A polynomial time approximation scheme for the MRCT problem*

Bang Ye Wu Kun–Mao Chao

1 Further improvement

Let S be a minimal δ-separator of $\widehat{T}$. The strategy of algorithms shown in our previous note on 3/2-approximation is to "guess" the structure of S and to construct a general star with the guessed structure as the core. If $T \in \text{star}(S)$, by the lemmas in the previous note,

$$C(T) \leq 2n \sum_{v \in V(G)} d_G(v, S) + (n^2/2)w(S),$$

and

$$C(\widehat{T}) \geq 2(1-\delta)n \sum_{v \in V} d_{\widehat{T}}(v, S) + 2\delta(1-\delta)n^2 w(S).$$

The approximation ratio, by comparing the two inequalities, is

$$\max\{\frac{1}{1-\delta}, \frac{1}{4\delta(1-\delta)}\}.$$

The ratio achieves its minimum when the two terms coincide, i.e., $\delta = 1/4$, and the minimum ratio is 4/3. In fact, by using a general star and a (1/4)-separator, it is possible to approximate an MRCT with ratio $(4/3) + \varepsilon$ for any constant $\varepsilon > 0$ in polynomial time. The additional error ε is due to the difference between the guessed and the true separators.

By this strategy, the approximation ratio is limited even if S was known exactly. The limit of the approximation ratio may be mostly due to that we consider only general stars. In a general star, the vertices are always connected to their closest vertices of the core. In extreme cases, roughly half of the vertices connected are to both sides of a costly edge. This results in the cost $(n^2/2)w(S)$ in the upper bound of a general star. To make a breakthrough, the restriction that each vertex is connected to the closest vertex of the core needs to be relaxed.

A metric graph is a complete graph with triangle inequality, i.e., each edge is a shortest path of its two endpoints. If the input graph is a metric graph, the core will be a two-edge path, and each vertex is adjacent to one of the "critical vertices"— a centroid and the two endpoints of a path separator. Define k-stars to be the trees with at most k internal vertices. The constructed approximation solution is a 3-star. More importantly, k-stars have no such restriction like general stars and can be used to approximate an MRCT more precisely. Later in this chapter we shall see how it works.

However, k-stars work only for metric graphs. The class of metric graphs is an important subclass of graphs. Solving the MRCT problem on metric graphs is itself meaningful. Before considering the approximation problem on metric graphs, two questions come to our minds:

*An excerpt from the book "Spanning Trees and Optimization Problems," by Bang Ye Wu and Kun-Mao Chao (2004), Chapman & Hall/CRC Press, USA.

- What is the computational complexity of the MRCT problem on metric graphs, NP-hard or polynomial-time solvable?
- If it is NP-hard, does its approximability differ from that of the general problem?

We shall answer the questions in the next section.

2 A Reduction to the Metric Case

In this section, we shall show that the MRCT problem on general inputs can be reduced to the same problem with metric inputs. The reduction is done by a transformation algorithm.

Definition 1: The metric closure of a graph $G = (V, E, w)$ is the complete graph $\bar{G} = (V, V \times V, \bar{w})$ in which $\bar{w}(u, v) = d_G(u, v)$ for all $u, v \in V$.

Let $G = (V, E, w)$ and $\bar{G} = (V, V \times V, \bar{w})$ be its metric closure. Any edge (a, b) in $\bar{G}$ is called a *bad edge* if $(a, b) \notin E$ or $w(a, b) > \bar{w}(a, b)$. For any bad edge $e = (a, b)$, there must exist a path $P = SP_G(a, b) \neq e$ such that $w(P) = \bar{w}(a, b)$. Given any spanning tree T of $\bar{G}$, the algorithm can construct another spanning tree Y without any bad edge such that $C(Y) \leq C(T)$. Since Y has no bad edge, $\bar{w}(e) = w(e)$ for all $e \in E(Y)$, and Y can be thought of as a spanning tree of G with the same routing cost. The algorithm is listed in the following.

Algorithm: REMOVE_BAD
Input: A spanning tree T of $\bar{G}$.
Output: A spanning tree Y of G such that $C(Y) \leq C(T)$.
Compute all-pairs shortest paths of G.
(I) **while** there exists a bad edge in T
Pick a bad edge (a, b). Root T at a.
/* assume $SP_G(a, b) = (a, x, ..., b)$ and y is the parent of x */
if b is not an ancestor of x **then**
$Y^* \leftarrow T \cup (x, b) - (a, b)$;$Y^{**} \leftarrow Y^* \cup \{(a, x)\} - \{(x, y)\}$;
else
$Y^* \leftarrow T \cup (a, x) - (a, b)$;$Y^{**} \leftarrow Y^* \cup \{(b, x)\} - \{(x, y)\}$;
if $C(Y^*) < C(Y^{**})$ **then** $Y \leftarrow Y^*$ **else** $Y \leftarrow Y^{**}$
(II) $T \leftarrow Y$

The algorithm computes Y by iteratively replacing the bad edges until there is no bad edge. It will be shown that the cost is never increased at each iteration and it takes no more than $O(n^2)$ iterations. We assume that the shortest paths obtained in the first step have the following property: If $SP_G(a, b) = (a, x, \ldots, b)$, then $SP_G(a, b) = (a, x) \cup SP_G(x, b)$. This assumption is not strong since almost all popular algorithms for all-pairs shortest paths output such a solution.

Proposition 1: The while loop in Algorithm REMOVE_BAD is executed at most $O(n^2)$ times.

Proof: For each bad edge $e = (a, b)$, let $h(e)$ be the number of edges in $SP_G(a, b)$ and $f(T) = \sum_{\text{bad } e} h(e)$. Since $h(e) \leq n - 1$, $f(T) < n^2$ initially. Since (a, x) is not a bad edge, it is easy to check that $f(T)$ decreases by at least 1 at each iteration. □

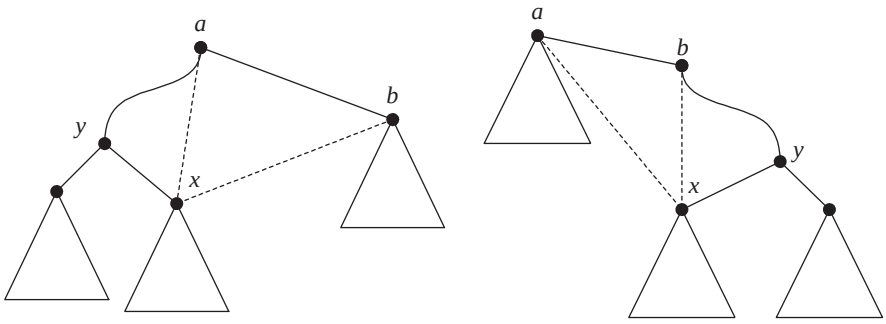

Figure 1: Remove bad edge (a, b). Case 1 (left) and Case 2 (right).

Proposition 2: Before instruction **(II)** is executed, $C(Y) \leq C(T)$.

Proof: For any node v, let $S_v = V(T_v)$. As shown in Figure 1, there are two cases. Case 2 is identical to Case 1 if the tree is re-rooted at b and the roles of a and b are exchanged. Therefore, only the inequality for Case 1 needs to be proved, i.e., $x \in S_a - S_b$.

If $C(Y^*) \leq C(T)$, the result follows. Otherwise, let $U_1 = S_a - S_b$ and $U_2 = S_a - S_b - S_x$. Since the distance does not change for any two vertices both in U_1 (or both in S_b), we have

$$\begin{aligned} C(T) &< C(Y^*) \\ \Rightarrow \sum_{u\in U_1}\sum_{v\in S_b} d_T(u,v) &< \sum_{u\in U_1}\sum_{v\in S_b} d_{Y^*}(u,v). \end{aligned}$$

Since for all $u \in U_1$ and $v \in S_b$, $d_T(u,v) = d_T(u,a) + \bar{w}(a,b) + d_T(b,v)$ and $d_{Y^*}(u,v) = d_T(u,x) + \bar{w}(x,b) + d_T(b,v)$,

$$\begin{aligned} & \sum_{u\in U_1}\sum_{v\in S_b} (d_T(u,a) + \bar{w}(a,b) + d_T(b,v)) \\ & < \sum_{u\in U_1}\sum_{v\in S_b} (d_T(u,x) + \bar{w}(x,b) + d_T(b,v)) \\ \Rightarrow \quad & |S_b| \sum_{u\in U_1} d_T(u,a) + |U_1||S_b|\bar{w}(a,b) \\ & < |S_b| \sum_{u\in U_1} d_T(u,x) + |U_1||S_b|\bar{w}(x,b) \\ \Rightarrow \quad & \sum_{u\in U_1} d_T(u,a) + |U_1|\bar{w}(a,b) < \sum_{u\in U_1} d_T(u,x) + |U_1|\bar{w}(x,b). \end{aligned}$$

