Swarm and Evolutionary Computation Base Data xxx (2017) 1–18

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation Base Data

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

Three pseudo-utility ratio-inspired particle swarm optimization with local search for multidimensional knapsack problem

Mingchang Chih

Department of Business Administration, National Chung Hsing University, South District, Taichung 402, Taiwan, ROC

ARTICLE INFO

Keywords:
Self-adaptive check and repair operator
Particle swarm optimization
Neighborhood local search
Multidimensional knapsack problem
Taguchi method

ABSTRACT

In this study, a three-ratio self-adaptive check and repair operator-inspired particle swarm optimization (3R-SACRO-PSO) with neighborhood local search is developed to solve the multidimensional knapsack problem (MKP). The proposed 3R-SACRO-PSO systematically alters substitute pseudo-utility ratios as the PSO method is executed. In addition, a local search scheme is introduced to improve solution quality. The proposed 3R-SACRO-PSO algorithm is tested using 168 different widely used benchmarks from the OR-Library to demonstrate and validate its performance. The control parameters for the performance test are determined through the Taguchi method. Experimental results parallel those of other PSO algorithms, and statistical test results show that the quality and efficiency of the proposed 3R-SACRO are better than those of the two-ratio SACRO method. Moreover, the proposed 3R-SACRO-PSO is on par with state-of-the-art PSO approaches. Thus, introducing the third pseudo-utility ratio into SACRO improves the performance of SACRO-based PSO. The neighborhood local search scheme further improves the solution quality in handling MKPs.

1. Introduction

The multidimensional knapsack problem (MKP) is a generalized model of the standard knapsack problem and is an intensively studied discrete NP-hard problem. MKPs are found in various applications, and several real-world problems are generally modeled as MKPs [1]. Mathematically, an m-dimensional knapsack problem with n items can be formulated as

$$\max \quad z = \sum_{j=1}^{n} c_j y_j, \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_{ij} y_j \le b_i, \quad i = 1, 2, \cdots, m, \tag{2}$$

$$y_j \in \{0, 1\}, \quad j = 1, 2, \cdots, n, \tag{3}$$

where each knapsack constraint i has the capacity b_i for $i = 1, 2, \ldots, m$. Each item j consumes a_{ij} units of resource in the ith constraint and yields c_j units of profit when it is selected. The goal of the mathematical programming for MKP is to seek a subset of items to maximize profit without violating the constrained knapsack capacities.

The MKP is a classic constrained combinatorial optimization problem [2] that has been intensively studied by researchers. Numerous solutions to MKPs have been recently proposed; these solutions include exact and heuristic algorithms. Exact approaches are not functional for MKPs because the search region of potential solutions exponentially grows as the problem size increases. Therefore, many soft computing algorithms, such as simulated annealing approach (SA) [3], Tabu search technique (TS) [4], genetic algorithm (GA) [5], ant colony optimization algorithm (ACO) [6], artificial immune system (AIS) [7], monarch butterfly optimization (MBO) [8,9], firefly algorithm (FA) [10], cuckoo search algorithm (CS) [11], and particle swarm optimization (PSO) [12,13], have been developed to solve knapsack problems.

PSO was initially designed by Kennedy and Eberhart [14] for

E-mail address: mcchih@dragon.nchu.edu.tw.

https://doi.org/10.1016/j.swevo.2017.10.008
Received 26 July 2017; Received in revised form 13 September 2017; Accepted 25 October 2017
Available online xxxx

considering continuous nonlinear optimization. However, many programming problems occur in a space that features binary or discrete decision variables. Kennedy and Eberhart [15] further designed a discrete version of the swarm intelligence method to solve these problems. To date, PSO has been successfully applied in several areas, such as pattern clustering [16], crew scheduling problems [17], multi-robot path planning [18], quality control [19], network reliability [20], inventory routing problems [21], time series forecasting [22], constrained shortest path problems [23], layer-packing problems [24], cost-sensitive attribute reduction [25], and MKPs [26].

In this work, a self-adaptive repair operator [27] concept with three pseudo-utility ratios and a local search neighborhood mechanism are introduced, and a novel PSO is proposed. The introduction of the third pseudo-utility ratio into the self-adaptive check and repair operator (SACRO) is inspired by the fact that the three different measures of pseudo-utility ratios must produce different ratio rankings and lead to various repairing results, thereby encouraging particles in PSO to explore different solution regions. The proposed PSO algorithm is paralleled with state-of-the-art PSO methods using the widely used MKP benchmarks from the OR-Library [28].

This paper is structured as follows. The background of the PSO algorithm is introduced in Section 2. The proposed three pseudo-utility ratios self-adaptive repair mechanism, the local search neighborhood mechanism, and the novel 3R-SACRO-PSO algorithm are presented in Sections 3 and 4. A comparative study on the performances of the proposed method based on test benchmarks is discussed in Section 5. The conclusion is given in Section 6.

