Approximate Min-Power Strong Connectivity

G. Calinescu *

Abstract

Given a directed simple graph \( G = (V, E) \) and a cost function \( c : E \rightarrow \mathbb{R}_+ \), the power of a vertex \( u \) in a directed spanning subgraph \( H \) is given by \( p_H(u) = \max_{uv \in E(H)} c(uv) \), and corresponds to the energy consumption required for wireless node \( u \) to transmit to all nodes \( v \) with \( uv \in E(H) \). The power of \( H \) is given by \( p(H) = \sum_{u \in V} p_H(u) \).

Power Assignment seeks to minimize \( p(H) \) while \( H \) satisfies some connectivity constraint. In this paper, we assume \( E \) is bidirected (for every directed edge \( e \in E \), the opposite edge exists and has the same cost), while \( H \) is required to be strongly connected. This is the original power assignment problem introduced by Chen and Huang in 1989 and since then the best known approximation ratio is 2 and is achieved by a bidirected minimum spanning tree. We improve this to 1.992 by combining techniques from Robins-Zelikovsky (2000) for Steiner Tree, Christofides (1976) for Metric Travelling Salesman, and Caragiannis, Flammini, and Moscardelli (2007) for the broadcast version of Power Assignment, together with a novel property on T-joins in hypergraphs that admit a strongly connected orientation.

1 Introduction

There has been a surge of research in Power Assignment problems since 2000 (among the earlier papers are [21, 26, 15]) This class of problems take as input a directed simple graph \( G = (V, E) \) and a cost function \( c : E \rightarrow \mathbb{R}_+ \). The power of a vertex \( u \) in a directed spanning subgraph \( H \) of \( G \) is given by \( p_H(u) = \max_{uv \in E(H)} c(uv) \), and corresponds to the energy consumption required for wireless node \( u \) to transmit to all nodes \( v \) with \( uv \in E(H) \). The power (or total power) of \( H \) is given by \( p(H) = \sum_{u \in V} p_H(u) \).

The study of the min-power power assignment was started by Chen and Huang [7], which consider, as we do, the case when \( E \) is bidirected (the case is sometimes called “symmetric” or “undirected” in the literature) while \( H \) is required to be strongly connected. We call this problem Min-Power Strong Connectivity. We use with the same name both the (bi)directed and the undirected version of \( G \). [7] prove that the bidirected version of a minimum (cost) spanning tree (MST) of the input graph \( G \) has power at most twice the optimum, and therefore the MST algorithm has approximation ratio at most 2. This is known to be tight (see Section 2).


*Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA. calinescu@iit.edu. Research supported in part by NSF grants CCF-0515088 and NeTS-0916743. These results were announced in an extended abstract at APPROX 2010.
the broadcast version of symmetric Power Assignment (assuming a bidirected $G = (V, E, c)$ and a "root" $u \in V$ is given, $H$ must contain a directed path from $u$ to every vertex of $G$), together with what we believe to be a new property on T-joins hypergraphs that admit a strongly connected orientation (see [11] or the next section for definitions). Familiarity with the fundamental NP-Hard optimization problems Steiner Tree and Travelling Salesman will help the reader make sense of this paper; however all our proofs are self-contained and only refer to previous papers for intuition.

Very restricted versions of Min-Power Strong Connectivity have been proven NP-Hard [17, 9, 6]. We are not aware of better than a factor of 2 approximation except for [6], (where $c : E \rightarrow \{A, B\}$, for $0 \leq A < B$; see also [4]), [2] (where $c$ is assumed to be a metric), and the exact (dynamic programming) algorithms [17] for the specific case where each vertex of $G$ maps to a point on a line, and $c( uv )$ is an increasing function of the Euclidean distance between the images of $u$ and $v$. A related version, also NP-Hard, asks for $H$ to be bidirected (also known as "undirected" or "symmetric"). This problem is called Min-Power Symmetric Connectivity, and the best known ratio of $5 / 3 + \epsilon$ [1] is obtained with techniques first applied to Steiner Tree; when $c : E \rightarrow \{A, B\}$ one gets $3 / 2$ with the same method [20]. In fact, many but not all power assignment algorithms use techniques from Steiner Tree variants (or direct reduction to Steiner Tree variants; these connections to Steiner Tree are not obvious and cannot be easily explained), and in particular Caragiannis et al [5] uses the relative greedy heuristic of Zelikovsky [28]. New interesting techniques were also developed for power assignment problems, as in [18], an improvement over [16].

The existing lower bound of the optimum, which we use, is the cost of the minimum spanning tree of $G$. Indeed, the optimum solution $OPT$ contains an in-arborescence rooted at $v$, for some $v \in \bar{V}$, and then, for all $u \in V \setminus \{v\}$, $p_{OPT}(u)$ is at least the cost of the directed edge connecting $u$ to its parent in the in-arborescence, whose total cost does not exceed the cost of the minimum spanning tree.

We also use a relative greedy method as in [28, 23]; Robins-Zelikovsky [23] is rarely used as a technique, and not by only citing the ratio (improved by now in [3]). We use the natural structures of [5] to improve over the minimum spanning tree. We are at a disadvantage as our new lower bound is not far from optimum (and the algorithm more than adds the two lower bounds); still the greedy Robins-Zelikovsky method allows an improvement over the factor of 2.

The new lower bound resembles the T-joins implicit in Christofides’ [8] approximation algorithm for Traveling Salesman Problem, however we have hypergraphs rather than graphs. Precisely, given an edge-weighted hypergraph $K = (V_K, E_K)$ an even-sized set of vertices $S \subseteq V_K$, a T-cut is a partition of $V$ into two parts $Q$ and $\bar{Q} := V \setminus Q$ such that $|Q \cap R|$ is odd. A T-join in $K$ for $R$ is a set of hyperedges $M \subseteq E_K$ such that for every T-cut $(Q, \bar{Q})$ there is a hyperedge $e \in M$ intersecting both $Q$ and $\bar{Q}$; such an edge is said to cross the T-cut. A minimum-weight T-join can be computed in polynomial-time if $K$ is a graph (Chapter 29 of [25]). The generalization of Minimum Weight Graph T-join to hypergraphs, which we call Hypergraph T-join, is however NP-hard (Section 2) and we cannot directly use the Christofides approach; instead we resort to Robins-Zelikovsky. For our new lower bound, we also need to know the supremum, over a special class of hypergraphs, of the minimum weight T-join divided by the weight of (all hyperedges in) the hypergraph. For (the class of) two-edge-connected graphs, this T-ratio is known (and not too hard to prove) to be $1 / 2$ [10]. For the special class of hypergraphs we obtain from power assignment solutions, we present a sequence of examples where the T-ratio converges to $2 / 3$, and prove it is at most $7 / 8$. This proof is long and very technical. In order to prevent an even longer paper, we only sketch an improvement to $4 / 5$, which implies an approximation ratio of 1.98 for the algorithm. Our special hypergraphs admit a strongly connected orientation - defined in the next section (see [11]), and we also sketch that the T-ratio is at most $4 / 5$ for the class of hypergraphs that
admit a strongly connected orientation.

