Available online at www.sciencedirect.com
SCIENCE DIRECT®
Discrete Applied Mathematics 143 (2004) 31–42
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

# Approximation algorithms for the optimal $p$-source communication spanning tree

Bang Ye Wu

*Department of Computer Science and Information Engineering, Shu-Te University, YanChau, Kaohsiung, Taiwan 824, ROC*

Received 14 August 2002; received in revised form 29 September 2003; accepted 8 October 2003

**Abstract**

The computational complexity and the approximation algorithms of the optimal $p$-source communication spanning tree ($p$-OCT) problem were investigated. Let $G$ be an undirected graph with nonnegative edge lengths. Given $p$ vertices as sources and all vertices as destinations, and also given arbitrary requirements between sources and destinations, we investigated the problem how to construct a spanning tree of $G$ such that the total communication cost from sources to destinations is minimum, where the communication cost from a source to a destination is the path length multiplied by their requirement. For any fixed integer $p \geqslant 2$, we showed that the problem is NP-hard even for metric graphs. For metric graphs of $n$ vertices, we show a 2-approximation algorithm with time complexity $O(n^{p-1})$. For general graphs, we present a 3-approximation algorithm for the case of two sources.


*Keywords:* Approximation algorithms; Network design; Spanning trees

## 1. Introduction

Consider the following *optimum communication spanning tree* (OCT) problem formulated by Hu [6]. Let $G=(V,E,w)$ be an undirected graph with nonnegative edge length function $w$. We are also given requirement $\lambda(u,v)$ for each pair $u$, $v$, of vertices. For any spanning tree $T$ of $G$, the communication cost between two vertices is defined to be the requirement multiplied by the path length of the two vertices on $T$, and the communication cost of $T$ is the total communication cost summed over all pairs of vertices. Our goal is to construct a spanning tree with minimum communication cost. That is, we want to find a spanning tree $T$ such that $\sum_{u,v\in V} \lambda(u,v)d_T(u,v)$ is minimized, where $d_T(u,v)$ is the distance between $u$ and $v$ on $T$.

The requirements in the OCT problem are arbitrary nonnegative numbers. By restricting the requirements, several special cases of the problem have been studied.

- $\lambda(u,v)=1$ for each $u,v\in V$. The problem is called the *minimum routing cost spanning tree* (MRCT) problem (also called the *shortest total path length spanning tree* problem), and is NP-hard [5,7]. The first constant ratio approximation algorithm for the MRCT problem consists in constructing a shortest path tree and showing that it is a 2-approximation [9]. The approximation ratio was improved to $(\frac{4}{3}+\epsilon)$ for any fixed $\epsilon>0$ [13], and then further improved to a *polynomial time approximation scheme* (PTAS) [10]. An exact algorithm for the problem was also studied [4]. Another related result concerns how to construct a subgraph with small routing cost and small size [14].
- $\lambda(u,v)=r(u)r(v)$ for each $u,v\in V$, where $r(v)$ is a given nonnegative vertex weight for each vertex $v$. This version of the problem is called the *optimal product-requirement communication spanning tree* (PROCT) problem. A 1.577-approximation algorithm for the PROCT problem was developed [11], and then improved to a PTAS [12].

*E-mail address:* bangye@mail.stu.edu.tw (B.Y. Wu).

0166-218X/$ - see front matter 
doi:10.1016/j.dam.2003.10.002

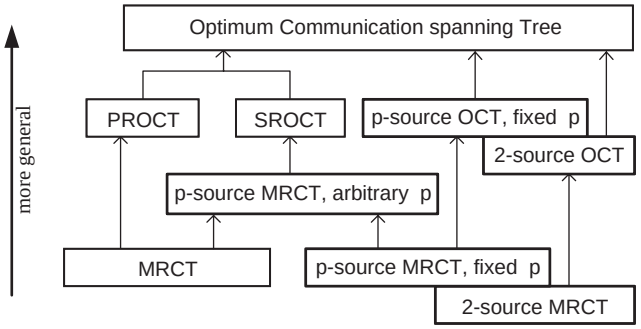


Fig. 1. The relationships between OCT problems.

- $\lambda(u,v)=r(u)+r(v)$ for each $u,v\in V$, where $r(v)$ is a given nonnegative vertex weight for each vertex $v$. This version of the problem is called the *optimal sum-requirement communication spanning tree* (SROCT) problem. A 2-approximation algorithm for the SROCT problem was shown [11].

In the *p-source MRCT* ($p$-MRCT) problem, we are given $p$ vertices as sources and all vertices (including the sources) as destinations. While the all-to-all distance is considered in the MRCT problem, the goal of the $p$-MRCT problem is to minimize the total distance from sources to destinations. The $p$-MRCT problem is a special case of the SROCT problem, in which the vertex weight of each source is one and the weights of all the other vertices are zeros. If there is only one source, it is always possible to find a spanning tree, called *shortest-path tree*, such that the path between the source and each vertex is a shortest path on the given graph. Therefore both the 1-MRCT and 1-OCT problems are polynomial-time solvable. However, the 2-MRCT problem was shown to be NP-hard even for metric graphs, and a PTAS for the problem was also proposed [15]. A metric graph is a complete graph with edge weights satisfying the triangle inequality.

In this paper, we investigate the *optimal p-source communication spanning tree* ($p$-OCT) problem. Let $G=(V,E,w)$ be the input graph. Given a set $S\subset V$ of $p$ vertices as sources and all vertices including the sources as destinations, the $p$-OCT of $G$ is a spanning tree $T$ of $G$ such that the total communication cost, defined by $\sum_{s\in S}\sum_{v}\lambda(s,v)d_T(s,v)$, is minimum. The $p$-OCT problem is a special case of the OCT problem, and in the meantime it is also a generalization of the $p$-MRCT problem. The previous result of the NP-hardness of the 2-MRCT problem only implies the NP-hardness of the 2-OCT problem, but it is not sufficient to show the computational complexity of the $p$-MRCT problem for other fixed $p$. It was pointed out that there is a simple reduction to show the NP-hardness of the $p$-MRCT problem for any *even* integer $p$, but the complexity for any odd integer $p$ is left open. In this paper, not surprisingly, we show the NP-hardness of the $p$-MRCT problem even for metric graphs and for any fixed $p\geqslant 2$. Then, the $p$-OCT problem is also shown to be NP-hard by a straightforward reduction. The proof generalizes the previous result and the reduction is more simple.

