Information Processing Letters 115 (2015) 828–832

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Approximate spanning cactus

Alak Kumar Datta *

Department of Computer & System Sciences, Visva-Bharati University, Santiniketan 731235, India

ARTICLE INFO

Article history:
Received 28 June 2014
Received in revised form 13 June 2015
Accepted 13 June 2015
Available online 18 June 2015
Communicated by S.M. Yiu

Keywords:
Spanning cactus
NP-completeness
Approximation algorithms
τ-Triangle inequality
Minimum spanning tree

ABSTRACT

We study minimum spanning cactus and bottleneck spanning cactus problems with τ-triangle inequality. Both problems are NP-Complete. No approximation algorithms are known for these problems. We present τ approximation algorithm for the first and $2\tau - 1$ approximation algorithm for the second problem.

1. Introduction

A graph $H = (V, E_H)$ is a *partial cactus* if every edge of the graph is contained in at most one cycle. A partial cactus is called a *cactus* if every edge is contained in exactly one cycle.

For a graph $G = (V, E)$, a subgraph $H = (V, E_H)$ is called a *spanning cactus* if H is a cactus. If $w(e)$ is the weight of an edge e of G then $\sum_{e \in E_H} w(e)$ is the weight of the spanning cactus H, say $w(H)$. *Minimum Spanning Cactus (MSC)* Problem is to compute a spanning cactus of minimum weight.

Spanning cactus of a graph may or may not exist. For example, the graph in Fig. 1 has no spanning cactus.

Bottleneck weight of a graph $G = (V, E)$, denoted by b(G), is defined as $max\{w(e)|e \in E\}$. For a graph $G = (V, E)$, *Bottleneck Spanning Cactus* (BSC) problem is to find a spanning cactus $H = (V, E_H)$ with minimum $b(H)$.

Cactus is a very simple graph. There are many applications in the areas of traffic estimation [12,18], genome comparison [14], representing cuts of a graph [6,7,9] that use this simple structure.

Minimum spanning cactus problem is NP-Complete, both for directed and undirected graphs [15,10]. The problem was studied for design of approximation algorithms. It was shown that designing an approximation algorithm for MSC problem has approximation hardness same as the traveling salesman problem, both for directed and undirected graphs, if the edge weights satisfy the triangle inequality [10,15]. However, this equivalence with the traveling salesman problem does not hold if the edge weights do not satisfy the triangle inequality [2]. Also, it was noted in [10] that the approximation equivalence to the traveling salesman problem does not remain true if the triangle inequality is replaced by τ-triangle inequality. This means, it still remains open to design an approximation algorithm for MSC with τ-triangle inequality. In this paper, we present a simple approximation algorithm for MSC with τ-triangle inequality with a relative error bound τ.

Next we study the BSC problem. We establish the fact that the problem is NP-Complete. Further, we show that approximation algorithm for MSC can be modified to get an approximation algorithm for the BSC problem with relative error $2\tau - 1$.

* Tel.: +(91) 9434755464.
E-mail addresses: alak.datta@visva-bharati.ac.in, alak.kumar.datta@gmail.com.

http://dx.doi.org/10.1016/j.ipl.2015.06.009
0020-0190/

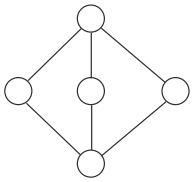

Fig. 1. A graph having no spanning cactus.

In Section 2, we present the approximation algorithm for MSC with τ-triangle inequality. Section 3 describes the approximation algorithm for BSC with τ-triangle inequality. Section 4 contains conclusion and scope for future work.

2. Approximate minimum spanning cactus

As in the case of traveling salesman problem, we assume, the input graph is complete and weighted. Weights satisfy τ-triangle inequality. This means that for three vertices x, y, z, $w((x, y)) \leq \tau[w(x, z) + w(z, y)]$, for some constant $\tau \geq 1$. Output of the algorithm is a spanning cactus H of G such that $w(H) \leq (1 + \tau)w(H^*)$, where H^* is the minimum spanning cactus of G.

2.1. Lower bound for MSC

Any constant bound approximation algorithm design needs to establish a good lower bound. For many problems, it has been found that weight of minimum spanning tree directly serves as lower bound [5,8,16,17]. We prove that a lower bound for the solution to MSC problem can be obtained from the minimum spanning tree.

