Swarm and Evolutionary Computation 32 (2017) 132–139

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Genetic algorithms to balanced tree structures in graphs ☆

Riham Moharam, Ehab Morsy *

Department of Mathematics, Suez Canal University, Ismailia 41522, Egypt

ARTICLE INFO

Article history:
Received 22 August 2015
Received in revised form
30 May 2016
Accepted 20 June 2016
Available online 22 June 2016

Keywords:
Minimum Spanning Tree
Shortest Path Tree
Tree Spanner
Balanced Spanning Tree
Minimum Maximum Stretch Spanning Tree
Genetic Algorithms

ABSTRACT

Given an edge-weighted graph $G=(V, E)$ with vertex set V and edge set E, we study in this paper the following related balanced tree structure problems in G. The first problem, called *Constrained Minimum Spanning Tree Problem* (CMST), asks for a rooted tree T in G that minimizes the total weight of T such that the distance between the root and any vertex v in T is at most a given constant C times the shortest distance between the two vertices in G. The *Constrained Shortest Path Tree Problem* (CSPT) requires a rooted tree T in G that minimizes the maximum distance between the root and all vertices in V such that the total weight of T is at most a given constant C times the minimum tree weight in G. The third problem, called *Minimum Maximum Stretch Spanning Tree* (MMST), looks for a tree T in G that minimize the maximum distance between all pairs of vertices in V. It is easy to conclude from the literatures that the above problems are NP-hard. We present efficient genetic algorithms that return (as shown by our experimental results) high quality solutions for these problems.

1. Introduction

Let $G=(V, E)$ be an edge-weighted connected graph with vertex set V and edge set E. Let n and m denote the cardinalities of V and E respectively (i.e., $|V|=n$ and $|E|=m$). A spanning tree T_m in G is a Minimum Spanning Tree (MST) if there is no other spanning tree in G that attains a total weight less than that of T_m. Kruskal's and Prim's algorithms are well known polynomial time algorithms for computing a minimum spanning tree in weighted graphs. For any vertex r in G, a spanning tree T_s rooted at r is a shortest path tree if, for every vertex $v \in V$, the distance between r and v in T_s equals the shortest distance between the two vertices in G. Dijkstra's algorithm is one of the well known polynomial time algorithms for computing shortest path tree in weighted graphs [44].

Tree structures are widely used in many applications like network routing, communication networks and distributed systems. In particular, the shortest path tree minimizes the delay from the source to every destination through a routing tree, and the minimum spanning tree minimizes the total routing cost along a tree. See [10,21,26,31,45] and the references therein. Thus, balanced tree structures are appropriate routing trees for communication networks that aim to balance the above two objectives. Also, in the case of concurrent requests through a routing tree, it is required to simultaneously minimize the distances between all vertex pairs in the constructed tree (see [14,25,36] and the references therein).

Note that, there exist weighted graphs in which the total weight of a shortest path tree may be much more than that of a minimum spanning tree, and vertices that are close to the designated root can be far away from the root in a minimum spanning tree (see [29] for an illustrative example). A rooted tree in a given graph balances a minimum spanning tree and a shortest path tree if its total weight is at most a constant times the minimum spanning tree weight, and the distance between the root and any vertex in the tree is at most a constant times the shortest distance between the two vertices in the graph. Formally, for any $\alpha, \beta \geq 1$, a rooted spanning tree T in G is called (α, β)-balanced spanning tree if, (i) for every vertex v, the distance between the root and v in T is at most α times the shortest distance between the two vertices in G, and (ii) the total weight of T is at most β times the minimum tree weight in G.

Given a constant $C \geq 1$, we first consider the following two variants of the (α, β)-balanced spanning tree problem. The Constrained Minimum Spanning Tree Problem (CMST) asks for a tree T in G that minimizes the total weight of T (i.e., minimizes β) such that the distance between the root and any vertex v in T is at most C times the shortest distance between the two vertices in G. A formal definition for the CMST can be described as follow. For any pair of vertices u and v in G, $d_T(u, v)$ (respectively, $d_G(u, v)$) denotes the shortest distance between u and v in T (respectively, G). For a

☆Preliminary versions of this paper appeared in the proceedings of the Federated Conference on Computer Science and Information Systems and the International Conference on Advanced Intelligent Systems and Informatics, 2015. This research was partially supported by Alexander von Humboldt Foundation.