To approximate the $p$-OCT, we first focused on the case of metric graphs. For metric graphs, we begin with a simple 2-approximation algorithm for the 2-OCT and then generalize the algorithm to the case of $p$ sources. For any fixed integer $p\geqslant 2$, the algorithm finds a 2-approximation of the $p$-OCT of a metric graph in $O(n^{p-1})$ time. For general graphs, we present a 3-approximation algorithm for the 2-OCT.

The relationship of the different versions of the OCT problems is illustrated in Fig. 1, and the currently best approximation ratios are summarized in Table 1.

The remaining sections are organized as follows: In Section 2, some definitions and notations are given. The computational complexity is shown in Section 3. The algorithm for the $p$-OCT of metric graphs is presented in Section 4, and the approximation algorithms for the 2-OCT of general graphs is shown in Section 5. Finally concluding remarks are in Section 6.

## 2. Preliminaries

By $G=(V,E,w)$, we denote a graph $G$ with vertex set $V$, edge set $E$, and edge length (or edge weight) function $w$. In this paper, we consider only connected undirected graphs with nonnegative edge lengths, and an edge between vertices $u$ and $v$ is denoted by $(u,v)$. A metric graph is a complete undirected graph and the edge lengths satisfy the triangle inequality. For any graph $G$, $V(G)$ denotes its vertex set and $E(G)$ denotes its edge set. Let $w$ be the edge length function

Table 1
The restrictions and currently best ratios of the OCT problems

| Problem | Restriction | Ratio | Reference |
|---|---|---|---|
| OCT | Nonnegative | $O(\log n \log\log n)$ | [10] |
| PROCT | $\lambda(u,v) = r(u)r(v)$ | PTAS | [12] |
| SROCT | $\lambda(u,v) = r(u) + r(v)$ | 2 | [11] |
| MRCT | $\lambda(u,v) = 1$ | PTAS | [10] |
| 2-MRCT | $\lambda(u,v) = r(u) + r(v)$<br>$r(s_1) = r(s_2) = 1$<br>$r(v) = 0$ for $v \notin \{s_1, s_2\}$ | PTAS | [15] |
| $p$-OCT | $\lambda(u,v) = 0$ for $u, v \notin S$<br>$\lvert S\rvert = p$ is a constant | 2 (metric graphs) | This paper |
| 2-OCT | $\lambda(u,v) = 0$ for $u, v \notin \{s_1, s_2\}$ | 3 (general graphs) | This paper |

of a graph $G$. For a subgraph $H$ of $G$, we define $w(H) = w(E(H)) = \sum_{e\in E(H)} w(e)$. We shall also use $n$ to denote $|V(G)|$ when there is no ambiguity.

**Definition 1.** Let $G = (V, E, w)$ be a graph. For $u, v \in V$, $SP_G(u,v)$ denotes a shortest path between $u$ and $v$ on $G$. The shortest path length is denoted by $d_G(u,v) = w(SP_G(u,v))$.

**Definition 2.** Let $H$ be a subgraph of $G$. For a vertex $v \in V(G)$, we use $d_G(v,H)$ to denote the shortest distance from $v$ to $H$, i.e., $d_G(v,H) = \min_{u\in V(H)} d_G(v,u)$. The definition also includes the case in which $H$ contains no edge.

Let $G = (V, E, w)$ be a graph and $m \in V$. The shortest path tree rooted at $m$ is a spanning tree $T$ such that $d_T(m,v) = d_G(m,v)$ for any $v \in V$. The shortest path tree problem has been well studied and efficient algorithms for various families of graphs have been developed [2]. For example, Dijkstra's algorithm [3] finds a shortest path tree for a graph with nonnegative weights in $O(|V|^2)$ time, and the time complexity can be improved to $O(|E| + |V|\log|V|)$ by using Fibonacci heaps. A recent result for undirected graphs with positive integer weights is an $O(|E|)$ algorithm [8]. Let $M \subset V$. A forest $F$ is a *shortest path forest* with roots $M$ if $d_F(v,M) = d_G(v,M)$ for any $v \in V$, i.e., each vertex is connected to the closest root by a shortest path. A shortest path forest can be constructed by an algorithm similar to the shortest path tree algorithm. First we create a dummy node and the multiple roots are connected to the dummy node by edges of zero weight. Then a shortest path tree rooted at the dummy node is constructed, and the shortest path forest is obtained by removing the dummy node and dummy edges. The time complexity is the same as the shortest path tree algorithm.

We now define the communication cost.

**Definition 3.** Let $T$ be a spanning tree of a graph $G$ and $S = \{s_1, s_2, \ldots, s_p\} \subset V(T)$ be the set of given sources. For any vertex $v \in V(T)$, the communication cost of $v$ on $T$ is defined by $c_T(v) = \sum_{i=1}^{p} r_i(v) d_T(v, s_i)$, where $r_i(v)$ is the given nonnegative requirement between $s_i$ and $v$. The communication cost of $T$ is defined by $c(T) = \sum_{v\in V(T)} c_T(v)$.

**Definition 4.** Given a graph, a set of $p$ sources, and the requirements, the optimal $p$-source communication spanning tree ($p$-OCT) is a spanning tree with minimum communication cost. The problem of finding the $p$-OCT is called as the $p$-OCT problem.

## 3. The NP-hardness

In this section, we shall discuss the computational complexity of the $p$-OCT problem. First we define a weighted version of the 2-MRCT problem. By a transformation from the well-known satisfiability problem, we show the weighted 2-MRCT problem is NP-hard. Then, we show that the $p$-MRCT problem for any fixed $p$ can be transformed from the weighted 2-MRCT problem, and the NP-hardness of the $p$-OCT problem is shown by a straightforward reduction.