Let $G = (V, E)$ be the complete graph with weights satisfying τ-triangle inequality. Also, let $H^* = (V, E')$ be its minimum spanning cactus and T^* be a minimum spanning tree of G. There are efficient polynomial time algorithms to compute T^* [11,13].

Minimum spanning cactus $H^* = (V, E')$ of $G = (V, E)$ consists of a set of cycles, say $C_1, C_2, ..., C_k$. Let $e_1, e_2, ..., e_k$ be the maximum weight edges in the cycles $C_1, C_2, ..., C_k$, respectively. Deletion of these edges from H^* leaves a spanning tree of G. Therefore, $w(H^* - \{e_1, e_2, ..., e_k\}) \geq$ weight of the minimum spanning tree $= w(T^*)$.

Therefore, $w(H^*) \geq w(T^*)$ + total weight of $\{e_1, e_2, ..., e_k\} \geq w(T^*) + w(e_i)$, where e_i is the maximum weighted edge in $\{e_1, e_2, \ldots, e_k\} \geq w(T^*) + w(e_{max})$, e_{max} is the maximum weighted edge in $T^* = w(T^*) + b(T^*)$.

$$\Rightarrow w(H^*) \geq w(T^*) + b(T^*) \tag{1}$$

Therefore,

Theorem 1. *Weight of the minimum spanning tree plus bottleneck of the minimum spanning tree is a lower bound for the weight of minimum spanning cactus.*

2.2. The approximation algorithm

We are now in a position to describe the algorithm. The algorithm first constructs a minimum spanning tree then adds some chords in it to get the approximate cactus. Chords are added in such a way that only cycles of length three are produced, except possibly one four cycle. Each three cycle contains two adjacent tree edges and one chord. We prove, by τ-triangle inequality that the weight of this chord is not too much. The four cycle, if included, contains two tree edges and two chords. In this case also, as in the case of three cycle, we prove that the weight of the two chords are within reasonable bounds. Steps of the algorithm are shown in Algorithm *Approx-MSC*.

Algorithm Approx-MSC

1 $C = \Phi$
2 Construct T^*, an MST of G.
3 Make a search on T^* to transform it to a rooted ordered tree with fixed root.
4 while $|T^*| \geq 4$ do //T^* contains more than three edges//

{$p =$ leftmost leaf
$r =$ parent of p
$q =$ next sibling of p,
if it exists else *parent(parent(p))*
$C = C \cup \{(r, p), (r, q), (p, q)\}$
$T^* = T^* - \{(r, p), (r, q), (p, q)\}$
}

5 If $|T^*| = 2$,
$C = C \cup \{(r, p), (r, q), (p, q)\}$, where (r, p), (r, q) are two tree edges.
6 If $|T^*| = 3$,
$C = C \cup \{(r, p), (r, q), (p, s), (s, q)\}$, where r is the only non-leaf vertex and p , s are two leaves.
7 return C

2.3. Correctness

Lemma 1. *At the end of execution, Approx-MSC returns a spanning cactus.*

Proof. Algorithm constructs an MST T^* at Step 2. At Step 4, the algorithm iteratively takes two adjacent edges out of T^*, constructs a cycle of length three which consists of these two tree edges and a unique chord, adds this cycle into the cactus. It continues the process until T^* contains at most three edges. If T^* contains only two edges then Step 5 is executed and one more cycle of length three is added to the cactus. On the other hand, if T^* contains three edges then a cycle of length four is added to C. At the end of Step 6, C contains a set of cycles. The graph induced by C is obviously connected; every edge is present in exactly one cycle and it spans over all the vertices of the original graph. Therefore, C provides a spanning cactus of G. □

Next we obtain an upper bound on the weight of each three cycle and four cycle in the cactus produced. Let C^3 be a three cycle in the cactus. Then we prove that

Lemma 2. $w(C^3) \leq (1 + \tau) \times [w(e_1) + w(e'_1)]$ *and* $b(C^3) \leq 2\tau b(T^*)$, *where* e_1 *and* e'_1 *are the spanning tree edges in* C^3.