Applied Soft Computing 19 (2014) 252–263

Contents lists available at ScienceDirect

Applied Soft Computing

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

Shuffled frog leaping algorithm and its application to 0/1 knapsack problem

Kaushik Kumar Bhattacharjee, S.P. Sarmah*

Department of Industrial Engineering and Management, Indian Institute of Technology, Kharagpur, WB 721302, India

ARTICLE INFO

Article history:
Received 9 February 2012
Received in revised form 5 November 2013
Accepted 9 February 2014
Available online 5 March 2014

Keywords:
Meta-heuristics
Discrete shuffled frog leaping algorithm
Knapsack problem
Genetic mutation

ABSTRACT

This paper proposes a modified discrete shuffled frog leaping algorithm (MDSFL) to solve 01 knapsack problems. The proposed algorithm includes two important operations: the local search of the 'particle swarm optimization' technique; and the competitiveness mixing of information of the 'shuffled complex evolution' technique. Different types of knapsack problem instances are generated to test the convergence property of MDSFLA and the result shows that it is very effective in solving small to medium sized knapsack problems. Further, computational experiments with a set of large-scale instances show that MDSFL can be an efficient alternative for solving tightly constrained 01 knapsack problems.

1. Introduction

The knapsack problem is one of the classical NP-hard optimization problem and the decision problem belongs to the class of NP-complete. It is thoroughly studied in the literature for last few decades. It offers many practical applications in vast field of different areas, such as project selection [1], resource distribution [2], network interdiction problem [3], investment decision-making [4] and so on. 01 knapsack problem is defined by: given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than a given limit and the total value is as large as possible. The most common formulation of the problem is the 01 knapsack problem, which restricts the number x_j of copies of each kind of item to zero or one.

$$\begin{aligned}
&\textit{Maximize} \quad f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} c_j x_j \\
&\textit{Subject to} \quad g(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j \le b, \\
&\qquad x_j \in \{0, 1\} \quad j = 1, 2, \ldots, n, \\
&\qquad c_j > 0, \quad a_j \ge 0, \quad b > 0.
\end{aligned} \tag{1}$$

The binary decision variables x_j are used to indicate whether item j is included in the knapsack or not. It may be assumed that all profits and weights are positive, and that all weights are smaller than the capacity b.

In recent times, many heuristic and meta-heuristic algorithms have been employed to solve 01 knapsack problems: Zhao et al. [5] proposed genetic algorithm to solve 01 knapsack problem. Greedy strategy combining with traditional genetic algorithm proved to be much more effective to handle difficult instances. Lin [6] used genetic algorithm to solve knapsack problem with imprecise weight, and he investigated the possibility of using genetic algorithms in solving the fuzzy knapsack problem without defining membership functions for each imprecise weight coefficient. Liu and Liu [7] proposed a schema-guiding evolutionary algorithm (SGEA) to solve 01 knapsack problems. Wanga et al. [8] proposed quantum swarm evolutionary algorithm to solve 01 knapsack problems. Shi [9] modified the parameters of the ant colony optimization (ACO) model to adapt itself to 01 knapsack problems. The improved ACO has strong capability of escaping from the local optimum through artificial interference. Li and Li [10] proposed a binary particle swarm optimization based on multi-mutation strategy (MMBPSO) to solve knapsack problem. The MMBPSO can effectively escape from the local optima to avoid premature convergence due to the utilization of Multi-Mutation strategy. Zou et al. [11] proposed a novel global harmony search algorithm to solve 01 knapsack problems. They utilized position updating and discrete genetic mutation strategy to avoid the premature convergence.

Although many 01 knapsack problems have been solved successfully by these algorithms, but some new and more difficult 01

* Corresponding author. Tel.: +91 3222 283734.
E-mail addresses: bhattacharjee.kaushik@gmail.com (K.K. Bhattacharjee), spsarmah@iem.iitkgp.ernet.in, sp_sarmah@yahoo.com (S.P. Sarmah).

http://dx.doi.org/10.1016/j.asoc.2014.02.010
1568-4946/© 2014 Elsevier B.V. All rights reserved.

knapsack problems hidden in the real world, so the research on this particular issue is still important. Many algorithms provide possible solutions for some 01 knapsack problems, but they may lose their efficiency on solving these difficult problems due to their own disadvantages and limitations. Most of these algorithm proposed recently are effective for solving 01 knapsack problem with very low dimension, but they may not be effective for 01 knapsack problems with high dimensional sizes.

The shuffled frog leaping algorithm (SFLA) is a meta-heuristic optimization method which is based on observing, imitating, and modeling the behavior of a group of frogs when searching for the location that has the maximum amount of available food [12]. SFLA, originally developed by Eusuff and Lansey in 2003, can be used to solve many complex optimization problems, which are nonlinear, non-differentiable, and multi-modal [13]. SFLA has been successfully applied to several engineering optimization problems such as water resource distribution [14], bridge deck repairs [15], job-shop scheduling arrangement [16], multi-mode resource-constrained project scheduling problem [17], unit commitment problem [18] and traveling salesman problem (TSP) [19]. The most distinguished benefit of SFLA is its fast convergence speed [20]. The SFLA combines the benefits of both the genetic-based memetic algorithm (MA) and the social behavior-based PSO algorithm [21].

