Artificial Intelligence 214 (2014) 1–25

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Potential-based bounded-cost search and Anytime Non-Parametric A*

Roni Stern [a,*], Ariel Felner [a], Jur van den Berg [b], Rami Puzis [a], Rajat Shah [c], Ken Goldberg [c]

[a] Information Systems Engineering, Ben Gurion University, Be'er Sheva, Israel
[b] School of Computing, University of Utah, Salt Lake City, UT, USA
[c] University of California, Berkeley, CA, USA

ARTICLE INFO

Article history:
Received 1 May 2012
Received in revised form 1 May 2014
Accepted 3 May 2014
Available online 10 May 2014

Keywords:
Heuristic search
Anytime algorithms
Robotics

ABSTRACT

This paper presents two new search algorithms: Potential Search (PTS) and Anytime Potential Search/Anytime Non-Parametric A* (APTS/ANA*). Both algorithms are based on a new evaluation function that is easy to implement and does not require user-tuned parameters. PTS is designed to solve bounded-cost search problems, which are problems where the task is to find as fast as possible a solution under a given cost bound. APTS/ANA* is a non-parametric anytime search algorithm discovered independently by two research groups via two very different derivations. In this paper, co-authored by researchers from both groups, we present these derivations: as a sequence of calls to PTS and as a non-parametric greedy variant of Anytime Repairing A*.

We describe experiments that evaluate the new algorithms in the 15-puzzle, KPP-COM, robot motion planning, gridworld navigation, and multiple sequence alignment search domains. Our results suggest that when compared with previous anytime algorithms, APTS/ANA*: (1) does not require user-set parameters, (2) finds an initial solution faster, (3) spends less time between solution improvements, (4) decreases the suboptimality bound of the current-best solution more gradually, and (5) converges faster to an optimal solution when reachable.

1. Introduction

Heuristic search algorithms are widely used to compute minimum-cost paths in graphs. Applications of heuristic search range from map navigation software and robot path planning to automated planning and puzzle solving. Different search algorithms return solutions of varying quality, which is commonly measured by a cost, where high quality solutions are those with a low cost. Ideally, one would like to find optimal solutions, i.e., those with minimum cost. Given an admissible heuristic, some search algorithms, such as A* [1] or IDA* [2], return optimal solutions. However, many problems are very hard to solve optimally [3] even with such algorithms.

In this paper we propose two algorithms: Potential Search (PTS) and Anytime Potential Search/Anytime Non-Parametric A* (APTS/ANA*). These algorithms are especially suited for cases where an optimal solution is hard to find. PTS is designed to solve problems where any solution with cost less than C, an input to the problem, is acceptable, while solutions with cost $\geq C$ are useless. We call such problems *bounded-cost search problems*. Bounded-cost search problems

* Corresponding author.

http://dx.doi.org/10.1016/j.artint.2014.05.002
0004-3702/

arise, for example, when the expense budget for a business trip is limited and the trip (e.g., flights and hotels) should be planned as quickly as possible but within the budget limit. If an online travel agency such as Expedia or Priceline is planning the trip, computational resources could be diverted to other clients once a plan is found within the budget limits (see Priceline's "Name Your Own Price" option).

The second algorithm we present, APTS/ANA*, is an anytime search algorithm, i.e., an algorithm "whose quality of results improves gradually as computation time increases" [4]. APTS/ANA* can be viewed as a translation of an anytime search algorithm into a sequence of bounded-cost search problems solved by PTS, or as an intelligent approach to avoid the parameter setting problem of Weighted A*-based anytime search algorithms. Setting parameters to bounded-suboptimal search algorithms is a known problem in the heuristic search literature [5]. A key benefit of APTS/ANA* is that it does not require users to set parameters, such as the w_0 and Δw parameters of ARA* [6]. Furthermore, experiments suggest that APTS/ANA* improves upon previous anytime search algorithms in most cases by (1) finding an initial solution faster, (2) spending less time between solution improvements, (3) decreasing the suboptimality bound of the current-best solution more gradually, and (4) converging faster to an optimal solution when reachable.

