Applied Soft Computing 11 (2011) 1983–1992

Contents lists available at ScienceDirect

Applied Soft Computing

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

A new evolutionary algorithm using shadow price guided operators

Gang Shen*, Yan-Qing Zhang

Department of Computer Science, Georgia State University, P.O. Box 3994, Atlanta, GA 30302-3994, USA

ARTICLE INFO

Article history:
Received 29 April 2009
Received in revised form 25 January 2010
Accepted 19 June 2010
Available online 25 June 2010

Keywords:
Genetic algorithm
Shadow price
Optimization
Traveling salesman problem

ABSTRACT

The genetic algorithm (GA) is a popular global search algorithm. It has been used successfully in many fields, however, it is still challenging for the GA to obtain optimal solutions for complex problems. Another problem is that the GA can take a very long time to solve difficult problems. This paper proposes a new evolutionary algorithm that uses the fitness value to measure overall solutions and shadow prices to evaluate components. New shadow price guided operators are used to achieve good measurable evolutions. The new algorithm is used first to solve a simple optimization function and then applied to the complex traveling salesman problem (TSP). Simulation results have shown that the new shadow price guided evolutionary algorithm is effective in terms of performance and efficient in terms of speed.

1. Introduction

Since Holland formally introduced the genetic algorithm in the early 1970s [15,16], it has been widely applied to many complex applications. Koza et al's survey [24] suggested that genetic programming produced human-competitive results in at least 36 fields.

GA, a branch of evolutionary algorithm (EA), mimics the evolution processes. Major traditional operators include selection, mutation, and crossover. Through the years, various new GA operators have been proposed, such as order crossover [29], partially mapped crossover [13], edge assembly crossover [27], and heuristic crossover operator [40]. These operators are very efficient in preserving the current solution information.

Different approaches have been used to improve the GA's efficiency. For example, using the nearest neighbor algorithm to generate the initial population to jump-start the algorithm [21], inserting local searches into the GA to achieve efficiency improvement [30,38], adding vaccine and immunity concepts to the process [20], injecting artificial chromosomes based on a dominance matrix [4], and adaptively controlling the algorithm parameters [1].

Most of the algorithm improvements focus on either tilting the search to some directions to preserve current information or making the next move based on a combination of historical information (pheromone trail) and current solution information. They all heavily rely on the fitness function, which can only describe the overall solution state. In this paper, we are proposing to exam the components of the solution and make decisions based on states of the components and the solution.

The traveling salesman problem (TSP) is a classic NP hard combinatorial problem. It has been routinely used as a benchmark to verify new algorithms. There are two major categories of algorithms used to solve the problem, exact or approximate algorithms. Exact algorithms, such as testing all permutations or branch and bound, typically either take very long time to compute or reach unsatisfied results. There are a lot approximate algorithms that achieve good results. In the Bio inspired algorithms area, genetic algorithm (GA) [5,21,28], ant colony optimization (ACO) [3,7,8,10,11,19], neural network (NN) [2,14,17,18,32], discrete particle swarm optimization (DPSO) [6,33,34,41,42], bee colony optimization (BCO) [37], simulated annealing (SA) [22], collective intelligence [25], and hybrid algorithms [26,39] have been used to solve the TSP and have achieved good results. We also use the TSP to validate our proposed algorithm and compare results with several of above-mentioned algorithms.

In the following sections, we start with the discussion of the fitness function, introducing the concept and the usage of the shadow price, followed by our proposal of applying the shadow price concept to EA. We demonstrate our new algorithm by solving a simple mathematic problem. Finally, we solve the TSP with our algorithm and compare results with other published results.

2. Shadow price based GA operators

2.1. Limitations of fitness function

In evolutionary based algorithms, a fitness function is used to assign a value to a solution. This value is used to compare and eval-

* Corresponding author.
E-mail address: gshen1@student.gsu.edu (G. Shen).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.06.014

uate solutions. It is the function to be minimized or maximized by the algorithm. For example, the total trip distance is the natural choice of the fitness function for the traveling salesman problem. The fitness function is the same as the objective function in the linear programming field.

Since the introduction of EA, most improvements focus on preserving existing information. Classic crossover operators concentrate on preserving relationships among components. Local optimization emphasizes on selecting good neighboring components.

Evolutionary operators work on the components of the solution(s). The method of component(s) selection directly impacts the algorithm's performance and the result. Since the fitness function can only evaluate the overall solution, traditional EA does not provide a way to intelligently select components. We believe it will be advantageous to directly compare components, their relationships, and contributions toward a better solution. Our research cannot locate any direct attempt to address the challenge.