Note that $S_b \neq \emptyset$ since the inequality holds. By the definition of the metric closure, $\bar{w}(a,b) = \bar{w}(a,x) + \bar{w}(x,b)$, and then

$$\sum_{u\in U_1} (d_T(u,a) - d_T(u,x)) < -|U_1|\bar{w}(a,x). \tag{1}$$

Now consider the cost of Y^{**}.

$$\left(C(Y^{**}) - C(T)\right)/2 \;=\; \sum_{u\in U_2}\sum_{v\in S_x} \left(d_{Y^{**}}(u,v) - d_T(u,v)\right)$$

$$+ \sum_{u \in U_1} \sum_{v \in S_b} \left(d_{Y^{**}}(u,v) - d_T(u,v) \right).$$

Since $d_{Y^{**}}(u,v) \leq d_T(u,v)$ for $u \in U_1$ and $v \in S_b$, the second term is not positive. By observing that $d_T(u,v) = d_T(u,x) + d_T(x,v)$ and $d_{Y^{**}}(u,v) = d_T(u,a) + \bar{w}(a,x) + d_T(x,v)$ for any $u \in U_2$ and $v \in S_x$, we obtain

$$\begin{aligned}
& \left(C(Y^{**}) - C(T) \right)/2 \\
\leq & \sum_{u \in U_2} \sum_{v \in S_x} \left(d_T(u,a) + \bar{w}(a,x) - d_T(u,x) \right) \\
= & |S_x| \sum_{u \in U_2} \left(d_T(u,a) + \bar{w}(a,x) - d_T(u,x) \right) \\
= & |S_x| \sum_{u \in U_2} \left(d_T(u,a) - d_T(u,x) \right) + |U_2||S_x|\bar{w}(a,x) \\
\leq & |S_x| \sum_{u \in U_1} \left(d_T(u,a) - d_T(u,x) \right) + |U_2||S_x|\bar{w}(a,x) \qquad (2) \\
< & -|U_1||S_x|\bar{w}(a,x) + |U_2||S_x|\bar{w}(a,x) \qquad (3) \\
\leq & 0.
\end{aligned}$$

(2) is obtained by observing that $U_1 - U_2 = S_x$ and $d_T(u,a) > d_T(u,x)$ for any $u \in S_x$. (3) is derived by applying (1). Therefore, $C(Y^{**}) < C(T)$ and the result follows. □

The next lemma follows Propositions 1 and 2, and that each iteration can be done in $O(n)$ time.

Lemma 1: For any spanning tree $\bar{T}$ of $\bar{G}$, it can be transformed into a spanning tree T of G in $O(n^3)$ time and $C(T) \leq C(\bar{T})$.

The above lemma implies that $C(\text{mrct}(G)) \leq C(\text{mrct}(\bar{G}))$. It is easy to see that $C(\text{mrct}(G)) \geq C(\text{mrct}(\bar{G}))$. Therefore, we have the following corollary.

Corollary 2: $C(\text{mrct}(G)) = C(\text{mrct}(\bar{G}))$.

Let ΔMRCT denote the MRCT problem with metric inputs. We have the next theorem.

Theorem 3: If there is a $(1+\varepsilon)$-approximation algorithm for ΔMRCT with time complexity $O(f(n))$, then there is a $(1+\varepsilon)$-approximation algorithm for MRCT with time complexity $O(f(n) + n^3)$.

Proof: Let G be the input graph for the MRCT problem. The metric closure $\bar{G}$ can be constructed in time $O(n^2 \log n + mn)$. If there is a $(1+\varepsilon)$-approximation algorithm for the ΔMRCT problem, a spanning tree T of $\bar{G}$ can be computed in time $O(f(n))$ such that $C(T) \leq (1+\varepsilon)C(\text{mrct}(\bar{G}))$. Using Algorithm REMOVE_BAD, a spanning tree Y of G can be constructed such that $C(Y) \leq C(T) \leq (1+\varepsilon)C(\text{mrct}(\bar{G})) = (1+\varepsilon)C(\text{mrct}(G))$. The overall time complexity is then $O(f(n) + n^3)$. □

Corollary 4: The ΔMRCT problem is NP-hard.

3 A Polynomial Time Approximation Scheme

3.1 Overview

We sketch a *Polynomial Time Approximation Scheme* (PTAS) for the MRCT problem in this section.

As described previously, the fact that the costs w may not obey the triangle inequality is irrelevant, since we can simply replace these costs by their metric closure. Therefore, in this section we may assume that $G = (V, E, w)$ is a metric graph.

We use k-stars, i.e., trees with no more than k internal nodes, as a basis of our approximation scheme. In Section 3.5 we show that for any constant k, a minimum routing cost k-star can be determined in polynomial time. In order to show that a k-star achieves a $(1+\varepsilon)$ approximation, we show that, for any tree T and constant $\delta \leq 1/2$:

1. It is possible to determine a δ-separator, and the separator can be cut into several δ-paths such that the total number of cut nodes and leaves of the separator is at most $\lceil \frac{2}{\delta} \rceil - 3$. (Lemma 5)

2. Using the separator, T can be converted into a $(\lceil \frac{2}{\delta} \rceil - 3)$-star X, whose internal nodes are just those cut nodes and leaves. The routing cost of X satisfies $C(X) \leq (1 + \frac{\delta}{1-\delta})C(T)$. (Lemma 7)

By using $T = \widehat{T}$ =mrct(G), $\delta = \frac{\varepsilon}{1+\varepsilon}$ and finding the best $(\lceil \frac{2}{\delta} \rceil - 3)$-star K, we obtain the desired approximation

$$C(K) \leq (1 + \frac{\delta}{1-\delta})C(\widehat{T}) = (1+\varepsilon)C(\widehat{T}).$$

Before going into the details of the general case, take a look at how to find a 3/2-approximation of an MRCT and its performance analysis.

Recall that a centroid of a tree is the vertex whose removal cuts the tree into components of no more than $n/2$ vertices. Let $\widehat{T}$ be an MRCT of a metric graph G. Root $\widehat{T}$ at its centroid r. For an edge (u, v) with parent v, the routing load of the edge is $2x(n-x)$, in which x is the number of descendants of u. (For convenience, we assume that a vertex is also a descendant of itself.) For a desired positive $\delta \leq 1/2$, removing all vertices with number of descendants no more than δn, we may obtain a connected subgraph S of T. The subgraph S is a minimal δ-separator. In the case that $\delta = 0.5$, S contains only the centroid. If the δ-separator S of the MRCT is given and, for each vertex not in S, its lowest ancestor in S is also known, we may easily construct a $1/(1-\delta)$-approximation Y of the MRCT as follows:

- $S \subset Y$.
- For each vertex u not in S, connect it to its lowest ancestor v in S by adding edge (u, v).

The approximation ratio of Y can be shown by the following two observations.

- For each edge in S, the routing load in Y is the same as that in $\widehat{T}$.
- For each edge (u, v) not in S, the routing load in Y is $2(n-1)$. However, there is a path from u to v in $\widehat{T}$ of which each edge has routing load no less than $2(1-\delta)n$, and the length of the path is at least the same as the edge (u, v).

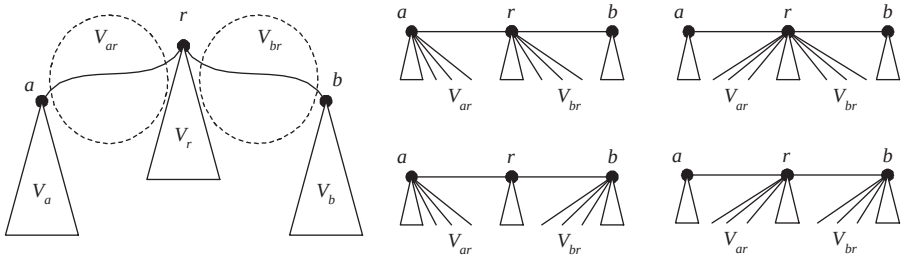

Figure 2: An MRCT and the four 3-stars.

For example, consider the simplest case that $\delta = 1/2$. A 1/2-separator contains only one vertex. Assume that r be such a vertex on an MRCT $\widehat{T}$. For every other vertex, the ancestor in the separator is r. Connecting all other nodes to r, we obtain a star Y. The routing cost of Y is the sum of all edges (v, r) multiplied by its routing load $2(n-1)$. For each vertex v, there is a path from v to r in $\widehat{T}$. Since r is a 1/2-separator of $\widehat{T}$, the routing load of the path is at least n and the path is no shorter than the edge (v, r) by the triangle inequality. Consequently Y is a 2-approximation of the MRCT $\widehat{T}$. Since the separator contains only one vertex, we may try all possible vertices and it leads to an $O(n^2)$ time 2-approximation of the MRCT problem on metric graphs.

But when $\delta < 1/2$, the separator may contain as many as $\Omega(n)$ vertices, and it is too costly to enumerate all possible separators. However, instead of the entire separator, we may achieve the same ratio by knowing only some *critical* vertices of the separator. In the following, we shall take $\delta = 1/3$ as an example to show that we only need to know three critical vertices of the separator to construct a 3/2-approximation.