Min-Power Strong Connectivity has a flavor similar to two more fundamental problems for which the best known ratio is 2: Min-Cost Strong Connectivity and Min-Cost Two Edge Connectivity and we hope this paper will renew the interest in those two problems.

2 Preliminaries

In directed graphs, we use arc to denote a directed edge. In a directed graph $K$, an incoming arborescence rooted at $x \in V(K)$ is a subgraph $T$ of $K$ such that the underlying undirected graph of $T$ is a tree and every vertex of $T$ other than $x$ has exactly one outgoing arc in $T$. The vertices of $T$ with no incoming arcs in $T$ are called leafs.

Given an arc $xy$, its undirected version is the undirected edge with endpoints $x$ and $y$. Arcs $xy$ and $yx$ are antiparallel, and the antiparallel arcs resulting from undirected edge $uv$ are $uv$ and $vu$; if undirected edge $uv$ has cost then each of the two antiparallel arcs resulting from undirected edge $uv$ have this cost.

An alternative definition of our problem (how it was originally posed) is: we are given a simple undirected graph $G = (V, E)$ and a cost function $c : E \rightarrow \mathbb{R}^+$. A power assignment is a function $p : V \rightarrow \mathbb{R}^+$, and it induces a simple directed graph $H(p)$ on vertex set $V$ given by $xy$ being an arc of $H(p)$ if and only if $\{x, y\} \in E$ and $p(x) \geq c(\{x, y\})$. The problem is to minimize $\sum_{u \in V} p(u)$ subject to $H(p)$ being strongly connected. To see the equivalence of the definition, given directed spanning subgraph $H$, define for each $u \in V$ the power assignment $p(u) = p_H(u)$.

The following example (see Figure 1) shows that the ratio of 2 for the MST algorithm is tight. Consider $2n$ points located on a single line such that the distance between consecutive points alternates between 1 and $\epsilon < 1$, and let the cost function $c$ be the square of the Euclidean distance Then the minimum spanning tree $\text{MST}$ connects consecutive neighbors and has power $p(\text{MST}) = 2n$. On the other hand, the tree $T$ with edges connecting each other node (see Figure 1(b)) has power equal $p(T) = n(1 + \epsilon)^2 + (n - 1)\epsilon^2 + 1$. When $n \to \infty$ and $\epsilon \to 0$, we obtain that $p(\text{MST})/p(T) \to 2$. On the other hand (Claim 2 of Theorem 3.2 of [17]), the power of the bidirected minimum spanning tree $T$ is at most

$$p(T) = \sum_{v \in V} \max_{u | vu \in E(T)} c(vu) \leq \sum_{v \in V} \sum_{u | vu \in E(T)} c(vu) = 2c(T) \leq \text{opt}$$

where $\text{opt} = p(\text{OPT})$ for an optimum solution $\text{OPT}$ (the last inequality is from the introduction).

The example above may give intuition on how Power Assignment (and even more specifically, Min-Power Symmetric Connectivity - the variant when $H$ must be bidirected mentioned in the introduction) relates to the $k$-restricted Steiner trees, with stars (trees of height 1) taking the place of restricted components. Another example from [1], (see Figure 2) and the following paragraph), shows how Min-Power Strong Connectivity differs from Min-Power Symmetric Connectivity, and may give intuition how Min-Power Strong Connectivity relates to Travelling Salesman, and also Min-Cost Strong Connectivity and Min-Cost Two-Edge Connectivity (a two-edge-connected graph has an edge orientation that makes it strongly connected - see for example Chapter 2, written by A. Frank, of [13]). However we cannot think of direct reductions either way, and, as we mention in conclusions, the methods we use (more precisely, the Christofides algorithm) only apply to certain instances of Min-Cost Strong Connectivity and Min-Cost Two-Edge Connectivity.
Figure 1: Tight example for the performance ratio of the MST algorithm. In both cases, the solution is bidirected and the undirected version of the arcs of the solution is given by solid edges. For each vertex, its power in the solution is next to it. (a) The MST-based power assignment needs total power $2n$. (b) Optimum power assignment has total power $n(1 + \epsilon)^2 + (n - 1)\epsilon^2 + 1 \rightarrow n + 1$.

The power of a Min-Power Strong Connectivity optimum solution can be almost half the power of an Min-Power Symmetric Connectivity optimum solution for the same instance: we present a series of examples illustrated in Figure 2. The $n(n + 1)$ vertices are embedded in the plane in $n$ groups of $n + 1$ points each. Each group has two “terminals” (represented as thick circles in Figure 2), and the $2n$ terminals are the corners of a regular $2n$-gon with sides of length 1. Each group has another $n - 1$ equally spaced points (dashes in Figure 2) on the line segment between the two terminals. The cost function $c$ is the square of the Euclidean distance. It is easy to see that a minimum power assignment ensuring strong connectivity assigns a power of 1 to one thick terminal in each group and a power of $\epsilon^2 = (1/n)^2$ to all other points in the group - the arcs going clockwise. The total power then equals $n + 1$. For symmetric connectivity it is necessary to assign power of 1 to all but two of the thick points, and of $\epsilon^2$ to the remaining points, which results in total power of $2n - 1 - 1/n + 2/n^2$. Also, keep in mind that the minimum spanning tree solution is a symmetric solution.

Let $K = (V_K, E_K)$ be an undirected hypergraph. A path in a hypergraph consists of an alternating sequence of vertices and hyperedges for which each hyperedge contains the two vertices which precede and follow it in the sequence. A hypergraph is two-edge-connected if there exist two hyperedge-disjoint paths between any two vertices. Note that Menger’s theorem holds for hypergraphs and a hypergraph is two-edge-connected if and only if the removal of any single hyperedge does not disconnect the graph. While it may be already known, it is easy to check that Hypergraph T-join is indeed NP-hard, by a reduction from 4-D Matching (which asks if a 4-regular hypergraph $K$ with $V(K)$ a multiple of 4, contains a perfect matching, that is, a set of disjoint hyperedges containing every vertex of the input hypergraph; see Garey and Johnson [12] problem SP1). It is easy to check that for $S = V(K)$, a T-join of size at most $|V(K)|/4$ must be a perfect matching.

A directed hypergraphs has for every hyperedge exactly one vertex, called the tail, and several vertices, that are heads (we reverse the tail-head notation of [11] to fit power assignment). A directed hypergraph is strongly connected if there exists at least one directed path between any two vertices of the hypergraph, where a directed path is defined to be a sequence of vertices and hyperedges for which each hyperedge has as tail the vertex preceeding it and as one of its heads the vertex following it in the sequence. Orienting an undirected hypergraph means choosing a tail for each hyperedge. It is not true
Figure 2: Total power for Min-Power Strong Connectivity can be half the total power for Min-Power Symmetric Connectivity. In both cases, the solution is represented by solid segments (arrows meaning not bidirected), and next to each vertex its power is given. (a) Minimum power assignment ensuring strong connectivity has total power \( n + n^2 \epsilon^2 = n + n^2 \frac{1}{n^2} = n + 1 \). (b) Minimum power assignment ensuring symmetric connectivity has total power \( (2n - 2) + (n^2 - n + 2) \epsilon^2 = 2n - 1 - \frac{1}{n} + \frac{2}{n^2} \), the solution is bidirected and the undirected version of the arcs of the solution is given by solid segments.

