Constraint Maintenance with Preferences and Underlying Flexible Solution

Contact Author: Paul Morris
Computational Sciences Division
NASA Ames Research Center M/S 269-1
Moffett Field, CA 94035, U.S.A.
pmorris@email.arc.nasa.gov

Ari Jonsson
Research Institute For Advanced Computer Science
NASA Ames Research Center
Moffett Field, CA 94035, U.S.A.
jonsson@email.arc.nasa.gov

John Bresina
Kanna Rajan
Computational Sciences Division
NASA Ames Research Center
Moffett Field, CA 94035, U.S.A.
{bresina|kanna}@email.arc.nasa.gov

SUBMITTED TO CP-2003

Constraint Maintenance with Preferences and Underlying Flexible Solution

John Bresina[2] Ari Jonsson[1] Paul Morris[2] Kanna Rajan[2]
1. Research Institute For Advanced Computer Science
2. Computational Sciences Division

NASA Ames Research Center, MS 269-1
Moffett Field, CA 94035, U.S.A.

Abstract. This paper describes an aspect of the constraint reasoning mechanism that is part of a ground planning system slated to be used for the Mars Exploration Rovers mission, where two rovers are scheduled to land on Mars in 2004.
The planning system combines manual planning software from JPL with an automatic planning/scheduling system from NASA Ames Research Center, and is designed to be used in a mixed-initiative mode. Among other things, this means that after a plan has been produced, the human operator can perform extensive modifications under the supervision of the automated system. For each modification to an activity, the automated system must adjust other activities as needed to ensure that constraints continue to be satisfied. Thus, the system must accommodate change in an interactive setting.
Performance is of critical importance for interactive use. This is achieved by maintaining an underlying flexible solution to the temporal constraints, while the system presents a fixed schedule to the user. Adjustments are then a matter of constraint propagation rather than completely re-solving the problem. However, this begs the important question of which fixed schedule (among the ones sanctioned by the underlying flexible solution) should be presented to the user. Our approach uses "least-change" and other preferences as a prism through which the user views the flexible solution.

1 Introduction

The Mars Exploration Rovers Mars 03 mission is one of NASA's most ambitious science missions to date. The rovers were launched in the summer of 2003 with each rover carrying instruments to conduct remote and in-situ observations to elucidate the planet's past climate, water activity, and habitability.

Science is the primary driver of MER and, as a consequence, making best use of the scientific instruments, within the available resources, is a crucial aspect of the mission. To address this critical need, the MER project has selected MAPGEN (Mixed-Initiative Activity Plan GENerator) as an activity planning tool.

MAPGEN combines two existing systems, each with a strong heritage: APGEN [6], the Activity Planning tool from the Jet Propulsion Laboratory and the Europa Planning/Scheduling system [4, 7] from NASA Ames Research Center. The Mixed-Initiative aspect means that after an initial plan has been produced, it undergoes a period of “tweaking” by the human operator. Thus, the system must accommodate change, and must do so rapidly enough for interactive use. As we will see, this is achieved by exploiting an underlying flexible solution in Europa so that fast temporal propagation methods can be used.

Flexible time means that instead of finding a single solution, the Planner preserves maximum temporal flexibility by maintaining a set of solutions that satisfy the constraints. This is represented internally as a Simple Temporal Network [3] (STN). As a result of propagation in the STN, each activity acquires a refined time window for its start time.

Among the advantages of preserving a flexible set of solutions is that the Planner may adapt to additional constraints by exploiting the flexibility, rather than completely re-solving the problem. However, this has to be reconciled with APGEN, which expects to see a fixed time schedule. Note that the presentation of temporal flexibility to a plan GUI would pose significant problems: it is difficult to provide a visual representation of flexible windows, as well as binary temporal relations such as BEFORE and AFTER. In our case, many tools associated with APGEN, such as those that do calculations of resource usage, require a fixed schedule of activities.

The approach we take is to present a single solution to the user in the APGEN GUI, while the Planner maintains the flexible set of solutions as a backup. This creates a problem of determining which fixed schedule to present to the user.

