Target-Value Search Revisited (Extended Abstract)

Carlos Linares López
Planning and Learning Group
Universidad Carlos III de Madrid
28911 Leganés (Madrid) - Spain
carlos.linares@uc3m.es

Roni Stern
SEAS
Harvard University
Cambrige, MA 02138 USA
roni.stern@gmail.com

Ariel Felner
Information Systems Engineering
Ben Gurion University
Beer-Sheva, Israel 85104
felner@bgu.ac.il

Abstract

This paper addresses the *Target-Value Search* (TVS) problem, which is the problem of finding a path between two nodes in a graph whose cost is as close as possible to a given target value T. This problem has been previously addressed only for directed acyclic graphs. In this work we develop the theory required to solve this problem optimally for any type of graphs. We modify traditional heuristic search algorithms for this setting, and propose a novel bidirectional search algorithm that is specifically suited for TVS. The benefits of this bidirectional search algorithm are discussed both theoretically and experimentally on several domains. A longer version of this work was accepted to IJCAI-2013 (Linares López *et al.* 2013)

Introduction

Given a target value T, the *Target-Value Search* problem (TVS) consists of finding a path in a search space from a start state s to a goal state t, such that the cost of that path is as close as possible to T. The problem was recently introduced (Kuhn *et al.* 2008; Schmidt *et al.* 2009), motivated by several significant applications, such as planning a tour of a given duration in a park. Previous work has addressed TVS only in the context of directed acyclic graphs (DAGs). In addition, they performed only small-scale experiments, with graphs having at most 70 nodes.

We describe how to modify A* and IDA* to solve TVS problems, and propose a novel bidirectional search algorithm especially suited for TVS. Then, we describe how to define and use abstraction-based heuristics for TVS, introducing a domain independent abstraction for TVS called the Induced Transition Graph (ITG). Experimental results on a number of classical search benchmarks show the benefit of the proposed bidirectional search algorithm coupled with the ITG heuristic over the other approaches.

A* and IDA* for TVS

In A*, the node with the lowest f value is expanded in every iteration. This is well suited for finding the lowest cost path. In TVS, the goal is to find a solution of cost T or as close as possible to T. Therefore, our adaptation of A* to TVS, called TVSA*, considers $\Delta_T(n) = |g(n) - T|$ instead of $f(n)$ to guide the search. In every iteration the node with the lowest $\Delta_T(n)$ is expanded, considering only the nodes with $g(n) \leq T$. If no such node exists, TVSA* chooses the node with the lowest Δ_T among all. Let $best_\Delta$ be the lowest Δ_T found so far for path to t. The search halts when either a perfect solution is found (i.e, a solution of cost T) or a node with $g(n) \geq T + best_\Delta$ is chosen from OPEN. Unlike A*, in TVSA* a node is expanded even if a shorter path to it has already been found (i.e, no duplicate detection is performed). This is because in TVS different paths to the same node with different costs are meaningful. See (Linares López *et al.* 2013) for more details. We also proposed an IDA* variant for TVS called: TVSIDA*. TVSIDA* is very similar to TVSA*, using the same node evaluation function $\Delta_T(n) = |g(n) - T|$ but using an iterative deepening framework rather than a best-first search.

Bidirectional TVS (BTVS)

Consider the differences between A* and TVSA*. A* searches the *state space*, while TVSA* searches a space that we call the *path space*, where every node corresponds to a path from s. The state space is substantially smaller than the path space. Thus, A* will find a path to t faster than TVSA*. However, A* may not find the path to t with the smallest Δ_T while TVSA* is guaranteed to find this path. To enjoy the complementary benefits of a state space and path space searches we proposed a unique bidirectional search called Bidirectional TVS (BTVS).

BTVS is composed of two alternating searches: a *forward search* and a *backward search*. The forward search is a state space search from s to t, while the backward search is a path space search from t to s. Importantly, the backward search, which potentially searches a substantially larger space than the forward search, is only allowed to consider paths that are composed of nodes and edges found by previous forward searches. BTVS alternates between the forward and the backward search until the best Δ_T is found.

We implemented the forward search as a uniform-cost search. Duplicate detection is performed, and for every visited node n we store the lowest g-value found from s to n. When a path to t is found (i.e., when t is expanded) the backward search is called.

The backwards search consists of running TVSIDA* from t to s but expanding only nodes that were expanded in the

forward search (i.e., the nodes in CLOSED of the forward search). We call the graph that is composed of these nodes and the edges between them the *induced graph* of the forward search. The backward search is done in the path space (i.e., no duplicate detection is done), possibly considering paths that were pruned by the forward search. The backward search halts when either $best_\Delta$ is proven to be the solution or when all paths in the induced graph were visited. The former case occurs when either $best_\Delta = 0$ (when a perfect solution is found) or when all the nodes with $g(n) \geq T + best_\Delta$ were expanded. If all paths in the induced graph were visited, we run the forward search again.

