Uncertain Reasoning for Dynamic Constraint Problems

Ken Brown and David Fowler

Department of Computing Science, University of Aberdeen, UK

Standard constraint-based techniques and tools provide little support for *dynamic* problems. Many real-world problems are dynamic, and change after we have begun executing solutions. For example, scheduling maintenance and repair operations, assigning deliveries to individual vehicles, or assigning incoming planes to arrival gates are all "on-line" problems - initial decisions must be made before the problem is completely specified, and new tasks keep arriving as the solution is being executed. Such problems may be tackled in three ways: (i) rely on a quick respecification and generation of a new solution when the change occurs; (ii) maintain extra information during the solution process to make the repair process easier; or (iii) take possible future changes into account when generating the initial solutions. There are three main criteria for judging different methods: *efficiency*, *optimality* and *robustness*. Efficient methods allow new solutions to be generated promptly after the changes occur - clearly, we need to find a new solution before it is too late to execute it. Optimal methods generate the best possible solution at each stage. Robust methods generate solutions that will require minimal modification when the change occurs. Robust solutions will tend to reduce the work to be carried out at each change, but may also increase user satisfaction - for example, a workforce may prefer to know a day's tasks in advance, rather than have their schedules disrupted every time a new job is required. These three criteria generally conflict, and so some compromises must be made.

Our research is concerned with dynamic resource allocation problems, where new tasks arrive as the solutions are being executed. We assume that we have knowledge of what these new tasks may be, in the form of a probabilistic tree of arrivals. We model this as the addition of variables to the problem. We are investigating methods of extending the CSP formalism to include such models of future changes, and extending the search methods to reason about likely changes. We aim to produce contingent solutions, specifying decisions for all likely developments of the problem. Further, our early decisions should be robust, in that they will lead to high quality final solutions regardless of how the problem develops. The main principle we are trying to implement is very simple: *if we expect to be given important tasks, then leave enough resources free to be able to complete them.*

Each node in our tree represents the possible arrival of a variable. Each of the possible paths through the tree from root to leaf thus specifies a possible sequence of arrivals. The same variable can appear in multiple nodes in the tree (but not twice in any one path). We aim to assign values to each node in the tree so that no constraint is violated over a path. A variable can be given different

values if it appears in different paths. Such an assignment is a contingent plan specifying what to do when each variable arrives. If it is not possible to construct such an assignment, then we search for sub-optimal solutions. We allow some variables to be rejected, and we associate a utility with each variable, which we gain if we assign it a value. If a variable is not assigned a value in a path, then we ignore any constraints on that variable in that path. Our aim is now to maximise the expected utility of the assignments, while satisfying the active constraints. Thus our solutions are robust - the initial decisions take into account all the likely future developments, and the decision that maximises utility over all of them is chosen. So far, we have defined the formalism, and developed a forward checking branch-and-bound algorithm, which finds the optimal solution [1]. We have shown that the corresponding formal decision problem is NP-complete [2]. We have also developed an incomplete algorithm, which sacrifices the guarantee of optimality to find good solutions quickly, by ignoring sub-trees with a low maximum expected utility [1] and extendeded it to an anytime algorithm [2]. The probabilistic tree looks like a finite-horizon Markov Decision Problem. We have shown that our complete method produces a solution faster than generating and solving the corresponding MDP [3].

Our system is not yet a practical tool, however. The most obvious deficiency is that we have only dealt with arrival sequences, and not arrival times. Uncertainty in the arrival times could be converted automatically into a tree, where each level is a time step. This will require temporal constraints, and temporal domains - the domain for a variable will then be dependent on its arrival time. Temporal reasoning will allow us to improve our anytime algorithms, and will also allow us to improve the robustness, by expressing constraints on the use of resources at particular times - for example, we would like to be able to state that any job to be carried out will be fixed 4 hours before the work starts. We are considering representations other than trees, to allow more sophisticated descriptions of future events. We are also investigating using directed cyclic graphs instead of trees, allowing arbitrarily long or even infinite horizons. These graphs will have the same relationship to general MDPs as our trees have to finite horizon MDPs. Finally, we would like to incorporate more of the known CSP techniques, including enhanced consistency maintenance during search, and more general techniques for soft constraints, optimisation and uncertainty handling.

References

1. Fowler, D. W. and Brown, K. N. "Branching constraint satisfaction problems for solutions robust under likely changes", *Proc CP2000*, 500–504, 2000.
2. Fowler, D. W. *Branching Constraint Satisfaction Problems*, PhD thesis in preparation, Department of Computing Science, University of Aberdeen, 2001.
3. Fowler, D. W. and Brown, K. N. "Branching constraint satisfaction problems and Markov Decision Problems compared" *Proc CP-AI-OR 2001, Wye College, UK*, pp81–96, 2001.