Information Sciences 332 (2016) 1–18

Contents lists available at ScienceDirect

Information Sciences

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

An effective initialization method for genetic algorithm-based robot path planning using a directed acyclic graph

Jaesung Lee, Dae-Won Kim*

School of Computer Science and Engineering, Chung-Ang University, 221, Heukseok-Dong, Dongjak-Gu, Seoul 156-756, Republic of Korea

ARTICLE INFO

Article history:
Received 9 January 2015
Revised 9 September 2015
Accepted 2 November 2015
Available online 6 November 2015

Keywords:
Robot path planning
Genetic algorithm
Initialization
Directed acyclic graph

ABSTRACT

The goal of robot path planning is to find a feasible path that proceeds from a starting point to a destination point without intersecting any obstacles in the given environment. Recently, genetic algorithm-based robot path planning methods have been widely considered in the intelligent robotics community. Because the initialization process significantly influences the performance of the genetic algorithm, an effective initialization method is required. However, investigation on this subject is still lacking. In this paper, we propose an effective initialization method for genetic algorithm-based robot path planning. Experimental results comparing genetic algorithms with conventional initialization methods and the proposed initialization method showed that the proposed method leads to high quality paths in a significantly shorter execution time.

1. Introduction

The goal of robot path planning (RPP) is to find a path which proceeds from a starting point to a destination point in a given environment. Although there are popular RPP methods based on A*, D*, Dijkstra's algorithm, and Floyd's algorithm [1–5], genetic algorithm-based robot path planning (GARPP) has recently received much attention in the intelligent robotics community; especially in application areas with complex objectives and complicated environments, such as unmanned bomb disposal, unmanned aerial vehicles, robot soccer, etc. [6–10]. In such areas, the obtained path must satisfy particular objectives, such as path feasibility, short path length, execution time to find a path, and path volatility [7,11–13].

Since the genetic algorithm (GA) starts its search for the final path from the initial path set, the initialization process is inevitably important for the effectiveness of GARPP methods [14–17]. Although it is already known that a good initial population significantly boosts the effectiveness of a GA [18–20], a serious investigation of the initialization process for the GARPP problem is still lacking. Therefore, GARPP methods suffer from performance degradation due to ineffective initial path sets.

In this paper, we propose an initialization method that specializes in GARPP problems. The proposed method creates a directed acyclic graph (DAG) by exploring a grid-based map encoded from a given environment, and then generates multiple paths for the initial GA path set from the obtained DAG. Because the proposed method generates multiple feasible solutions with short path lengths, GARPP methods initialized by the proposed method are able to output high quality paths in a significantly shorter execution time.

* Corresponding author. Tel.: +82 2 820 5304; fax: +82 2 820 5301.
E-mail address: dwkim@cau.ac.kr (D.-W. Kim).

http://dx.doi.org/10.1016/j.ins.2015.11.004
0020-0255/

Table 1
Comparison of path creation methods.

Method	Efficiency	Feasible path	Short path	Multiple paths	No. of points in path
Random	Fast	No	No	No	Large
Key cells	Fast	No	No	No	Small
Random walk	Slow	Yes	No	No	Large
CBPRM	Slow	Yes	No	No	Large
RRT	Fast	Yes	Yes	No	Small

2. Related works

GARPP methods find a path using a population-based stochastic search that consists of initialization, reproduction, evaluation, and a natural selection process [10]. In the early stages of GARPP research, many researchers focused on the reproduction process to enhance the effectiveness of the genetic search. This task was accomplished by adding new operators to fine-tune the paths to improve the effectiveness of GAs on RPP problems [4,15,16,21–23]. Although these studies formed a major trend for GARPP, researchers have paid less attention to the initialization process, even though it is known that an effective initial population significantly improves the GA search capability [18,20].

Initialization has shown to be effective in other GA applications and is considered an important issue because it affects the performance of GAs. However, in most GARPP studies, the initial path set is obtained by iteratively executing a simple random path creation method [16,17,24–26]. Because paths created by this method may include many intermediate points if the given environment size is large [6], the key cells-based path creation method was proposed as an alternative [12]. This method limits the maximum number of points in a path, based on the height, width, and number of obstacles in the environment. A common drawback of random- and key cells-based path creation methods is that they frequently create infeasible paths; an infeasible path intersects with an obstacle. This is problematic because mating two infeasible paths in a GA is likely to create a new infeasible path [16,27].

To ensure the created path is feasible, the clearance-based probabilistic road map method (CBPRM) can be used to create a path by executing a random walk-based path creation method [16] after enlarging the size of the obstacles in the environment [15]. However, the path thus created is likely to be longer in the worst case, because new points are added to the end of the point sequence until the destination point is included, resulting in a path that wanders all over the environment (roaming). To avoid roaming, the rapidly-exploring random tree (RRT)-based path creation method can be considered [28]. The RRT method creates a path by randomly selecting a point and connecting it to the nearest point in the sequence, creating a short, feasible path. However, RRT can be unscalable for initializing a GARPP, because it creates only a single path for each tree. Thus, the computational burden for initialization can be expensive if each run of RRT incurs a significant computational cost for exploring a map with a large number of nodes. In addition, the computational cost for initialization can be unacceptably high when the size of the path set is large, because RRT must be run repeatedly. A few other path creation methods for GARPP have also been proposed [4,17,29,30]. However, conventional methods suffer from a lack of generality to the environment, meaning that feasible paths may not be created [31].

Table 1 summarizes the characteristics of conventional initialization methods. Our goal is to provide a new initialization method that efficiently generates multiple feasible solutions with short path lengths. In addition, to the best of our knowledge, this is the first investigation of the quality of conventional initialization methods for the GARPP problem.

3. Proposed initialization method

3.1. Preliminaries

In this section, we explain the concept behind our proposed initialization method for GARPP. The steps for our multiple feasible path creation method are (1) gridize a given environment into a set of nodes, (2) create a DAG that connects the starting point to the destination point using a set of nodes, and (3) generate multiple paths based on the obtained DAG of the map. To illustrate our initialization method, we first instantiated the pre-processing step of the given environment in Fig. 1a; the environment is gridized and then each node, or cell, on the grid in the obtained map is numbered sequentially. In Fig. 1a, white cells represent obstacle-free nodes, gray cells represent obstacle nodes, and red cells represent the starting and destination nodes. (See Fig. 1 in the color-printed paper or PDF.)

Fig. 1b shows the proposed idea for creating a graph with multiple paths. The selected node and connected edges are illustrated on the left side of the figure; the selection sequence of each node and its connected nodes are represented in the table on the right. For example, node 20, selected at the third iteration, is reached from nodes 13 and 32 (selected in the first and second iterations, respectively). Fig. 1c shows a simple illustration of the graph obtained from Fig. 1b. The obtained graph includes a set of nodes that have multiple incoming edges on the graph, such as nodes 20 and 24. Table 2 demonstrates the multiple paths that can be generated from the graph in Fig. 1c. Fig. 1 and Table 2 show multiple paths that can be quickly obtained because the graph is created only once, and the back-tracking process for making paths does not consume significant computational resources.