[11] that every two-edge-connected hypergraphs admits a strongly connected orientation (for graphs, this property holds and is known since at least 1939 [22]). We sketch in Subsection 3.3 a proof that the T-ratio is at most \( \frac{4}{5} \) for the class of hypergraphs that admit a strongly connected orientation; also we show that for two-edge-connected hypergraphs, the T-ratio is 1.

Our greedy approach in the next section uses the fundamental combinatorial optimization concept of submodularity (see for example Chapter 44 of [25]). Precisely, given a ground set \( X \) of \( n \) elements and a set function \( f : 2^X \rightarrow \mathbb{R}_+ \), \( f \) is submodular iff

\[
f(A \cup B) + f(A \cap B) \leq f(A) + f(B)
\]

for all \( A, B \subseteq X \). A set function \( f \) is monotone (by which we mean non-decreasing) if \( f(A) \leq f(B) \) for all \( A \subseteq B \) and \( f(\emptyset) = 0 \). For monotone set function \( f \) and \( A, S \subseteq X \), define \( f_A(S) = f(A \cup S) - f(A) \). It is well known ([25]) and easy to check that \( f_A \) is submodular if \( f \) is submodular. Also ([19, Proposition 7]; see also Theorem 44.1 of [25]) a monotone function \( f \) is submodular if and only if for any \( S \subseteq T \subseteq X \), we have

\[
\sum_{i \in T \setminus S} f_S(i) \geq f_S(T)
\]

3 The Approximation Algorithm

This section is dedicated to proving the main result of this paper:

**Theorem 1** There exists a polynomial time algorithm for Min-Power Strong Connectivity with approximation ratio 1.992.
3.1 The Algorithm

Our algorithm uses a greedy approach similar to [28, 23, 5]. Let $T$ be the undirected minimum spanning tree of $G$. We will not use that $T$ is minimum except to note $opt \geq c(T)$, where $opt = p(OPT)$ for an optimum solution $OPT$. Let $\tilde{T}$ be the bidirected version of $T$.

For $u \in V$ and $r \in \{c(uv) \mid uv \in E\}$, let $S(u, r)$ be the directed star with center $u$ containing all the arcs $uv$ with $c(uv) \leq r$; note that $r$ is the power of $S$. For a directed star $S$, let $E(S)$ be its set of arcs and $V(S)$ be its set of vertices.

For given $S(u, r)$, let $Q(u, r)$ be the set of edges of $T$ on a path in $T$ between some $x, y \in V(S(u, r))$. Let $\tilde{Q}(u, r)$ be the set of arcs of $\tilde{T}$ on a directed path from $u$ to some $x \in V(S(u, r))$; it is easy to verify that the undirected version of $\tilde{Q}(u, r)$ is $Q(u, r)$.

For a collection $\mathcal{A}$ of directed stars $S(u_i, r_i)$, define $Q(\mathcal{A}) = \bigcup_{S(u_i, r_i) \in \mathcal{A}} Q(u_i, r_i)$ and $f(\mathcal{A}) = \sum_{e \in Q(\mathcal{A})} c(e)$. We will use later that this function is monotone and submodular, a well-known fact which appears as an example in Subchapter 44.1.a (pages 768-9) of [25]. Also define $w(\mathcal{A}) = \sum_{S(u_i, r_i) \in \mathcal{A}} r_i$, the total power used by the stars in $\mathcal{A}$. For $S = S(u, r)$, recall that $f(\mathcal{A}) = f(\mathcal{A} \cup \{S\}) - f(\mathcal{A}) = \sum_{e \in Q(u, r)} Q(\mathcal{A}) c(e) = \sum_{e \in I(\mathcal{A})} c(e)$, where $I(\mathcal{A})$ is defined to be those arcs of $Q(u, r)$ for which the undirected version is not in $Q(\mathcal{A})$.

The algorithm starts with $M = \tilde{T}$ as the set of arcs, and greedily adds directed stars to collection $\mathcal{A}$ (initially empty) while removing arcs from $M$ to improve the following quantity which is an upper bound on the power of our output: the cost the arcs in $M$ plus the total power of the stars in $\mathcal{A}$. To simplify later proofs, we make changes even if our quantity stays the same. To be precise:

**Algorithm Greedy:**

\[
\begin{align*}
\mathcal{A} & \leftarrow \emptyset, \; M \leftarrow \tilde{T} \\
\text{While} \; (f(\mathcal{A}) < c(T) \;) \; \text{do} \\
& \quad (u, r) \leftarrow \text{argmax}_{(u', r')} f(\mathcal{A})(S(u', r'))/r' \\
& \quad M \leftarrow M \setminus I(\mathcal{A})(S(u, r)) \\
& \quad \mathcal{A} \leftarrow \mathcal{A} \cup \{S(u, r)\} \\
\text{Output} \; \cup_{S \in \mathcal{A}} E(S) \cup M
\end{align*}
\]

Figure 3 shows two iterations of the algorithm.

Note that unless $f(\mathcal{A}) = c(T)$ a star $S(u, r)$ always exists for which $f(\mathcal{A})(S(u, r)) > 0$ and $f(\mathcal{A})(S(u, r))/r \geq 1$. Indeed, as long as a pair of antiparallel arcs $e'$ and $e''$ are in $M$, we can pick as next star $S(u, r)$ the one given by $u$ being the tail of $e'$ and $r = c(e')$; this star will be added to $\mathcal{A}$ while $e'$ is removed from $M$.

**Lemma 2** The output is a spanning strongly connected subgraph of $G$.

**Proof.** We prove the following invariant: $X := \cup_{S \in \mathcal{A}} E(S) \cup M$ gives a spanning strongly connected subgraph whenever the while condition is checked by the algorithm. Moreover, suppose we remove from $T$ all edges for which both antiparallel arcs appear in $M$, splitting $T$ in components with vertex sets $T_i$, for some range of $i$. We prove that for every $i$ and every $x, y \in T_i$, there exists a directed path $P$ from $x$ to $y$ using only vertices of $T_i$ and arcs from $X$.

The invariant is true before the first iteration, when each $T_i$ has just one vertex, so consider the moment a star $S = S(u, r)$ is added to $\mathcal{A}$. Figure 4 may provide intuition. We add arcs $uz$, for $z \in V(S) \setminus \{u\}$, while removing from $M$ (and from $X$) arcs $xy$ if $yx \in M$ and there is some $z$ such
that $xy$ is on the directed simple path from $u$ to $z$ in $\tilde{T}$. The same effect is obtained if we do this change for each $z \in V(S) \setminus \{u\}$ one after the other.

Let $P$ be the simple path in $T$ from $u$ to $z$, and let $x_iy_i$, for $1 \leq i \leq k$, be, in order, the arcs of $M$ on $P$ such that also $y_ix_i \in M$. Thus the change to $X$ consists of adding the arc $uz$ and removing all arcs $x_iy_i$; note that if $k = 0$ no arc is removed and our induction step is complete. Let $M' = M \setminus \{x_1y_1, \ldots, x_ky_k\}$ and $X' = X \setminus \{x_1y_1, \ldots, x_ky_k\} \cup \{uz\}$. We need to show that $X'$ and $M'$ satisfy the conditions from the induction hypothesis.

