Future Generation Computer Systems 78 (2018) 77–86

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A novel binary artificial bee colony algorithm for the set-union knapsack problem

Yichao He [a], Haoran Xie [b], Tak-Lam Wong [b], Xizhao Wang [c,*]

[a] College of Information and Engineering, Hebei GEO University, Shijiazhuang 050031, China
[b] Department of Mathematics and Information Technology, The Education University of Hong Kong, Hong Kong Special Administrative Region
[c] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

HIGHLIGHTS

- A novel bee colony method based on the full mapping function is proposed.
- Infeasible solutions are addressed by using a greedy strategy for Knapsack problems.
- The method has better results than extant approximation algorithms to solve SUKP.
- The proposed generic model can be integrated with other evolutionary algorithms.

ARTICLE INFO

Article history:
Received 3 September 2016
Received in revised form 19 April 2017
Accepted 19 May 2017
Available online 15 June 2017

Keywords:
Set-union knapsack problem
Artificial bee colony
Infeasible solution
Repairing and optimization

ABSTRACT

This article investigates how to employ artificial bee colony algorithm to solve Set-Union Knapsack Problem (SUKP). A mathematical model of SUKP, which is to be easily solved by evolutionary algorithms, is developed. A novel binary artificial bee colony algorithm (BABC) is also proposed by adopting a mapping function. Furthermore, a greedy repairing and optimization algorithm (S-GROA) for handling infeasible solutions by employing evolutionary technique to solve SUKP is proposed. The consolidation of S-GROA and BABC brings about a new approach to solving SUKP. Extensive experiments are conducted upon benchmark datasets for evaluating the performance of our proposed models. The results verify that the proposed approach is significantly superior to the baseline evolutionary algorithms for solving SUKP such as A-SUKP, ABC_{bin} and binDE in terms of both time complexity and solution performance.
© 2017 Published by Elsevier B.V.

1. Introduction and background

The Set-Union Knapsack Problem (SUKP) [1,2], a natural extension of the standard 0–1 Knapsack Problem (0–1 KP), is an NP-complete problem. In spite of the difficulty, SUKP has been identified to be valuable in various domain-specific applications such as financial decision making [2,3], flexible manufacturing machine [1,4,5], database partitioning [6,7], smart city [8] and data stream compression [9]. In particular, a popular application of SUKP is to build public key prototype (PKC) [10]. To enhance the security in building PKC based on SUKP, researchers attempt to hidden the trace of public key through many iterations. Therefore, intruders are unable to adopt Lenstra integer programming algorithms to break the key. It is worth to pointing out that Evolutionary Algorithms (EAs) are essentially a random search scheme in which the search performance is irrelevant to properties of the problem. Many researchers therefore believe that EA-based PKC is a promising technique. The extant studies show that the crucial techniques are how to design quick and efficient algorithm to solve SUKPs. It is commonly acknowledged that studies about approaches to solving SUKP based on EAs are quite important to the area of information security.

Goldschmidt et al. proposed a Dynamic Programming (DP) algorithm for addressing SUKP accurately based on the hypergraph theory [1]. However, the time complexity of the algorithm is exponential so that it is infeasible for real-world applications. Moreover, Ashwin developed an approximation algorithm called A-SUKP to solve SUKP based on the greedy strategy [2]. The approximation rate of A-SUKP is $1/(1-e^{-1/d})$, where $d(d \geq 2)$ is the upper-bound of the occurrence of all the elements. Apparently, the approximation solution of A-SUKP is relatively unsatisfactory and inefficient when d becomes large.

