Expert Systems with Applications 42 (2015) 2722–2730

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Uninformed pathfinding: A new approach

Kai Li Lim [a,*], Kah Phooi Seng [b], Lee Seng Yeong [a], Li-Minn Ang [b], Sue Inn Ch'ng [a]

[a] Department of Computer Science & Networked System, Sunway University, Malaysia
[b] School of Engineering, Edith Cowan University, WA 6027, Australia

ARTICLE INFO

Article history:
Available online 15 November 2014

Keywords:
Pathfinding
Uninformed search
Bidirectional search
Parallel search
Multi-goal search

ABSTRACT

This paper presents a new pathfinding algorithm called the boundary iterative-deepening depth-first search (BIDDFS) algorithm. The BIDDFS compromises the increasing memory usage of the Dijkstra's algorithm, where the memory clears enables the BIDDFS to consume less memory than the Dijkstra's algorithm. The expansion redundancy of the iterative-deepening depth-first search (IDDFS) is also compensated; it is faster than the IDDFS in all of the testing instances conducted. The BIDDFS is further enhanced for bidirectional searching to allow expanding to fewer nodes and reducing pathfinding time. The bidirectional BIDDFS and the parallel bidirectional BIDDFS are also proposed. The proposed BIDDFS is further extended to the multi-goal BIDDFS, which is able to search for multiple goals present on the map in a single search. Simulation examples and comparisons have revealed the good performance of the proposed algorithms.

1. Introduction

Pathfinding is described as the process of calculating a path between two points located on a map or environment. This process is performed using pathfinding algorithms (Cai, Yan, & Tie-Song, 2011; Dijkstra, 1959; Hart, Nilsson, & Raphael, 1968; Korf, 1988; Korf, 1985; Stentz, 1994; Wilt, Thayer, & Ruml, 2010). The calculated routes are normally optimized for shortest distance, fastest traveling time, and/or lowest cost. The eventual goal of pathfinding is to calculate a route from point to point.

Pathfinding algorithms consist of a search to calculate the cost to the goal, expanding nodes, and to find the path from the goal to start using the route of lowest cost. The search algorithms can be informed or uninformed. Informed pathfinding involve the use of a heuristic function to locate the goal during pathfinding, and the direction of pathfinding is guided towards that location, making informed searches typically faster than uninformed searches, it is used in situations where the goal location is known. Uninformed pathfinding does not use heuristics for pathfinding, and are also known as blind searches, it typically search in all directions, usually in a radial pattern originating from the starting node, it is used in situations where the goal location is unknown. Examples of uninformed pathfinding algorithms are the Dijkstra's algorithm (Dijkstra, 1959) and the iterative-deepening depth-first search (IDDFS) (Korf, 1985). Examples of informed pathfinding algorithms includes the A* search (Hart et al., 1968) and the IDA* search (Korf, 1985).

Dijkstra's algorithm is an uninformed search algorithm proposed by E.W. Dijkstra. This algorithm uses a cost calculation feature to guarantee the shortest path from pathfinding. This algorithm is often redeveloped and improved upon its classical model – this includes implementing the algorithm in road navigation (Yin & Wang, 2010) and improving the algorithm by its storage structure and searching area (DongKai & Ping, 2010). This algorithm suffers a drawback where an increasing size of map will introduce an exponential increase in memory requirements. The performance of this algorithm is often measured and compared with other algorithms such as the breadth first search (BFS), depth-first search (DFS) and the A* algorithm (Terzimehic, Silajdzic, Vajnberger, Velagic, & Osmic, 2011). Adding to that, the Dijkstra's algorithm has also been tested for performance over different architectures of implementation – serial and parallel (Jasika et al., 2012). This algorithm also assumes that the working environment is static and not dynamic. Since every old node is not back referred, a changing environment will render this algorithm ineffective.

The iterative-deepening depth-first search (IDDFS) was derived to solve the memory drawback of the Dijkstra's algorithm. This algorithm searches the map like the Dijkstra's algorithm sans the memory to record the location that it has expanded. The algorithm to prevent over-searching the map uses a threshold. An iteration of

* Corresponding author.
E-mail addresses: 12057642@sunway.edu.my, kaililim@theiet.org (K.L. Lim), jasmine.seng@gmail.com (K.P. Seng), leesengy@sunway.edu.my (L.S. Yeong), li-minn.ang@ecu.edu.au (L.-M. Ang), sueinnc@sunway.edu.my (S.I. Ch'ng).

http://dx.doi.org/10.1016/j.eswa.2014.10.046
0957-4174/© 2014 Elsevier Ltd. All rights reserved.

the IDDFS begins at the starting point; this introduces search redundancy where the starting point is searched multiple times before the goal is found. At its very basic implementation, the IDDFS is widely demonstrated as a tree-searching algorithm (Smit & Stroulia, 2011a, 2011b). Hence, rather than deriving its methods from Dijkstra's algorithm as did the IDA* from the A*, the IDDFS was derived from the depth-first search (DFS) (Korf, 1985). For this implementation, the IDDFS was adapted for use on a grid-based map. Still, as a tree-search algorithm, it was not designed to return a solution route from the goal node back to the starting node. To allow the shortest route to be calculated, a cost variable has to be factored in, and this is borrowed from the Dijkstra's algorithm. Literatures employing the IDDFS usually do so due to its lower memory requirements compared to the Dijkstra's algorithm, which is usually used in video games (Maggiore et al., 2013; Zhao, Schiffel, & Thielscher, 2009).

