# Potential Search: A New Greedy Anytime Heuristic Search

**Roni Stern**
Information Systems Engineering
Ben Gurion University
Beer-Sheva, Israel
roni.stern@gmail.com

**Rami Puzis**
Deutsche Telekom Laboratories,
Information Systems Engineering
Ben Gurion University
Beer-Sheva, Israel
puzis@bgu.ac.il

**Ariel Felner**
Deutsche Telekom Laboratories,
Information Systems Engineering
Ben Gurion University
Beer-Sheva, Israel
felner@bgu.ac.il

## Abstract

In this paper we explore a novel approach for anytime heuristic search, in which the node that is most probable to improve the *incumbent solution* is expanded first. This is especially suited for the "anytime aspect" of anytime algorithms - the possibility that the algorithm will be be halted anytime throughout the search. The potential of a node to improve the incumbent solution is estimated by a custom cost function, resulting in *Potential Search* , an anytime best-first search. Experimental results on the 15-puzzle and on the *key player problem in communication networks* (KPP-COM) show that this approach is competitive with state-of-the-art anytime heuristic search algorithms, and is more robust.

## Introduction

Anytime algorithms are:"algorithms whose quality of results improves gradually as computation time increases" (Zilberstein 1996). A typical run of an anytime algorithm can be divided into three phases. First, a suboptimal solution is quickly found. Then, the search continues, better solutions are found until an optimal solution is reached. Finally, the algorithm verifies that no better solution exists, returning the optimal solution. After the first solution has been found, anytime algorithms are designed to always return a solution if they are halted. The best solution found at each stage is referred to as the *incumbent solution*.

Unlike *contract algorithms* which are given a runtime limit a priori, *anytime algorithms* can be halted at anytime. Therefore, rapidly improving the cost of the incumbent goal should be the main focus of anytime algorithms. Assume that it is possible to calculate the probability of any given node to lead to a goal with lower cost than the incumbent goal cost. We propose the following approach: expand the node that is most probable to lead to a solution with lower cost than the cost of the incumbent solution. This is different from most algorithms that use variants of $f_w = g + w \cdot h$ as they prioritize nodes that are closest to any goal without considering the cost of the incumbent solution.

For example, consider Figure 1 where node $S$ is the initial node and $G$ is the goal. Nodes $A$ and $B$ are in the *open list* and a path from $S$ to $G$ with cost 100 has been previously found (cost of incumbent solution = 100). Let $\hat{f}$ be the cost function used by the given algorithm, e.g., $f_w$ for $w \geq 1$. In addition, assume that $p(n)$ is the probability of finding a better cost than 100, via node $n$. Most anytime heuristic search algorithms will choose to expand node $B$ due to its lower $\hat{f}$ cost. We propose to expand node $A$, because it is more probable to lead to a solution with lower cost than the incumbent solution. This especially suits anytime behavior, as a solution may be requested anytime during the search.

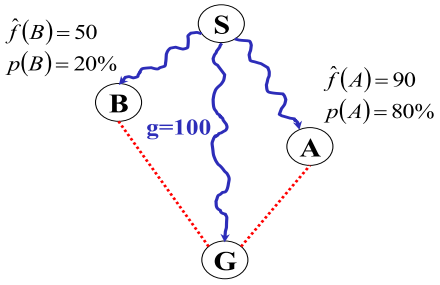


Figure 1: Example of expansion dilemma

## Potential Search

We refer to the probability that a node $n$ will lead to a better solution than the incumbent solution as the *potential* of a node and denote it as $p(n)$. This probability can be formally defined as follows. Let $g(n)$ be the cost of the lowest cost path found so far from the initial state to $n$, $h^*(n)$ as the cost of the lowest cost path from $n$ to a goal and $UB$ as the cost of the incumbent solution. A node will improve the incumbent goal if $g(n) + h^*(n) < UB$. If $h^*(n)$ is known then the *potential* is a binary function, returning one if $g(n) + h^*(n) < UB$ and zero otherwise. However, $h^*(n)$ is usually unknown (otherwise finding the shortest path would be trivial). This leads to the following natural definition of the *potential* of a node:

**Definition 1 Potential.** *The* potential *of node $n$, denoted $p(n)$, is the probability that $g(n) + h^*(n) < UB$.*

If the error of $h(n)$ with respect to $h^*(n)$ grows linearly with $h(n)$, then the potential of a node can be estimated accurately with the cost function $p_{lr}(n) = \frac{h(n)}{UB - g(n)}$. The intuition behind $p_{lr}(n)$ is as follows. $UB - g(n)$ is an upper

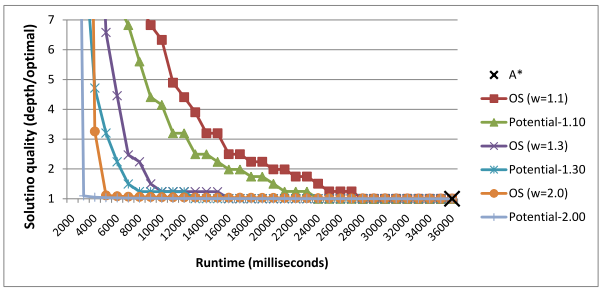


Figure 2: 15-puzzle, solution quality Vs. Runtime

bound on the distance to a goal from node $n$ that may improve the incumbent solution.[1] $h(n)$ is a lowerbound on the search distance to a goal from $n$. Therefore a node with low $\frac{h(n)}{UB-g(n)}$ is more likely to improve the incumbent goal cost. It can be proven that if $h^*(n) = X \cdot h(n)$ for an independent random variable $X$, then using $p_{lr}$ as the cost function that determines which nodes are expanded next, results in an expansion order that is exactly the same as using $p(n)$. *Potential Search* is a best-first search implementation that uses $p_{lr}$ as a cost function. Note that until the first path to a goal is found $UB$ is unknown, and some search algorithm that can find a solution quickly will be used (e.g. WA*).