Without a unified means to evaluate components, it would be difficult for EA operators to select high potential component(s) to generate better solutions. At the macrolevel, solutions can be compared based on their fitness values. At the microlevel, components of the solution need to be compared for EA operators' usage. It is necessary to design a systematic method to measure components.

2.2. Duality and shadow price in linear programming

Dantzig [9] stated "The linear programming model needs an approach to finding a solution to a group of simultaneous linear equations and linear inequalities that minimize a linear form." Linear programming is the algorithm to search for an optimal value for a linear objective function that satisfies linear equations and linear inequalities.

Kolman and Beck [23] defined the standard form for linear programming as:

For values of $x_1, x_2, \ldots, x_n$ which will maximize:

$$z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

Subject to the constraints:

$$\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le b_1\\
&a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le b_2\\
&\vdots\\
&a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{m1n}x_n \le b_m\\
&\quad x_j \ge 0, j = 1, 2, \ldots, n
\end{aligned}$$

More conveniently, we can use a matrix notation. Let

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n}\\ a_{21} & a_{22} & \cdots & a_{2n}\\ \vdots & \vdots & & \vdots\\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad b = \begin{bmatrix} b_1\\ b_2\\ \vdots\\ b_m \end{bmatrix}, \quad x = \begin{bmatrix} x_1\\ x_2\\ \vdots\\ x_n \end{bmatrix},$$

$$c = \begin{bmatrix} c_1\\ c_2\\ \vdots\\ c_n \end{bmatrix}$$

A linear programming standard form can be rewritten as:

$$\begin{array}{ll} \text{Maximize} & z = c^Tx\\ \text{Subject to} & Ax \le b\\ & x \ge 0 \end{array}$$

The duality theorem states that there is an equivalent linear programming problem for every linear programming problem. One is called the primal problem and the other is called the dual problem. Dantzig [9] proved the duality theorem. The dual problem for the above standard form is given below.

For values of $y_1, y_2, \ldots, y_m$ which will minimize:

$$z' = b_1y_1 + b_2y_2 + \cdots + b_my_m$$

Subject to the constraints:

$$\begin{aligned}
&a_{11}y_1 + a_{21}y_2 + \cdots + a_{m1}y_m \ge c_1\\
&a_{12}y_1 + a_{22}y_2 + \cdots + a_{m2}y_m \ge c_2\\
&\vdots\\
&a_{1n}y_1 + a_{2n}y_2 + \cdots + a_{mn}y_m \ge c_n\\
&\quad y_j \ge 0, j = 1, 2, \ldots, m
\end{aligned}$$

The matrix representation is

$$\begin{array}{ll} \text{Minimize} & z' = b^Ty\\ \text{Subject to} & A^Ty \ge c\\ & y \ge 0 \end{array}$$

$$\text{where } y = \begin{bmatrix} y_1\\ y_2\\ \vdots\\ y_m \end{bmatrix}.$$

The duality theorem also states that if the primal problem has an optimal solution (x_0) and the dual problem has an optimal solution (y_0), then

$$z = c^Tx_0 = z' = b^Ty_0$$

So, solving one linear programming problem is equivalent to solving its dual problem. Kolman et al. [23] described the shadow prices as:

The jth constraint of the dual problem is

$$\sum_{i=1}^{m} a_{ij}y_i \ge c_j$$

The coefficient a_{ij} represents the amount of input i per unit of output j, and the right-hand side is the value per unit of output j. This means that the units of the dual variable y_i are the "value per unit of input i"; the dual variables act as prices, costs, or values of one unit of each of the inputs. They are referred as dual prices, fictitious prices, shadow prices, etc.

The shadow price is the contribution to the objective function that can be made by relaxing a constraint by one unit. Different constraints have different shadow prices, and every constraint has a shadow price. Each constraint's shadow price changes along with the algorithm's progress of searching for the optimal solution.

2.3. Using shadow prices in linear programming

In the operations research field, linear programming has been used widely in various industrial fields. With a concrete mathematical model, it provides direct relationships among profit and constraints, output and constraints, other goals and constraints, etc. It is a very efficient tool for operations management. The linear models can be solved efficiently. Dantzig's [9] simplex method is one of them.

As a strict mathematical model, linear programming requires all constraints and all possible activities that meet the constraints listed in the tabular format. This is not a problem when the number of possible activities is small, such as maximizing profit for a small manufacturer. Constraints are material and labor. The objective function is defined as profit. It is rather straightforward to define the linear constraints, construct the linear objective function and search for optimal solutions for this category of problems.