We first introduce the satisfiability problem. Let $U = \{u_1, u_2, \ldots, u_n\}$ be a set of Boolean *variables*. A *truth assignment* for $U$ is a function mapping each variable to TRUE or FALSE. If $u$ is a variable in $U$, then $u$ and $\bar{u}$ (the negation of $u$) are literals over $U$. A *clause* over $U$ is a set of literals over $U$, which represents the disjunction of those literals and is *satisfied* by a truth assignment if and only if at least one of its members is assigned TRUE. For a set $X$ of clauses over

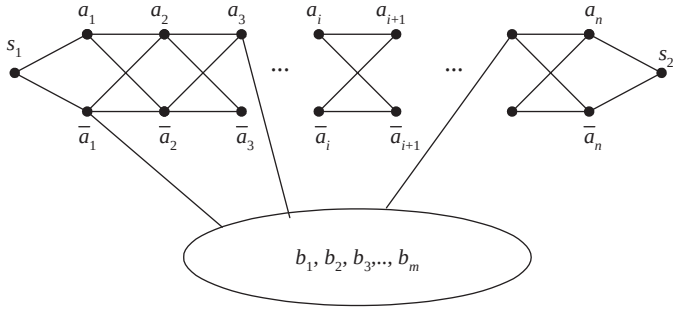


Fig. 2. The transformation from the SAT problem to the 2-MRCT($\alpha$) problem.

$U$, a truth assignment $t$ is called a *satisfying truth assignment* for $X$ if all the clauses in $X$ are simultaneously satisfied by $t$. A set of clauses is *satisfiable* if and only if there exists some satisfying truth assignment.

**Definition 5.** Given a set $U$ of variables and a set $X$ of clauses over $U$, the SATISFIABILITY (SAT) problem consists in determining if there is a satisfying truth assignment for $X$.

**Definition 6.** Let $G=(V,E,w)$ be a graph and $s_1,s_2 \in V$ be two given sources. For any integer $\alpha \geqslant 1$, the 2-MRCT($\alpha$) problem consists in finding a spanning tree $T$ of $G$ such that the weighted routing cost $c(T,\alpha)=\sum_{v\in V}(\alpha d_T(v,s_1)+d_T(v,s_2))$ is minimum.

We shall transform the SAT problem to the 2-MRCT($\alpha$) problem (Fig. 2). Given a set $U=\{u_1,u_2,\ldots,u_n\}$ of Boolean variables and a set $X=\{x_1,x_2,\ldots x_m\}$ of clauses as an instance of the SAT problem, we construct a graph $G=(V,E,w)$ as follows:

- Let $A=\{a_i,\bar{a}_i|1\leqslant i\leqslant n\}$ and $B=\{b_i|1\leqslant i\leqslant m\}$. The vertex set $V=\{s_1,s_2\}\cup A\cup B$, in which $s_1$ and $s_2$ are the two sources. The vertices $a_i$ and $\bar{a}_i$ correspond to variable $u_i$ and negated variable $\bar{u}_i$, respectively, and $b_i$ corresponds to clause $x_i$.
- The edge set $E$ contains the following subsets:
  (1) $E_1=\{(s_1,a_1),(s_1,\bar{a}_1),(s_2,a_n),(s_2,\bar{a}_n)\}$.
  (2) $E_2=\bigcup_{1\leqslant i<n}\{(a_i,a_{i+1}),(a_i,\bar{a}_{i+1}),(\bar{a}_i,a_{i+1}),(\bar{a}_i,\bar{a}_{i+1})\}$.
  (3) $E_3=\{(a_i,b_j)|u_i\in x_j,\forall i,j\}\cup\{(\bar{a}_i,b_j)|\bar{u}_i\in x_j,\forall i,j\}$.
- For any $e\in E_1\cup E_2$, $w(e)=1$. For any $(a_i,b_j)$ or $(\bar{a}_i,b_j)$ in $E_3$, the edge weight is $L-(\alpha-1)/(\alpha+1)i$, in which $L=(\alpha+1)mn$.

We shall show that the SAT problem has a satisfying truth assignment if and only if there is a spanning tree $T$ of $G$ such that the weighted routing cost $c(T,\alpha)\leqslant\beta$, where

$$\beta=(n+1)^2\alpha+(n^2+4n+1)+((\alpha+1)L+n+1)m.$$

**Proposition 1.** *If there is a truth assignment satisfying $X$, there exists a spanning tree $Y$ of $G$ such that $c(Y,\alpha)=\beta$.*

**Proof.** We may construct a corresponding spanning tree $Y$ of $G$ as follows.

- The path $P_Y$ between the two sources has the form

$$(s_1=v_0,v_1,v_2,\ldots,v_n,s_2),$$

  in which, for $1\leqslant i\leqslant n$, $v_i=a_i$ if $u_i$ is assigned true and $v_i=\bar{a}_i$ otherwise.
- For all $1\leqslant i\leqslant n$, if $v_i=a_i$, connect $\bar{a}_i$ to $v_{i-1}$. Otherwise, connect $a_i$ to $v_{i-1}$.
- For all $1\leqslant i\leqslant m$, connect $b_i$ to one of the vertices on $P_Y$. Such an edge always exists since at least one of the literals in $x_i$ is assigned true.

For any $v \in \{a_i, \bar{a}_i\}$, if $v$ is on the path $P_Y$, $d_Y(v,s_1) = i$ and $d_Y(v,s_2) = n+1-i$, and otherwise $d_Y(v,s_1) = i$ and $d_Y(v,s_2) = n+3-i$. For any $b_i$, since it is connected to some $a_j$ (a similar argument holds for $\bar{a}_j$) on $P_Y$,

$$\begin{aligned}\alpha d_Y(b_i,s_1) + d_Y(b_i,s_2) &= (\alpha+1)w(b_i,a_j) + \alpha j + (n+1-j)\\ &= (\alpha+1)L - (\alpha-1)j + (\alpha-1)j + (n+1)\\ &= (\alpha+1)L + n + 1.\end{aligned}$$

Note that the cost depends only on whether $b_i$ is directly connected to path $P_Y$ or not, but not on which vertex of the path it is connected to.

