Applied Soft Computing 38 (2016) 817–830

Contents lists available at ScienceDirect

Applied Soft Computing

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

An improved monkey algorithm for a 0-1 knapsack problem

Yongquan Zhou [a,*], Xin Chen [a,b], Guo Zhou [c]

[a] College of Information Science and Engineering, Guangxi University for Nationalities, Nanning, Guangxi 530006, China
[b] School of Software, Dalian University of Technology, Dalian 116024, China
[c] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100081, China

ARTICLE INFO

Article history:
Received 12 August 2013
Received in revised form 26 August 2015
Accepted 18 October 2015
Available online 28 October 2015

Keywords:
Monkey algorithm
Binary version of the monkey algorithm
Knapsack problem
Cooperation process
Greedy strategy

ABSTRACT

The 0-1 knapsack problem is a classic combinational optimization problem. However, many exiting algorithms have low precision and easily fall into local optimal solutions to solve the 0-1 knapsack problem. In order to overcome these problems, this paper proposes a binary version of the monkey algorithm where the greedy algorithm is used to strengthen the local search ability, the somersault process is modified to avoid falling into local optimal solutions, and the cooperation process is adopted to speed up the convergence rate of the algorithm. To validate the efficiency of the proposed algorithm, experiments are carried out with various data instances of 0-1 knapsack problems and the results are compared with those of five metaheuristic algorithms.

1. Introduction

Knapsack problems (KPs) were first proposed by Dantzig in the 1950s [1] and subsequently attracted many scholars. Zou et al. described a novel global harmony search (HS) algorithm that includes two important operations, position updating and genetic mutation, for solving 0-1 knapsack problems [2]. The experiment shows the proposed algorithm has good performance for solving large-scale knapsack problems. Xiang et al. also considered a novel global-best harmony search algorithm that combines a two-phase repair operator to repair an infeasible harmony vector and to further improve a feasible solution [3]. Changdar et al. considered an improved ant colony optimization (ACO) approach to solve 0-1 knapsack problems in a fuzzy environment [4]. The proposed algorithm creates n candidate groups for n objects; each ant selects a candidate value from each group. Bansal et al. proposed a modified particle swarm optimization called MBPSO based on binary particle swarm optimization (BPSO) [5] for solving 0-1 knapsack problems and multidimensional knapsack problems [6]. Azad et al. proposed a simplified binary version of the artificial fish swarm algorithm (AFSA) for solving 0-1 quadratic knapsack problems [7]; the random heuristic drop_item procedure is used to make the points feasible, and the heuristic add_item is also implemented to improve the quality of the solutions. Glover used an improved greedy algorithm and surrogate constraints for linear and quadratic knapsack problems [8]. Sitarz studied the relations between multiple criteria dynamic programming (MCDP) and the multiple knapsack problem [9]; the paper also showed how to use MCDP methods to solve multiple knapsack problems. Gao et al. presented a novel quantum-inspired artificial immune system called MOQAIS, which is composed of a quantum-inspired artificial immune algorithm (QAIS) and an artificial immune system (BAIS) for multiobjective knapsack problems [10]. QAIS is responsible for exploration of the search space, and BAIS is applied for exploitation of the search space. García-Martínez proposed a tabu-enhanced destruction mechanism for an iterated greedy search in studying quadratic multiple knapsack problems [11]; the method records the last removed objects and avoids removing them again in subsequent iterations. Baykasoğlu et al. used a priority-based encoding technique for a firefly algorithm (FA) to construct feasible solutions and prevent infeasibility for solving dynamic multidimensional knapsack problems [12]. Hifi et al. used a dichotomous search-based exact method based on decomposing the original problem into a series of knapsack problems and introduced new upper bounds and incremental valid lower bounds in the interval search [13]. Levin et al. studied the stochastic behavior of the knapsack problem and defined a variant problem in which item values are deterministic and item sizes are independent random variables [14]. Zhao used a nonlinear reductive dimension approximate algorithm for the knapsack problem [15]. Liu et al. removed useless solution regions before applying simulated annealing (SA) to solve a knapsack problem and extracted the most possible part of

* Corresponding author. Tel.: +86 13607882594.
E-mail address: yongquanzhou@126.com (Y. Zhou).

http://dx.doi.org/10.1016/j.asoc.2015.10.043
1568-4946/

the optimal solution space from the whole optimal solution space [16]. Saraç et al. used a genetic algorithm (GA) and a hybrid solution approach for solving a developed model called the generalized quadratic multiple knapsack problems (G-QMKP) [17]. The experiment shows the proposed hybrid solution approach can obtain good solutions in a reasonable time for a large-scale problem. Lin investigated using GA in solving the fuzzy knapsack problem [18]; this method simulates a fuzzy number by distributing it into partition points and uses GA to evolve the values in each partition point.