Both PTS and APTS/ANA* are based on a new evaluation function $u(n) = \frac{C-g(n)}{h(n)}$ that was discovered independently by two research groups via two very different derivations. In this paper, co-authored by researchers from both groups, we present both derivations. The first derivation of $u(\cdot)$ is based on a novel concept called the *potential* of a node. The potential of a node n is defined with respect to a given value C, and is the probability that a node n is part of a solution of cost lower than C. We prove that the node with the highest $u(\cdot)$ is the node with the highest potential, under certain probabilistic relation between the heuristic function and the cost it estimates.

The second derivation of $u(\cdot)$ is based on the desire for a non-parametric version of the Anytime Repairing A* algorithm [6]. We show that expanding the node with the highest $u(\cdot)$ has the same effect as setting the parameters of ARA* dynamically to improve the best solution found so far as fast as possible. In addition, we show that $u(\cdot)$ bounds the suboptimality of the current solution.

We compare PTS and APTS/ANA* with previous anytime algorithms on five representative search domains: the 15-puzzle, robot motion planning, gridworld navigation, Key player problem in communication (KPP-COM), and multiple sequence alignment. As mentioned above, the experimental results suggest that APTS/ANA* improves upon previous anytime search algorithms in terms of key metrics that determine the quality of an anytime algorithm. As a bounded-cost algorithm, our results suggest that PTS, which is specifically designed for bounded-cost problems, outperforms competing algorithms for most cost bounds and exhibits an overall robust behavior.

This paper extends our preliminary work [7–9] by (1) providing a substantially more rigorous theoretical analysis of the presented algorithms, (2) extending the experimental results, (3) adding a comprehensive discussion on the relation between bounded-cost search and anytime search, and (4) discussing the limitations of PTS and APTS/ANA*.

The structure of this paper is as follows. First, we provide background and related work. In Section 3, we introduce the bounded-cost search problem and present Potential Search (PTS). In Section 4, we present Anytime Potential Search (APTS) [7,8] and Anytime Non-Parametric A* (ANA*) [9], showing that they are equivalent and discussing their theoretical properties. Section 5 presents experimental results comparing PTS and APTS/ANA* with previous algorithms. Finally, we discuss a generalization of PTS (Section 6) and conclude with suggestions for future work (Section 7).

2. Background and related work

Search algorithms find a solution by starting at the initial state and traversing the *problem space graph* until a goal state is found. Various search algorithms differ by the order in which they decide to traverse the problem space graph. Traversing the problem space graph involves *generating* and *expanding* its nodes. The term *generating* a node refers to creating a data structure that represents it, while *expanding* a node means generating all its children.

One of the most widely used search frameworks is best–first search (BFS) [10]. BFS keeps two lists of nodes: an *open list* (denoted hereafter as *OPEN*), which contains all the generated nodes that have not been expanded yet, and a *closed list* (denoted hereafter as *CLOSED*), which contains all the nodes that have been previously expanded. Every generated node in *OPEN* is assigned a value by an evaluation function. The value assigned to a node is called the *cost* of the node. In every iteration of BFS, the node in *OPEN* with the lowest cost is chosen to be expanded. This lowest-cost node is moved from *OPEN* to *CLOSED*, and the children of this node are inserted to *OPEN*. The purpose of *CLOSED* is to avoid inserting nodes that have already been expanded into *OPEN*. *CLOSED* is also used to help reconstruct the solution after a goal is found. Once a goal node is chosen for expansion, i.e., it is the lowest-cost node in *OPEN*, BFS halts and that goal is returned.[1] Alternatively, BFS can be defined such that every node is assigned a value, and in every iteration the node with the highest value is expanded.

BFS is a general framework, and many well-known algorithms are special cases of it. For example, Dijkstra's single-source shortest-path algorithm [12] and the A* algorithm [1] are both special cases of BFS, differing only in their evaluation function.[2] Dijkstra's algorithm is BFS with an evaluation function that is $g(n)$, which is the shortest path found so far from

[1] This is the textbook version of BFS [10]. However, there are variants of BFS where the search is halted earlier (e.g., BFS with lookaheads [11]).

[2] In this paper we consider Dijkstra's algorithm in its best–first search variant, which is also known as Uniform Cost Search. It has been shown [13] that this variant of Dijkstra is more efficient than the implementation of Dijsktra detailed in the common algorithm textbook [14].