Contents lists available at ScienceDirect

Information Sciences

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

A discrete gravitational search algorithm for solving combinatorial optimization problems

Mohammad Bagher Dowlatshahi [a], Hossein Nezamabadi-pour [b,*], Mashaallah Mashinchi [c]

[a] Department of Computer Science, Shahid Bahonar University of Kerman, Kerman, Iran
[b] Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran
[c] Department of Statistics, Shahid Bahonar University of Kerman, Kerman, Iran

A R T I C L E I N F O

Article history:
Received 22 November 2011
Received in revised form 11 May 2013
Accepted 13 September 2013
Available online 25 September 2013

Keywords:
Metaheuristics
Gravitational search algorithm
Combinatorial optimization
Traveling salesman problem
Path re-linking

A B S T R A C T

Metaheuristics are general search strategies that, at the exploitation stage, intensively exploit areas of the solution space with high quality solutions and, at the exploration stage, move to unexplored areas of the solution space when necessary. The Gravitational Search Algorithm (GSA) is a stochastic population-based metaheuristic that was originally designed for solving continuous optimization problems. It has a flexible and well-balanced mechanism for enhancing exploration and exploitation abilities. In this paper, a Discrete Gravitational Search Algorithm (DGSA) is proposed to solve combinatorial optimization problems. The proposed DGSA uses a Path Re-linking (PR) strategy instead of the classic way in which the agents of GSA usually move from their current position to the position of other agents. The proposed algorithm was tested on a set of 54 Euclidean benchmark instances of TSP with sizes ranging from 51 to 2392 nodes. The results were satisfactory and in the majority of the instances, the results were equal to the best known solution. The proposed algorithm ranked ninth when compared with 54 different algorithms with regard to quality of the solution.

1. Introduction

Combinatorial optimization problems arise in many areas of computer science and other disciplines such as artificial intelligence, operations research, bioinformatics and electronic commerce, and typically involve finding the best possible grouping, ordering or assignment of a finite set of objects satisfying certain conditions or constraints. Algorithms for solving these problems may be classified as either "complete" or "approximate" [7]. Complete algorithms obtain optimal solutions and guarantee their optimality in a bounded time for every finite size instance of a combinatorial optimization problem. However, it has been shown that for combinatorial optimization problems that are NP-hard, no polynomial time algorithm exists unless P = NP [12,29]. Therefore, complete algorithms for combinatorial optimization problems often need exponential computation time and this makes them impractical for most real-world applications. Thus, approximate algorithms to solve NP-hard combinatorial optimization problems have been receiving increasing attention. In approximate methods, the guarantee of finding exact optimal solutions is sacrificed for the sake of obtaining good solutions in a significantly reduced amount of time [7,42].

Approximate algorithms can be categorized into three important classes: approximation algorithms, problem specific heuristics, and metaheuristics. Approximation algorithms, unlike problem specific heuristics and metaheuristics, provide provable solution quality and provable run-time bounds for finding a non-optimal solution. Problem specific heuristics

* Corresponding author. Tel./fax: +98 341 3235900.
E-mail address: nezam@mail.uk.ac.ir (H. Nezamabadi-pour).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.09.034

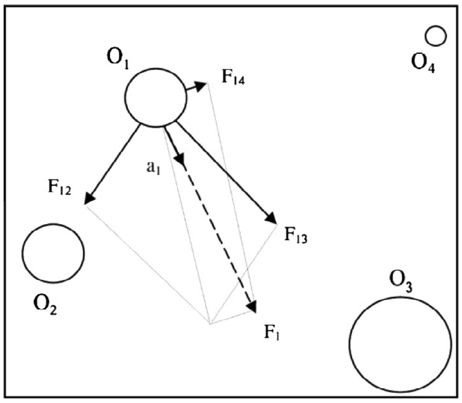

Fig. 1. Every object accelerates in the direction of the resultant force that acts on it from the other objects.

are problem dependent and are designed for a particular problem whereas metaheuristics represent more general approximate algorithms and are applicable to a large variety of optimization problems. Metaheuristics solve problems that are believed to be hard by exploring the large solution space and achieve this goal by effectively reducing the size of this space and exploiting the reduced space efficiently [7,14,42]. This class of algorithms includes, but is not restricted to, Ant Colony Optimization (ACO) [10], Evolutionary Computation (EC) such as Genetic Algorithms (GA) [21,22,43], Greedy Randomized Adaptive Search Procedure (GRASP) [11], Iterated Local Search (ILS) [32,40], Variable Neighborhood Search (VNS) [33], Simulated Annealing (SA) [27], and Tabu Search (TS) [17].

Recently, a stochastic population-based metaheuristic, called Gravitational Search Algorithm (GSA), which is motivated by the laws of gravity and motion has been proposed. Originally, GSA was designed for solving continuous optimization problems [36] and, like most metaheuristics, has a flexible and well-balanced mechanism for enhancing exploration and exploitation abilities. The search strategy of GSA is to move the members of population towards the K best solutions of population using the laws of gravity and motion, where K is a number between one and the size of the population. Motivated by the success of the GSA with variant optimization problems [5,18,19,35,37,39,44], this paper proposes a Discrete GSA (DGSA) to solve combinatorial optimization problems. As a test, this algorithm was applied to the Traveling Salesman Problem (TSP). The results were satisfactory and for the majority of the instances the results were equal to the best known solution.

The rest of the paper is organized as follows. Section 2 provides a review of original GSA. In Section 3, the main concepts and components behind GSA to accomplish search is investigated. In Section 4, details of the proposed DGSA are described. An implementation of the proposed algorithm for solving TSP is presented in Section 5. Computational results are presented in Section 6. Finally, in Section 7 conclusions and suggestions for future research are given.

2. Gravitational search algorithm

In physics, gravitation is the tendency of objects with mass to accelerate towards each other. In the Newton gravitational law, each object attracts every other object by a gravitational force. As an example, consider a 2-dimensional space including the objects O_1, O_2, O_3, and O_4. As seen in Fig. 1, F_{1j} ($j \in \{2,3,4\}$) is the force acting on O_1 from O_j ($j \in \{2,3,4\}$), and F_1 is the overall force that acts on O_1 from all other objects and generates acceleration a_1 based on Newton's second law.[1]

In the basic model of the GSA, which has been originally designed to solve continuous optimization problems, a set of objects (agents) are introduced in the n-dimensional solution space of the problem to find the optimum solution by simulation of the Newtonian laws of gravity and motion. The position of each agent in GSA demonstrates a candidate solution to the problem, hence each agent is represented by the vector X_i in the solution space of the problem. Agents with a higher performance get a greater gravitational mass, because a heavy agent has a large effective attraction radius and hence a great intensity of the attraction. During the lifetime of GSA, each agent successively adjusts its position X_i toward the positions of K best agents of the population.

To describe GSA in more detail, consider an n-dimensional space with s searcher agents in which the position of the ith agent is defined as follows:

$$X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^n); \quad i = 1, 2, \ldots, s, \tag{1}$$

where x_i^d presents the position of the ith agent in the dth dimension. Based on [36], gravitational mass of the ith agent is calculated after computing the current population's fitness as follows:

[1] Newton's second law states that when a force is applied to an object, the acceleration of this object depends only on the force and gravitational mass of this object. For example, suppose O_i is an object, F_i is a force which acts on O_i, and a_i is the acceleration of O_i. The value of a_i will be obtained as follows: $a_i = \frac{F_i}{\text{gravitational mass of } O_i}$.