The routing cost of $Y$ is given by

$$\begin{aligned}&c(Y,\alpha)\\ &=(\alpha+1)d_Y(s_1,s_2) + \sum_{v\in A}(\alpha d_Y(v,s_1) + d_Y(v,s_2))\\ &\quad + \sum_i (\alpha d_Y(b_i,s_1) + d_Y(b_i,s_2))\\ &=(\alpha+1)(n+1) + \alpha\sum_{i\leqslant n}(2i) + \sum_{i\leqslant n}(2n+4-2i) + m((\alpha+1)L+n+1)\\ &=(n+1)^2\alpha + (n^2+4n+1) + ((\alpha+1)L+n+1)m\\ &=\beta. \qquad \square\end{aligned}$$

**Proposition 2.** *Let $T$ be an optimal solution of 2-MRCT($\alpha$) on $G$. If $c(T,\alpha) \leqslant \beta$, the path $P_T$ from $s_1$ to $s_2$ on $T$ has the form $(s_1, v_1, v_2, \ldots, v_n, s_2)$, in which $v_i \in \{a_i, \bar{a}_i\}$ for $1 \leqslant i \leqslant n$.*

**Proof.** If the proposition was false, we would have that $P_T$ contains some vertex in $B$ or that it contains more than $n$ vertices in $A$. In the following, we show that both cases lead to contradictions.

- Suppose that $P_T$ contains some $b_i$. This implies that $d_T(s_1,s_2) > 2L - n$ since $d_G(b_i,s_1) > L$ and $d_G(b_i,s_2) > L - n$. For any vertex $v \in A$,

  $$\alpha d_T(v,s_1) + d_T(v,s_2) \geqslant d_T(s_1,s_2) > 2L - n.$$

  For any vertex $v \in B$,

  $$\alpha d_T(v,s_1) + d_T(v,s_2) \geqslant \alpha d_G(v,s_1) + d_G(v,s_2) > (\alpha+1)L - n.$$

  Therefore, since $L = (\alpha+1)mn$,

  $$\begin{aligned}c(T,\alpha) &> 2n(2L-n) + m((\alpha+1)L - n)\\ &= 4nL - 2n^2 - mn + (\alpha+1)mL\\ &> 4n^2m\alpha + n^2m + (\alpha+1)mL.\end{aligned}$$

  Comparing with

  $$\beta = (n+1)^2\alpha + (n^2+4n+1) + ((\alpha+1)L + n + 1)m,$$

  we have $c(T,\alpha) > \beta$ when $m,n > 2$, and it is a contradiction.
- Suppose that the path $P_T$ contains more than $n$ vertices in $A$. It implies that $d_T(s_1,s_2) > n+1$ and that there exists the smallest $i$ such that both $a_i$ and $\bar{a}_i$ are on the path. By the definition of $G$, without loss of generality, we may assume that the path $P_T = (\ldots, a_i, a_{i+1}, \bar{a}_i, \bar{a}_{i+1}, \ldots)$. However, if we replace edge $(a_{i+1}, \bar{a}_i)$ with $(a_i, \bar{a}_{i+1})$, we may obtained another spanning tree and its cost is smaller than $T$ since the costs for $a_{i+1}$ and $\bar{a}_i$ are decreased and the costs for any other vertex is not increased. $\square$

**Proposition 3.** *Let $T$ be an optimal solution of 2-MRCT($\alpha$) on $G$. If $c(T,\alpha) \leqslant \beta$, there is a truth assignment satisfying $X$.*

**Proof.** By Proposition 2, the path $P_T$ from $s_1$ to $s_2$ on $T$ has the form $(s_1, v_1, v_2, \ldots, v_n, s_2)$, in which $v_i \in \{a_i, \bar{a}_i\}$ for $1 \leqslant i \leqslant n$. For any $a_i$ (a similar argument holds for $\bar{a}_i$) on $P_T$,

$$\alpha d_T(a_i, s_1) + d_T(a_i, s_2) = i\alpha + n + 1 - i.$$

For any $a_i$ (a similar argument holds for $\bar{a}_i$) not on $P_T$,

$$\begin{aligned}
&\alpha d_T(a_i, s_1) + d_T(a_i, s_2) \\
&= \min_{v \in V(P_T)} \{(\alpha+1) d_T(a_i, v) + \alpha d_T(v, s_1) + d_T(v, s_2)\} \\
&\geqslant (\alpha+1) + \alpha(i-1) + (n+1-(i-1)) \\
&= i\alpha + n + 3 - i.
\end{aligned}$$

The lower bound is obtained when $a_i$ is connected to $v_{i-1}$ on $P_T$. For any $b_i$, similar to the proof of Proposition 1,

$$\alpha d_T(b_i, s_1) + d_T(b_i, s_2) \geqslant (\alpha+1)L + n + 1.$$

The lower bound is obtained when $b_i$ is connected to some vertex on $P_T$. Consequently,

$$\begin{aligned}
&c(T, \alpha) \\
&\geqslant (\alpha+1)(n+1) + \sum_{i=1}^{n} ((i\alpha + n + 1 - i) + (i\alpha + n + 3 - i)) \\
&\quad + ((\alpha+1)L + n + 1)m \\
&= (n+1)^2\alpha + (n^2 + 4n + 1) + ((\alpha+1)L + n + 1)m = \beta.
\end{aligned}$$

The lower bound is obtained when each vertex in $B$ is connected to some $v_i$ on $P_T$. It implies that, for each clause in $X$, there is a literal assigned TRUE, and hence $X$ is satisfiable. □

**Theorem 1.** *For any fixed integer $\alpha \geqslant 1$, the 2-MRCT($\alpha$) problem is NP-hard.*

**Proof.** By Propositions 1 and 3, we have transformed the SAT problem to the 2-MRCT($\alpha$) problem. Given an instance of the SAT problem, in polynomial time, we can construct an instance of the 2-MRCT($\alpha$) problem. If there is a polynomial time algorithm finding the optimal solution of the 2-MRCT($\alpha$) problem, it can also be used to solve the SAT problem. Therefore the 2-MRCT($\alpha$) problem is NP-hard since the SAT problem is NP-complete [1,5]. □

The NP-hardness result can be easily extended to metric graphs.