Artificial Bee Colony (ABC), which is a swarm intelligence approach developed in 2005 [11–13], mimics the bee colony to search for quality honey in the natural environment. Karaboga and Basturk analyzed the characteristics and conducted comprehensive

* Corresponding author.
E-mail address: xzwang@szu.edu.cn (X. Wang).

http://dx.doi.org/10.1016/j.future.2017.05.044
0167-739X/© 2017 Published by Elsevier B.V.

comparisons between ABC and a number of Evolutionary Algorithms (EAs) including Genetic Algorithm (GA) [14], Differential Evolution (DE) [15], Particle Swarm Optimization (PSO) [16], etc. The experimental results illustrate that the ABC has better than or similar performance to other population-based algorithms. In addition, ABC has the advantage of employing a few tuning parameters. Recently, ABC has achieved promising results in several optimization problems. For example, Karaboga proposed a method for digital infinite impulse response (IIR) filters based on ABC [17]. Karaboga and Ozturk exploited ABC for training network networks [18]. Kang et al. proposed an algorithm called Rosenbrock Artificial Been Colony algorithm to improve the accuracy of ABC in numerical optimization problems [19]. Karaboga and Ozturk proposed a novel clustering approach based on ABC [20]. Pan et al. developed a Discrete Artificial Bee Colony algorithm to address the lot-streaming flow shop scheduling problem [21]. Tsai adapted ABC for tackling the constrained optimization problems [22]. Kiran et al. developed a variable search strategy for continuous optimization problems [23]. Banitalebi et al. presented an approach called Enhanced Compact Artificial Bee Colony, which improves the performance of ABC [24]. Kiran proposed a new approach for the incapacitated facility location problem based on Continuous Artificial Bee Colony (ABC_{bin}) algorithm [25]. Ozturk et al. proposed a Binary Artificial Bee Colony (BABC) algorithm for solving the 0–1 KP problem by adopting genetic operators [26]. Secui designed an enhanced ABC for solving the economic dispatch problem [27]. Due to the large number of applications of ABC to optimization problems, this article aims at developing an approach to solving SUKP based on ABC. We firstly introduce a mathematical model of SUKP. One characteristic of the model is that EAs can also be readily integrated to solve SUKP. A novel Binary Artificial Bee Colony (BABC) algorithm is proposed. Furthermore, we present a novel method for solving SUKP by an efficient algorithm (named S-GROA) to tackle the issue of the infeasible solution in SUKP.

The remaining parts of the paper is organized as follows. Section 2 presents the definition of SUKP and the mathematical model of SUKP, which facilitate the use of EAs. Section 3 reviews the principle of ABC and proposes BABC algorithm. In Section 4, we firstly present an efficient and generic an efficient algorithm (named S-GROA) to tackle the issue of the infeasible solution in SUKP. In addition, we applied S-GROA to BABC to accomplish a new approach for solving SUKP. It follows by a large-scale comparison between BABC and other methods including A-SUKP, GA, ABC_{bin} and binDE [28] for solving three real-world problems in Section 5. The experimental results show that BABC is superior to A-SUKP with respect to efficiency and performance. Moreover, the performance of BABC is more efficient than one of GA, ABC_{bin} and binDE. We draw the conclusion and discuss the future direction of research in Section 6.

2. Definition and mathematical models

Definition 1 (Set-union Knapsack Problem, SUKP) [1,2]: Given a set of elements $U = \{1, 2, \ldots, n\}$ and a set of items $S = \{1, 2, \ldots, m\}$, such that each item $i \in S$ $(i = 1, 2, \ldots, m)$ corresponds to a subset $U_i \subseteq U$ and associates with a value $p_i > 0$. Each element $j \in U$ $(j = 1, 2, \ldots, n)$ has a weight $w_j > 0$. For an arbitrary non-empty set $A \subseteq S$, the profit and the weight of A are defined as $P(A) = \sum_{i \in A} p_i$ and $W(A) = \sum_{j \in \bigcup_{i \in A} U_i} w_j$ respectively. The objective of SUKP is to find the subset $S^* \subseteq S$, such that $P(S^*)$ is maximized, subject to $W(S^*) \leq C$ where C is the capacity of knapsack.