The monkey algorithm (MA) is a new type of swarm intelligence based algorithm. It was proposed by Zhao and Tang in 2008 and is derived from simulation of the mountain-climbing processes of monkeys [19]. It consists of three processes: the climb process, watch–jump process and somersault process. The climb process is designed to gradually improve the objective function value. However, MA will spend considerable computing time searching for local optimal solutions in the climb process. To reduce the computing time and speed up the convergence rate, Chen and Zhou introduced the inertial step in the climb process and combined the simple method after the somersault process [20]. The watch–jump process can speed up the convergence rate of the algorithm, the purpose of the somersault process is to make monkeys find new search domains to avoid falling into local search. The algorithm has the advantages of simple structure, strong robustness, and not easy falling into local optimal solutions. Therefore, MA has been successfully applied in solutions to various optimization problems, such as transmission network expansion planning [21], intrusion detection technology [22], optimal sensor placement in structural health monitoring [23], the optimization of gas filling station project scheduling problem [24], the clustering analysis problem [25], etc. In this paper, the 0-1 knapsack problem will be studied, and a binary version of the monkey algorithm which combines cooperation process and greedy strategy (CGMA) is proposed. The algorithm improves the calculation accuracy and increases the convergence speed of the algorithm to a certain degree. The numerical experiment results show that the proposed algorithm has good performance in solving the 0-1 knapsack problem. It can be an efficient alternative for solving the 0-1 knapsack problem.

2. The 0-1 knapsack problem

The 0-1 knapsack problem is a typical NP-hard problem in operations research [26]. The problem is defined as follows:

Given a set of items $O = \{o_1, o_2, \ldots, o_n\}$, each with a weight w_i and a value p_i, determine the number of each item to include in a collection so that the total weight WX is less than or equal to a given limit and the total value PX is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. Its mathematical model is as follows:

$$\begin{cases} \text{maximize } f(x_1, x_2, \ldots, x_n) = PX = \sum_{i=1}^{n} p_i x_i \\ \text{subject to } WX = \sum_{i=1}^{n} w_i x_i \le V \end{cases} \quad (1)$$

and $x_j \in \{0, 1\}$, $j = 1, 2, \ldots, n$. Where $P = (p_1, p_2, \ldots, p_n)$, $W = (w_1, w_2, \ldots, w_n)$ represent the value vector and weight vector of all items. V is the maximum capacity of the knapsack. $x_i = 1$ indicates that item i is included in the knapsack and $x_i = 0$ that it is not.

3. Description of modified monkey algorithm

The monkey algorithm was first proposed to solve numerical optimization problems as a new swarm intelligence based algorithm stemmed from the mountain-climbing behavior of monkeys [19]. Assume that there are many mountains in a given field. At the beginning, the monkeys climb up from their respective positions to find the mountaintops (this action is called climb process). When a monkey get the top of its mountain, it will find a higher mountain within the sight and jump somewhere of the mountain from the current position (this action is called watch–jump process), then repeat the climb process. After repetitions of the climb process and the watch–jump process, each monkey will somersault to a new search domain to find a much higher mountaintop (this action is called somersault process).

This paper proposed a binary version of the monkey algorithm where the greedy algorithm is used to correct the infeasible solutions and to improve the quality of the feasibility, the somersault process is modified to avoid falling into local search, the cooperation process is implemented to speed up the convergence rate, and the control parameter is used to keep the population diversity. The algorithm consists of 5 parts, the climb process, watch–jump process, greedy strategy repair process, cooperation process and somersault process.

3.1. Coding method

For the 0-1 knapsack problem, each item has two different status, namely the item has been included in the knapsack or it remains out. First, M is defined as the population size of monkeys. For monkey i, its position is denoted as a vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, and this position will be employed to express a solution of the 0-1 knapsack problem, where $x_{ij} \in \{0, 1\}$ and $j = 1, 2, \ldots, n$, n is the number of the items. $x_{ij} = 1$ indicates the item j is included in the knapsack and $x_{ij} = 0$ indicates it is not.

3.2. Initial population

In CGMA, the initial population is randomly generated. The random initialization process of M monkeys and n items is as follows:

```
for i = 1 to M do
  for j = 1 to N do
    x[i][j] = rand( ) ;
    If x[i][j] < 0.5
      x[i][j] = 0 ;
    else
      x[i][j] = 1 ;
        endif
      endfor
endfor
```

where $x_{i,j}$ represents the jth component in the vector X_i.

3.3. Climb process

According to the idea of pseudo-gradient-based simultaneous perturbation stochastic approximation (SPSA) [27], the climb process is a step-by-step procedure to improve the objective function by choosing a better one between two positions that are generated around the current position. For the monkey i, its position is $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $i = 1, 2, \ldots, M$, respectively. $f(X_i)$ is the corresponding objective function value. The improved climb process is given as follows:

(1) Randomly generate two vectors $\Delta x'_i = (\Delta x'_{i1}, \Delta x'_{i2}, \ldots, \Delta x'_{in})$ and $\Delta x''_i = (\Delta x''_{i1}, \Delta x''_{i2}, \ldots, \Delta x''_{in})$, where

$$\Delta x'_{ij}, \Delta x''_{ij} = \begin{cases} a & \text{with probability } \frac{1}{2} \\ -a & \text{with probability } \frac{1}{2} \end{cases} \quad (2)$$