A Transparent Interface to State-Space Search Programs

Steven L. Tanimoto*
University of Washington

Stefano Levialdi†
University of Rome "La Sapienza"

Abstract

We present a visual interface for a variety of programs that employ state-space search methods. The interface displays a tree of the states visited so far, and it permits user interaction through manipulation of the tree. The interface is implemented in a domain-independent Python module. We give illustrations of its application in three domains: classical problem solving, image processing, and composition of musical motifs. The interface exemplifies a class of techniques for bringing a form of transparency to computer systems.

CR Categories: H.5.2 [User Interfaces]: Interaction Styles—Theory and Methods, Screen Design; J.6 [Computer-Aided Engineering]: Computer-Aided Design—Collaborative Design.

Keywords: transparent interface, state-space search, tree layout, interactive search, computer-aided design, collaborative problem solving

1 Introduction

As software systems become more complex, the challenge of making systems understandable to users becomes harder. One approach to meeting the challenge is to "open up" more of the inner workings of the system to the user purview. Computer systems that perform "intelligent" processing are a case in point, where problem complexity or the vastness of the space of candidate solutions can confuse users.

We present a visual interface to one class of intelligent computation methods – state space search. The interface is based on the display of the tree of states explored so far, starting with a given initial state. The interface serves as a starting point for the implementation of transparent software tools for a variety of domains. A *transparent* interface is one that reveals some of the inner workings of the system [Tanimoto 2004]. This project is motivated in part by ideas in H. Simon's *The Sciences of the Artificial* [Simon 1996].

2 The Transparent STate-space search ARchitecture (T-STAR)

In order to facilitate the design and implementation of transparent tools for problem solving and design in a variety of application domains, we have developed a software module called T-STAR using the Python programming language. T-STAR (which can also be written T*) stands for "Transparent STate-space search ARchitecture."

The module, **TStar.py**, implements a variety of domain-independent features including: representation of tree nodes and arcs, display of trees, interactive features for managing the use of screen space by the tree, organization of state representation, operators, and code that applies operators. It is written in the Python language, using the Tkinter GUI toolkit [Lundh 1999].

3 T-STAR in Classical Problem Solving

Our first illustration of T-STAR is in the traditional problem-solving domain of combinatorial search. The Missionaries and Cannibals puzzle has an initial state in which three missionaries and three cannibals are on the west bank of a river, along with a canoe. They must get across the river. However, there are constraints on how they may move. The canoe holds a maximum of three people, and it must always contain a missionary to guide it. The missionaries must never be outnumbered by cannibals at either the west bank, east bank or in the canoe, for if they are, they will be eaten[Russell and Norvig 2003]. The role of the T-STAR software is to manage the states explored by the user and to display, in a flexible fashion, those states. It also enforces the constraints. In the example, T-STAR does not automatically solve the puzzle but manages the moves made by a human user.

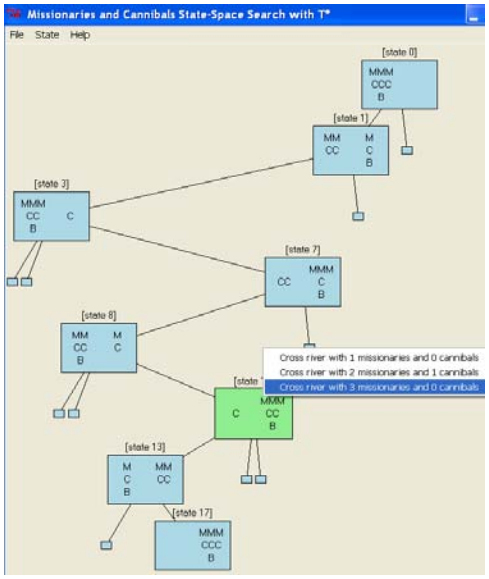

Figure 1: A display of the T-STAR interface for the Missionaries and Cannibals Puzzle.

*e-mail: tanimoto@cs.washington.edu
†e-mail: levialdi@di.uniroma1.it

SOFTVIS 2006, Brighton, United Kingdom, September 04–05, 2006.
© 2006 ACM 1-59593-464-2/06/0009 $5.00

4 Examples in Processing of Images and Music

Next we show an application of T-STAR to image processing. In figure 2, a screen shot is presented that portrays a program that extends T-STAR with operators for digital image processing. This program, called TRAIPSE, uses a digital image to represent each state, and any operator takes one state to another by making an image transformation.