Formally, SUKP can be notated as follows:

$$\textit{Maximize} \quad P(A) = \sum_{i \in A} p_i \tag{1}$$

$$\text{subject to} \quad W(A) = \sum_{j \in \bigcup_{i \in A} U_i} w_j \leq C, A \subseteq S. \tag{2}$$

We name the above mathematical model as SUKP-M1. Without loss of generality, let p_i $(i = 1, 2, \ldots, m)$, w_j $(j = 1, 2, \ldots, n)$ and C be positive integers. Let $\boldsymbol{V} = \{U_1, U_2, \ldots, U_m\}$ be the cover of U, such that $U_i \subset U (i = 1, 2, \ldots, m)$ and $U_i \neq \Phi$. In addition, $W(S) > C$ and $\sum_{j \in U_i} w_j \leq C$ for all $i \in S$. EAs are not suitable to use SUKP-M1. Therefore, we developed an integer programming model named as SUKP-M2, which can facilitate for solving SUKP by using EAs.

Let $Y = [y_1, y_2, \ldots, y_m] \in \{0, 1\}^m$ be an m-dimension 0–1 vector. $A_Y = \{i | y_i \in Y, y_i = 1, 1 \leq i \leq m\} \subseteq S$, then for an arbitrary $i = 1, 2, \ldots, m$, $y_i = 1$ if and only if $i \in A_Y$. Apparently, the 0–1 vector Y and the subset $A_Y \subset S$ are one-to-one mappings. By using the one-to-one mapping of Y and A_Y, SUKP can be modeled as a new integer programming model (SUKP-M2) as follows.

$$\textit{Maximize} \quad f(Y) = \sum\nolimits_{i=1}^{m} y_i p_i \tag{3}$$

$$\text{subject to} \quad W(A_Y) = \sum\nolimits_{j \in \bigcup_{i \in A_Y} U_i} w_j \leq C. \tag{4}$$

According to SUKP-M2, all 0–1 vectors $Y = [y_0, y_1, \ldots, y_m] \in \{0, 1\}^m$ are the only possible solutions of SUKP. Solutions which satisfy constraint (4) are feasible solutions of SUKP, while the solutions which cannot satisfy constraint (4) are infeasible solutions. As an instance of SUKP is consisted of the parameters n and m, the profit set $P = \{p_i | 1 \leq i \leq m\}$, the weight set $W = \{w_j | 1 \leq j \leq n\}$, the subset family $\boldsymbol{V} = \{U_1, U_2, \ldots, U_m\}$ and the capacity C, an instance of SUKP will be denoted as $\mathbf{INS}(n, m, P, W, C, \boldsymbol{V})$ in the remaining sections.

3. Binary ABC (BABC) algorithm

3.1. Artificial bee colony algorithm

The Artificial Bee Colony (*ABC*) [11,12] is a population-based meta-heuristic algorithm for optimizing numerical problems. It was inspired by the intelligent foraging behavior of honey bees when they are seeking a quality food source. In the *ABC*, each candidate solution to the optimization problem is associated with a food source. It is represented by an D-dimensional real-coded vector, where D is the dimension of the optimization problem. The quality (i.e., fitness) of a solution corresponds to the amount of nectar in that food source.

The population of *ABC* includes three categories of bees: employed bees, onlooker bees and scout bees. Each of them carrying on various activities to find a better food source. Employed bees take charge of exploring the solution space to search for food sources and then sharing various pieces of information with other bees. Onlooker bee exerts a probabilistically modification on the solution (food source) for finding a new solution and tests the fitness amount of the new solution. Scout bees work to help *ABC* escape from local optimums. To sum up, the employed and onlooker bees are responsible for exploitation, whereas the scout bees handle exploration.

Let $X = [x_1, x_2, \ldots, x_D] \in [l_j, u_j]^D$ represents a food source, *fit* (X) is the fitness value of a food source as shown in the following Eq. (5).

$$\textit{fit}(X) = \begin{cases} 1/(1 + f(X)), & \textit{if } f(X) \geq 0 \\ 1 + |f(X)|, & \textit{otherwise} \end{cases} \tag{5}$$

where $f(X)$ is the objective function of X, l_j and u_j represent minimum and maximum of the jth variable respectively. In the *ABC*,

Download English Version:

https://daneshyari.com/en/article/4950261

Download Persian Version:

https://daneshyari.com/article/4950261

Daneshyari.com