Split $T$ into components by removing all the undirected edges $xy$ with both antiparallel arcs $xy$ and $yx$ in $M'$; in particular all $x_iy_i$, for $1 \leq i \leq k$.

Then we know by induction that $X$ contains directed paths: $P_1$ from $x_1$ to $u$, $P_2$ from $x_2$ to $y_1$, $\ldots$, $P_k$ from $x_k$ to $y_{k-1}$, and $P_{k+1}$ from $z$ to $y_k$, and none of these paths uses $x_iy_i$ or $y_ix_i$. Putting together these paths, the arcs $y_ix_i$, for $1 \leq i \leq k$, and the arc $uz$, we have a directed cycle $C$ containing none of the arcs $x_iy_i$, for $1 \leq i \leq k$. Any arc removed can be replaced, when discussing connectivity, with a path around the cycle $C$, and so $(V, X')$ is strongly connected, as required.

We now split $T$ into components by removing all the undirected edges $xy$ with both $xy$ and $yx$ in $M'$, obtaining components $T'_i$, with $i$ in some range. Note that none of $P_i$, $1 \leq i \leq k$, from above, has an arc with endpoints in two distinct components $T'_i$ (as $T'_i$ is the union of several $T_j$). As all the edges on the path from $u$ to $z$ in $T$ do not have anymore both antiparallel arcs in $M'$. all the vertices on this path including $u, z$ and all $x_i, y_i$ are in the same component $T'_i$ of the split $T$. Thus all the arcs of $C$ have their endpoints in the same component of the split $T$.

We prove that for every $i$ and every $x, y \in T'_i$, there exists a directed path $P$ from $x$ to $y$ using only vertices of $T'_i$ and arcs from $X'$. Indeed, we use the path $P$ from $X$ but replace if necessary arcs of $X \setminus X'$ by arcs of $C$, staying as indicated above in the same component of $T$ split by $M'$.

This completes the induction step. ■
Figure 4: Rounded rectangles show the components $T_i$, dashed before $S$ is added to $A$ ($T$ split by $M$) and solid afterwards ($T$ split by $M'$). Arcs of $M$ crossing from one component to another are given by thin arcs, $S$ by the four thick arcs $uz_1, uz_2, uz_3, uz_4$, and the dashed arcs are those removed from $M$ when $S$ is added.

### 3.2 Approximation Ratio Analysis

**Lemma 3** There exists a collection of stars $B$ with $f(B) = c(T)$ and $w(B) \leq (7/8) \text{opt}$, where $\text{opt}$ is the power of the optimum solution.

The factor of 7/8 above cannot be improved to a constant better than $1/2$, as we can see by looking at the example in Figure 1. To “cover” the whole minimum spanning tree (that is, have $f(B) = c(T)$) one needs to select every second star of optimum (that is, use with power $(1 + \epsilon)^2$ nodes 3, 7, 11 $\ldots$).

Our proof is based on T-joins and the $7/8$ above cannot be improved to more than $2/3$ with the same method, as we present an example later.

**Proof.** Let $(S_v)_{v \in V}$ be the directed stars of $OPT$, with $S_v$ centered at $v$, and let $A$ be collection of these stars. Let $K = (V_K, E_K)$ be the (undirected) hypergraph defined by $V_K = V$ and $E_K = \{V(S) \mid S \in A\}$. Define the weight of an hyperedge to be the power of the corresponding directed star. Recall from the introduction that, with given $R \subseteq V$ with $|R|$ even, a T-cut is a partition of $V$ into two parts $Q$ and $\bar{Q} := V \setminus Q$ such that $|Q \cap R|$ is odd. A T-join in $K$ for $R$ is a set of hyperedges $M \subseteq E_K$ such that for every T-cut $(Q, \bar{Q})$ there is a hyperedge $e \in M$ intersecting both $Q$ and $\bar{Q}$.

The following is the equivalent of Christofides’ method:

**Claim 4** Let $R$ be the vertices of the tree $T$ of odd degree (note that $|R|$ is even). Let $D$ be a set of stars such that the corresponding hyperedges form a T-join in $K$ for $R$. Then $f(D) = c(T)$.

**Proof.** Note that when it comes to computing $f$, for each star $S$ only $V(S)$ counts, and which vertex is the center is not relevant. For star $S = S(u, r)$, let $Q(S) := Q(u, r)$, defined previously as be the set of edges of $T$ on a path in $T$ between some $x, y \in V(S)$.

We need to show that for every $e \in T$, there is a star $S \in D$ with $e \in Q(S)$. Indeed, if we remove $e$ from $T$, we create two subtrees $T_u$ and $T_v$, where $u$ and $v$ are the endpoints of $e$. Then $|R \cap V(T_u)|$ is odd, since if we take $T_u$ and add the vertex $v$ and the edge $uv$, we have an even number of vertices of odd degree, of which one is $v$. Thus $(V(T_u), V(T_v))$ is a T-cut for $R$ and the T-join given by $D$ must
have an hyperedge intersecting both \( V(T_u) \) and \( V(T_v) \), and thus a star \( S \in D \) with \( e \in Q(S) \). ■

Based on the lemma above, it is enough for us to construct, for any arbitrary \( R \), a T-join in \( K \) with weight at most \((7/8)w(K)\).

Before continuing with the proof of Lemma 3 (the main technical difficulty of this paper) it is instructive to see how \( 2/3 \) is the best we can hope instead of \( 7/8 \). As mentioned before, if we were dealing with graphs rather than hypergraphs the ratio would be \( 1/2 \). First a \( 3/5 \) small example: (see Figure 5) nine vertices \( x, y_0, y_1, y_2, z_0, z_1, z_2, u_1, u_2, \) edges of cost \( 2 \): \( xy_0 \) and \( xz_0 \), edges of cost \( 1 \): \( y_1y_2, z_1z_2 \), and \( u_1u_2 \), and edges of cost \( 0 \): \( y_0y_1, z_0z_1, y_2u_1, z_2u_1 \), and \( u_2x \). OPT has cost \( 5 \): \( x \) has power \( 2 \) and \( y_1, z_1 \), and \( u_1 \) each have power \( 1 \). The minimum spanning tree has the five edges of cost \( 0 \) and the three edges of cost \( 1 \). \( R \) consists of the vertices \( x, y_0, z_0, u_1, \) and one can check by inspection that any T-join has weight \( 3 \) (here one can “cover” the minimum spanning tree with just the star of power \( 2 \) rooted at \( x \), but our proof method relies on T-joins!).

![Figure 5: All edges have their cost written: thinnest edges have cost 0, medium thick have cost 1, and thickest edges have cost 2. Arrows indicate the optimum power assignment solution. Solid edges give the minimum spanning tree, and its vertices of odd degree are solid and form the set \( R \). An example of a hypergraph T-join for \( R \) is given by the hyperedges represented by the three rounded shapes.](image)

Both the example below and the reminder of the proof of Lemma 3 are very long and technical and it makes sense to skip them in a first reading.