* Corresponding author.
E-mail addresses: riham.moharam@science.suez.edu.eg (R. Moharam), ehabmorsy@science.suez.edu.eg (E. Morsy).

http://dx.doi.org/10.1016/j.swevo.2016.06.005
2210-6502/

subgraph G' of G, let $w(G')$ denote the total weight of G' (i.e., the total weights of all edges in G').

Constrained Minimum Spanning Tree Problem (CMST):

Input: An edge-weighted graph $G=(V,E)$, a root $r \in V$, and a constant $C \geq 1$.

Feasible solution: A spanning tree T in G such that $d_T(r,v) \leq C \cdot d_G(r,v)$ for all $v \in G$.

Goal: Find a feasible solution that minimizes $w(T)/w(T_m)$.

Similarly, the Constrained Shortest Path Tree Problem (CSPT) requires a tree T in G that minimizes the maximum distance between the root and all vertices in G (i.e., minimizes α) such that the total weight of T is at most C times the minimum tree weight in G. This problem can be defined formally as follows.

Constrained Shortest Path Tree Problem (CSPT):

Input: An edge-weighted graph $G=(V,E)$, a root $r \in V$, and a constant $C \geq 1$.

Feasible solution: A spanning tree T in G such that $w(T) \leq C \cdot w(T_m)$.

Goal: Find a feasible solution that minimizes $\max_{v \in V} d_T(r,v)$.

A *tree t-spanner* problem is a well known balanced tree structure problem that looks for a tree T in G such that the distance between every pair of vertices in T is at most t times the shortest distance between the two vertices in G. In other words, for any $t \geq 1$, a spanning tree T of G is called tree t-spanner if, for any two vertices u and v in G, it holds that $d_T(u,v) \leq t \cdot d_G(u,v)$. The value of t, called the stretch factor of T, estimates the goodness of the distance approximation of T. In this paper, we are concerned with the Minimum Maximum Stretch Spanning Tree (MMST) problem, the problem of finding a *tree t-spanner* with the smallest possible stretch factor t [20,35]. A formal description for the MMST is given as follows.

Minimum Maximum Stretch Spanning Tree Problem (MMST):

Input: An edge-weighted graph $G=(V,E)$.

Feasible solution: A spanning tree T in G.

Goal: Find a feasible solution T that minimizes the stretch factor t.

It is well known that, for any $\alpha, \beta \geq 1$, the problem of deciding whether G contains an (α,β)-balanced spanning tree is NP-complete [29]. Consequently, the CMST and CSPT problems are NP-hard problems. Also, for any $t \geq 1$, the problem of deciding whether G contains a tree t-spanner is NP-complete [9], and hence the MMST problem is NP-hard. In this paper, we present efficient genetic algorithms for the above defined problems. Up to our knowledge, these are the first evolutionary algorithms for these problems. Our experimental results show that the proposed algorithms return high quality solutions for the problems.

The rest of this paper is organized as follows. Section 2 reviews some results on related problems. Section 3 presents the proposed genetic algorithms. Section 4 evaluates our algorithms by applying it to randomly generated instances of the two problems. Section 5 makes some concluding remarks.

2. Related work

In this section, we present results on related problems. Let $G=(V,E)$ be a given edge-weighted graph. The eccentricity of a vertex $v \in V$ is the greatest distance between v and any other vertex in V. The radius of G is the minimum eccentricity of any vertex. The diameter of G is maximum eccentricity of any vertex in the graph.

Bharath-Kumar and Jaffe [30] studied the problem of finding a rooted tree in G such that the total distances from the root to all vertices is at most a constant times the minimum total distances from the root to all vertices.

A tree in G is called shallow-light tree if its diameter is at most a constant (greater than or equal 1) times the diameter of G and with total weight at most a constant times the minimum spanning tree weight. Awerbuch et al. [2] proved that each graph has a shallow-light tree.