Root the MRCT $\widehat{T}$ at its centroid r. There are at most two subtrees which contain more than $n/3$ nodes. Let a and b be the lowest vertices with at least $n/3$ descendants. We ignore the cases where $r = a$ or $r = b$. For such cases, the ratio can be shown similarly. Let P be the path from a to b. Obviously the path contains r and is a minimal 1/3-separator. We shall transform T into a 3-star with internal nodes a, b, and r such that its routing cost is no more than 3/2 of that of $\widehat{T}$. As shown in Figure 2, let us partition all the vertices into V_a, V_b, V_r, V_{ar}, and V_{br}. The sets V_a, V_b, and V_r contain the vertices whose lowest ancestor on P is a, b, and r, respectively. The set V_{ar} (and V_{br}) consists of the vertices whose lowest ancestor on P is between a and r (and between b and r, respectively).

First replace P with edges (a, r) and (b, r). For each vertex v in V_a (or V_b, V_r), edge (v, a) (or (v, b), (v, r) respectively) is added. For the vertices in V_{ar}, we consider two cases. Either all of them are connected to a or all of them are connected to r. The vertices in V_{br} are connected similarly. The four possible 3-stars are illustrated in Figure 2.

Now consider the routing cost of $\widehat{T}$. For each vertex v, the routing load of the path from v to P is no less than $4n/3$ since P is a 1/3-separator. For each edge e of P, since there are at least $n/3$ nodes on either side of it, the routing load is no less than $2(n/3)(2n/3) = 4n^2/9$. Therefore we have the following lower bound of the routing cost of the MRCT:

$$C(\widehat{T}) \geq (4n/3) \sum_v d_{\widehat{T}}(v, P) + (4/9)n^2 w(P).$$

In either of the 3-stars constructed above, for each vertex v in $V_a \cup V_r \cup V_b$, the routing load of the edge incident to v is $2(n-1)$, and the edge length is at most the same as the path from v to

P on T. For each node v in V_{ar}, by the triangle inequality, we have

$$(w(v,a) + w(v,r))/2 \leq d_{\widehat{T}}(v,P) + d_{\widehat{T}}(a,r)/2.$$

Note that there are no more than $n/6$ nodes in V_{ar}. For edge (a,r), the routing load is no more than $2(n/2)(n/2) = n^2/2$. (Note: For this simple case that $\delta = 1/3$, this bound is enough. However, a more precise analysis of the incremental routing load is required for smaller δ.)

For the nodes in V_{br}, we may obtain a similar result. In summary, the minimum routing cost of the constructed 3-stars is no more than

$$\begin{aligned} & 2(n-1)\sum_v d_{\widehat{T}}(v,P) + (n^2/6)(d_{\widehat{T}}(a,r) + d_{\widehat{T}}(b,r)) + (1/2)n^2 w(P) \\ \leq \; & 2n\sum_v d_{\widehat{T}}(v,P) + (2/3)n^2 w(P). \end{aligned}$$

The approximation ratio, by comparing with the lower bound of the optimal, is 3/2.

The method can be extended to any $\delta \leq 0.5$. Let S be a δ-separator of T. The critical vertex set, defined as the *cut and leaf set* in the next section, to construct a $1/(1-\delta)$-approximation k-star consists of the following vertices.

- The leaves of S as a and b in the above example.
- The vertices with more than two neighbors on S.
- Some additional vertices such that all the critical vertices cut the separator into edge-disjoint paths and the number of vertices whose lowest ancestors on S belong to the same path is no more than $\delta n/2$. Such a path is defined as a δ-path in the next section. In the above example, r is such a vertex. The vertices a, b and r cut the separator into two paths, and the number of vertices in either V_{ar} or V_{br} is no more than $n/6$.

We shall show that the number of the necessary critical vertices is at most $2/\delta - 3$ for any $\delta \leq 0.5$. Consequently there exists a $(2/\delta - 3)$-star which is an approximation of an MRCT with ratio $1/(1-\delta)$. The PTAS is to construct the $(2/\delta - 3)$-star of minimum routing cost.

The core of a tree is the subgraph obtained by removing all its leaves. The core of a k-star contains no more than k vertices and therefore the number of all possible cores is polynomial to k. For each possible core, the algorithm finds the best way to connect the leaves to one of the vertices of the core. A k-component integer vector $(n_1, n_2, \ldots, n_k)$ is used to indicate how many leaves will be connected to each of the k vertices of the core, in which $\sum_i n_i = n - k$. There are $O(n^{k-1})$ such vectors. For each core and each vector, the routing load on each core edge is fixed since the number of vertices on both sides of the edge are specified by the core and the vector. Therefore the best leaf connection is determined by the leaf edges subject to the numbers of leaves to be connected to the vertices of the core. Such a problem can be solved by solving an assignment problem in $O(n^3)$ time. Consequently the minimum routing cost k-star can be constructed in time polynomial to k and n.

Consider an example for $k = 3$. For a 3-star, the core is a 3-vertex path. There are $O(n^3)$ possible cores. For each possible core (a,b,c), use a three-component integer vector (x,y,z) to indicate how many leaves will be connected to a, b, and c, in which $x+y+z = n-3$ and x, y, z are nonnegative integers. There are $O(n^2)$ such vectors. For a specified core and a specified vector, the routing load on each core edge is also fixed. That is, the routing load on (a,b) is $2(x+1)(n-x-1)$ and on (b,c) is $2(z+1)(n-z-1)$. Therefore the best leaf connection is determined by the leaf

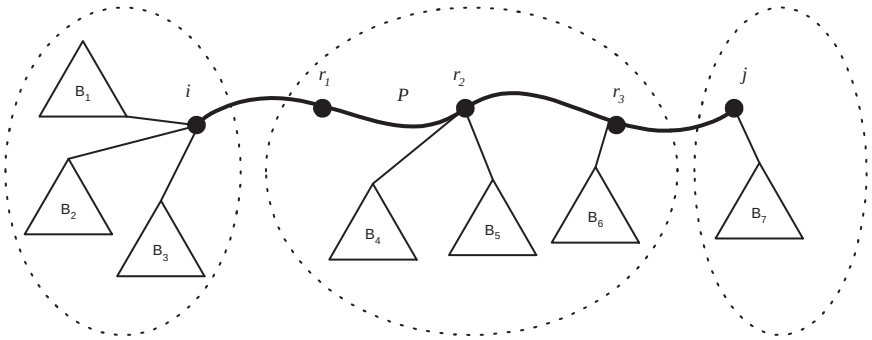

Figure 3: $B_1, \ldots, B_7$ are branches of P. $VB(T, P, i) = \{i\} \cup V(B_1) \cup V(B_2) \cup V(B_3)$. P^c is the number of vertices in $\{r_1, r_2, r_3\} \cup V(B_4) \cup V(B_5) \cup V(B_6)$.

edges subject to the numbers of leaves to be connected to a, b, c. Such a problem can be solved by solving an assignment problem in $O(n^3)$ time. The total time complexity is $O(n^{2k+2})$, which is polynomial for constant k. In fact we need not solve the individual assignment problem for the best leaf connection of each vector. The best leaf connection of one vector can be found from that of another vector by solving a shortest path problem if the two vectors are adjacent. Two vectors are adjacent if one can be obtained from the other by increasing a component by one and decreasing a component by one, e.g., $(5, 4, 2)$ and $(5, 3, 3)$. With this result, the time complexity is reduced to $O(n^{2k})$.

3.2 The δ-spine of a tree

Let $P = SP_T(i, j)$ in which $|VB(T, P, i)| \geq |VB(T, P, j)|$. We shall use the following notations to simplify the expressions.

- $P^a = |VB(T, P, i)|$.
- $P^b = |VB(T, P, j)|$.
- $P^c = n - |VB(T, P, i)| - |VB(T, P, j)|$.
- $Q(P) = \sum_{1 \leq x \leq h} |VB(T, P, r_x)| \times d_T(r_x, i)$, where $P = (i, r_1, r_2, \ldots, r_h, j)$.

P^a and P^b are the numbers of vertices that are hanging at the two end points of the path. Note that we always assume $P^a \geq P^b$. In the case that P contains only one edge, $P^c = 0$. The notations are illustrated in Figure 3.

Definition 2: Let $1 \leq k \leq n$. A k-star is a spanning tree of G which has no more than k internal nodes. An *optimal* k-star is the k-star with the minimum routing cost.

We now turn to the notions of δ-paths and δ-spines. Informally, a δ-path is a path such that not too many nodes (at most $\delta n/2$) are hanging at its internal nodes. A δ-spine is a set of edge-disjoint δ-paths, whose union is a minimal δ-separator. That is, a δ-spine is obtained by cutting a minimal δ-separator into δ-paths. In the case that the minimal δ-separator contains just one node, the only δ-spine is the empty set.