For a series of examples approaching \( 2/3 \), start (see Figure 6 for an illustration) with a complete binary tree of height \( h \) with nodes \( i \), \( 1 \leq i \leq 2^{h+1} - 1 \) (as in a binary heap, the children of node \( j \)), with \( j < 2^h \), are \( 2j \) and \( 2j+1 \).

Replace each node \( i \) with nodes \( y_i, z_i \), connected by an edge of cost \( 0 \), and, for \( i < 2^h \), add edges of cost \( 2 \) \( z_iy_{2i} \) and \( z_iy_{2i+1} \). Call the resulting tree \( B \); assume it is rooted at \( y_1 \), and for \( y_i \), let \( B_i \) be (the vertex set of) the subtree of \( B \) consisting of \( y_i \) and all its descendents. Add vertex \( u \) and edge of cost \( 1 \) \( uy_1 \). Add another \( 2^h \) vertices \( x_1, \ldots, x_{2^h} \) and edges of cost \( 1 \) \( x_iz_i+2^h-1 \) and of cost \( 0 \) \( x_iu \). Optimum has, for \( i = 1, 2, \ldots, 2^h - 1 \), \( p(z_i) = 2 \), for \( i = 2^h, \ldots, 2^{h+1} - 1, p(z_i) = 1 \), and \( p(u) = 1 \), with all the other vertices having power \( 0 \). The total cost of this solution is \( 2(2^h-1)+2^h+1 = 3 \cdot 2^h - 1 \) and it is optimum indeed, as its power is only \( 2 \) more than the cost of minimum spanning tree cost, described below, and one has, for the most costly edge \( e \) of the MST, \( p(OPT) \geq c(MST) + c(e) \), as proved in the remaining of this paragraph. Let \( U, W \) be the partition of the vertex set defined by MST after removing \( e \). Since OPT is strongly connected, there must exist an arc \( a = (u, w) \) in \( E(OPT) \) with \( u \in U \) and \( w \in W \). If \( p_{E(OPT)}(u) < c(e) \) then \( c(a) \leq p_{E(OPT)}(u) < c(e) \), so the tree \( T' = (MST \setminus e) \cup a \) is cheaper than
MST, a contradiction. Hence $p_{E(OPT)}(u) \geq c(e)$. Using $u$ as the root of a spanning in-arborescence inside $OPT$, we obtain as in the introduction that in fact $\sum_{v \in V \setminus \{u\}} p_{E(OPT)}(v) \geq c(MST)$.

One minimum spanning tree $T$ includes all the edges of cost 1 and 0, as well as the edges of cost 2 except the last level of the complete binary tree, that is $T$ contains the edges $z_iy_2$ and $z_iy_{2i+1}$, for $i < 2^{h-1}$. $R$, the set of vertices of odd degree consists of $u$, $z_i$, for $i < 2^h$, and $y_{j_i}$, for $2^h \leq i < 2^{h+1}$. While one can “cover” $T$ using the stars of power $2$ rooted at $z_i$, for all $i$ with $2^{h-1} \leq i < 2^h$, this cover does not give us a T-join for $R$.

Now we show that every T-join $M$ for $R$ has weight at least $2(2^h - 1)$. Assume that for some $i$ $1 \leq i < 2^h$, $M$ does not use the star with power $2$ rooted at $z_i$ (or else, we are done). Look at the subtrees $B = B_{2i}$ and $B' = B_{2i+1}$, each having an odd number of vertices of $R$. Now apply to $B$ the following “pruning” procedure: if some $z_j \in B$ and $j < 2^h$ also is such that $M$ does not use the star with power $2$ rooted at $z_j$, then remove from $B$ the vertices of $B_{j} \setminus \{y_j, z_j\}$; note that $B$ continues to have an odd number of vertices of $R$, since $B_{j}$ has an odd number of vertices of $R$, and therefore $B_{j} \setminus \{y_j, z_j\}$ had an even number of vertices of $R$. After doing this for all possible $j$, $B$ still has an odd number of vertices of $R$ and thus a hyperedge of $K$ must have an endpoint in $B$ and one outside - this cannot be the star rooted at $z_i$ or some pruned $z_j$ (it must be the edge $z_kx_{k-2^h+1}$ for some $k$ with $z_k$ decendant of $i$ in $B$), and this hyperedge must have weight at least 1. Similarly, after pruning, another hyperedge of weight at least 1 is obtained crossing $B'$; associate these two hyperedges to $z_i$. Notice
that for \( j \neq i \), we cannot associate the same hyperedge to both \( z_i \) and \( z_j \) since such an (hyper)edge \( z_k x_{k-2^h+1} \) will have \( z_k \) as descendant in \( B \) of both \( z_i \) and \( z_j \) - but then pruning will make sure that the higher (in \( B \)) of \( z_i \) and \( z_j \) cannot use \( z_k x_{k-2^h+1} \).

Thus whenever \( M \) does not use the star with power 2 rooted at \( z_i \), for some \( 1 \leq i < 2^h \), it must use two (hyper)edges of weight 1, not shared with another \( i \). We conclude that indeed \( w(M) \geq 2(2^h-1) \geq (2/3 - \epsilon)opt \). Note that this 2/3 lower bound holds for the T-ratio in hypergraphs that admit strongly connected orientations.

![Figure 7: The vertices of \( H_i \), a strongly connected subgraph, are in the ellipse. We select \( x_i \) to construct \( H_{i+1} \). \( \tilde{S}_i \) is represented by thick edges, with four leafs \( u_1, u_2, u_3, u_4 \). The path \( P_1 \) is represented by dashed arrows, \( P_2 \) and \( P_3 \) use solid arrows, while \( P_3 \) uses dash-dots arrows. Altogether, \( \tilde{S}_i \) and these paths are added to \( H_i \) to make \( H_{i+1} \).](image-url)

We resume the proof of Lemma 3. Recall that \( (S_v)_{v \in V} \) are the directed stars of \( OPT \), with \( S_v \) centered at \( v \), and \( \mathcal{A} \) is collection of these stars. \( K = (V_K, E_K) \) is the (undirected) hypergraph defined by \( V_K = V \) and \( E_K = \{ V(S) \mid S \in \mathcal{A} \} \). The weight of an hyperedge is the power of the corresponding directed star.

Remove from \( G \) all the arcs not in any \( (S_v)_{v \in V} \), and for the edges remaining set \( c(e) = p(S_v) \) if \( e \) has tail \( v \). As \( B \) will be a subset of \( \mathcal{A} \), these modifications do not change anything since for any \( v \), \( p(S_v) \) does not change. For intuition, we mentioned that \( \mathcal{A} \) is now a strongly connected, weighted, directed hypergraph, except that as opposed to standard notation [11], hyperedges have multiple heads and only one tail.

