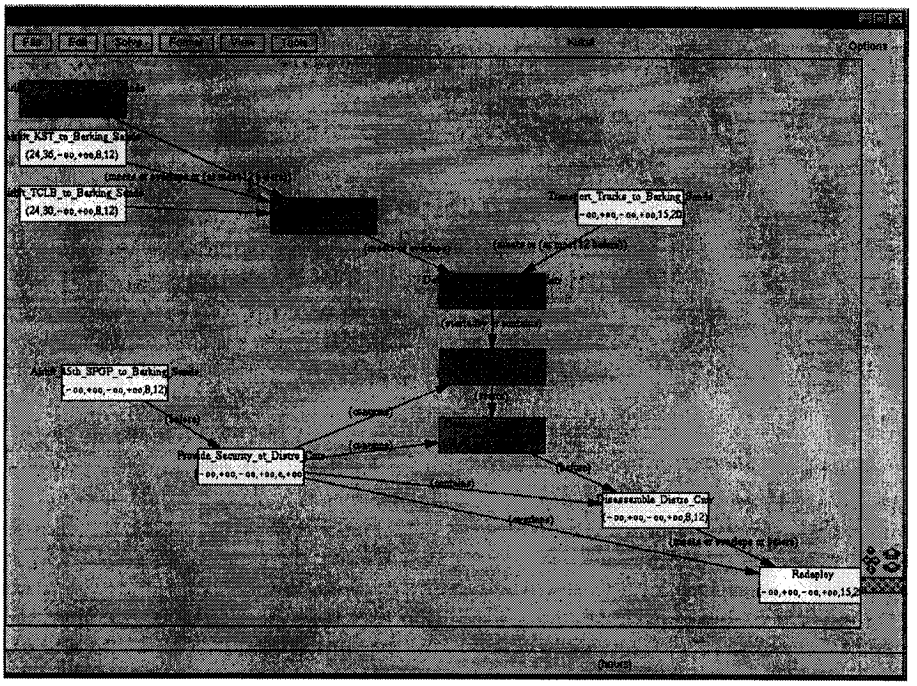

Figure 6: Tachyon tracks down points of inconsistency, and presents them to the user for action.

Arthur, R., Deitsch, A., and Stillman, J., 1993. "Tachyon: A Constraint-based Temporal Reasoning Model and its Implementation," SIGART Bulletin, 4:3.

Cormen, T.H., Leiserson, C.E., and Rivest, R.L., 1990. *Introduction to Algorithms*, MIT Press.

Dean, T.L., and McDermott, D.V., 1987. "Temporal data base management," *Artificial Intelligence*, **32**, pp. 1-55.

Dechter, R., Meiri, I. and Pearl, J., 1991. "Temporal Constraint Networks," *Artificial Intelligence*, **49**, pp. 61-95.

Ishima, K., Tsukiyama, S., and Shinoda, S., 1990. "An Algorithm to Detect Positive Cycles in a Constraint Graph for Layout Compaction," in Proc. IEEE Transactions on Circuits and Systems.

Montanari, U., 1974. "Networks of Constraints: Fundamental Properties and Applications to Picture Processing," *Information Sciences*, **7**, pp. 95-132.

Reeves, C., 1993. *Modern Heuristic Techniques for Combinatorial Problems*, Wiley, New York.

Rit, J.-F., 1986. "Propagating temporal constraints for scheduling," *Proceedings of AAAI*, pp. 383-388.

Stillman, J., Arthur, R., and Deitsch, A., 1994. "Temporal Reasoning with Tachyon: Model, Implementation, and Experience," GE Corporate Research & Development Center Technical Report, available from the authors.

Stillman, J., et al., 1993. "Spatio-temporal Data Management," in Proceedings of the Symposium on Advanced Information Processing and Analysis, 2-4 March 1993.

Stillman, J., 1993. "An Approach to Spatio-Temporal Retrieval and Reasoning," in Proceedings of GIS '93, Washington, DC.

Valdés-Pérez, R.E., 1986. "Spatio-temporal reasoning and linear inequalities," MIT Artificial Intelligence Laboratory Memo 875.

Van Beek, P., and Cohen, R., 1990. "Exact and approximate reasoning about temporal relations," *Computational Intelligence*, **6.3**, pp. 132-144

Van Benthem, 1983. *The Logic of Time*, Reidel.

Vilain, M.B., 1982. "A System for Reasoning About Time," *Proceedings of AAAI*, pp. 197-201.