2 Earliest Time Solution

The theory of Simple Temporal Networks guarantees that a solution is obtained by assigning to each event the earliest time in its time window. This seems like an obvious candidate for the solution to be presented to APGEN. However, this creates certain undesirable artifacts.

Consider for example an activity that has a flexible start time and flexible duration, but the end time is fixed by a constraint. The earliest time solution will cause the duration to be stretched to its maximum extent.

In our application, the durations are generally not flexible. Nevertheless, more subtle forms of this problem can occur, as indicated by the following example. The most critical time for the solar-powered Mars Rover in terms of energy is often in the early morning period before the batteries have fully recharged. The CPU is a primary user of energy, and it is required to be on to enable most of the activities. Thus, one way of economizing on energy use is to have greater overlap between CPU-using activities. However, if two overlapping activities are such that the later one is “anchored” in time, while the earlier one is flexible, then they will tend to be pulled apart by the earliest time solution, thus increasing the energy demand.

In the above example, the requirement that fixes the time of an activity is an explicit temporal constraint, and the overlap restriction is a derived preference that may arise from resource limitations. While the earliest time solution may be acceptable as a general default, interactions between constraints and preferences may require departures from this to satisfy specific user needs.

3 Constraints and Preferences

In this application, there is a variety of constraints and preferences that arise from engineering restrictions and scientific need, many of which may not be recognised until specific circumstances arise in operation.

The explicit temporal constraints fall into three categories: *model* constraints, *daily* constraints, and *expedient* constraints. The model constraints encompass definitional constraints and some flight rules. For example, the decomposition of activities into sub-activities specifies temporal relations between the parent and its children. Some activities might be restricted to the day or the night. The daily constraints comprise "on the fly" temporal relations between elements of scientific observations, depending on what scientific hypotheses are being investigated. For example, an image may be taken before using a specific instrument in some circumstances, but not in others. The expedient constraints are those imposed by the Europa planner to guarantee compliance with some higher level constraint that cannot be directly expressed in an STN. For example, a flight rule might specify that two activities are mutually exclusive (such as taking a picture while the rover is moving). This is really a disjunctive constraint, but the planner will satisfy it by placing the activities in some arbitrary order. This has important implications for the tweaking process.

There are also preferences that arise from varied sources. Some are based on engineering or scientific considerations such as desiring calibrations to be close to measurements, or wanting separate observations to occur in similar lighting conditions. Perhaps most are derived from the need to solve problems related to resources. In general, the tweaking process is driven by a desire to fit as much "science" as possible into the plan, while steering it on a course that avoids running aground on competing resource limitations.

The resource calculations are complex. For example, they may involve thermal modelling performed by legacy software. There are often complex options available for reducing usage. For example, in imaging, there are several ways of reducing data volume, such as reducing resolution, or using fewer filters. Choosing between these may require human-level scientific judgement. This means that many of the preferences have an ephemeral nature driven by short-term solutions to transient problems.

These considerations rule out formal modelling of these preferences and dictate the need for a process of informal tweaking by a human operator. The preferences are implicit in the modifications made during this period. However, the modifications interact with the hard constraints discussed above. The automated system must prevent these from being violated. Within this framework, a policy of minimal change provides a reasonable approach for respecting the implicit preferences.

A dramatic illustration of the need for the minimal change occurs when switching from a native APGEN mode, where users are free to modify activities at will, unimpeded by constraints, to the mode where constraints are enforced. To satisfy constraints, some activities must be moved, but arbitrary reorganization of the plan is undesirable.

4 Accommodating Change

Assume that a plan has been produced, and no preferences have yet been expressed to modify the solution. Then the initial solution presented is the earliest time one discussed earlier. During the subsequent tweaking phase, MAPGEN provides a GUI feature, called *constrained move*, that allows dragging an activity to a new location. When the mouse button is released, other activities are also moved to maintain the integrity of the constraints.