2. Background

2.1. Classic PSO

PSO is a population-based stochastic optimization technique that is inspired by the metaphor of social interaction and communication, such as the flocking of birds. Each particle represents a potential candidate solution to the programming model. In PSO, local and global searches are combined to acquire high exploration efficiency. Initially, particles are randomly located in the programming search region. In every iteration, each particle moves by considering not only its best position but also the best position in the swarm population. The first "best" value, designated *pbest*, is the best result found by the particle. The other "best" value, designated *gbest*, is utilized in PSO as the present best value achieved by any particle in the whole swarm. After both "best" values are determined, the velocity and position of particles are updated on the bases of Eqs. (4) and (5).

$$v_{ij}^{t} = w \cdot v_{ij}^{t-1} + c_1 \cdot Rand() \cdot \left(p_j^{t-1} - x_{ij}^{t-1}\right) + c_2 \cdot Rand() \cdot \left(g_j^{t-1} - x_{ij}^{t-1}\right), \quad (4)$$

$$\text{and } x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t}, \quad (5)$$

where v_{ij}^{t} and x_{ij}^{t} are the velocity and position of the ith particle at the tth iteration; p_j^{t-1} and g_j^{t-1} are the *pbest* and *gbest*, respectively; *Rand*() is a pseudo random number between [0, 1]; c_1 and c_2 are cognition learning factor and social learning factor, respectively; and w is the inertia weight.

Shi and Eberhart [29] significantly improved the performance of PSO using a linearly time-varying inertia weight. Their simulation and evaluation results illustrated that the proposed PSO converges quickly, and its performance is more robust and insensitive to the population size. The mathematical representation of their proposed method is expressed as

$$w = (w_{\max} - w_{\min}) \frac{t_{\max} - t}{t_{\max}} + w_{\min}. \quad (6)$$

In addition to the time-varying inertia weight, Ratnweera et al. [30] introduced time-varying acceleration coefficients (TVAC) in PSO for local search navigation and convergence toward the global optima. The simulation results demonstrated that TVACs improve global exploration during the early part of search and boost the solutions to converge to the global best position at the end of the optimization. TVACs can be defined and formulated as

$$c_1 = (c_{1f} - c_{1i}) \frac{t-1}{t_{\max}} + c_{1i}, \quad (7)$$

$$c_2 = (c_{2f} - c_{2i}) \frac{t-1}{t_{\max}} + c_{2i}. \quad (8)$$

To solve MKPs, Chih et al. [26] introduced time-varying inertia weight and learning factors driven by uniform random number and chaotic logistic map, respectively, into their proposed PSO algorithms. The composite velocity updating function can be formulated as

$$\begin{aligned} v_{ij}^{t} = w \cdot v_{ij}^{t-1} &+ \left\{ (c_{1f} - c_{1i}) \frac{t-1}{t_{\max}} + c_{1i} \right\} \cdot Rand() \cdot \left(p_j^{t-1} - x_{ij}^{t-1}\right) \\ &+ \left\{ (c_{2f} - c_{2i}) \frac{t-1}{t_{\max}} + c_{2i} \right\} \cdot Rand() \cdot \left(g_j^{t-1} - x_{ij}^{t-1}\right). \end{aligned} \quad (9)$$

2.2. Binary PSO

PSO was initially developed to optimize continuous functions where velocity and position are formulated as real-valued numbers. Hence, the major drawback of utilizing a PSO in real-world applications is its continuous nature. That is, it cannot cope with a binary constrained optimization problem, such as MKPs. Consequently, Kennedy and Eberhart [15] designed the first discrete version of PSO, called BPSO, to resolve its problem with binary variables. In BPSO, a particle is defined by binary formulation, and velocity is formulated in terms of the probability that a binary decision variable will take a value of one. A sigmoid function, which is shown in Eq. (10), is then utilized by BPSO to transform all real-valued velocities to the interval [0, 1].

$$S\left(v_{ij}^{t}\right) = \frac{1}{1 + \exp\left(-v_{ij}^{t}\right)}, \quad (10)$$

where $S(v_{ij}^{t})$ denotes the probability of bit v_{ij}^{t} taking the value of 1.

Given that the disadvantage observed with the sigmoid function is the non-monotonic curve of the transforming probability density function of a bit, the position update equation has been recently employed in the BPSO [26,31]. The conception of the position update function is straightforwardly obtained from the equation of PSO for continuous position updating. If the lower and upper bounds for velocity are $-V_{\max}$ and $V_{\max}$, then the value of $x_{ij}^{t-1} + v_{ij}^{t}$ is limited within $(0 - V_{\max} = -V_{\max})$ and $(1 + V_{\max})$ because x_{ij}^{t} in Eq. (5) must be a binary value of 0 or 1. Accordingly, the position update function can be expressed as

$$x_{ij}^{t} = \begin{cases} 1, & \text{if } U(-V_{\max}, 1 + V_{\max}) < x_{ij}^{t-1} + v_{ij}^{t}; \\ 0 & \text{otherwise.} \end{cases} \quad (11)$$

Given that $U(0,1) = \frac{U(-V_{\max}, 1+V_{\max}) + V_{\max}}{(1+2V_{\max})}$; thus, Eq. (11) can be rewritten as

$$x_{ij}^{t} = \begin{cases} 1, & \text{if } U(0,1) < \dfrac{x_{ij}^{t-1} + v_{ij}^{t} + V_{\max}}{1 + 2V_{\max}} \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

where $U(0,1)$ is a uniform variable within [0, 1], and $U(-V_{\max}, 1 + V_{\max})$ is a uniform variable within $[-V_{\max}, 1 + V_{\max}]$.

2.3. Check and repair operator for MKP

The MKP is a constrained combinatorial optimization problem. Thus,