**Corollary 2.** *For any fixed integer $\alpha \geqslant 1$, the 2-MRCT($\alpha$) problem is NP-hard even for metric graphs.*

**Proof.** Let $G$ be the constructed graph in Theorem 1, and $\bar{G} = (V, V \times V, \bar{w})$ be the metric closure of $G$, that is, $\bar{w}(u, v) = d_G(u, v)$ for all $u, v \in V$. Clearly $\bar{G}$ is a metric graph. The transformation is similar to Propositions 1–3. □

Let $p \geqslant 1$ be any fixed integer. We can easily transform the 2-MRCT($p$) problem to the $p$-MRCT by duplicating $p$ copies of the source $s_1$. The next corollary is obvious and the proof is omitted.

**Corollary 3.** *For any fixed integer $p \geqslant 1$, the $p$-MRCT is NP-hard even for metric inputs.*

Since the OCT problem includes the MRCT problem as a special case, we have the following result.

**Corollary 4.** *For any fixed integer $p \geqslant 1$, the $p$-OCT is NP-hard even for metric inputs.*

## 4. Approximating $p$-OCT on metric graphs

In this section, we show a 2-approximation algorithm for the $p$-OCT problem with metric inputs. A metric graph is a complete graph for which the edge between any pair of vertices is a shortest path. We start with a simple algorithm for the case of two sources, and then generalize to $p$-OCT for any fixed integer $p$.

### 4.1. Approximating the 2-OCT

To approximate the 2-OCT, our algorithm starts at the edge $(s_1, s_2)$, and then inserts other vertices one by one in arbitrary order. In each iteration, we greedily connect a vertex $v$ to either $s_1$ or $s_2$ depending on the communication cost. We shall show the approximation ratio is two. The algorithm is given in the following.

**Algorithm A1**
**Input:** A metric graph $G$, two vertices $s_1, s_2$, and requirements $r_1(v), r_2(v)$, for each vertex $v \in V(G)$.
**Output:** A spanning tree $T$ of $G$.
Initially $T$ contains only one edge $(s_1, s_2)$.
For each vertex $v \in V$ do
/* Connect each $v$ to either $s_1$ or $s_2$ */
If $(r_1(v) + r_2(v))w(v, s_1) + r_2(v)w(s_1, s_2)$
$\leqslant (r_1(v) + r_2(v))w(v, s_2) + r_1(v)w(s_1, s_2)$
insert edge $(v, s_1)$ into $T$;
else
insert edge $(v, s_2)$ into $T$;
endif
Output $T$.
For convenience, we define some notations.

**Definition 7.** Let $Y$ be the 2-OCT and $P$ be the path between $s_1$ and $s_2$ on $Y$. We define $f_1(v) = d_Y(v, s_1) - d_Y(v, P)$ and $f_2(v) = d_Y(v, s_2) - d_Y(v, P)$ for each vertex $v$.

The next lemma gives a formula of the optimal cost. The formula directly follows the above notations and the definition of the communication cost. We omit the proof.

**Lemma 5.** *Let $Y$ be the 2-OCT and $P$ be the path between $s_1$ and $s_2$ on $Y$.* $c(Y) = \sum_{v \in V} ((r_1(v) + r_2(v))d_Y(v, P) + r_1(v)f_1(v) + r_2(v)f_2(v))$.

The next theorem shows that Algorithm A1 is an approximation algorithm. Here we assume that the input graph is already in the memory.

**Theorem 6.** *Algorithm A1 computes a 2-approximation of the* 2-OCT of a metric graph in O($n$) *time.*

**Proof.** Using adjacency lists to store the tree, the insertion of an edge takes only constant time. Therefore, the time complexity of the algorithm is obviously O($n$). We now prove the performance ratio. Let $Y$ be the 2-OCT and $P$ be the path between $s_1$ and $s_2$ on $Y$. By the triangle inequality, $w(v, s_1) \leqslant d_Y(v, s_1) = d_Y(v, P) + f_1(v)$. We have

$$\begin{aligned}
&(r_1(v) + r_2(v))w(v, s_1) + r_2(v)w(s_1, s_2) \\
&\quad \leqslant (r_1(v) + r_2(v))(d_Y(v, P) + f_1(v)) + r_2(v)(f_1(v) + f_2(v)) \\
&\quad = (r_1(v) + r_2(v))d_Y(v, P) + (f_1(v)r_1(v) + f_2(v)r_2(v)) + 2f_1(v)r_2(v) \\
&\quad = c_Y(v) + 2f_1(v)r_2(v).
\end{aligned}$$

That is, if $v$ is connected to $s_1$, the cost of $v$ is increased by at most $2f_1(v)r_2(v)$. Similarly $w(v, s_2) \leqslant d_Y(v, s_2) = d_Y(v, P) + f_2(v)$ and we have

$$\begin{aligned}
&(r_1(v) + r_2(v))w(v, s_2) + r_1(v)w(s_1, s_2) \\
&\quad \leqslant (r_1(v) + r_2(v))(d_Y(v, P) + f_2(v)) + r_1(v)(f_1(v) + f_2(v)) \\
&\quad = (r_1(v) + r_2(v))d_Y(v, P) + (f_1(v)r_1(v) + f_2(v)r_2(v)) + 2f_2(v)r_1(v) \\
&\quad = c_Y(v) + 2f_2(v)r_1(v).
\end{aligned}$$

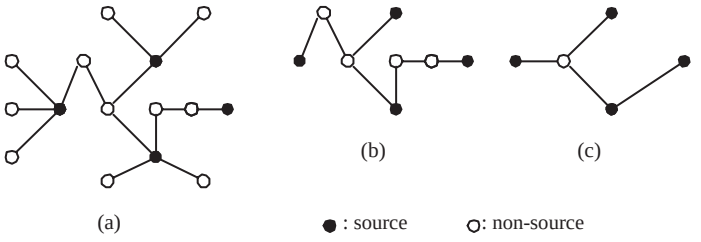


Fig. 3. (a) A tree with four sources, (b) the skeleton and (c) the reduced skeleton.