Recent literatures covering iterative-deepening searches often continues to improve on the IDA*. For example, Mencía, Sierra, & Varela (2013) described a search technique which combines the IDA* with a partially informed depth-first search, resulting in faster searches for small to medium instances. Burns & Ruml (2013) also described an algorithm that improves on the IDA* that is able to select the nodes to be expanded, capable of on-line training. Faster iterative-deepening searching was also achieved by Sharon, Felner, & Sturtevant (2014) with the proposal of the Exponential Deepening A* (EDA*) search, which introduces an exponentially increasing threshold to decrease the number of iterations needed for pathfinding, resulting in faster pathfinding. On the other hand, Valenzano, Arfaee, Thayer, Stern, & Sturtevant, (2013) published a work on finding suboptimal bounds in heuristic search, noting that most implementations including the IDA* are optimal. The additive bounding technique that they proposed notes that it is not feasible to find optimal solutions given practical runtime and memory constrains, and aims to find a suboptimal tradeoff for that solution. An in-depth analysis of the IDA* was recently published by Lelis, Zilles, & Holte (2013), where they also proposed algorithms to predict the number of nodes expanded by the IDA* search. It is observable that most literature proposing new iterative-deepening algorithms attempt to improve upon the IDA*, which is an informed search algorithm. There is hence a lack of proposal of algorithms to improve upon the uninformed IDDFS algorithm, where the heuristic usage of informed search algorithms often result in faster pathfinding from the reduced search area. However, a similar need to improve the IDDFS for the same reasons of the IDA* is present as there exist applications that benefit from the implementation of uninformed search algorithms over informed search algorithms. For example, the IDDFS was proposed due to the Dijkstra's algorithm exhibiting an exponential memory requirement increase with map size. While the IDDFS minimizes the memory footprint of the IDDFS by not storing data regarding expanded nodes, it suffers a drawback whereby searching becomes redundant whereby nodes are repeatedly re-expanded at every new iteration. These problems motivated the research here to develop a new approach on uninformed pathfinding, addressing the search redundancy of the IDDFS while retaining its lower memory footprint.

In this paper, the boundary iterative-deepening depth-first search (BIDDFS) is proposed to compensate the memory requirements of the Dijkstra's algorithm and the search redundancy of the IDDFS. This algorithm utilizes the iterative-deepening feature of the IDDFS to reduce the memory requirements for pathfinding and the cost calculation feature of the Dijkstra's algorithm to guarantee the shortest path. A portion of memory is allocated for the algorithm to store the location of the boundary to minimize search redundancy. This means that while the IDDFS perpetually repeats its search from the starting point, the BIDDFS repeats its search from the saved boundary locations. The BIDDFS exhibits faster single-node expansion speed compared to the A* search, IDA* search and fringe search (Björnsson, Enzenberger, Holte, & Schaeffer, 2005), and it is faster than the IDDFS due to its minimized redundancy. To allow faster searching, the BIDDFS was enhanced for bidirectional searching. Bidirectional searching searches back and forth between the starting and goal locations, and these two searches meet at a distance in between and subsequently the route is then plotted from the meeting location. This approach of searching is able to reduce pathfinding times due to lesser number of nodes searched. Furthermore, the bidirectional BIDDFS is tested with a parallel approach to search from both the starting and goal locations simultaneously. The BIDDFS is also enhanced to search for multiple goals on a map. A multi-goal BIDDFS is able to search for all goals on a map in a single search, whereas single-goal algorithms will need to repeat searching for different goal locations, introducing search redundancy. This eventually allows the multi-goal BIDDFS to save time and redundancy searching multiple goals.

The remainder paper is organized as follows: Section 2 proposes the BIDDFS. Section 3 describes a bidirectional BIDDFS. Section 4 introduces a parallel bidirectional BIDDFS. Section 5 proposes a multi-goal BIDDFS. Section 6 shows the experiments for the algorithms. Section 7 discusses the possible applications for the algorithms. Finally, the concluding remark is given in Section 8.

2. The proposed boundary iterative-deepening depth-first search (BIDDFS)

This section presents BIDDFS, an uninformed boundary search algorithm. The BIDDFS is a newly proposed algorithm aiming to address the memory drawback of the conventional Dijkstra's algorithm (Dijkstra) and the searching redundancy of the IDDFS. Its main concept utilizes a boundary search. The BIDDFS explores the compromise between the IDDFS redundancy and the Dijkstra's algorithm's increasing memory requirements on larger maps.

A list is maintained in the memory to store information of the boundary nodes. The expansion pattern for this algorithm with IDDFS and the Dijkstra's algorithm is the same, so long as the map and cost environment remains the same. Since this algorithm is indirectly based off the Dijkstra's algorithm, this algorithm, along with all other algorithms discussed in this section should return the same, shortest path back to the starting node from the goal node.

The BIDDFS also operates using a threshold to compensate for the lack of memory, when a sub-runtime reaches its boundary, the boundary nodes are saved into the memory and accessed when the threshold increases and the search process restarts. To derive the BIDDFS from the IDDFS, a variable is initialized as an array to store the list of boundary node locations. This variable is written at the end of every threshold where the algorithm is unable to locate a goal; the nodes stored in this variable will be flagged as the locations where pathfinding on a new threshold is started, unlike the IDDFS that starts pathfinding at every threshold from the IDDFS. When the memory is cleared at every threshold, the variable's contents are not cleared. Fig. 1. shows an example of pathfinding using the BIDDFS. The algorithm does not re-expand the starting node and the expanded node. Since memory is allocated to store the boundary nodes, node expansions begins at every threshold from the boundary node. The pseudocode in Fig. 2 describes the BIDDFS algorithm.

Essentially, this algorithm follows the procedure below:

(1) Increasing threshold by 1
  (a) Calculating the cost of surrounding nodes from the location node.
  (b) Update surrounding OPEN nodes' cost.