Expert Systems With Applications 71 (2017) 319–331

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A new weighted pathfinding algorithms to reduce the search time on grid maps

Zeyad Abd Algfoor[a,*], Mohd Shahrizal Sunar[a,*], Afnizanfaizal Abdullah[b]

[a] UTM-IRDA Digital Media Centre, Media and Game Innovation Centre of Excellence, Institute of Human Centred Engineering, Universiti Teknologi Malaysia, 81310 Skudai Johor, Malaysia

[b] Synthetic Biology Research Group, Faculty of Computing, Universiti Teknologi Malaysia, 81310 Skudai Johor, Malaysia

ARTICLE INFO

Article history:
Received 11 July 2016
Revised 16 November 2016
Accepted 2 December 2016
Available online 3 December 2016

Keywords:
Pathfinding
JPS, A*
Bi-A*
Weight techniques
Pathfinding benchmarks maps

ABSTRACT

Artificial Intelligence (AI) techniques are utilized widely in the field of Expert Systems (ES) - as applied to robotics, video games self-driving vehicles and so on. Pathfinding algorithms are a class of heuristic algorithms based on AI techniques which are used in ES as decision making functions for the purpose of solving problems that would otherwise require human competence or expertise. ES fields that use pathfinding algorithms and operate in real-time face many challenges: for example time constraints, optimality and memory overhead for storing the paths which are found. For these algorithms to work, appropriate problem-specific maps must be constructed. In relation to this, the uniform-cost grid set-up is the most appropriate for ES applications. In this method, each node in a graph is represented as a tile, and the weight "between" tiles is set at a constant value, usually this is set to 1. In the state-of-the-art heuristic algorithms used with this data structure, multiplying the heuristic function by a weight greater than one is well-known technique. In this paper, we present three new techniques using various weights to accelerate heuristic search of grid maps. The first such technique is based on the iteration of a heuristic search algorithm associated with weight-set w. The second technique is based on the length between the start node and goal node, which is then associated with w. The last technique is based on the travel cost and is associated with a weight-set α. These techniques are applicable to a wide class of heuristic search algorithms. Therefore, we implement them, here, within the A*, the Bidirectional A* (Bi-A*) and Jump Point Search (JPS) algorithms; thus obtaining a family of new algorithms. Furthermore, it is seen that the use of these new algorithms results in significant improvements over current search algorithms. We evaluate them in path-planning benchmarks and show the amended JPS technique's greater stability, across weight values, over the other two techniques. However, it is also shown that this technique yields poor results in terms of cost solution.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

The use of pathfinding algorithms represents a key task in many domains and these algorithms are widely employed in areas such as artificial intelligence (Atallah & Blanton, 2009), robotics (Algfoor, Sunar, & Kolivand, 2015), video games (Algfoor et al., 2015), metabolic pathways (Algfoor, Sunar, Abdullah, & Kolivand, 2016; Croes, Couche, Wodak, & Van Helden, 2005; Planes & Beasley, 2008), and the resolving of problems relating to vehicular traffic (Bleiweiss, 2008; Silver, 2005). Pathfinding problems are divided into two major categories, multi-agent and single-agent. The multi-agent pathfinding (MAPF) problem is a generalization of the single-agent pathfinding problem. In the multi-agent situation, a number of agents simultaneously search for their destinations. Pathfinding algorithms begin exploring the graph from the root node and continue until a solution is found. In mathematical terms, graphs are represented as a set of vertices with edges connecting them.

To use and evaluate the performance of pathfinding algorithms, a suitable environment must be set-up. The most popular and most widely used method for representing pathfinding environments is the ubiquitous undirected uniform-cost grid map (Harabor, Botea, & others, 2010; Pochter, Zohar, Rosenschein, & Felner, 2010). Moreover, our focus, in this article, is on single-agent pathfinding problems, and our aim is to reduce the search space these have to explore to find an optimal solution.

There have been many techniques which have been suggested for the solution of pathfinding problems. One of the most powerful and simple techniques that has been proposed is that of the weighted heuristic search. This technique is extensively used with heuristic search algorithms. The weighted A* algorithm

* Corresponding author.
E-mail addresses: zeyadiraq1982@gmail.com (Z.A. Algfoor), shahrizal@utm.my (M.S. Sunar), afnizanfaizal@utm.my (A. Abdullah).

http://dx.doi.org/10.1016/j.eswa.2016.12.003
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

(Hart, Nilsson, & Raphael, 1968) is a well-known algorithm for single-agent search problems. It is based on multiplying the heuristic function by a constant weight. In the standard A* algorithm, the evaluation function f (*s*) = g (*s*) + *wh (s)* is used to rank a state which is on the search frontier. Where *g* (*s*) is the cost value incurred to reach *s, h* (*s*) is a (heuristic) estimate of the cost to reach a solution from *s*, and *w* is the weight value (greater than or equal to one). This algorithm, using this weighting technique, can find the solution much faster than a conventional A* algorithm. However, a drawback of this technique is that if the heuristic h is admissible, it will be far from optimal by at most a factor w in the majority of cases.

There are three types of problems which heuristic search algorithms suffer from: memory overhead, run-time constraints, and the optimality of solutions. For instance, the A* algorithm requires a memory allocation which is exponential to the length of the solution path; thus the algorithm may run out of memory before producing a solution, or it will spend an impractical time generating, storing and revisiting the stored search information. On the other hand, if the aim of the algorithm is to find optimal solutions only, a comprehensive exploration makes sense; consequently the search space used will be expanded in order to find an optimal solution in a shorter time. In point of fact, this strategy results in instability and is not always practicable. Hence, heuristic algorithms should pursue at least two objectives simultaneously, run-time limitation and optimality. These objectives are often in a trade-off relation, whereby the first biases the agent in favor of attempted actions, and the second causes the agent to aggressively try to keep the current quality of solution. Dealing with this trade-off adequately is the key to resolving the problems of these algorithms, and this depends on the requirements of the field of application.