That is, if $v$ is connected to $s_2$, the cost of $v$ is increased by at most $2f_2(v)r_1(v)$. Since the vertex $v$ is connected to either $s_1$ or $s_2$ by choosing the minimum of the two costs,

$$c_T(v) \leqslant c_Y(v) + \min\{2f_1(v)r_2(v), 2f_2(v)r_1(v)\}.$$

Since the minimum of two number is no more than their weighted mean, we have

$$\begin{aligned} & c_Y(v) + \min\{2f_1(v)r_2(v), 2f_2(v)r_1(v)\} \\ & \quad \leqslant c_Y(v) + \frac{r_1(v)^2}{r_1(v)^2 + r_2(v)^2} 2f_1(v)r_2(v) + \frac{r_2(v)^2}{r_1(v)^2 + r_2(v)^2} 2f_2(v)r_1(v) \\ & \quad = c_Y(v) + \frac{2r_1(v)r_2(v)}{r_1(v)^2 + r_2(v)^2} \times (f_1(v)r_1(v) + f_2(v)r_2(v)). \end{aligned} \tag{1}$$

Since $r_1(v)^2 + r_2(v)^2 - 2r_1(v)r_2(v) = (r_1(v) - r_2(v))^2 \geqslant 0$, we have

$$c_T(v) \leqslant c_Y(v) + f_1(v)r_1(v) + f_2(v)r_2(v) \leqslant 2c_Y(v). \tag{2}$$

We have shown that $c_T(v) \leqslant 2c_Y(v)$ for any vertex $v$. Therefore $c(T) = \sum_{v \in V} c_T(v) \leqslant 2 \sum_{v \in V} c_Y(v) = 2c(Y)$, and $T$ is a 2-approximation of the optimal. □

### 4.2. The reduced skeleton of a tree

To analyze the approximation algorithm, we define the *reduced skeleton* of a tree as follows.

**Definition 8.** Let $T$ be a spanning tree of a metric graph $G$ and $S \subset V(T)$ be the set of sources. The *S-skeleton* of $T$ is a tree $Y$ defined by $\bigcup_{u,v \in S} SP_T(u,v)$. The *reduced S-skeleton* of $T$ is a tree $X = (V(X), E(X), w)$ defined by the following:

- $V(X)$ is the union of the source set and the set of vertices whose degrees on the skeleton $Y$ are not two.
- For $u, v \in V(X)$, $(u,v) \in E(X)$ if and only if the path $SP_Y(u,v)$ contains no other vertex in $V(X)$.
- For each $e \in E(X)$, $w(e) = d_G(u,v)$.

An example of the $S$-skeleton and reduced $S$-skeleton of a tree is illustrated in Fig. 3. The $S$-skeleton may be easily obtained by repeatedly removing the non-source leaves from the tree, i.e., the leaves of the skeleton must be sources. The reduced skeleton is obtained from the skeleton by eliminating the non-source vertices with degree two. Our approximation algorithm tries to guess the reduced $S$-skeleton $X$ of the OCT, and the other vertices are connected to one of the vertex of $X$ by making the cost as small as possible. Similar to the case of two sources, it can be shown that the approximation ratio is two.

When there are only two sources, the reduced skeleton is the edge between the two sources and can be easily determined. But the structure of the reduced skeleton becomes more complicated as the number of sources increases. The reduced skeleton is a tree spanning $S$ and possibly some other vertices. In the next lemma, we show that the number of vertices of $X$ is bounded by $2|S| - 2$. In other words, there are at most $|S| - 2$ non-source vertices in $X$. For a constant $p = |S|$, in polynomial time one can check all trees spanning the $p$ given sources and at most $(p-2)$ non-source vertices in order to guess $X$.

**Lemma 7.** *Let $X$ be the reduced $S$-skeleton of $T$. For any $u, v \in V(X)$, $d_X(u,v) \leqslant d_T(u,v)$. Furthermore $|V(X)| \leqslant 2|S| - 2$.*

**Proof.** By definition, the $S$-skeleton of $T$ is the union of the path between any pair of vertices in $S$. For any two vertices of the skeleton, the distance on the skeleton remains the same as the one on the tree $T$. Since the graph $G$ is metric, the edge weight is no more than the length of any path between the two endpoints. Consequently $d_X(u,v) \leqslant d_T(u,v)$ for any $u, v \in V(X)$.

Let $n_1$ and $n_2$ be the number of leaves and internal vertices, respectively. By definition, the leaf set of $X$ is a subset of $S$, and therefore $n_1 \leqslant |S|$ and there are $(|S| - n_1)$ source vertices which are internal vertices. Since $X$ is a tree, the number of edges is $(n_1 + n_2 - 1)$ and the sum of degrees of all vertices is $2(n_1 + n_2 - 1)$. Since the reduced skeleton contains no source vertex of degree two, we have

$$n_1 + 3(n_2 - (|S| - n_1)) + 2(|S| - n_1) \leqslant 2(n_1 + n_2 - 1)$$

and therefore,

$$n_1 + n_2 \leqslant n_1 + |S| - 2 \leqslant 2|S| - 2. \qquad \square$$

### 4.3. Approximating the p-OCT

In this subsection, we generalize the 2-approximation algorithm for the 2-OCT to the case of $p$ sources, where $p \geqslant 2$ is a constant. Our approximation algorithm is shown below.