Cong et al. [12] proposed a model of timing-driven global routing for cell-based design to improve the construction of a shallow-light tree based on the idea of finding minimum spanning tree with bounded radius. They designed an algorithm to find, for any constant $\epsilon > 0$, a spanning tree with radius $(1+\epsilon) \cdot R$ (using an analog of the classical Prim's minimum spanning tree structure), where R is the minimum possible tree radius. That is, they found a smooth trade-off between the radius and the cost of the tree. Afterwards, they proposed a new method [13] to improve their previous algorithm based on a provably good algorithm that simultaneously minimizes both the total weight and the longest interconnection path length of the tree. More specifically, their algorithm produced a tree with radius at most $(1+\epsilon) \cdot R$ and of total weight at most $(1+2/\epsilon)$ times the minimum tree weight.

Given any $\alpha \geq 1$, Awerbuch et al. [3] proposed an algorithm that approximates a minimum spanning tree and a shortest paths tree in G. Namely, they modified the algorithm described in [2,12] to compute an $\left(\alpha, 1+\frac{4}{\alpha-1}\right)$-balanced spanning tree in $O(m+n\log n)$ time. Afterwards, Khullar et al. [29] improved the above result and presented a constructive linear time algorithm that outputs an $\left(\alpha, 1+\frac{2}{\alpha-1}\right)$-balanced spanning tree in G. In other words, for any $\gamma > 0$, the algorithm of Khullar et al. [29] outputs an $\left(1+\sqrt{2}\gamma, 1+\frac{\sqrt{2}}{\gamma}\right)$-balanced spanning tree in linear time.

For unweighted graphs (i.e., all edges in G have unit edge weights), Cai and Corneil [9] produced a linear time algorithm to find *a tree t-spanner* in G for any given $t \geq 2$. Moreover, they showed that, for any $t \geq 4$, the problem of finding *a tree t-spanner* in G is NP-complete. Brandstädt et al. [7] substantially strengthened the hardness result in [9]. Namely, they showed that, for any $t \geq 4$, *tree t-spanner* problem is NP-complete even on chordal graphs of diameter at most $t+1$ (respectively, $t+2$) if t is even (respectively, odd). Afterwards, they proved that the *tree t-spanner* problem is NP-complete even for chordal bipartite graphs for $t \geq 5$ [8].

In [22], Fekete and Kremer showed that it is NP-hard to determine a minimum value for t for which a *tree t-spanner* exists even for planar unweighted graphs. They designed a polynomial time algorithm that decides if the planar unweighted graphs with bounded face length contains *a tree t-spanner* for any fixed t. Moreover, they proved that for $t=3$, it can be decided whether the unweighted planar graph has *a tree t-spanner* in polynomial time. The problem was left open whether a *tree t-spanner* is polynomial time solvable in case of $t \geq 4$. Later, this problem is solved by Dragan et al. [15]. For any fixed t, they showed that it is possible in polynomial time not only to decide if a planar graph G has a tree t-spanner, but also to decide if G has a t-spanner of bounded treewidth. In particular, for every fixed values of t and k, they showed that the problem, for a given planar graph G to decide if G has a t-spanner of treewidth at most k, is not only polynomial time solvable, but is fixed parameter tractable (with k and t being the parameters).

Recently, Dragan and Köhler [16] examined the *tree t-spanner* on chordal graphs, generalized chordal graphs, and general graphs. For unweighted graphs G, they proposed a new algorithm that constructs a tree $(2\lceil t/2 \rceil \lfloor \log_2 n \rfloor)$-spanner in $O(mn \log^{2n})$ time or a tree $(6t\lfloor \log_2 n \rfloor)$-spanner in $O(m\log n)$ time for the arbitrary graphs. After that, they improved these results and constructed a tree $(2\lfloor \log_2 n \rfloor)$-spanner in $O(m\log n)$ time for chordal graphs. Also, they constructed a tree $(2\rho\lfloor \log_2 n \rfloor)$-spanner in $O(m \log^{2n})$ time or a tree $(12\rho\lfloor \log_2 n \rfloor)$-spanner in $O(m\log n)$ time for graphs that confess a Robertson–Seymour's tree-decomposition [16]. In