Applied Soft Computing 9 (2009) 625–631

Contents lists available at ScienceDirect

Applied Soft Computing

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

An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem

Alok Singh *

Department of Computer and Information Sciences, University of Hyderabad, Hyderabad 500046, Andhra Pradesh, India

ARTICLE INFO

Article history:
Received 4 March 2008
Received in revised form 16 July 2008
Accepted 1 September 2008
Available online 9 September 2008

Keywords:
Artificial bee colony algorithm
Constrained optimization
Leaf-constrained minimum spanning tree
Swarm intelligence

ABSTRACT

Given an undirected, connected, weighted graph, the leaf-constrained minimum spanning tree (LCMST) problem seeks on this graph a spanning tree of minimum weight among all the spanning trees of the graph that have at least ℓ leaves. In this paper, we have proposed an artificial bee colony (ABC) algorithm for the LCMST problem. The ABC algorithm is a new metaheuristic approach inspired by intelligent foraging behavior of honeybee swarm. We have compared the performance of our ABC approach against the best approaches reported in the literature. Computational results demonstrate the superiority of the new ABC approach over all the other approaches. The new approach obtained better quality solutions in shorter time.

1. Introduction

Given an undirected, connected, weighted graph with n nodes and a positive integer $\ell(2 \leq \ell < n-1)$, the leaf-constrained minimum spanning tree (LCMST) problem seeks on this graph a spanning tree that contains at least ℓ leaves and has minimum total weight among all such spanning trees. Formally, let $G = (V, E)$ be an undirected, connected graph, where V denotes the set of nodes and E denotes the set of edges. Given a non-negative weight function $w : E \rightarrow \mathbb{R}^+$ associated with its edges and a positive integer $\ell(2 \leq \ell < n-1)$, the LCMST problem seeks a spanning tree $T \subseteq E$ that has at least ℓ leaves and that minimizes:

$$W(T) = \sum_{e \in T} w(e)$$

In general, this problem is NP-Hard [1]. If the leaf constraint ℓ is less than the number of leaves in an unconstrained minimum spanning tree (MST), then this MST is also a LCMST. Therefore, the LCMST problem can be solved in polynomial time in this case. However, if ℓ is larger than the number of leaves in any MST, then the LCMST problem is NP-Hard. Usually, the difficulty of the problem increases with increase in ℓ.

The LCMST problem has several practical applications. It finds applications in facilities location, circuit and network designs [2]. This problem can be considered as an extension of the p-median problem, where medians are also connected among themselves [3].

Several heuristics for the LCMST problem work by first computing a MST and then transforming this MST into a leaf-constrained spanning tree (LCST), i.e., a spanning tree with at least ℓ leaves. Deo and Micikevicius [1] presented one such heuristic. It iteratively tries to transform a MST into a LCST. During each iteration, it interchanges a tree edge with a non-tree edge so that the number of leaves in the tree increases with minimum increase in tree's weight. This process is repeated until either the number of leaves in the tree reaches ℓ or no edge interchange is possible that increases the number of leaves. Therefore, this heuristic some times fails to find a LCST. Its time complexity is $O(n^4)$. Julstrom [4] proposed another heuristic called ML that also begins by computing a MST. Then it iteratively transforms this MST into a LCST. During each iteration, it selects an interior node (a node that is not a leaf) from the set of current interior nodes and converts it into a leaf. The interior node is selected according to the following criterion. It tries each interior node as leaf one by one. During each trial it finds a MST on the graph induced by the remaining interior nodes and then connects all leaves including the new one to the nearest interior node. The interior node for which the resulting spanning tree is of minimum weight is chosen for conversion to leaf. This process is repeated until the number of leaves in the spanning tree reaches ℓ. If the underlying graph is complete then ML always finds a LCST. The complexity of ML is also $O(n^4)$. Julstrom [4] observed that for $\ell = 0.6n$, ML finds lower-weight

* Tel.: +91 40 23134011.
E-mail address: alokcs@uohyd.ernet.in.

1568-4946/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2008.09.001

LCSTs than the heuristic of Deo and Micikevicius [1] does and for $\ell = 0.9n$ the latter heuristic always fails to find a LCST.

Edelson and Gargano [5] developed a permutation-coded genetic algorithm for the LCMST problem that encodes a LCST using $3n - \ell - 2$ symbols. A two-level decoder converts this encoding first into Prüfer code [6] and then this Prüfer code into equivalent LCST. This scheme represents only the feasible solutions. However, chromosome length is considerably large with this scheme. Julstrom [2] presented two generational genetic algorithms. One of these two genetic algorithms uses the Blob code [7], whereas the other uses the subset coding. The subset coding represents a LCST by the set of its interior nodes only. A two-step procedure converts this coding into its equivalent LCST. The first step constructs a MST on the graph induced by the set of interior nodes, whereas the latter step connects each leaf to its nearest interior node. Like the encoding of Edelson and Gargano [5], both of these encodings also represent feasible solutions only. At the same time their chromosome lengths are much smaller. The chromosome length with Blob code is $n - 2$, whereas that with subset coding is $n - \ell$. The genetic operators for these two genetic algorithms are designed in such a way that they generate feasible solutions only. Computational results on the test instances considered in [2] showed that subset-coded genetic algorithm always finds the LCST of lowest weight followed by ML heuristic. The Blob-coded genetic algorithm performed even worse than ML heuristic. Hereafter, the subset-coded genetic algorithm will be referred to as SCGA.