Definition 3: Given a spanning tree T of G, and $0 < \delta \leq 0.5$, a δ-path of T is a path P such that $P^c \leq \delta n/2$.

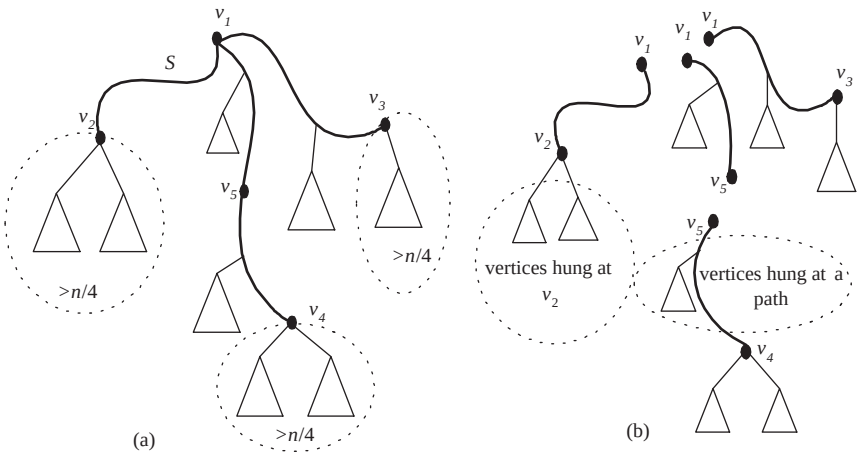

Figure 4: Separator, spine and CAL. Each triangle represents a subtree with no more than $n/4$ vertices.

Definition 4: Let $0 < \delta \leq 0.5$. A δ-spine $Y = \{P_1, P_2, ..., P_h\}$ of T is a set of pairwise edge-disjoint δ-paths in T such that $S = \bigcup_{1 \leq i \leq h} P_i$ is a minimal δ-separator of T. Furthermore, for any pair of distinct paths P_i and P_j in the spine, we require that either they do not intersect or, if they do, the intersection point is an endpoint of both paths.

Definition 5: Let Y be a δ-spine of a tree T. $CAL(Y)$ (which stands for the cut and leaf set of Y) is the set of the endpoints of the paths in Y. In the case that Y is empty, the cut and leaf set contains only one node which is a δ-separator of T. Formally $CAL(Y) = \{u, v | \exists P = SP_T(u, v) \in Y\}$ if Y is not empty, and otherwise $CAL(Y) = \{u | u$ is a minimal δ-separator $\}$.

Example 1: In Figure 4(a), S (bold lines) is a minimal 1/4 separator of the tree. Vertex v_1 is a centroid, and vertices v_2, v_3, and v_4 are leaves in S. In (b), the separator is cut into a 1/4-spine of the tree. The CAL of the spine is $\{v_1, v_2, v_3, v_4, v_5\}$. The path between v_1 and v_4 is cut at vertex v_5 to ensure that the number of vertices hung at each path is no more than $n/8$.

Lemma 5: For any constant $0 < \delta \leq 0.5$, and a spanning tree T of G, there exists a δ-spine Y of T such that $|CAL(Y)| \leq \lceil 2/\delta \rceil - 3$.

Proof: Let S be a minimal δ-separator of T. S is a tree. Let U_1 be the set of leaves in S, U_2 be the set of vertices which have more than two neighbors in S, and $U = U_1 \cup U_2$. Let $h = |U_1|$. Clearly, $|U| \leq 2h - 2$. Let Y_1 be the set of paths obtained by cutting S at all the vertices in U_2. For example, for the tree on the right side of Figure 4, $U_1 = \{v_2, v_3, v_4\}$; $U_2 = \{v_1\}$; Y_1 contains $SP_T(v_1, v_2)$, $SP_T(v_1, v_3)$ and $SP_T(v_1, v_4)$. For any $P \in Y_1$, if $P^c > \delta n/2$ then P is called a *heavy* path. It is easy to check that Y_1 satisfies the requirements of a δ-spine except that there may exist some heavy paths. Suppose P is not a δ-path. We can break it up into δ-paths by the following process. First find the longest prefix of P starting at one of its endpoints and ending at some internal vertex, say i, in the path, that determines a δ-path. Now we break P at vertex i. Then we repeat the breaking process on the remaining suffix of P starting at i, stripping off the next δ-path

and so on. In this way P can be cut into δ-paths by breaking it up at no more than $\lceil 2P^c/(\delta n)\rceil - 1$ vertices. Since there are at least δn nodes hung at each leaf,

$$\sum_{P\in Y_1} P^c < n - h\delta n.$$

Let U_3 be the minimal vertex set for cutting the heavy paths to result in a δ-spine Y of T. We have

$$|U_3| \le \lceil 2\,(n - h\delta n)\,/\,(\delta n)\rceil - 1 = \lceil 2/\delta\rceil - 2h - 1.$$

So, $|CAL(Y)| = |U| + |U_3| \le \lceil 2/\delta\rceil - 3$. □

Example 2: For any tree, there always exist a (1/3)-spine and a (1/4)-spine whose cut and leaf set contains no more than 3 and 5 vertices, respectively, as illustrated in Figure 4. Taking $\delta = 1/5$, it follows that there exists a (1/5)-spine whose cut and leaf set has no more than 7 vertices.

3.3 Lower bound

We are going to establish a more precise lower bound than Lemma **??** (Section **??**). While proving Lemma **??**, we substituted $l(T,e)$ by $2\delta(1-\delta)n^2$ in (**??**),

$$C(T) \ge 2(1-\delta)n\sum_{v\in V} d_T(v,S) + \sum_{e\in E(S)} l(T,e)w(e).$$

Now we give a more careful analysis of the last term. Let Y be a δ-spine of a spanning tree T of G and $S = \bigcup_{P\in Y} P$ be a minimal δ-separator of T. Rewriting in terms of δ-spine and recalling $l(T,e) = 2e^a e^b$, we have

$$\sum_{e\in E(S)} l(T,e)w(e) = 2\sum_{P\in Y}\sum_{e\in P} e^a e^b w(e).$$

Assume $P = (r_0, r_1, r_2, \ldots, r_h)$ in which $|VB(T,P,r_0)| \ge |VB(T,P,r_h)|$. Let $|VB(T,P,r_i)| = n_i$ for $1 \le i \le h-1$ and $e_i = (r_{i-1}, r_i)$ for $1 \le i \le h$.

$$\begin{aligned}
&\sum_{e\in P} e^a e^b w(e)\\
=\;& \sum_{i=1}^{h}\left(P^a + P^c - \sum_{j=i}^{h-1} n_j\right)\left(P^b + \sum_{j=i}^{h-1} n_j\right) w(e_i)\\
\ge\;& \sum_{i=1}^{h} P^b\,(P^a + P^c)\,w(e_i) + (P^a - P^b)\sum_{i=1}^{h}\sum_{j=i}^{h-1} n_j w(e_i)\\
&+ \sum_{i=1}^{h}\left(\sum_{j=i}^{h-1} n_j\right)\left(P^c - \sum_{j=i}^{h-1} n_j\right) w(e_i)\\
\ge\;& P^b(P^a + P^c)w(P) + (P^a - P^b)\sum_{j=1}^{h-1} n_j\left(\sum_{i=1}^{j} w(e_i)\right)\\
=\;& P^b\,(P^a + P^c)\,w(P) + (P^a - P^b)Q(P).
\end{aligned}$$

Note that $Q(P)$ is defined at the first paragraph of Section 3.2. From this and (**??**), we obtain the next lemma.

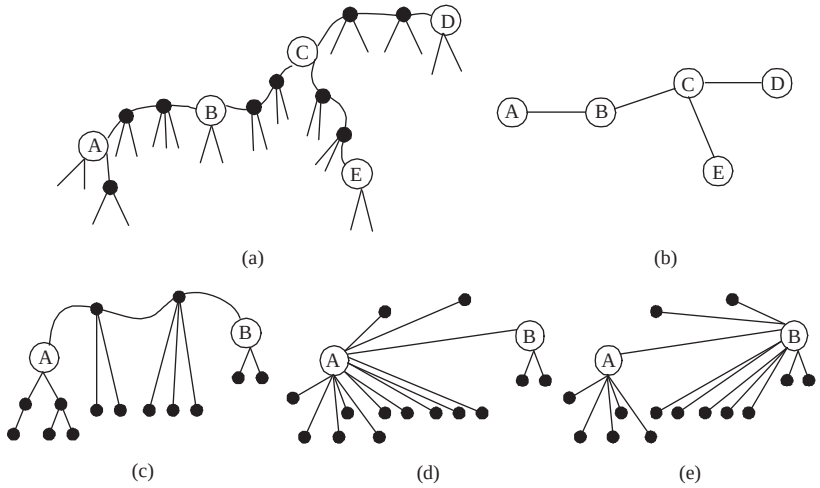

Figure 5: Constructing the k-star from an optimal tree.

