# Automatic State Abstraction for Pathfinding in Real-Time Video Games

Nathan Sturtevant, Vadim Bulitko, and Michael Buro

Department of Computing Science, University of Alberta, Edmonton, Alberta, T6G 2E8, Canada
{nathanst|bulitko|mburo}@cs.ualberta.ca

**Abstract.** Real-time video games are a unique domain for pathfinding and search. Traditional approaches to search have usually assumed static worlds with a single agent. But, in real-time video games there are multiple cooperative and adversarial agents. While the search space in most games is relatively small, algorithms are expected to plan in mere milliseconds. Thus, techniques such as abstraction are needed to effectively reason and act in these worlds. We provide an overview of the research we have completed in this area, as well as areas of current and future work.

## 1 Introduction

The worlds of popular real-time video games, such as Warcraft III or Command and Conquer are much richer than many traditional domains used as test-beds for Artificial Intelligence research. The most basic task in these games is to get one or more units from their current location to some destination. These games are not designed to showcase algorithms from AI, but to be interesting for people to play. This means that standard approaches from AI and search are usually too slow or memory-intensive to be effective in practice. We believe that abstraction is the key technique for meeting the real-time and other constraints posed by these domains.

## 2 Previous Work

We first provide an overview of completed work on applying abstraction to search. First, we have developed a simple method for automatically building abstractions from an underlying octile map. For the purposes of this discussion, we consider any map to be a graph, where a node is a tile in the underlying map, and an edge means an agent can pass directly between two tiles without going through another tile first.

Given a graph representation of a map, we build an abstract version of the map by reducing connected components into abstract nodes. Instead of using the approach of [1] where nodes are abstracted along with their neighbors, we only abstract groups of nodes that are fully connected cliques. In practice, this means that at most four nodes are abstracted in any one step. When building an abstraction offline, we process the space in a uniform manner so that the abstract space more closely represents the space it abstracts. This process can also be applied in an online fashion by assuming unknown portions of the map are empty and applying local repair as an agent explores the map.

Given an abstract representation of a search space, there are many methods that leverage this abstraction for quick pathfinding. The approach of Botea et. al. [2] is to abstract large sectors in the original space. When doing refinement, local smoothing is applied to account for error introduced by large sector sizes. Smoothing, however, is only applied to complete paths. Because we are interested in dynamic worlds, we do not always want to compute complete paths. Instead, we build partial paths at each level of abstraction refining them locally as needed. This allows us to spread the computational cost of path following evenly across path execution. So, if we are interrupted, we reduce lost computation efforts. These methods are described in detail in [3].

Abstraction necessarily introduces error, so we would like to learn about errors and correct them over repeated pathfinding experiences. Thus, we have taken the abstract search that is applied at each level of our abstraction and replaced the A* search with a learning search [4]. This allows us to learn much better heuristics in abstract space.

## 3 Current and Future Work

We are currently working towards four goals with regard to abstraction and search. First, a variety of ideas have been suggested for building abstractions of search spaces, including reductions based on cliques, local neighborhoods, large sectors, and triangulation. We are attempting to incorporate a more flexible abstraction module into our simulation framework so that we can more precisely quantify the advantages and trade-offs of different methods.

Second, we would like to generalize existing work to a single algorithm that can parameterize methods for search, so we can better evaluate which parameters for search work best on which problems and abstraction methods.

Third, general work on learning better heuristics is computationally expensive, because we must keep a large table of heuristic information between every pair of nodes in our search space. Storing heuristics in abstract space reduces this cost somewhat, but we are also looking into ways to reduce this cost further by selectively storing learned heuristic information at each level of abstraction. To this end we are actively developing high-performance learning methods for real-time heuristic search [5].

Finally, we are building stochastic and dynamic environments in which there are multiple cooperative and competitive units that must interact, so that we can measure how existing techniques scale to environments typical of real-time video games.

## References

1. Holte, R., Perez, M., Zimmer, R., MacDonald, A.: Hierarchical A*: Searching abstraction hierarchies efficiently. In: AAAI/IAAI Vol. 1. (1996) 530–535
2. Botea, A., Müller, M., Schaeffer, J.: Near optimal hierarchical path–finding. J. of Game Develop. **1(1)** (2004) 7–28
3. Sturtevant, N., Buro, M.: Partial pathfinding using map abstraction and refinement. In: Under Review. (2005)
4. Bulitko, V., Sturtevant, N., Kazakevich, M.: Speeding up learning in real-time search via state abstraction. In: Under Review. (2005)
5. Bulitko, V., Lee, G.: Learning in real-time heuristic search: A unifying framework. Under Review (2005)