This raises an issue with respect to the expedient constraints. Since these arise from disjunctive constraints that could be satisfied by different arbitrary choices, a mode is provided in which the expedient constraints are relaxed. When this relaxed mode is exited, there is a need to re-establish constraints in a way that minimizes the disturbance to the existing plan. A similar need arises when passing from the native APGEN mode to the constraint-maintenance mode. Also, the input files presented to MAPGEN are implicitly in the APGEN mode, and require a similar assimilation to the constraint-maintenance mode.

In this section, we describe the algorithm that is used to modify the solution presented to APGEN by the Europa system. In this interactive application, efficiency considerations seem to rule out the seeking of true optimality (even the tractable kind discussed in [5]). Instead, we have adopted a greedy algorithm that locally minimizes the amount of change from the existing positions of activities.

It is convenient to use a special set of unary singleton constraints to store the current positions of the start and end times of activities. Then the algorithm can be outlined as shown in figure 1.

```
1. Save all the current positions in a temporary list.
2. Remove all the current position constraints and repropagate.
3. For each saved position t of timepoint x  do
      if t is within the STN bounds for x then
         add a position constraint setting x to t
      else if t < the lower bound for x then
         add a position constraint setting x to the lower bound
      else if t > the upper bound for x then
         add a position constraint setting x to the upper bound
      Propagate the effect of the new constraint;
```

Fig. 1. Constrained Update Algorithm

We see that each step that reinstalls a position constraint tries to minimize the departure from the previous position while maintaining consistency. However, the greedy nature of the algorithm means that the order in which activities are considered may affect the outcome. For example, suppose that activity A is constrained to end before activity B starts. If an APGEN file is loaded where activity A is initially simultaneous with B, then one of A or B must be moved. Which of these occurs will depend upon the order in which A and B are considered for the position update in step 3.

There are certain situations in which the user needs to ensure that a particular activity prevails in the update lottery. For example, after a constrained move, clearly the activity that is moved should be held to its new position. This is easily done by considering it first. (The new position is guaranteed to be within the STN bounds because a visual indication of these bounds is given during the move, and attempts to move the activity outside that range are ineffective.)

For more general situations, a *pinning* mechanism is provided that allows the user to lock specified activities at their current positions. This is achieved by applying additional constraints. There is a visual indication of which activities are pinned, and they can be unpinned on request.

Note that step 1 of the algorithm requires a repropagation of the network after deletions. In general, propagations of an STN after deletions are more costly than the simple incremental propagations that occur after additions. However, in the constrained-move case, the deletions are from a consistent state, so the $O(N \log N)$ Dijkstra algorithm can be used rather than the more costly $O(EN)$ Bellman-Ford [2] (where N is the number of nodes and E the number of edges in the STN). With typical networks of 1000 nodes, the propagation appears instantaneous to the interactive user. (If greater speed were needed, then more elaborate techniques for incrementality after deletions [1] are available in the literature.)

When switching from relaxed mode to strict mode (see above), it is possible for the current set of constraints (including pins) to be inconsistent. If inconsistency is encountered during the greedy update, the system removes the most recent implicated activity from the plan, and places it in a temporary storage area called the *hopper*, where it can be inspected by the operator, and possibly reinserted into the plan after further modifications.

References

1. R. Cervoni, A. Cesta, and A. Oddi. Managing dynamic temporal constraint networks, in artificial intelligence planning systems. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94)*, 1994.
2. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT press, Cambridge, MA, 1990.
3. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, May 1991.
4. Ari K. Jonsson, Paul H. Morris, Nicola Muscettola, Kanna Rajan, and Benjamin D. Smith. Planning in interplanetary space: Theory and practice. In *Artificial Intelligence Planning Systems*, 2000.
5. L. Khatib, P. Morris, R. Morris, and B. Venable. Tractable pareto optimization of temporal preferences. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acupulco, Mexico, 2003. Morgan Kaufmann, San Francisco, CA.
6. Maldague, Ko, Page, and Starbird. Apgen: A multi-mission semi-automated planning tool. In *Proceedings of the 1st International Workshop on Planning and Scheduling for Space*, Oxnard, California, 1997.
7. N. Muscettola, P.P. Nayak, B. Pell, and B.C. Williams. Remote agent: to boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–48, August 1998.