Based on this observation, we present three new approaches to reducing the space required for the search, finding optimal solutions, and furthermore avoiding the inherent intractability which is present in most artificial intelligence problems. Each approach has its own equation which generates the values which are to be multiplied by the result of the heuristic function. The generated values will represent a bias in the selection of the next step along the path. The approaches (equations) are associated with two constant weight sets. The first two approaches are associated with weights $w > 0$ and the other approaches are associated with weights $\alpha \geq 2$. Furthermore, we have implemented all these approaches within three heuristic search algorithms: A*, Bidirectional A*(Sint & de Champeaux, 1977), and the JPS Jump Point Search (D. D. Harabor, Grastien, & others, 2011). We evaluate these implementations using standard benchmark maps extracted from a video-game (Sturtevant, 2012).

The rest of the paper is organized as follows. The next section describes related work on pathfinding approaches. In Section 3 we describe the three weighting approaches as implemented within the A*, Bidirectional A* and JPS heuristic algorithms. Section 4 presents our three proposed approaches and in the next section we describe the mechanism of these approaches. We then visualize these mechanisms in relation to grid maps in Section 6. Finally, in Section 8, we evaluate our algorithms via various convergence rate measurements commonly used in pathfinding. We then discuss the relevant aspects of the performance of the algorithms we are analyzing. The paper ends with a conclusion and suggestions for future work.

2. Related work

A number of weighting related techniques have been used to reduce the time taken to find solution paths. In practical terms, the time constraints can, fairly easily, be satisfied by the use of short distances between the start and goal nodes. Thus, the impact of any weight set can be more clearly seen when long distances intervene between the initial and the goal states. On the other hand, of course, when longer distances are involved, the number of obstacles and the distribution of these obstacles will incur more time and memory overhead (Cowling et al., 2013; Sturtevant, 2007). It should be noted that most previous work considers the cost of travelling between any two neighbors nodes to be constant. With this assumption, in the forward and backward directions the cost will be 1 and for diagonal movements, the cost value is $\sqrt{2}$.

The weighted A* (Pohl, 1970) method was one of the first attempts to find a relation between the weightings and the heuristic term. The best weight values, $1 \geq w \geq 0.5$, were selected based on observations associated with the heuristic value. The learning real-time A* algorithm (LRTA*) (Bulitko & Lee, 2006; Shimbo & Ishida, 2003) was designed taking into account the behavior and flexibility of the real-time search problem for autonomous agents. The weight values $2 \geq w > 0$ were selected to deal with the heuristic value. The drawbacks of the LRTA* algorithm are excessive exploration and the instability of the solution quality which is produced. In the same context, the suboptimal solution online/offline kNNLRTA* algorithm (Bulitko, Björnsson, Sturtevant, & Lawrence, 2011) utilizes large-scale multi-agent pathfinding with weight values $1 \geq w > 0$. This latter provides a new concept in terms of how real-time search agents can learn heuristics. The drawback of KNNLRTA* is that generating offline databases (as it does) takes additional time, and game companies do not find this acceptable. The wLSS-LRTA* and wLRTA*-LS algorithms (Rivera, Baier, & Hernández, 2013) combine two techniques in order to achieve a more efficient solution cost and total search time. The weights are incorporated into the lookahead search phase and the edges of the search graph during the learning phase.

We have noted that the previous work uses a set of weight values within certain limits based on experimental observations. This methodology for choosing weights is not sufficient and does not always provide optimal solutions. The reason for this is that the value of a certain weight can reduce the search time and give an optimal solution for some cases, but perhaps not in other cases. Therefore, we cannot base a weighting approach on a specific weight value which is always to be multiplied by the heuristic value. We have also noted that the weight values do not deal with cost travel value directly and without reference to the heuristic value. Consequently, we have implemented and tested three new techniques which generate values which 'cooperate' with constant weight values. Two of these techniques manipulate the heuristic value, and the other one manipulates the travel cost value.

3. Notation and terminology

In this work, the state space is defined as a quintuple (S, A, c, s, G), where (S, A) represents the search space in a strongly connected graph. Set S is a non-empty finite set of states (nodes), and A represents the set of all available actions (edges) $A \subset S \times S - \{(s, s) \mid s \in S\}$. Each edge is labeled with a function relating to the cost of travel from one node to another C: $A \rightarrow R+$, $\forall s \in S$ and a set $G \subset S$ of goal states. In this paper, a state space is an undirected graph where c (*s, t*) = c (*t, s*) for any (*s, t*) ∈ A.

A path is a non-empty sequence and a finite sequence of states (s0, s1, s2 …); we define the set of (immediate) successors of s by Succ (*s*) = {t | (*s, t*) ∈ A}, where *t* represents a candidate node. We calculate the distance between *s* and *t, d (s, t)* where *d* here denotes the direct distance of the shortest path between *s* and *t,* ignoring obstacles. We assume, in the state space, that there is at least one path from the non-goal state to the goal state, $s \in S - G$; the cost of the action of travelling from the initial state to the goal state is the summation of all successive states along the path. A heuristic function h: $S \rightarrow [0, \infty)$ undertakes to approximate the

Download English Version:

https://daneshyari.com/en/article/4943544

Download Persian Version:

https://daneshyari.com/article/4943544

Daneshyari.com