Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)

# Solving the Target-Value Search Problem

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

**Roni Stern** and **Ariel Felner**
Information Systems Engineering
Ben Gurion University
Beer-Sheva, Israel 85104
roni.stern@gmail.com,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: first, for directed acyclic graphs; second, for general graphs under the assumption that nodes can be revisited given that the same edge can not be traversed twice. In this work we focus on a more restrictive variant of the same problem where nodes can not be revisited. We prove that this variant is NP-complete and discuss novel theoretical properties and provide empirical results to solve this problem optimally.

## Introduction

In the *Target-Value Search* problem (TVS) we are given a graph $G = (V, E)$, two states $s, t \in V$ and a *target value* $T \in \mathbb{N}$. The task is to find a path from $s$ to $t$ such that its cost 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. Previous work first, addressed TVS only in the context of directed acyclic graphs (Kuhn *et al.* 2008) with small-scale experiments with graphs having at most 70 nodes. Secondly, for general graphs under the assumption that nodes can be revisited given that the same edge can not be traversed twice (Linares López *et al.* 2013).

In this work, we address a more restrictive variant of the same problem for general graphs where nodes can not be revisited. If $T \leq g^*(s,t)$, where $g^*(s,t)$ is the cost of the shortest path between $s$ and $t$, the problem can be solved in polynomial time since it suffices to compute the shortest path. Otherwise, it can be shown with a reduction from the subset-sum problem that TVS is NP-hard. Given an instance $\langle\{s_1,\ldots,s_n\},T\rangle$ of the subset-sum problem, construct a weighted graph $G_T(V_T, E_T)$, where $V_T = \{v_0,\ldots,v_n,v'_1,\ldots,v'_n\}$ and there is an edge $(v_{i-1}, v_i)$ with a weight $s_i$. Additionally, add the edges $(v_{i-1}, v'_i)$ and $(v'_i, v_i)$ with a null weight. Clearly, there is a simple path between $v_0$ and $v_n$ of cost equal to $T$ if and only if there is a subset of $\{s_1,\ldots,s_n\}$ whose sum is equal to $T$. Since the TVS problem is clearly in NP, it is NP-complete[1].

[1]Thanks to Yuval Filmus for providing this proof.

We succintly describe the algorithms previously used in the second variant mentioned above and discuss a simple modification that shows that this variant can be effectively solved also.

## A$^*$ and IDA$^*$ for TVS

A$^*$ and IDA$^*$ can be easily adapted to deal with this variant of the TVS problem. Importantly, no duplicate detection is performed with these algorithms since with TVS different paths to the same node with different costs are meaningful.

The 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 nodes exists, TVSA$^*$ chooses the node with the lowest $\Delta_T(n)$ among all. 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, where $best_\Delta$ is the minimum deviation wrt $T$ found so far. IDA$^*$ was also adapted in a variant called TVSIDA$^*$. It is 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. See (Linares López *et al.* 2013) for more details.

## 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 $s$. We use the term ***BTVS iteration*** to denote a single call for the forward and backward search.

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 $n$ which is denoted as $g_f(n)$. When a path to $t$ is found we store its $\Delta_T$ in $best_\Delta$ and call the backward search.

The backwards search consists of running TVSIDA$^*$ from $t$ to $s$ but expanding only nodes that were expanded in 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. During the backward search $best_\Delta$ is updated if a solution is found with $\Delta_T < best_\Delta$. The backward search halts when either $best_\Delta$ is proven to be the solution or when all paths in the induced graph were visited. In the latter case, we run the forward search again until the $f$-value of all nodes is larger or equal than $T + best_\Delta$. In addition, the backwards search is equipped with additional pruning and ordering criteria (Linares López *et al.* 2013).

**Theorem 1** *BTVS is sound and complete and after precisely two BTVS iterations the path with the lowest $\Delta_T$ is guaranteed to be found.*

## The Induced Transition Graphs

Experiments with BTVS have shown it to have better performance than TVSA* and TVSIDA*. However, it is still feasible to speed it up by choosing smartly when should the backward search be run, and when the forward search should continue to search and reveal more nodes. The original idea consisted of introducing a novel heuristic called *Induced Transition Graph* (ITG) (Linares López *et al.* 2013).

An ITG is an abstraction of the state space where all the nodes with the same $g$ value are mapped into a single node. Every node in the ITG is labelled according to 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). For this particular variant, let $l_i$ denote the number of nodes in the original state space with $g_f(\cdot) = i$. A path $P'$ in the ITG is called an ITG traversal if it is a path from $s'$ to $t'$ such that every node $i$ exists in $P'$ at most $l_i$ 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, one can enumerate all the ITG traversals and check whether there is an ITG traversal $P'$ such that $|T - |P'|| < best_\Delta$. This technique was used to restrict the backward searches in BTVS: a backward search is only applied if there is an ITG traversal $P'$ such that $|T - |P'|| < \Delta_T(P_{best})$. We denote with T* to BTVS with ITGs used in this way.

In the particular case that interests us here where only single paths are allowed, this idea however does not suffice because it implicitly assumes that the same node $n$ in the original state space can be used to move between two nodes with the same $g_f$ value. Thus, a simple modification consists of recording in the ITG the number of nodes with two or more parents. We demonstrate it in domains with unit edge cost where dynamic programming can be used to allow BTVS to use the ITG with constant overhead, in a similar way as it was previously done (Linares López *et al.* 2013).

**Theorem 2** *Let us denote with $best^0_\Delta$ the minimum deviation of the best solution found after the first iteration of BTVS. The number of iterations of T* is bounded by $(T + best^0_\Delta - g_f(s,t))$.*

### Experimental Results

We evaluated the performance of TVSA*, TVSIDA* BTVS and T* on the 4-connected grid pathfinding, the sliding-tile puzzle and the pancake puzzle. All the experiments have been performed on an Intel Xeon 2.93 GHz Quad core processor (64 bits) using Linux with a time cutoff of 120 seconds and 2 Gb of memory. A small part of these experimental results can be viewed in Table 1, which shows the number of instances solved, out of 400, for a wide range of even and odd target values, $T$. As can be seen, BTVS and T* are able to solve substantially more instances than its competitors in this domain as well as in the others.

| | even | | | | |
|---|---|---|---|---|---|
| | 5 | 6 | 7 | 8 | 9 |
| TVSA* | 353 | 359 | 117 | 18 | 10 |
| TVSIDA* | 202 | 102 | 69 | 40 | 23 |
| BTVS | 377 | 373 | 380 | 390 | 390 |
| T* | 400 | 373 | 380 | 390 | 390 |
| | odd | | | | |
| | 5 | 6 | 7 | 8 | 9 |
| TVSA* | 351 | 357 | 115 | 19 | 10 |
| TVSIDA* | 196 | 103 | 69 | 40 | 23 |
| BTVS | 371 | 372 | 380 | 390 | 390 |
| T* | 400 | 372 | 380 | 390 | 390 |

Table 1: # instances solved for various sizes of the $N$-pancake puzzle

## 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 revisited. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pages 601–607, Beijing (China), August 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.