Singh and Baghel [8] presented two metaheuristic approaches, one based on ant-colony optimization (ACO) and the other based on tabu search, for the LCMST problem and compared their approaches with the subset-coded genetic algorithm and ML heuristic [2]. Both of these methods use the subset coding [2] to represent a LCST and construct the equivalent LCST according to the two-step procedure described above. The ACO approach constructs a LCST by first identifying its $(n - \ell)$ interior nodes through ant and then constructs the complete LCST by the two-step procedure. Starting from a random initial solution, the tabu search iteratively moves from one solution to another by exchanging an interior node with a leaf node until the stopping criterion is satisfied. During each iteration, among all valid exchange moves, the move which results in a LCST of least cost is selected (even if its cost is more than the current solution). Once a move has been performed its exactly reverse move is forbidden for the duration determined by tabu tenure. This tabu search also has an in-built cycle detection mechanism. Computational results demonstrated the superiority of these approaches over subset-coded genetic algorithm and ML heuristic. The tabu search and ACO performed quite similar in terms of solution quality, but tabu search was much faster than the ACO approach. Hereafter, these tabu search and ACO approaches will be, respectively referred to as TS-LCMST and ACO-LCMST.

In this paper, we have proposed an artificial bee colony (ABC) algorithm for the LCMST problem. The artificial bee colony algorithm is a new metaheuristic approach, proposed by Karaboga [9]. It is inspired by the intelligent foraging behavior of honeybee swarm. We have compared our ABC approach against the ACO-LCMST and TS-LCMST approaches proposed in [8] and SCGA proposed in [2]. Computational results demonstrate the effectiveness of our approach in comparison to these approaches.

The rest of this paper is organized as follows: Section 2 provides an introduction to the artificial bee colony algorithm. Section 3 describes our ABC algorithm for the LCMST problem. Computational results are presented in Section 4, whereas Section 5 outlines some conclusions.

2. The artificial bee colony algorithm

The artificial bee colony algorithm is a new population-based metaheuristic approach proposed by Karaboga [9] and further developed by Karaboga and Basturk [10–13]. This approach is inspired by the intelligent foraging behavior of honeybee swarm. The foraging bees are classified into three categories—employed, onlookers and scouts. All bees that are currently exploiting a food source are classified as "employed". The employed bees bring loads of nectar from the food source to the hive and may share the information about food source with onlooker bees. "Onlookers" are those bees that are waiting in the hive for the information to be shared by the employed bees about their food sources and "scouts" are those bees which are currently searching for new food sources in the vicinity of the hive. Employed bees share information about food sources by dancing in a common area in the hive called dance area. The duration of a dance is proportional to the nectar content of the food source currently being exploited by the dancing bee. Onlooker bees which watch numerous dances before choosing a food source tend to choose a food source according to the probability proportional to the quality of that food source. Therefore, the good food sources attract more bees than the bad ones. Whenever a bee, whether it is scout or onlooker, finds a food source it becomes employed. Whenever a food source is exploited fully, all the employed bees associated with it abandon it, and may again become scouts or onlookers. Scout bees can be visualized as performing the job of exploration, whereas employed and onlooker bees can be visualized as performing the job of exploitation.

Motivated by this foraging behavior of honeybees, Karaboga [9] proposed the artificial bee colony algorithm. In the ABC algorithm, each food source represents a possible solution to the problem under consideration and the nectar amount of a food source represents the quality of the solution represented by that food source. In this algorithm also colony of artificial bees (bees for short) has same three types of bees—employed, onlookers and scouts. First half of the bee colony comprises employed bees, whereas the latter half contains the onlookers. The ABC algorithm assumes that there is only one employed bee for every food source, i.e., the number of food sources is same as the number of employed bees. The employed bee of an abandoned food source becomes a scout and as soon as it finds a new food source it again becomes employed. The ABC algorithm is an iterative algorithm. It starts by associating all employed bees with randomly generated food sources (solution). Then during each iteration, every employed bee determines a food source in the neighborhood of its currently associated food source and evaluates its nectar amount (fitness). If its nectar amount is better than that of its currently associated food source then that employed bee moves to this new food source leaving the old one, otherwise it retains its old food source. When all employed bees have finished this process, they share the nectar information of the food sources with the onlookers, each of whom selects a food source according to a probability proportional to the nectar amount of that food source. The probability p_i of selecting a food source i is determined using the following expression:

$$p_i = \frac{f_i}{\sum_{j=1}^{m} f_j}$$

where f_i is the fitness of the solution represented by the food source i and m is the total number of food sources. Clearly, with this scheme good food sources will get more onlookers than the bad ones. After all onlookers have selected their food sources, each of them determines a food source in the neighborhood of his chosen food source and computes its fitness. The best food source among all the neighboring food sources determined by the onlookers associated with a particular food source i and food source i itself,