Lemma 6: Let Y be a δ-spine of a spanning tree T of G and $S = \bigcup_{P \in Y} P$ be a minimal δ-separator of T. Then

$$C(T) \geq 2(1-\delta)n \sum_{v \in V} d_T(v, S) + 2 \sum_{P \in Y} \Big(P^b(P^a + P^c)w(P) + (P^a - P^b)Q(P) \Big).$$

3.4 From trees to stars

Let $\delta \leq 1/2$ and $k = \lceil 2/\delta \rceil - 3$. We now decompose an optimal solution $\widehat{T}$ and construct a k-star whose routing cost is upper bounded by $\frac{k+3}{k+1} C(\widehat{T})$.

Let $Y = \{P_i | 1 \leq i \leq h\}$ be a δ-spine of $\widehat{T}$ in which $|CAL(Y)| \leq \lceil 2/\delta \rceil - 3$. Note that the set of all the edges in Y forms a δ-separator S. Assume $P_i = SP_{\widehat{T}}(u_i, v_i)$ and $|VB(\widehat{T}, P_i, u_i)| \geq |VB(\widehat{T}, P_i, v_i)|$. By the following steps, we construct a k-star whose internal nodes are exactly the cut and leaf set of the δ-spine we just identified.

1. We connect these nodes by short-cutting edges along the spine to construct a tree spanning these nodes with the same skeletal structure as the spine.

2. All vertices in subtrees hanging at the cut and leaf nodes of the spine are connected directly to their closest node in the spine.

3. Along a δ-path in the spine, all the internal nodes and nodes in subtrees hanging at internal nodes are connected to one of the two endpoints of this path (note that both are in the cut and leaf set of the spine) in such a way as to minimize the resulting routing cost.

This is the k-star used to argue the upper bound on the routing cost.

Example 3: In Figure 5, we illustrate how to construct the desired k-star from an optimal tree. Frame (a) is an optimal tree in which the separator is shown and the cut and leaf set is $\{A, B, C, D, E\}$. Frame (b) is the tree spanning the cut and leaf nodes, which has the same skeletal

structure as the spine. Frames (c), (d) and (e) illustrate how to connect other nodes to the cut and leaf nodes. Frame (c) exhibits the nodes hanging at a δ-path. These nodes will be connected as in either Frame (d) or (e). The nodes hanging at the endpoints of the path will be connected to the endpoints in either case. All the internal nodes of the path and nodes hanging at the internal nodes will be connected to one of the two endpoints. Note that they are connected to the same endpoint either as Frame (d) or Frame (e), but not connected to the two endpoints partially.

More formally, construct a subgraph $R \subset G$ with vertex set $CAL(Y)$ and edge set $E_r = \{(u_i, v_i) | 1 \leq i \leq h\}$. Trivially, R is a tree. Let $f(i)$ be an indicator variable such that

$$f(i) = \begin{cases} 1 & \text{if } \left(P_i^a - P_i^b\right) P_i^c w(P_i) \geq n \left(2Q(P_i) - P_i^c w(P_i)\right) \\ 0 & \text{otherwise} \end{cases}$$

The indicator variable $f(i)$ determines the endpoint of P_i to which all the internal nodes and nodes hanging at such internal nodes will be directly connected. We construct a spanning tree X of G where the edge set E_x is determined by the following rules:

1. $R \subset X$.
2. If $q \in VB(\widehat{T}, S, r)$ then $(q, r) \in E_x$, for any $r \in \{u_i, v_i | 1 \leq i \leq h\}$.
3. For the vertex set $V_i = V - VB(\widehat{T}, P_i, u_i) - VB(\widehat{T}, P_i, v_i)$, if $f(i) = 1$ then $\{(q, u_i) | q \in V_i\} \subset E_x$, else $\{(q, v_i) | q \in V_i\} \subset E_x$. That is, the vertices in V_i are either all connected to u_i or all connected to v_i.

Proposition 3: X is a k-star.

The above proposition is trivial; consider the cost of X.

Proposition 4: $C(X) \leq \frac{1}{1-\delta} C(\widehat{T})$.

Proof: Since any edge in $E_x - E_r$ is incident with a leaf,

$$\frac{C(X)}{2} = \sum_{e \in E_r} e^a e^b w(e) + (n-1) \sum_{e \in E_x - E_r} w(e).$$

First, for any $e = (u_i, v_i) \in E_r$,

$$\begin{aligned} e^a e^b w(e) &\leq \left(P_i^a + f(i) P_i^c\right) \left(P_i^b + (1 - f(i)) P_i^c\right) w(P_i) \\ &= P_i^a P_i^b w(P_i) + \left(f(i) P_i^b + (1 - f(i)) P_i^a\right) P_i^c w(P_i). \end{aligned}$$

Second, by the triangle inequality and recalling that for subgraph $S \subset \widehat{T}$, $d_{\widehat{T}}^S(i, j)$ stands for $w(SP_{\widehat{T}}(i, j) \cap S)$, we have

$$\begin{aligned} \sum_{e \in E_x - E_r} w(e) &\leq \sum_{v \in V} d_{\widehat{T}}(v, S) + \sum_{i=1}^{h} \sum_{v \in V_i} \left(f(i) d_{\widehat{T}}^S(v, u_i) + (1 - f(i)) d_{\widehat{T}}^S(v, v_i)\right) \\ &= \sum_{v \in V} d_{\widehat{T}}(v, S) + \sum_{i=1}^{h} \left(f(i) Q(P_i) + (1 - f(i)) \left(P_i^c w(P_i) - Q(P_i)\right)\right). \end{aligned}$$

Thus,

$$\begin{aligned}\frac{C(X)}{2} &\leq \sum_{i=1}^{h} P_i^a P_i^b w(P_i) + n\sum_{v\in V} d_{\widehat{T}}(v,S)\\ &+ \sum_{i=1}^{h} \min\{P_i^b P_i^c w(P_i) + nQ(P_i), P_i^a P_i^c w(P_i) + n(P_i^c w(P_i) - Q(P_i))\}.\end{aligned}$$

Since the minimum of two numbers is not larger than their weighted mean, we have

$$\begin{aligned}&\min\{P_i^b P_i^c w(P_i) + nQ(P_i), P_i^a P_i^c w(P_i) + n\left(P_i^c w(P_i) - Q(P_i)\right)\}\\ \leq\ &\left(P_i^b P_i^c w(P_i) + nQ(P_i)\right)\frac{P_i^a}{P_i^a + P_i^b}\\ &+ \left(P_i^a P_i^c w(P_i) + n\left(P_i^c w(P_i) - Q(P_i)\right)\right)\frac{P_i^b}{P_i^a + P_i^b}.\end{aligned}$$

Then,

$$\begin{aligned}\frac{C(X)}{2} &\leq \sum_{i=1}^{h} P_i^a P_i^b w(P_i) + n\sum_{v\in V} d_{\widehat{T}}(v,S) + \sum_{i=1}^{h}\frac{\left(2P_i^a P_i^b P_i^c + nP_i^b P_i^c\right)w(P_i)}{P_i^a + P_i^b}\\ &+ \sum_{i=1}^{h}\frac{(P_i^a - P_i^b)nQ(P_i)}{P_i^a + P_i^b}\\ &= n\sum_{v\in V} d_{\widehat{T}}(v,S) + \sum_{i=1}^{h}\frac{w(P_i)}{P_i^a + P_i^b}\left(\left(P_i^a P_i^b + P_i^b P_i^c\right)n + P_i^a P_i^b P_i^c\right)\\ &+ \sum_{i=1}^{h}\frac{(P_i^a - P_i^b)nQ(P_i)}{P_i^a + P_i^b}.\end{aligned}$$

The simplification in the last inequality uses the observation that for any i, we have $P_i^a + P_i^b + P_i^c = n$. By Lemma 6,

$$C(X) \leq C(\widehat{T})\max_{1\leq i\leq h}\left\{\frac{1}{1-\delta}, \frac{n}{P_i^a + P_i^b} + \frac{P_i^a P_i^c}{(P_i^a + P_i^b)(P_i^a + P_i^c)}\right\}.$$

Since $P_i^c \leq \delta n/2$,

$$\begin{aligned}&\frac{n}{P_i^a + P_i^b} + \frac{P_i^a P_i^c}{(P_i^a + P_i^b)(P_i^a + P_i^c)}\\ \leq\ &\frac{n}{P_i^a + P_i^b} + \frac{P_i^c}{P_i^a + P_i^b}\\ =\ &\frac{n + P_i^c}{n - P_i^c} \leq \frac{2+\delta}{2-\delta} \leq \frac{1}{1-\delta}.\end{aligned}$$

This completes the proof. □

For any integer $k \geq 1$, we take $\delta = \frac{2}{k+3}$. By the above two propositions, we have the next lemma.