**Algorithm A2**
**Input:** A metric graph $G = (V, E, w)$, a set $S \subset V$ of $p$ sources,
and requirement $r_i(v)$ for each $s_i \in S$ and $v \in V$.
**Output:** A spanning tree $T$ of $G$.

```
For each set V1 of (p − 2) vertices in V \ S do
        For each tree X with vertex set S ∪ V1 do
                Initially T = X;
                For each vertex v ∉ V(X) do
/* Connect each v to a vertex in V(X) */
                        For each u ∈ V(X), compute Σ_i r_i(v)(w(v,u) + d_X(u,s_i))
                        and choose the vertex u* minimizing the cost;
                        Insert edge (v,u*) into T;
                Compute the cost of T and keep the best tree found so far;
Output T.
```

**Theorem 8.** *For a metric graph, Algorithm A2 finds a 2-approximation of the $p$-source OCT in $O(n^{p-1})$ time, where $p \geqslant 2$ is a constant.*

**Proof.** The algorithm tries each tree $X$ spanning the $p$ sources and $(p-2)$ other vertices. The total number of such trees is $\binom{n-p}{p-2}(2p-2)^{2p-4}$. For each $X$ and each $v \in V \setminus V(X)$, it takes $O(p)$ time to determine a vertex $u^* \in V(X)$ and insert edge $(v, u^*)$. The total time complexity is therefore $O(n^{p-1})$ since $p$ is a constant.

Let $Y$ be the optimal solution and $\bar{X}$ be the reduced $S$-skeleton of $Y$. By Lemma 7, $|V(\bar{X})| \leqslant 2p - 2$. First we show the approximation ratio for the case that $|V(\bar{X})| = 2p - 2$, and the other case, i.e. $|V(\bar{X})| < 2p - 2$, will be explained later. Since the algorithm tries all possible trees spanning the sources and $(p - 2)$ other vertices, it must happen that $X = \bar{X}$ in some iteration. It is sufficient to show the approximation ratio of the tree constructed in this iteration because the algorithm outputs the best of the trees constructed in all iterations.

Let $(u_1, u_2) \in E(X)$ and $P = SP_Y(u_1, u_2)$. Removing the edges of $P$ from $Y$, the tree is cut into several components. Let $Y_1$ and $Y_2$ be the components containing $u_1$ and $u_2$ respectively, and $B(u_1, u_2) = V(Y) \setminus (V(Y_1) \cup V(Y_2))$ be the set of the vertices not in $Y_1$ or $Y_2$. Also let $S_1$ and $S_2$ denote the set of the sources in $T_1$ and $T_2$, respectively. Since $X$ is the reduced $S$-skeleton, $S = S_1 \cup S_2$ and $B(u_1, u_2)$ contains no source. The subsets $V(X)$ and $B(u_1, u_2)$ for all $(u_1, u_2) \in E(X)$ form a partition of the vertex set $V(Y)$. Since $d_T(u,v) = d_Y(u,v)$ for any $u, v \in V(X)$ by Lemma 7, we have $c_T(v) = c_Y(v)$ for each $v \in V(X)$. We shall prove that, for any $(u_1, u_2) \in E(X)$ and any vertex $v \in B(u_1, u_2)$, $c_T(v) \leqslant 2c_Y(v)$, and therefore $T$ is a 2-approximation of $Y$.

To simplify the proof, we use the following notations. For $v \in V$, let $R_1 = \sum_{s_i \in S_1} r_i(v)$ and $R_2 = \sum_{s_i \in S_2} r_i(v)$ be the total requirements between $v$ and all sources in $S_1$ and $S_2$, respectively. As in the previous subsection, we define $f_1(v) = d_Y(v,u_1) - d_Y(v,P)$ and $f_2(v) = d_Y(v,u_2) - d_Y(v,P)$. Since the vertex $v$ is connected to one of the vertices in $V(X)$ by making the cost as small as possible, the cost $c_T(v)$ is no more than what it would be in the case that $v$ is connected to $u_1$.

$$c_T(v) \leqslant (R_1 + R_2)w(v,u_1) + \sum_{s_i \in S_1} r_i(v)d_T(u_1,s_i) + R_2(v)w(u_1,u_2) + \sum_{s_i \in S_2} r_i(v)d_T(u_2,s_i). \tag{3}$$

Since $w(v,u_1) \leqslant d_Y(v,P) + f_1(v)$ and $w(u_1,u_2) \leqslant f_1(v) + f_2(v)$,

$$(R_1 + R_2)w(v,u_1) + R_2 w(u_1,u_2) \leqslant (R_1 + R_2)d_Y(v,P) + (f_1(v)R_1 + f_2(v)R_2) + 2f_1(v)R_2. \tag{4}$$

By definition, the cost $c_Y(v)$ can be computed as follows:

$$c_Y(v) = (R_1 + R_2)d_Y(v,P) + f_1(v)R_1 + f_2(v)R_2 + \sum_{s_i \in S_1} r_i(v)d_T(u_1,s_i) + \sum_{s_i \in S_2} r_i(v)d_T(u_2,s_i). \tag{5}$$

By Eqs. (3)–(5):

$$c_T(v) \leqslant c_Y(v) + 2f_1(v)R_2. \tag{6}$$

Similarly, the cost $c_T(v)$ is no more than what it would be in the case that $v$ is connected to $u_2$, and we have

$$c_T(v) \leqslant c_Y(v) + 2f_2(v)R_1. \tag{7}$$

By taking the minimum of the bounds in Eqs. (6) and (7), and using the similar technique of Eqs. (1) and (2) in the proof of Theorem 6, we obtain

$$c_T(v) \leqslant c_Y(v) + \min\{2f_1(v)R_2, 2f_2(v)R_1\} \leqslant 2c_Y(v).$$

Therefore $T$ is a 2-approximation of $Y$.