We do the following ear-decomposition of \( OPT \) (see Figure 7 for an illustration): start with one arbitrary directed cycle (graph) \( H_1 \) inside \( OPT \). We will construct strongly connected \( H_{i+1} \) out of \( H_i \), stopping only when \( V(H_i) = V \), as follows: Since \( OPT \) is strongly connected, there exist \( x_i \in V(H_i) \).
such that \( V(S_{x_i}) \) contains vertices not in \( H_i \). Let \( \hat{S}_i \) be the maximal substar of \( S_{x_i} \), whose leaves are not in \( H_i \). Let \( u_1, u_2, \ldots, u_k \) be the vertices of \( V(\hat{S}_i) \setminus \{x_i\} \). For \( j = 1 \) to \( k_i \), find a minimal path \( P_j^i \) in \( OPT \) from \( u_j \) to either a vertex in \( H_i \) or a vertex on a some \( P_q^i \) with \( q < j \). (strong connectivity guarantees the existence of these paths. Intuitively, the nice thing about these paths (and arborescences) is that their power equals their cost. Add \( B_i := \cup_j P_j^i \) to \( H_i \) to make \( H_{i+1} \). Let \( \tilde{i} \) be such that \( V(H_{\tilde{i}}) = V \), our last subgraph \( H \).

We have that \( H_i \) is a subgraph of \( OPT \), but not necessarily \( H_i \) is exactly the subgraph of \( OPT \) induced by \( V(H_i) \), as for example some \( u_j \) may have two arcs of \( OPT \) going to vertices of \( H_i \), and only one is included in \( H_{i+1} \).

Note that a vertex \( v \) has outdegree one when it joins its first \( H_i \); we call \( e_v \) the unique arc out of \( v \) in this \( H_i \). Also note that an \( x_i \) as above is not used twice in the ear decomposition. Let \( e_i \) be \( e_{x_i} \) (also depicted in Figure 7). For such an \( x_i \), let \( \hat{S}_i = \hat{S}_{x_i} \) be the star that contains \( e_i \) and all the arcs of \( \hat{S}_i \).

Let \( K_i \) be the following hypergraph: \( V(K_i) = V(H_i) \) and \( E(K_i) \) consists of the undirected version of the arcs of \( E(H_i) \) and, if \( i > 1 \), the hyperedges \( V(\hat{S}_j) \), for \( 1 \leq j \leq i - 1 \).

We use recursion to obtain a \( T \)-join \( J_i \) in \( K_i \), and an accounting scheme to prove that \( J_i \) has low weight. When processing \( H_i \), we are given the set \( R_i \) for which we must find a \( T \)-join \( J_i \) in hypergraph \( K_i \), and costs \( c_i \) on the edges of \( H_i \); for \( H_i \), \( R_i := R \) and \( c_i := c \). Costs \( c_i \) give power function \( p_i \) on \( H_i \), and as we will see when we set up the recursion, \( c_i \) may differ from \( c \) only on edges \( e_j \) for \( j \geq i \), for which \( c_i \) may be 0; if the recursion picks one such edge \( e \) of cost \( c_i(e) = 0 \), then the proof (later) makes sure that \( e \) will be removed at some point and not used in the final \( T \)-join; moreover \( e \) does not appear in any star of \( H_i \) with more than \( e \) as arcs (as \( e = e_v \) for some \( v \)).

Moreover, vertices \( v \) of \( V(H_i) \) can each have debt: \( \text{debt}_i(v) \), where \( \text{debt}_i(v) = 0 \) for all \( v \in V \). For \( K_i \), the weight of a hyperedge is obtained with respect to cost function \( c_i \). If \( i = 1 \), we will obtain (later):

\[
\sum_{v \in V(H_i)} \text{debt}_1(v) \leq (7/8)p_1(H_1) \quad (2)
\]

For \( i > 1 \), we will carefully (later) select \( R_{i-1} \) and \( c_{i-1} \), and recurse. Then we will construct \( J_i \), a \( T \)-join in \( K_i \) for \( R_i \) from \( J_{i-1} \) and some hyperedges of \( E(K_i) \setminus E(K_{i-1}) \) to satisfy:

\[
w_i(J_i) - w_{i-1}(J_{i-1}) + \sum_{v \in V(H_i) \setminus V(H_{i-1})} \text{debt}_i(v) \\
\leq (7/8)(p_i(H_i) - p_{i-1}(H_{i-1})) + \sum_{v \in V(H_{i-1})} (\text{debt}_{i-1}(v) - \text{debt}_i(v)). \quad (3)
\]

By summing up Inequations 2 and 4, one gets for all \( i \):

\[
w_i(J_i) + \sum_{v \in V(H_i)} \text{debt}_i(v) \leq (7/8)p_i(H_i), \quad (5)
\]

which is exactly what we need once we plug in \( i = \tilde{i} \). What actually happens when we look at the cases later is that only for \( v = x_{i-1} \), we can have \( \text{debt}_{i-1}(v) \neq \text{debt}_i(v) \), so one can also think as “\( x_{i-1} \) gets into debt for the operation and for retiring the debt of those nodes in \( H_i \) but not \( H_{i-1} \)”.

This way of thinking is also correct since \( x_i \neq x_j \) for \( i \neq j \), so \( x_{i-1} \) had no debt before we recourse from \( H_i \) to \( H_{i-1} \). Thus we think, when doing a recursive step, that we have \((7/8)(p_i(H_i) - p_{i-1}(H_{i-1})) \) cash in
hand, to pay for the operation and retiring the debt of those nodes in $V(H_i) \setminus V(H_{i-1})$; if this cash is not enough we borrow from (or, in other words, place a debt on) $x_{i-1}$.

We will prove that our recursion also maintains the following invariant: vertices have no debt except for those $v \in V(H_i)$ (for some $i$) such that $v = x_j$ for some $j \geq i$, for which

$$\text{debt}_i(v) \leq \frac{1}{8} c_i(e_v),$$

(6)

where recall that $e_v$ is the unique arc out of $v$ in $H_i$, i.e. if $v = x_j$, $e_v = e_j$.

If $v$ is added in $H_i$ (or $v \in V(H_i) \setminus V(H_{i-1})$, is in our last “ear”), then as implied before, $v$ carries no debt. Also, recall that $c_i(e) = c(e)$ for every edge $e$. For the maintenance of these invariants and the definition of $c_i$, we look at three cases.

In the first case, $i = 1$, and we deal with $H_1$, which is a directed cycle. We have $(7/8)p_1(H_1) = (7/8)c_1(H_1)$ cash (with outdegree 1 for every vertex, its power equals the cost of the outgoing arc). Exactly as in Christofides’ analysis, the arcs of $H_1$ are partitioned into two T-joins, $D_0$ and $F_0$ of $K_1$: go around the cycle and change T-join whenever meeting a vertex of $R_1$. That is, start with an arc arbitrarily and put it in $D_0$, and then process each arc of $C$ as follows: if the preceeding arc $e' \in D_0$ and the tail of $e$ is not in $R_1$, put $e \in D_0$; if $e' \in D_0$ and the tail of $e$ is in $R_1$, put $e \in F_0$; if $e' \in F_0$ and the tail of $e$ is in $R_1$, put $e \in D_0$; if $e' \in F_0$ and the tail of $e$ is not in $R_1$, put $e \in F_0$.