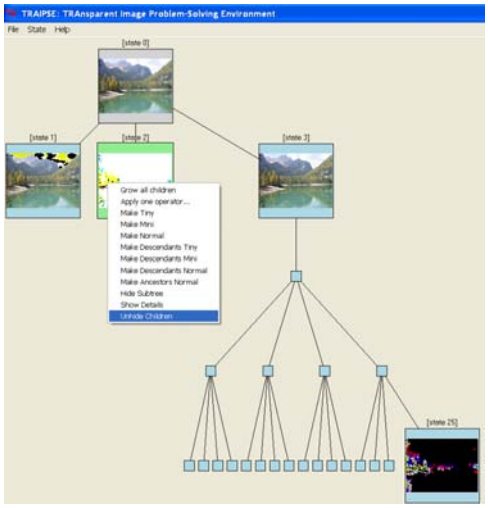

Figure 2: The TRAnsparent Image Problem Solving Environment based on T-STAR.

The domain of musical composition provides some counterpoint to the previous domains. In Figure 3, the T-STAR system has been extended with a means to display musical motifs as states, and a set of permutation operators has been incorporated in order to allow a computer (user) to make arbitrary sequences of transformations to the initial motif, which is defined in the example to be a diatonic scale. The program has also been given a means to play the musical motifs.

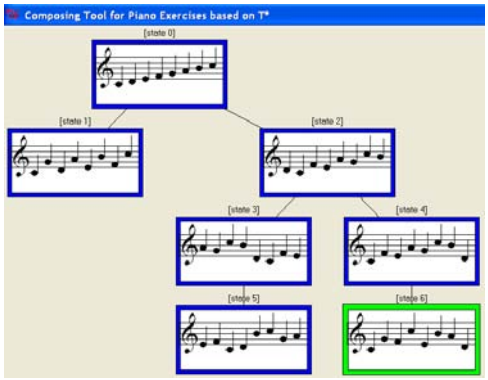

Figure 3: Application of T-STAR to the composition of musical motifs using permutation operators.

5 Issues

This project addresses a number of important issues having to do with software visualization.

The behavior of a state-space search program can involve visiting very large numbers of states. Facilities for managing screen real-estate are clearly important in situations like these. The ability to minimize and maximize nodes' displays is important, and it is helpful to have a variety of ways for expressing how this is to be done. There may be different size options for each node, and there may be different ways to specify sets of nodes that are to be adjusted. T-STAR allows not only 3 different levels of detail for node displays but also allows specifying sets of nodes in terms of ancestors and descendants of any selected node.

Another design issue for T-STAR is how to make it easy to customize T-STAR for different applications. Our approach has been to encapsulate generic functionality in methods of a class, so that the methods can be overridden in subclasses that are application-specific. The state display method is an example of such a method.

Some application domains require large numbers of operators. Operators may be parameterized, too. In the future, we plan to incorporate into T-STAR facilities for managing sets of operators and for managing their parameterization.

Next, there is the issue of incorporating automatic services such as the running of search algorithms, application of evaluation criteria, etc. T-STAR currently provides just a small amount of functionality of this nature – enough to show a straightforward way to incorporate more such functionality; a menu command is available to request the automatic generation of a subtree of depth 2 from the currently selected node.

Finally, we mention the issue of supporting collaboration in problem solving and design by teams of users. The T-STAR interface can serve to provide both common views and private views of portions of the state space. An extention of T-STAR called CO-STAR is currently under development to provide affordances for collaboration in design activities.

6 Acknowledgements

Thanks to P. Bottoni, L. Cinque, A. Malizia and W. Winn for constructive comments about the T-STAR software. Preparation of this extended abstract was funded in part by NSF Grant IIS-0537322.

References

LUNDH, F. 1999. *Introduction to Tkinter*.

RUSSELL, S., AND NORVIG, P. 2003. *Artificial Intelligence: A Modern Approach, 2nd ed.* Prentice Hall.

SIMON, H. 1996. *The Sciences of the Artificial, 3rd ed.* MIT Press.

TANIMOTO, S. 2004. Transparent interfaces: Model and methods. In *Proceedings of the Workshop on Invisible and Transparent Interfaces*.