Lemma 7: An optimal k-star of a metric graph is a $(k+3)/(k+1)$ approximation of an MRCT.

Example 4: Taking $\delta = 1/3$ in Lemma 7, it follows that there exists a 3-star which is a 1.5-approximation of an MRCT, which coincides with the result stated in Section 1. Taking $\delta = 0.2$, it follows that there exists a 7-star which is a 1.25-approximation.

In the following section we will show that it is possible to determine an optimal k-star of a graph in polynomial time. In fact, we have the following lemma.

Lemma 8: An optimal k-star of a graph G can be constructed in $O(n^{2k})$ time.

The proof is delayed to the next section. The following theorem establishes the time-complexity of our PTAS.

Theorem 9: There exists a PTAS for the ΔMRCT problem, which can find a $(1+\varepsilon)$-approximation solution in $O(n^{\rho})$ time complexity where $\rho = 2\lceil 2/\varepsilon \rceil - 2$.

Proof: By Lemma 7, there exists a k-star which is a $(k+3)/(k+1)$ approximation of an MRCT. For finding a $(1+\varepsilon)$-approximation solution, we take $k = \lceil 2/\varepsilon \rceil - 1$ and find an optimal k-star. The time complexity is $O(n^{\rho})$ where $\rho = 2\lceil 2/\varepsilon \rceil - 2$ from Lemma 8. □

The result in Theorem **??** is immediately derived from Theorems 9 and 3.

3.5 Finding an optimal k-star

In this section we describe an algorithm for finding an optimal k-star in G for a given value of k. As mentioned before, given an accuracy parameter $\varepsilon > 0$, we apply this algorithm for $k = \lceil \frac{2}{\varepsilon} - 1 \rceil$, and return an optimal k-star as a $(1+\varepsilon)$-approximate solution.

For a given k, to find an optimal k-star, we consider all possible subsets S of vertices of size k, and for each such choice, find an optimal k-star where the remaining vertices have degree one.

3.5.1 A polynomial-time method

First, we verify that the overall complexity of this step is polynomially bounded for any fixed k. Any k-star can be described by a triple $(S, \tau, \mathcal{L})$, where $S = \{v_1, \ldots, v_k\} \subseteq V$ is the set of k distinguished vertices which may have degree more than one, τ is a spanning tree topology on S, and $\mathcal{L} = (L_1, \ldots, L_k)$, where $L_i \subseteq V - S$ is the set of vertices connected to vertex $v_i \in S$. For any $r \in Z^+$, an r-vector is an integer vector with r components. Let $l = (l_1, \ldots, l_k)$ be a nonnegative k-vector such that $\sum_{i=1}^{k} l_i = n - k$. We say that a k-star $(S, \tau, \mathcal{L})$ has the configuration (S, τ, l) if $l_i = |L_i|$ for all $1 \leq i \leq k$.

Example 5: For the 3-star shown in Figure 6(a), $S = \{v_1, v_2, v_3\}$, τ is shown in (b), $L_1 = \{u_1, u_2, u_3\}$, $L_2 = \{u_4\}$, $L_3 = \{u_5\}$, and $l = (3, 1, 1)$.

For a fixed k, the total number of configurations is $O(n^{2k-1})$ since there are $\binom{n}{k}$ choices for S, k^{k-2} possible tree topologies on k vertices, and $\binom{n-1}{k-1}$ possible such k-vectors. (To see this, observe that every such vector can be put in correspondence with picking $k-1$ among $n-1$ linearly ordered elements, and using the cardinalities of the segments between consecutively picked segments as the components of the vector.) Note that any two k-stars with the same configuration have the same routing load on their corresponding edges. We define $\alpha(S, \tau, l)$ to be an optimal k-star with configuration (S, τ, l).

Note that any vertex v in $V - S$ that is connected to a node $s \in S$ contributes a term of $w(v, s)$ multiplied by its routing load of $2(n-1)$. Since all these routing loads are the same, the

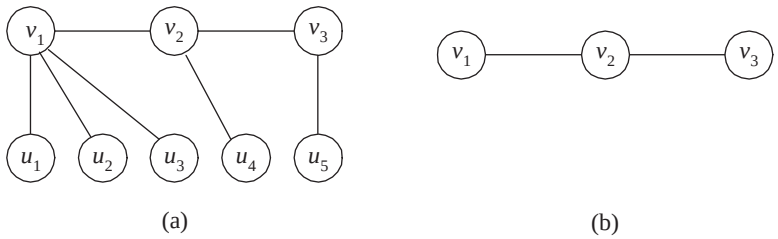

Figure 6: The configuration of a 3-star.

best way of connecting the vertices in $V - S$ to nodes in S is obtained by finding a minimum-cost way of matching up the nodes of $V - S$ to those in S which obeys the degree constraints on the nodes of S imposed by the configuration, and the costs are the distances w. This problem can be solved in polynomial time for a given configuration by a straightforward reduction to an instance of minimum-cost perfect matching. The above minimum-cost perfect matching problem, also called the *assignment* problem, has been well studied and can be solved in $O(n^3)$ time (cf. [1]). Therefore, the overall complexity is $O(n^{2k+2})$ for finding an optimal k-star.

3.5.2 A faster method

In the PTAS, we need to solve many matching problems, each for one configuration. It takes polynomial time to solve these problems individually and this result is sufficient for showing the existence of the PTAS. Although there is no obvious way to reduce the time complexity for one matching problem, the total time complexity can be significantly reduced when considering all these matching problems together. The key point is that two optimal k-stars of similar configurations share a large common portion in their structures. By carefully ordering the matching problems for the configurations and exploiting the common structure of two consecutive problems, we can obtain an optimal solution of any configuration in this order by performing a single augmentation on the optimal solution of the previous configuration. Thus, we show (Lemma 10) how to compute $\alpha(S, \tau, l)$ for a given configuration in $O(nk)$ time.

Let W_{ab} be the set of all nonnegative a-vectors whose entries add up to a constant b. In $W_{ab} \times W_{ab}$, we introduce the relation $\sim$ as $l \sim l'$ if there exist $1 \leq s, t \leq a$ such that

$$l'_i = \begin{cases} l_i - 1 & \text{if } i = s \\ l_i + 1 & \text{if } i = t \\ l_i & \text{otherwise} \end{cases}$$

For a pair l and l' such as the above, we say that l' *is obtained from* l *by* s *and* t.

Let $r = |W_{ab}| = \binom{a+b-1}{a-1}$. The following proposition shows that the elements of W_{ab} can be linearly ordered as $l^1, \ldots, l^r$ so that $l^{i+1} \sim l^i$ for all $1 \leq i \leq r - 1$.

Proposition 5: For all positive integers a, b, there exists a permutation $\pi^{a,b}$ of W_{ab} such that $\pi_1^{a,b}$ is the lexicographic minimum, $\pi_r^{a,b}$ is the lexicographic maximum, and $\pi_{i+1}^{a,b} \sim \pi_i^{a,b}$ for all $i = 1, \ldots, r - 1$.

Proof: By induction. The claim is clearly true when $a = 1$ for any b. Assume the claim is true for all b when $a = m - 1$. For $a = m$ construct the ordering as follows. First the elements for which

$l_1 = 0$ ordered applying $\pi^{a-1,b}$ to $(l_2, \ldots, l_a)$. Then the elements for which $l_1 = 1$, ordered according to *decreasing* $\pi^{a-1,b-1}$. In general each block for which $l_1 = h$ is ordered by applying $\pi^{a-1,b-h}$ to $(l_2, \ldots, l_a)$, forward or backwards according to the parity of h. Note that $\pi^{a,b}_{i+1} \sim \pi^{a,b}_i$ within one block. Furthermore, at block boundaries the part $(l_2, \ldots, l_a)$ is either a lexicographic minimum or maximum so that it is feasible to increase by one l_1. Finally, it is obvious that the first and the last of the constructed ordering are the lexicographic minimum and maximum respectively. □

Example 6: The ordering of $W_{3,4}$ is as follows:

$$\begin{array}{lllll} (0,0,4) & (0,1,3) & (0,2,2) & (0,3,1) & (0,4,0) \\ (1,3,0) & (1,2,1) & (1,1,2) & (1,0,3) & (2,0,2) \\ (2,1,1) & (2,2,0) & (3,1,0) & (3,0,1) & (4,0,0) \end{array}$$

According to Proposition 5 we can order the elements of $W_{k,(n-k)}$ as $l^1, \ldots, l^r$, where $r = \binom{n-1}{k-1}$. Note that $l^1 = (0, \ldots, 0, n-k)$ and $l^r = (n-k, 0, \ldots, 0)$. We shall show how to obtain $\alpha(S, \tau, l^{i+1})$ from $\alpha(S, \tau, l^i)$.

Lemma 10: $\alpha(S, \tau, l^{i+1})$ can be computed from $\alpha(S, \tau, l^i)$ in $O(nk)$ time.