We use for our T-join $D_0$ if $c_1(D_0) \leq c_1(F_0)$; otherwise we use $F_0$. Our cash pays for the hyperedges we use as well for retiring the debt of all $v \in V(H_1)$: indeed this debt does not exceed $(1/8)(c_1(D_0) + c_1(F_0))$ provided the invariant is maintained. In other words, we get Inequation 2 using Invariant 6.

In the second case, $i > 1$ and $p_i(\hat{S}_{i-1}) \geq 2(c_i(B_{i-1}))$. We pick $J_i$, the T-join in $K_i$ for $R_i$, as follows: all the hyperedges of $K_i$ obtained from $B_{i-1}$ and all the hyperedges of $J_{i-1}$, a recursively-obtained a T-join in $K_{i-1}$ for $R_{i-1} \subseteq V(K_{i-1})$, where $R_{i-1}$ is constructed as follows: We set $R_{i-1} = R_i$, but then we modify it below, keeping in mind we must at the end have $R_{i-1} \subseteq V(H_{i-1})$ and $|R_{i-1}|$ even. $B_{i-1}$ consists of a collection of vertex-disjoint incoming arborescences $A_{i-1}^1$, each with its own distinct root $r_{i-1}^1$ in $V(H_{i-1})$. If $A_{i-1}^1$ has, including its root, an even number of vertices of $R_i$, remove those vertices from $R_{i-1}$ and add $r_{i-1}^1$ in $R_{i-1}$. If $A_{i-1}^1$ has, including its root, an even number of vertices of $R_i$, remove those vertices from $R_{i-1}$. Both transformation keep $R_{i-1}$ even-sized. Also, the final $R_{i-1}$ is a subset of $V(H_{i-1})$.

Moreover, the union of $B_{i-1}$ and a T-join in $K_{i-1}$ for this $R_{i-1}$ is indeed a T-join in $K_i$ for $R_i$, as we argue below. An arbitrary T-cut $(Q, \hat{Q})$ in $K_i$ for $R_i$ is covered by (the edges of) $B_{i-1}$ unless, for each index $j$, $Q$ contains all the arborescence $A_{i-1}^j$ or $\hat{Q}$ contains all the arborescence $A_{i-1}^j$. If the T-cut $(Q, \hat{Q})$ has this property, then replacing $R_i$ by $R_{i-1}$ does not change the parity of $Q \cap R_i$. Thus $(Q \cap V(K_{i-1}), \hat{Q} \cap V(K_{i-1}))$ is a T-cut in $K_{i-1}$ for $R_{i-1}$, and is therefore covered by the recursively constructed T-join in $K_{i-1}$ for $R_{i-1}$.

Also, in this second case, we keep $c_i(\hat{e}_i) = c_i(e)$ for every $e \in E(H_{i-1})$, and in particular $c_i(e_i) = c(e_i)$, as $e_i$ was not considered for a costs modification before this recursive step. We need to pay for the hyperedges obtained from the arcs of $B_{i-1}$ as well as debt accumulated by the vertices of $V(H_i) \setminus V(H_{i-1})$, which is, by Invariant 6, at most $(9/8)c_i(B_{i-1})$.

Our cash in hand is $(7/8)(p_i(H_i) - p_{i-1}(H_{i-1})) = (7/8)c_i(B_{i-1})$. We also put on $x_{i-1}$ a debt of $(1/8)c_i(e_i)$ (thus satisfying Invariant 6), and use this amount for the payment.

Using the fact that in this (second) case, $p_i(\hat{S}_{i-1}) \geq 2(c_i(B_{i-1}))$, and that $p_i(\hat{S}_{i-1}) = c_i(e_i) = c_i(e_i)$, we can immediatly verify that the cash in hand plus the one taken as a loan from the debt
on $x_{i-1}$ is enough to do the payment. Precisely, we verified that:

$$\frac{9}{8}c_i(B_{i-1}) \leq \frac{7}{8}c_i(B_{i-1}) + \frac{1}{8}c(e_{i-1}),$$

(7)

or in other words Inequality 4 holds.

In the third case, $i > 1$ and $p_i(\hat{S}_{i-1}) < 2c_i(B_{i-1})$. In this case we plan to use $\hat{S}_{i-1}$ as well as some edges from $B_{i-1}$ in addition to a T-join in $K_{i-1}$ for carefully defined $R_{i-1}$ and cost $c_{i-1}$, as described below. We set $R_{i-1} = R_i$, but then we modify it below, keeping in mind we must at the end have $R_{i-1} \subseteq V(H_{i-1})$ and $|R_{i-1}|$ even. Consider, one by one the vertex-disjoint arborescences in $B_{i-1}$, that is, for each $j$, $A_{i-1}^j$, and let $R_{i-1}^j = R_i \cap V(A_{i-1}^j)$. Make $A_{i-1}^j$ undirected, and add to it, if $r_{i-1}^j \neq x_{i-1}$, the vertex $x_{i-1}$ and the edge of weight 0: $r_{i-1}^j x_{i-1}$. For an edge/arc of $A_{i-1}^j$, have its weight equal its cost $c_i$. Add to $A_{i-1}^j$ the undirected version of the arcs of the star $\hat{S}_{i-1}$ with head in $A_{i-1}^j$ (tail is $x_{i-1}$ for all such arcs), each with weight 0.

This way we create a two-edge-connected undirected graph $Z_{i-1}^j$. Indeed, there are two edge-disjoint paths between any two vertices of $Z_{i-1}^j$, as explained in the reminder of this paragraph. If one vertex is the ancestor of the other in $A_{i-1}^j$, one path is in $A_{i-1}^j$, and the other goes from the lower of the two vertices to a leaf of $A_{i-1}^j$ to $x_{i-1}$ to $r_{i-1}^j$ to the highest of the two vertices. If none is the ancestor of the other, one path is obtained by going up from both vertices in $A_{i-1}^j$ until the least common ancestor, the other path by going down to leafs of $A_{i-1}^j$ and passing through $x_{i-1}$.

If $|R_{i-1}^j|$ is even, let $\hat{R}_{i-1}^j = R_{i-1}^j$, else $\hat{R}_{i-1}^j = R_{i-1}^j \otimes r_{i-1}^j$. In all cases, $\hat{R}_{i-1}^j$ is even-sized. There exists a minimal T-join $Y_{i-1}^j$ in $Z_{i-1}^j$ for $\hat{R}_{i-1}^j$ of weight at most $\frac{1}{2}w(E(Z_{i-1}^j))$. If this $Y_{i-1}^j$ contains the edge ( of weight 0) $r_{i-1}^j x_{i-1}$, then set $\hat{Y}_{i-1}^j = Y_{i-1}^j$ without this edge; otherwise $\hat{Y}_{i-1}^j := Y_{i-1}^j$. Also take out of $\hat{Y}_{i-1}^j$ the edges of $\hat{S}_{i-1}$; we are left only with the undirected version of arcs of $A_{i-1}^j$, a subgraph of $B_{i-1}$. Also, modify $R_{i-1}$ as indicated in the four subcases below.