In addition, the backward search applies the following pruning rule. Let $g_f(n)$ be the cost of the lowest cost from s to n found by the forward search, and let $g_b(n)$ be the cost of the backward path from t to n. If $g_f(n)+g_b(n) \geq T+best_\Delta$ then the backward search prunes n, as continuing the backward search from n to s will only find paths at least as large as $g_f(n)$, and thus a path with a better Δ_T than $best_\Delta$ can not be found. Node ordering techniques were also helpful in speeding the backward search (Linares López *et al.* 2013).

The Induced Transition Graphs

BTVS as defined above shown worse performance than TVSIDA*. The root cause of this is the backward search. The backward search runs on the path space, and is thus much more computationally demanding than the forward search. The performance of BTVS could then be greatly improved if one had an oracle that would know when running the backward search will find a better solution (i.e., one with $\Delta_T < best_\Delta$). Thus, the key to speeding up BTVS is to choose smartly when should the backward search be run, and when the forward search should continue to search and reveal more nodes.

Next, we describe a novel heuristic method that suggests when the backward search might lead to an improved solution. Furthermore, we prove that this method can detect with certainty cases where the backward search cannot lead to an improved solution, thus saving redundant calls to the backward search. This method is based on the *Induced Transition Graph* (ITG), defined next.

An ITG is an abstraction of the state space where all the nodes with the same g_f value are mapped into a single node. Every node in the ITG is labeled with the g_f value of the nodes in the original search space that are mapped to it. An edge between nodes i and j in the ITG exists iff the forward search encountered such an edge (between nodes with the corresponding g value). Every edge (i, j) in the ITG is labeled with the number of times in the forward search that a node with $g = i$ generated a descendant with $g = j$. This is denoted by $\#(i, j)$.

A path P' in the ITG is called an ITG traversal if it is a path from s' to t' such that every edge (i, j) exists in P' at most $\#(i, j)$ times. It is easy to see that every path from s to t that is found by the backwards search has a corresponding ITG traversal of the same length. Thus, instead of performing the backwards search to check if it contains a better path than P_{best}, one can enumerate all the ITG traversals and check whether there is an ITG traversal P' such that $|T - |P'|| < best_\Delta$, where $|P'|$ is the length of P'.

	Perfect				Not-Perfect			
Blocked	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4
TVSA*	213	180	199	201	40	43	45	50
TVSIDA*	250	250	248	239	81	88	97	123
T*	250	250	250	250	240	240	240	240

Table 1: # Instances solved, 512x512 grid

Theorem 1 *If for every ITG traversal P' it holds that $|T - |P'|| \geq best_\Delta$ then the backwards search will not provide any solution better than Δ_T.*

Theorem 1 can be used to restrict the backward searches in BTVS: a backward search is only applied if there is an ITG traversal P' such that $|T - |P'|| < best_\Delta$. BTVS with ITGs used in this way is called T*. Note that the ITG concept can be applied to any state space abstraction that preserves edges. We demonstrate it on the ITG abstractions as a domain-independent solution. In domains with unit edge cost, dynamic programming can be used to allow BTVS to use ITG with constant overhead (Linares López *et al.* 2013)

Experimental Results

We evaluated the performance of TVSA*, TVSIDA* and T* on three standard search benchmarks: 4-connected grid pathfinding, the tile puzzle and the pancake puzzle. All the experiments have been performed on a Linux computer with a time cutoff of 120 seconds and 2 Gb of memory. In all domains T* substantially outperformed TVSA* and TVSIDA*. Table 1 shows the results of the 4-connected grid pathfinding domain, where a perfect solution exists and where it does not, for a range of T values. As can be seen, T* is always able to solve substantially more instances than its competitors.

References

Lukas Kuhn, Tim Schmidt, Bob Price, Johan de Kleer, Rong Zhou, and Minh Do. Heuristic search for target-value path problem. In *The First International Symposium on Search Techniques in Artificial Intelligence and Robotics*, 2008.

Carlos Linares López, Roni Stern, and Ariel Felner. Target-value search for general graphs. In *International Joint Conference on Artificial Intelligence (IJCAI), to appear*, 2013.

Tim Schmidt, Lukas Kuhn, Bob Price, Johan de Kleer, and Rong Zhou. A depth-first approach to target-value search. In *Symposium on Combinatorial Search (SOCS-09)*, 2009.