Proof: We shall show that $\alpha(S, \tau, l^{i+1})$ can be found from $\alpha(S, \tau, l^i)$ by means of a shortest path computation. A similar argument is used for solving a minimum cost flow problem given the solution of another minimum cost flow problem which differs by only one unit capacity arc (see Exercise 10.20 in [1]).

For convenience, let us rename the vertices so that $S = \{1, \ldots, k\}$. Let $l^i = (|L_1|, \ldots, |L_k|)$ and $(S, \tau, \mathcal{L}) = \alpha(S, \tau, l^i)$. Let us define an auxiliary weighted digraph $D(\mathcal{L}) = (V, A, \delta)$ in which the arc set is

$$A = \{(u, v) | u \in V - S, v \in S\} \cup \{(u, v) | u \in S, v \in L_u\},$$

and $\delta(u, v) = w(u, v)$ if $u \notin S$, and $\delta(u, v) = -w(u, v)$ if $u \in S$. For a node in S, the weight on an outgoing arc reflects the cost reduction for removing a leaf from its neighbors, and the weight on an incoming arc reflects the increase in cost for connecting a leaf to the node.

Any cycle (not necessarily simple) in the graph describes a way of changing $(S, \tau, \mathcal{L})$ into another k-star with the same configuration, and the difference in cost between the new and the old k-stars is given by $2(n-1)$ times the length of the cycle. Since $(S, \tau, \mathcal{L})$ is optimal for its configuration, we conclude that there is no negative length cycle in $D(\mathcal{L})$. Consequently any shortest path between two vertices must be a simple path.

Similarly, if l^{i+1} is obtained from l^i by s and t, then any path from s to t in the auxiliary graph changes $(S, \tau, \mathcal{L})$ into a k-star with configuration (S, τ, l^{i+1}). Conversely, any k-star with configuration (S, τ, l^{i+1}) can be obtained by a path (not necessarily simple) from s to t. Since the contributed cost of a path is proportional to its length, it is clear that there is a shortest path from s to t, changing $\alpha(S, \tau, l^i)$ into $\alpha(S, \tau, l^{i+1})$. We now show how such a shortest path can be computed in $O(kn)$ time.

Consider any shortest path $(u_1, v_1, u_2, \ldots, v_{h-1}, u_h)$ between two nodes $u_i \in S$ and $v_i \in V - S$ in $D(\mathcal{L})$. Take two consecutive edges (u_i, v_i) and (v_i, u_{i+1}) in the path. Since the path is shortest, v_i must be such as to minimize the sum of the two edge lengths. Recall that $\delta(u_i, v_i) = -w(u_i, v_i)$ and $\delta(v_i, u_{i+1}) = w(v_i, u_{i+1})$. Then, we have that the sum of the two edge lengths is $\min_{v_i \in L_i}\{w(v_i, u_{i+1}) - w(u_i, v_i)\}$. To find the shortest path from s to t on $D(\mathcal{L})$, we construct a complete digraph $D'(\mathcal{L})$ with vertex set S and lengths δ', in which

$$\delta'(i, j) = \min_{v \in L_i}\{w(v, j) - w(i, v)\}.$$

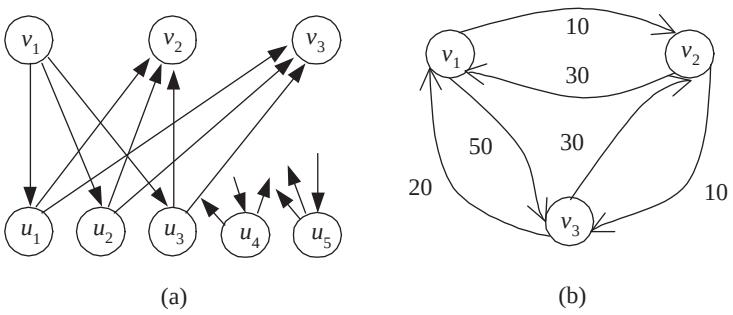

Figure 7: The auxiliary graphs for the faster method.

It is easy to see that the length of the shortest path from s to t on $D'(\mathcal{L})$ is the same as the one on $D(\mathcal{L})$.

Given the graph $D'(\mathcal{L})$, a shortest s-t path (and also the corresponding path on $D(\mathcal{L})$) can be found in $O(k^2)$ time. Finally, to construct $D'(\mathcal{L})$, for each vertex $i \in S$, we have to find $k-1$ minima (one for each other $j \in S$), each over a set of l_i elements. Adding up, the total time complexity is

$$(k-1)\sum_{i=1}^{k} l_i = (k-1)(n-k) = O(nk).$$

□

Example 7: Suppose that the edge lengths $w(v_i, u_j)$ in Figure 6 are given in the following matrix.

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{ccccc} u_1 & u_2 & u_3 & u_4 & u_5 \end{array} \\ \left[\begin{array}{ccccc} 10 & 10 & 30 & 50 & 50 \\ 50 & 60 & 40 & 20 & 60 \\ 80 & 70 & 80 & 30 & 30 \end{array}\right] \end{array}$$

The 3-star in Figure 6(a) is optimal for $l = (3, 1, 1)$ and we want to compute an optimal for vector $(2, 1, 2)$ which is obtained from l by v_1 and v_3. The auxiliary digraph $D(\mathcal{L})$ is shown in Figure 7(a) and $D'(\mathcal{L})$ is shown in (b). Since $L_1 = \{u_1, u_2, u_3\}$, the length $\delta'(v_1, v_2)$ is obtained by

$$\min\left\{\begin{array}{l} w(v_2,u_1) - w(v_1,u_1) \\ w(v_2,u_2) - w(v_1,u_2) \\ w(v_2,u_3) - w(v_1,u_3) \end{array}\right\} = w(v_2,u_3) - w(v_1,u_3) = 10,$$

which corresponds to the cost of moving u_3 from L_1 to L_2. The shortest path from v_1 to v_3 in $D'(\mathcal{L})$ is (v_1, v_2, v_3) and has length 20. Thus the best leaf connection of configuration $(S, \tau, (2, 1, 2))$ is $L_1 = \{u_1, u_2\}$, $L_2 = \{u_3\}$ and $L_3 = \{u_4, u_5\}$.

We are now able to prove Lemma 8 which states that an optimal k-star can be found in $O(n^{2k})$ time.

Proof: When S and τ are fixed, to find an optimal k-star we begin by $\alpha(S, \tau, l^1)$, which is readily obtained by setting $L_k = V - S$. Then, using Lemma 10, we compute optimal k-stars for configurations $l^2, \ldots, l^r$, and report the best overall.

Since we have $\binom{n}{k}$ choices for S, k^{k-2} possible tree topologies, and for each fixed S and τ, we have $\binom{n-1}{k-1}$ configurations, where an optimal for each configuration can be computed in $O(nk)$ time, therefore, in total we can find an optimal k-star in $O(n^k) \times O(n^{k-1}) \times O(n) = O(n^{2k})$ time. □

4 Applications

4.1 Network design

An obvious application of an MRCT is in network design. The length of an edge reflects the cost of routing along the edge. The input graph is the underlying graph in which the edges represent all possible links between nodes. One may want to construct a fixed network such that message or cargo may be transported on the network from sources to destinations.

Trees are an important network structure because of the following two features:

1. Trees are connected subgraphs with minimum number of edges.
2. The routing algorithm on a tree is very simple.

An MRCT is a spanning tree minimizing the total routing cost when the communication requirements for all pairs of vertices are equal. Or in the stochastic model, an MRCT is a spanning tree having the minimum expected routing cost if the requirement between any pair of vertices has equal probability.

4.2 Computational biology

Besides the obvious connection to network design, trees with small routing cost also find application in the construction of good multiple sequence alignments in computational biology. To explicate the application, we shall first briefly introduce the concept of multiple sequence alignments. More details can be found in textbooks for computational biology such as [8].

4.2.1 Multiple sequence alignments

Multiple sequence alignments are important tools for finding patterns common to a set of genetic sequences in computational biology. A multiple alignment of a set of n strings involves inserting gaps (blanks) in the strings and arranging their characters into columns with n rows, one from each string. The order of characters along a row corresponding to string s_i is the same as that in s_i, with possibly some blanks inserted. The following is an example of an alignment of three strings `TCCGATG`, `CCGGACG` and `TCGACG`.

```
T  C  C  -  G  A  T  -  G
-  C  C  G  G  A  -  C  G
T  C  -  -  G  A  -  C  G
```

The intent of identifying common patterns is represented by attempting as much as possible to place the same character in every column.

The multiple sequence alignment problem has typically been formalized as an optimization problem in which some explicit objective function is minimized or maximized. One of the most popular objective functions for multiple alignment generalizes ideas from alignment of two sequences. The pairwise-alignment problem [13] can be phrased as that of finding a minimum mutation path between two sequences. Formally, given costs for inserting or deleting a character and for substituting one character of the alphabet for another, the problem is to find a minimum-cost mutation path from one sequence to the other. The cost of this path is known as the *edit distance* in computer science. An optimal alignment of two sequences of length l can be computed effectively by dynamic programming [13] in $O(l^2)$ time. The generalization to multiple sequences leads to the sum-of-pairs objective.