Finally, we show the approximation ratio for the case $|V(\bar{X})| < 2p-2$, i.e. the reduced skeleton has less than $(2p-2)$ vertices. There exists $X$ constructed in some iteration such that $E(\bar{X}) \subset E(X)$ and, for each $v \in V(X) \setminus V(\bar{X})$, $v$ is connected to the vertex of $\bar{X}$ to minimize the cost of $v$. In other words, the case is included in the case $|V(\bar{X})| = 2p-2$ and the approximation ratio is two. □

## 5. Approximating 2-OCT on general graphs

In this section, we present the approximation algorithm of the 2-OCT problem in the case that the input is a general graph. Our approximation algorithm consists in finding a shortest path between the two sources and then constructing a shortest path forest with all the vertices of the path as the multiple roots. The output tree is the union of the forest and the path. The algorithm is exactly the same as the one which finds a 2-approximation of the 2-MRCT [15]. However, for the 2-OCT problem, we shall show that the approximation ratio is three. The algorithm is listed below.

**Algorithm A3**
**Input:** A graph $G = (V,E,w)$, two sources $s_1, s_2$, and requirements $r_1(v), r_2(v)$.
**Output:** A spanning tree $T$ of $G$.
Find a shortest path $X$ between $s_1$ and $s_2$ on $G$.
Find the shortest path forest with multiple roots in $V(X)$.
Let $T$ be the union of the forest and $X$. Output $T$.

We now show the approximation in the next lemma, in which $T$ is the spanning tree obtained by the approximation algorithm.

**Lemma 9.** *Let $Y$ be the 2-OCT. For any vertex $v$, $c_T(v) \leqslant 3c_Y(v)$.*

**Proof.** Suppose that $v$ is connected to the path $X$ at vertex $x$ of $X$. In other words, among the trees of the shortest path forest, $x$ is the root of the tree containing $v$. By the property of shortest path forests, $d_T(v,x) = d_G(v,x) \leqslant d_G(v,s_1)$. Since $X$ is a shortest path between $s_1$ and $s_2$, $d_T(s_1,x) = d_G(s_1,x) \leqslant d_G(s_1,v) + d_G(v,x)$. Therefore,

$$d_T(v,s_1) = d_T(v,x) + d_T(x,s_1) \leqslant 2d_G(v,x) + d_G(s_1,v) \leqslant 3d_G(v,s_1).$$

Similarly, $d_T(v,s_2) \leqslant 3d_G(v,s_2)$. By the definition of the communication cost:

$$\begin{aligned} c_T(v) &= r_1(v)d_T(v,s_1) + r_2(v)d_T(v,s_2) \\ &\leqslant 3r_1(v)d_G(v,s_1) + 3r_2(v)d_G(v,s_2) \\ &\leqslant 3(r_1(v)d_Y(v,s_1) + r_2(v)d_Y(v,s_2)) \\ &= 3c_Y(v). \qquad \square \end{aligned}$$

The following theorem summarize our result for the 2-OCT problem on general graphs.

**Theorem 10.** *The algorithm A3 computes a 3-approximation of the 2-OCT of a general graph $G = (V,E,w)$ in $O(|E| + |V|\log|V|)$ time. The time complexity can be reduced to $O(|E|)$ if the edge weights are integers.*

**Proof.** By Lemma 9, $c_T(v) \leqslant 3c_Y(v)$ for any vertex $v$. Since $c(T) = \sum_v c_T(v)$, $c(T) \leqslant 3c(Y)$. The time complexity is dominated by the time of finding the shortest path forest. As mentioned in Section 2, the shortest path forest can be found in $O(|E| + |V|\log|V|)$ time, and the time complexity can be reduced to $O(|E|)$ if the edge weights are all integers. $\square$

## 6. Concluding remarks

The main open question left in the paper is how to approximate the $p$-OCT of general graphs. Algorithm A2 only works for metric graphs, and we did not find a similar result for general graphs. The time complexity of Algorithm A2 is $O(n^{p-1})$ which is not efficient at all for large $p$. It would be interesting to find efficient algorithms to approximate the $p$-OCT with good ratio. Another interesting approach concerns the development of an approximation scheme, by which one can control the trade-off between the time complexity and the approximation ratio.

## References

[1] S.A. Cook, The complexity of theorem-proving procedures, Proceedings of the Third Annual ACM Symposium on Theory of Computing, 1971, pp. 151–158.
[2] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1994.
[3] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[4] M. Fischetti, G. Lancia, P. Serafini, Exact algorithms for minimum routing cost trees, Networks 39 (2002) 161–173, DOI 10.1002/net.10022.
[5] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, San Francisco, 1979.
[6] T.C. Hu, Optimum communication spanning trees, SIAM J. Comput. 3 (1974) 188–195.
[7] D.S. Johnson, J.K. Lenstra, A.H.G. Rinnooy Kan, The complexity of the network design problem, Networks 8 (1978) 279–285.
[8] M. Thorup, Undirected single source shortest paths with positive integer weights in linear time, J. ACM 46 (1999) 362–394.
[9] R. Wong, Worst-case analysis of network design problem heuristics, SIAM J. Algebraic Discrete Methods 1 (1980) 51–63.
[10] B.Y. Wu, A polynomial time approximation scheme for the two-source minimum routing cost spanning trees, J. Algorithms 44 (2002) 359–378 doi:10.1016/S0196-6774(02)00205-5.
[11] B.Y. Wu, K.M. Chao, C.Y. Tang, Approximation algorithms for some optimum communication spanning tree problems, Discrete Appl. Math. 102 (2000) 245–266.

[12] B.Y. Wu, K.M. Chao, C.Y. Tang, Approximation algorithms for the shortest total path length spanning tree problem, Discrete Appl. Math. 105 (2000) 273–289.
[13] B.Y. Wu, K.M. Chao, C.Y. Tang, A polynomial time approximation scheme for optimal product-requirement communication spanning trees, J. Algorithms 36 (2000) 182–204, doi:10.1006/jagm.2000.1088.
[14] B.Y. Wu, K.M. Chao, C.Y. Tang, Light graphs with small routing cost, Networks 39 (2002) 130–138, DOI 10.1002/net.10019.
[15] B.Y. Wu, G. Lancia, V. Bafna, K.M. Chao, R. Ravi, C.Y. Tang, A polynomial time approximation scheme for minimum routing cost spanning trees, SIAM J. Comput. 29 (2000) 761–778.