## Experimental results

We first evaluated the performance of *Potential Search* on the standard 100 random instances of the 15-puzzle. We compared *Potential Search* to AWA* (Hansen and Zhou 2007) and Optimistic search (Thayer and Ruml 2008) (labeled hereafter as OS). AWA* is an anytime variant of WA* that continues to run WA* with the same weight even after a solution is found. OS uses two cost functions: an admissible heuristic $h$ and an inadmissible heuristic $\hat{h}$ that more accurately estimates the cost to the goal. OS chooses to expand the node with the lowest $g + \hat{h}$, but switches to using $g + h$ if all the nodes in the open list will not improve the incumbent solution according to $\hat{h}$. In our experiments we used $\hat{h} = w \times h$ with $w$ values of 1.1, 1.3 and 2 where $h$ is Manhattan Distance. WA* with the same weights was used for finding the first solution in *Potential Search* (the cost of this solution was then used to initialize $UB$).

Figure 2 shows the results of *Potential Search* Vs. OS. The x-axis denotes the runtime in milliseconds, and the y-axis displays the solution quality (the cost of goal found divided by the optimal goal cost). The X mark at the rightmost point of the x-axis denotes the runtime required for A*to find the optimal solution. As can be seen, when using the same weight *Potential Search* always outperforms Optimistic Search. In this domain, AWA* and *Potential Search* displayed similar performance.

Our second test domain is the *key player problem in communication networks* (KPP-COM) (Puzis, Elovici, and Dolev 2007). This is the problem of finding the set of $k$ nodes with the highest group betweenness centrality

[1] We assume here a domain with unit edges cost, but this can be easily generalized.

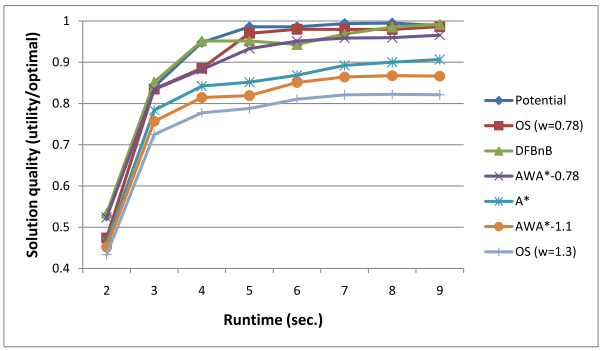


Figure 3: KPP-COM, 600 nodes, density 2

(GBC) (Everett and Borgatti 1999) in a graph. GBC is a metric for centrality of a group of nodes which is ratio of shortest paths that pass through these group of nodes. This problem has several real-life applications in networking and security. While this problem is NP-Complete, search techniques can be used to find high quality solutions. In addition there exists effective admissible heuristics for this problem, see (Puzis, Elovici, and Dolev 2007) for more detailed discussion on this heuristic and the KPP-COM problem. In addition, besides *Potential Search* , AWA* and OS we also experimented with DFBnB in this domain.

Since the main motivation for KPP-COM problem is in communication network domains, scale free graphs generated according to the Barabasi-Albert model (Barabasi and Albert 1999) were used in our experiments. This model is a well-used model of Internet topology and the web graph. Figure 3 presents a small representing subset of the experiments, displaying the average results on 100 different graphs with 600 nodes, density of 2 and a desired group size of 20. The x-axis denotes the runtime and the y-axis the solution quality (best utility found divided by the optimal utility). As can be seen, *Potential Search* outperforms all other algorithm, albeit only slightly better than DFBnB.

There are many future direction for this work. Improving potential function to incorporate search effort, solution quality estimates as well as more complex error functions is one such direction. Another direction is to combine *Potential Search* with a contract algorithm when some knowledge of the halting time is given.

## References

Barabasi, A. L., and Albert, R. 1999. Emergence of scaling in random networks. *Science* 286(5439):509–512.

Everett, M. G., and Borgatti, S. P. 1999. The centrality of groups and classes. *Journal of Mathematical Sociology* 23(3):181–201.

Hansen, E. A., and Zhou, R. 2007. Anytime heuristic search. *J. Artif. Intell. Res. (JAIR)* 28:267–297.

Puzis, R.; Elovici, Y.; and Dolev, S. 2007. Finding the most prominent group in complex networks. *AI Commun.* 20(4):287–296.

Thayer, J. T., and Ruml, W. 2008. Faster than weighted a*: An optimistic approach to bounded suboptimal search. In *ICAPS*, 355–362.

Zilberstein, S. 1996. Using anytime algorithms in intelligent systems. *AI Magazine* 17(3):73–83.