The *sum-of-pairs* or SP objective for multiple alignment is to minimize the sum, over all pairs of sequences, of the pairwise distance between them *in the alignment* (where the distance of two sequences in an alignment with l columns is obtained by adding up the costs of the pairs of characters appearing at positions $1, \ldots, l$).

Pioneering work of Sankoff and co-authors [12] led to an exponential-time dynamic programming solution to the SP-alignment problem. A straightforward implementation requires time proportional to $2^n l^n$, for a problem with n sequences each of length at most l. Considering that in typical real-life instances l can be a few hundred, the basic dynamic programming approach turns out to be infeasible for all but very small problems.

4.2.2 Approximation algorithms via routing cost trees

The first approximation algorithm for the SP-alignment problem was due to Gusfield [7] with a performance ratio of $2 - \frac{2}{n}$ where n is the number of sequences aligned. This was slightly improved to $2 - \frac{3}{n}$ by Pevzner [11]. The best known approximation algorithm for this problem is due to Bafna, Lawler and Pevzner [2] which achieves a ratio of $2 - \frac{r}{n}$ for any fixed value of r. The running time is exponential in r. Notice that this is not a PTAS for the problem, and no polynomial time approximation scheme is known yet for the SP-alignment problem.

Gusfield's approximation algorithm for the SP-alignment problem is based on the 2-approximation for minimum routing cost trees due to Wong [14]. The algorithm uses a folklore approach to multiple alignment guided by a tree, due to Feng and Doolittle [4]: Given a spanning tree on the complete graph on the sequences to be aligned, the multiple alignment guided by the tree is built recursively as follows. First, remove a leaf sequence l in the tree attached to sequence v by a tree edge (l, v), and align the remaining sequences recursively. Then, insert back the leaf sequence in the alignment guided by an optimal pairwise alignment between the pair l and v. If this optimal pairwise alignment introduces a gap in v, insert the same gap in the recursively computed alignment for the tree without the leaf. Since the cost of aligning a blank to a blank is assumed zero, the resulting alignment has the property that for every pair related by a tree edge, the cost of the induced pairwise alignment equals to their edit distance. By the triangle inequality on edit-distances, the SP-cost of the alignment derived from this spanning tree is upper-bounded by the routing cost of the tree.

Wong's 2-approximation algorithm is the one introduced in Section **??**. For graphs with metric distances obeying the triangle inequality, every shortest-paths tree is isomorphic to a star. Furthermore, in this case, Wong's analysis shows that the best star has routing cost at most twice the total cost of the graph itself. The cost of the graph in this case is the sum of pairwise edit distances between sequences, which is a lower bound on the SP-cost. Thus, Gusfield observed that a multiple alignment derived from the best center-star gives a 2-approximation for the SP-alignment problem.

4.2.3 Tree-driven SP-alignment

Despite the popularity of the SP-objective, most of the currently available methods for finding alignments use a *progressive* approach of incrementally building the alignment adding sequences one at a time with no performance guarantee on the SP-cost. The Feng-Doolittle procedure can be viewed as one such procedure. The advantages of such approaches is their low running time, but the shortcoming is that the order in which the sequences are merged into the alignment determines its cost.

In trying to define a middle ground between the SP-objective and the more practical progressive methods, the tree-driven SP-alignment method was proposed: apply the Feng-Doolittle procedure

to the *best possible* spanning tree in the complete graph on the sequences. By the reasoning above, the tree that gives the best upper bound on the SP-cost of the alignment is the one with the minimum routing cost. Thus, the PTAS for routing cost trees may be useful in finding good trees for applying any progressive alignment method such as the Feng-Doolittle procedure.

5 Summary

In this chapter, we investigate how to approximate a minimum routing cost spanning tree of a graph. The first algorithm approximates an MRCT by constructing a shortest-paths tree rooted at some vertex. By comparing with a trivial lower bound, the cost is shown within two times the optimal. By the technique of solution decomposition, we can have another proof of the approximation ratio. More importantly the technique can be extended to designing better approximation algorithms. It is exhibited by approximation algorithms with ratio 15/8 and 3/2, which use general stars instead of shortest-paths trees to approximate an MRCT.

However, a difficulty is encountered while attempting better approximation ratios for general graphs. A transformation algorithm is introduced to resolve the difficulty. The algorithm Remove_Bad can construct a spanning tree of a graph from a spanning tree of its metric closure without increasing the routing cost. The transformation not only shows the NP-hardness of the MRCT problem on metric graphs but also reduces the approximation problem on general graphs to the metric case.

By decomposing an optimal solution, it is shown that there exists a k-star which is a good approximation of an MRCT of a metric graph. The approximation ratio approaches to 1 as k is increased. A PTAS is then presented by showing how to construct an optimal k-star in polynomial time for fixed k.

Finally the applications of an MRCT in network design and in computational biology are discussed.

Bibliographic Notes and Further Reading

Te Chiang Hu [9] formulated a general version of the routing cost spanning tree problem that he called optimum communication spanning trees (OCT), cf. [ND7] in [6]. In this problem, in addition to the costs on edges, a requirement value λ_{ij} is specified for every pair of vertices i, j. The communication cost between a pair in a spanning tree is the cost of the path between them in the tree multiplied by their requirement λ_{ij}. Thus the routing cost is a special case of the communication cost when all the requirement values are one. He used the term "optimum distance spanning trees" to denote trees with a minimum routing cost and derived a weak condition under which the optimum routing cost tree is a star.

David S. Johnson *et al.* [10] showed that the MRCT problem on a general graph is NP-hard in the strong sense. Robert Dionne *et al.* [3] studied the exact and the heuristic algorithms. The MRCT problem is also known by the name *shortest total path length spanning tree problem*, cf. [ND3] in [6]. R. Wong [14] showed that there exists a shortest-paths tree which is a 2-approximation of an MRCT and gave a worst-case analysis of the approximation algorithm. Bang Ye Wu, Kun-Mao Chao, and Chuan Yi Tang [15] made a breakthrough on the approximation ratio by using the general stars and giving a lower bound with the solution decomposition technique. They showed that it is possible to approximate an MRCT with ratio $(4/3) + \varepsilon$ for any positive constant ε in polynomial time. The idea was soon generalized to the PTAS by Bang Ye Wu *et al.* [16]. The term

"MRCT" first appeared in the paper. Matteo Fischetti *et al.* [5] studied the techniques for finding the exact solution while avoiding an exhaustive search.

Several extensions of the MRCT problem will be discussed in the next chapter, including the OCT problem, the sum-requirement OCT problem and the product-requirement OCT problem. Some variants where not all vertices are sources are also included.

More details of the multiple sequence alignment as well as other problems in computational biology can be found in [8].

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows – Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

[2] V. Bafna, E.L. Lawler, and P. Pevzner. Approximation algorithms for multiple sequence alignment. In *Proceedings of the 5th Combinatorial Pattern Matching conference*, LNCS 807, pages 43–53. Springer-Verlag, 1994.

[3] R. Dionne and M. Florian. Exact and approximate algorithms for optimal network design. *Networks*, 9(1):37–60, 1979.

[4] D. Feng and R. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, 25:351–360, 1987.

[5] M. Fischetti, G. Lancia, and P. Serafini. Exact algorithms for minimum routing cost trees. *Networks*, 39:161–173, 2002.

[6] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, 1979.

[7] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bull. Math. Biol.*, 55:141–154, 1993.

[8] D. Gusfield. *Algorithms on Strings, Trees, and Sequences – Computer Science and Computational Biology*. Cambridge University Press, 1997.

[9] T.C. Hu. Optimum communication spanning trees. *SIAM J. Comput.*, 3:188–195, 1974.

[10] D.S. Johnson, J.K. Lenstra, and A.H.G. Rinnooy Kan. The complexity of the network design problem. *Networks*, 8:279–285, 1978.

[11] P. Pevzner. Multiple alignment, communication cost, and graph matching. *SIAM J. Appl. Math.*, 52:1763–1779, 1992.

[12] D. Sankoff and J. B. Kruskal, editors. *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison Wesley, 1983.

[13] M.S. Waterman. *Introduction to Computational Biology*. Chapman & Hall, CRC Press, 1995.

[14] R. Wong. Worst-case analysis of network design problem heuristics. *SIAM J. Algebra. Discr.*, 1:51–63, 1980.

[15] B.Y. Wu, K.-M. Chao, and C.Y. Tang. Approximation algorithms for the shortest total path length spanning tree problem. *Discrete Appl. Math.*, 105:273–289, 2000.

[16] B.Y. Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi, and C.Y. Tang. A polynomial time approximation scheme for minimum routing cost spanning trees. *SIAM J. Comput.*, 29:761–778, 2000.