Available online at www.sciencedirect.com

ScienceDirect

Applied Soft Computing 8 (2008) 1643–1653

Applied Soft Computing

www.elsevier.com/locate/asoc

Solving shortest path problem using particle swarm optimization

Ammar W. Mohemmed, Nirod Chandra Sahoo [*], Tan Kim Geok

Faculty of Engineering and Technology, Multimedia University, 75450 Melaka, Malaysia

Received 22 August 2006; received in revised form 25 December 2007; accepted 15 January 2008
Available online 19 January 2008

Abstract

This paper presents the investigations on the application of particle swarm optimization (PSO) to solve shortest path (SP) routing problems. A modified priority-based encoding incorporating a heuristic operator for reducing the possibility of loop-formation in the path construction process is proposed for particle representation in PSO. Simulation experiments have been carried out on different network topologies for networks consisting of 15–70 nodes. It is noted that the proposed PSO-based approach can find the optimal path with good success rates and also can find closer sub-optimal paths with high certainty for all the tested networks. It is observed that the performance of the proposed algorithm surpasses those of recently reported genetic algorithm based approaches for this problem.
© 2008 Elsevier B.V. All rights reserved.

Keywords: Shortest path problem; Particle swarm optimization; Path encoding

1. Introduction

The shortest path (SP) problem concerns with finding the shortest path from a specific origin to a specified destination in a given network while minimizing the total cost associated with the path. This problem has widespread applications. Some important applications of the SP problem include vehicle routing in transportation systems [1], traffic routing in communication networks [2] and path planning in robotic systems [3]. Furthermore, the shortest path problem also has numerous variations such as the minimum weight problem, the quickest path problem, and so on.

The SP problem has been investigated extensively. The well-known algorithms for solving this problem include the Bellman's dynamic programming algorithm for directed networks, the Dijkstra labeling algorithm and Bellman–Ford successive approximation algorithm for networks with non-negative cost coefficients. The details of these algorithms can be found in [4]. These traditional algorithms have major shortcomings; firstly, they are not suitable for networks with negative weights of the edges, i.e., in communication networks, the link weights represent the transmission line capacity and negative weights correspond to links with gain rather than loss. Secondly, the algorithms search only for the shortest route, but they cannot determine any other similar/non-similar short routes (which is commonly referred to as the *k*th SP problem). Thirdly, they exhibit high computational complexity for real-time communications involving rapidly changing network topologies such as wireless ad hoc networks. Therefore, new techniques have been continuously under investigation.

Artificial neural networks (ANN) have been examined to solve the SP problem relying on their parallel architecture to provide a fast solution [5–7]. However, the ANN approach has several limitations. These include the complexity of the hardware which increases considerably with increasing number of network nodes; at the same time, the reliability of the solution decreases. Secondly, they are less adaptable to topological changes in the network graph [7], including the cost of the arcs. Thirdly, the ANNs do not consider sub-optimal paths. Among other approaches for this problem, the powerful evolutionary programming techniques have considerable potential to be investigated in the pursuit for more efficient algorithms because the SP problem is basically an optimal search problem. In this direction, genetic algorithm (GA) has shown promising results [8–11]. The most recent notable results have been reported in [10]. Their algorithm shows better performance compared to those of ANN approach and overcomes the limitations mentioned above.

It is apparent that there is always a great need for more efficient optimization algorithms for the SP problem. Among the

[*] Corresponding author: Present address: Department of Electrical Engineering, Indian Institute of Technology, Kharagpur 721302, India.
Tel.: +91 3222 283052.
E-mail address: ncsahoo@ee.iitkgp.ernet.in (N.C. Sahoo).

1568-4946/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2008.01.002

notable algorithms for path finding optimization problems in network graphs, successful use of GA and Tabu Search (TS) has been reported [12–14]. The success of these evolutionary programming approaches promptly inspires investigative studies on the use of other similar (and possibly more powerful) evolutionary algorithms for this problem. Particle Swarm Optimization is one such evolutionary optimization technique [15], which can solve most of the problems solved by GA with less computation cost [16]. It is to be noted that GA and TS demand expensive computational cost. Some more comparative studies of the performances of GA and PSO have also been reported [17–20]. All these studies have firmly established similar effectiveness of PSO compared to GA. Even for some applications, it has been reported that the PSO performs better than other evolutionary optimization algorithms in terms of success rate and solution quality. The most attractive feature of PSO is that it requires less computational bookkeeping and, generally, a few lines of implementation codes. The basic philosophy and science behind PSO is based on the social behavior of a bird flock and a fish school etc. Because of the specific algorithmic structure of PSO (updating of position and velocity of particles in a continuous manner), PSO has been mainly applied to many continuous optimization problems with few attempts for combinatorial optimization problems. Some of the combinatorial optimization problems that have been successfully solved using PSO are: task assignment problem [21], traveling salesman problem [22,23], sequencing problem [24] and permutation optimization problem [25], etc.