2. Discrete shuffled frog leaping algorithm

In SFLA, the population consists of a set of frogs (solutions) that is partitioned into subsets referred to as memeplexes. The different memeplexes are considered as different cultures of frogs, each performing a local search. Within each memeplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of memetic evolution. After a defined number of memetic evolution steps, ideas are passed among memeplexes in a shuffling process [22]. The local search and the shuffling processes continue until defined convergence criteria are satisfied [14].

An initial population of P frogs is created randomly. For S-dimensional problems (S variables), a frog i is represented as $X_i=(x_{i1}, x_{i2}, \ldots, x_{iS})$. Afterwards, the frogs are sorted in a descending order according to their fitness. Then, the entire population is divided into m memeplexes, each containing n frogs (i.e. $P=m\times n$). In this process, the first frog goes to the first memeplex, the second frog goes to the second memeplex, frog m goes to the mth memeplex, and frog $m+1$ goes back to the first memeplex, etc.

Within each memeplex, the frogs with the best and the worst fitnesses are identified as X_b and X_w, respectively. Also, the frog with the global best fitness is identified as X_g. Then, a process similar to PSO is applied to improve only the frog with the worst fitness (not all frogs) in each cycle. Accordingly, the position of the frog with the worst fitness is adjusted as follows:

$$D_i = Rand()\times(X_b - X_w), \tag{2}$$

where D_i is the change in ith frog position and new position is given by:

$$\begin{aligned} &X_w(new) = X_w + D_i,\\ &-D_{\max} \le D_i \le D_{\max}; \end{aligned} \tag{3}$$

where $Rand()$ is a random number($Rand()\sim U(0, 1)$); and D_{max} is the maximum allowed change in a frog's position. If this process produces a better solution, it replaces the worst frog. Otherwise, the calculations in Eqs. (2) and (3) are repeated but with respect to the global best frog (i.e. X_g replaces X_b). If no improvement possible in this case, then a new solution is randomly generated to replace that frog. The calculations then continue for a specific number of iterations [14].

For handling integer programming problems the discrete version of the SFLA is used, called discrete shuffled frog leaping algorithm (DSFLA). The worst frog within each memeplex is updated [12] according to

$$\begin{aligned} &D_i = \begin{cases} \min\{\text{int}[Rand\times(X_b - X_w)], D_{\max}\} & \text{for a positive step,}\\ \max\{\text{int}[Rand\times(X_b - X_w)], -D_{\max}\} & \text{for a negative step;}\end{cases}\\ &X_w(new) = X_w + D_i. \end{aligned} \tag{4}$$

Like SFLA, DSFLA also follows same steps to replace the worst frog. If Eq. (4) does not produce a better solution, then X_b is replaced by the global best frog i.e. X_g; and if in this case also we replace the worst frog by a new randomly generated solution, if Eq. (4) does not produce a better solution. Accordingly, the main parameters of DSFLA are: number of frogs P; number of memeplexes m; number of generation for each memeplex before shuffling n; number of shuffling iterations it; and maximum number of iterations $iMax$. The pseudocode for the DSFLA is given in Algorithm 1.

Algorithm 1. Pseudocode for a DSFLA procedure

```
Generate random population of $P$ solutions (frogs)
for each individual $i \in P$ do
  calculate fitness($i$)
end for
Sort the population $P$ in descending order of their fitness
Divide $P$ into $m$ memeplexes
for each memeplex do
  Determine the best and worst frogs
  Improve the worst frog position using Eq. (4)
  Repeat for a specific number of iterations
end for
Combine the evolved memeplexes
Sort the population $P$ in descending order of their fitness
if termination = true then
  Return best solution
end if
```

3. Modified DSFLA for 01 knapsack

01 knapsack problem cannot be handled directly by SFLA or DSFLA because of its particular structure. For this reason original DSFLA is modified and applied to solve 01 knapsack problems as discussed below. Shuffled frog leaping algorithm has the most advantageous property of fast convergence speed. But at the same time it looses the searching capability of divergent field and sometimes trapped within a local optima. To make a balance between the convergent and divergent property we further modified DSFLA by hybridizing which include the genetic mutation property of divergent category. The modified discrete shuffled frog leaping algorithm (MDSFLA) is discussed in this section in full details.

3.1. *Construction of individual frog*

01 knapsack problem is an integer programming problem. There are two possible values for the decision variable x_j, zero or one. The individual frog is represented by a n-bit binary string, where n is the dimension of the problem. The initial population is created randomly to achieve sufficient diversification.

3.2. *Process for discrete variables*

In this paper we have used three kinds of discretization to solve the 01 knapsack problems.