To the best knowledge of the authors, there is no reported work on the use of PSO for solving the core shortest path problem without any use of the classical algorithms such as the Dijkstra and Bellman–Ford algorithms. The purpose of this paper is to investigate on the applicability and efficiency of PSO for this problem. In this regard, this paper reports the use of particle swarm optimization to solve the shortest path problem, where a modified indirect encoding is used to represent the particle (position). In addition, a novel heuristic operator has been used for reducing the possibility of loop formation during potential path constructions (search procedure) from an origin node to a specific destination node in the network graph. The proposed algorithm has been tested by exhaustive simulation experiments on various random network topologies. The analysis of the results indicates the superiority of the PSO-based approach over those using GA [10,11].

The paper is organized as follows. In Section 2, PSO paradigm is briefly discussed. The particle encoding mechanism is presented Section 3 followed by the overall flow of the PSO algorithm for solving the SP problem being provided in Section 4. The results from computer simulation experiments are discussed in Section 5. Section 6 concludes the paper.

2. Particle swarm optimization: a brief overview

Particle swarm optimization is a population based stochastic optimization technique inspired by the social behavior of bird flock (and fish school, etc.), as developed by Kennedy and Eberhart [15]. As a relatively new evolutionary paradigm, it has grown in the past decade and many studies related to PSO have been published [26]. The algorithmic flow in PSO starts with a population of particles whose positions, that represent the potential solutions for the studied problem, and velocities are randomly initialized in the search space. The search for optimal position (solution) is performed by updating the particle velocities, hence positions, in each iteration/generation in a specific manner as follows. In every iteration, the fitness of each particle's position is determined by some defined fitness measure and the velocity of each particle is updated by keeping track of two "*best*" positions. The first one is the best position (solution) a particle has traversed so far. This value is called *pBest*. Another "*best*" value is the best position (solution) that any neighbor of a particle has traversed so far. This best value is a neighborhood best and is called *nBest*. When a particle takes the whole population as its neighborhood, the neighborhood best becomes the global best and is accordingly called *gBest*. A particle's velocity and position are updated as follows.

$$v_{id} = v_{id} + c_1 r_1 (b_{id} - x_{id}) + c_2 r_2 (b^n_{id} - x_{id});\; i = 1, 2, \ldots, N_s \quad \text{and} \quad d = 1, 2, \ldots, D \tag{1}$$

$$x_{id} = x_{id} + v_{id} \tag{2}$$

where c_1 and c_2 are positive constants, called *acceleration coefficients*, N_s is the total number of particles in the swarm, D is the dimension of problem search space, i.e., number of parameters of the function being optimized, r_1 and r_2 are two independently generated random numbers in the range [0,1] and "n" represents the index of the best particle in the neighborhood of a particle. The other vectors are defined as: $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ is the position of ith particle; $\mathbf{v}_i = [v_{i1}, v_{i2}, \ldots, v_{iD}]$ is the velocity of ith particle; $\mathbf{b}_i = [b_{i1}, b_{i2}, \ldots, b_{iD}]$ is the best position of the ith particle ($pBest_i$), and $\mathbf{b}^n_i = [b^n_{i1}, b^n_{i2}, \ldots, b^n_{iD}]$ is the best position found by the neighborhood of the particle i ($nBest_i$). The pseudo-codes for general algorithmic flow of PSO are listed in Fig. 1.

Eq. (1) calculates a new velocity for each particle based on its previous velocity, the particle's position at which the best possible fitness has been achieved so far, and the neighbors' best position achieved. Eq. (2) updates each particle's position in the solution hyperspace.c_1 and c_2 are two learning factors, which control the influence of *pBest* and *nBest* on the search process. In all initial studies of PSO, both c_1 and c_2 are taken to be 2.0 yielding good results [15]. However, in most cases, the velocities quickly attain very large values, especially for particles far from their global best. As a result, particles have larger position updates with particles leaving boundary of the search space. To control the increase in velocity, velocity clamping is used in Eq. (1). Thus, if the right side of Eq. (1) exceeds a specified maximum value $V^{\max}_d$, then the velocity on that dimension is clamped to $V^{\max}_d$. Many improvements have been incorporated into this basic algorithm. A review of these modifications can be seen in [27].

The commonly used PSOs are either global version or local version of PSO. In global version, all other particles influence the velocity of a particle, while in the local version of PSO, selected

Download English Version:

https://daneshyari.com/en/article/497158

Download Persian Version:

https://daneshyari.com/article/497158

Daneshyari.com