In Subcase 1, $|R_{i-1}^j|$ is even, and $Y_{i-1}^j$ contains the edge ( of weight 0) $r_{i-1}^j x_{i-1}$ (so $x_{i-1} \neq r_{i-1}^j$); then set $R_{i-1} = (R_{i-1} \setminus R_{i-1}^j) \otimes \{x_{i-1}\} \cup \{r_{i-1}^j\}$. Note that whether $r_{i-1}^j \in R_{i-1}$ or not, $R_{i-1}$ stays even-sized.

In Subcase 2, $|R_{i-1}^j|$ is even and $Y_{i-1}^j$ does not contain the edge ( of weight 0) $r_{i-1}^j x_{i-1}$ (this is also the case when $r_{i-1}^j = x_{i-1}$); then set $R_{i-1} = R_{i-1} \setminus R_{i-1}^j$. Note that $R_{i-1}$ stays even-sized.

In Subcase 3, $|R_{i-1}^j|$ is odd and $Y_{i-1}^j$ contains the edge ( of weight 0) $r_{i-1}^j x_{i-1}$ (so $x_{i-1} \neq r_{i-1}^j$); then set $R_{i-1} = (R_{i-1} \setminus R_{i-1}^j) \otimes \{x_{i-1}\}$. Note that whether $r_{i-1}^j \in R_{i-1}$ or not, $R_{i-1}$ stays even-sized.

In Subcase 4, $|R_{i-1}^j|$ is odd and $Y_{i-1}^j$ does not contain the edge ( of weight 0) $r_{i-1}^j x_{i-1}$, (this is also the case when $r_{i-1}^j = x_{i-1}$); then set $R_{i-1} = (R_{i-1} \setminus R_{i-1}^j) \cup \{r_{i-1}^j\}$. Note that whether $r_{i-1}^j \in R_{i-1}$ or not, $R_{i-1}$ stays even-sized.

In all four cases, the vertices of $A_{i-1}^j$ other that $r_{i-1}^j$ are removed from $R_{i-1}$. Thus the final $R_{i-1} \subseteq V(H_{i-1})$. After we finish this for all $j$ ($x_{i-1}$ may enter and exit $R_{i-1}$ several times), set $c_{i-1}(e_{i-1}) = 0$ (for all the other edges $e$, keep $c_{i-1}(e) = c_i(e)$). Thus the final $R_{i-1} \subseteq V(H_{i-1})$.

Recourse in $K_{i-1}$, obtaining T-join $J_{i-1}$. Now we construct $J_i$, our desired (but not proven yet to be one) T-join in $K_i$ for $R_i$, as follows: $J_i = (J_{i-1} \setminus \{e_{i-1}\}) \cup \{\hat{S}_{i-1}\} \cup \bigcup \hat{Y}_{i-1}^j$. That is, we use the whole star of $x_{i-1}$, and if recursion uses the arc out of $x_{i-1}$ of cost $c_{i-1}$ zero, we give it up (since it is included in the star anyway). Note that in the end, all the arcs selected at artificial (reduced by the procedure) cost 0 are removed and replaced by a bigger star/hyperedge.
We need the following fact, for which we could only find a very long proof by case analysis despite the fact that this fact may be intuitively clear to the reader. Again, it makes sense to delay reading the proof.

**Claim 5** In all cases, $J_l$ is a T-join in $K_i$ for $R_l$.

**Proof.** If $e_{i-1} \notin J_{i-1}$, we used $\tilde{S}_{i-1}$ in $J_l$ instead of $e_{i-1}$ and $\tilde{S}_{i-1}$. However, with hyperedges $e_{i-1}$ and $\tilde{S}_{i-1}$ sharing vertex $x_{i-1}$, using $\tilde{S}_{i-1}$ is equivalent, for covering T-cuts, to using $e_{i-1}$ and $\tilde{S}_{i-1}$.

Let us look again at the construction of $R_{i-1}$. We started with $R_{i-1}(0) = R_i$ (please do not confuse $R_{i-1}(k)$ with $R_k$, they are not the same set). We processed one by one the arborescences $A^j_{i-1}$, for $j = 1, 2, \ldots, q$ (for some $q = q_i$), constructing set of edges $Y^j_{i-1}$, and $R_{i-1}(j)$ from $R_{i-1}(j-1)$, until $R_{i-1} = R_{i-1}(q)$ is the subset of $V(H_{i-1})$ used for the T-join $J_{i-1}$ in $K_{i-1}$.

Thus it is enough to show that $J_l = J_{i-1} \cup \{\tilde{S}_{i-1}\} \cup \left(\cup_{q=0}^{q_i} \tilde{Y}^q_{i-1}\right)$ is a T-join in $K_i$ for $R_l$ (since, if $e_{i-1} \notin J_{i-1}$, we make the proof with $\tilde{S}_{i-1}$ instead of the larger set $\tilde{S}_{i-1}$ as a hyperedge). Let $M_l := J_{i-1} \cup \{\tilde{S}_{i-1}\} \cup \left(\cup_{q=0}^{q_i} \tilde{Y}^q_{i-1}\right)$ (with $M_0 := J_{i-1} \cup \{\tilde{S}_{i-1}\}$), and note that we need to prove that $M_l$ is a T-join in $K_i$ for $R_l$. We prove by induction on $l$ that: $M_l$ is a T-join for $R_{i-1}(q-l)$ in $K_i$. Applying this with $l = q$ yields the claim.

For the base case ($l = 0$), let $(Q, \tilde{Q})$ be an arbitrary T-cut for $R_{i-1}(q) = R_{i-1}$. Then $(Q \cap V(H_{i-1}), \tilde{Q} \cap V(H_{i-1}))$ is a T-cut for $R_{i-1}$ in $K_{i-1}$, and therefore a hyperedge of the T-join $J_{i-1}$ crosses this T-cut, and $(Q, \tilde{Q})$ in $K_i$ as well. Thus $M_0$ is a T-join in $K_i$ for $R_{i-1}(q-0)$.

For the inductive case, proving for $l + 1$ assuming it holds for $l$, we must look at how $Y^q_{i-1}$ and $R_{i-1}(q-l)$ are constructed from $A^q_{i-1}$ and $R_{i-1}(q-l-1)$. To simplify notation, in the rest of the proof, let $x := x_{i-1}$, $r := r^q_{i-1}$, $Z := Z^q_{i-1}$, $R := R_{i-1}(q-l-1)$, $R' := R_{i-1}(q-l)$, and $Y := Y^q_{i-1}$, the minimal T-join in $Z$ for $R'_{i-1}$. To prove below that $M_{l+1}$ is a T-join for $R$ in $K_i$, we use that $M_l$ is a T-join for $R'$ in $K_i$.

To further simplify notation let $\tilde{S} := \tilde{S}_{i-1}$, $\tilde{R} := \tilde{R}^{q-1}_{i-1}$, $\tilde{R} := R_{i-1} \setminus \{r\}$, and $\tilde{Y} := \tilde{Y}^{q-1}_{i-1}$. Note that $M_{l+1} = M_l \cup \tilde{Y}$, that $\tilde{R} = R \cap (V(Z) \setminus \{x, r\})$, that $\tilde{R} = \tilde{R}$ or $R = R \cup \{r\}$ (whichever makes $|\tilde{R}|$ even), and that in all four subcases, $R' \subseteq (R \cup \{x, r\}) \setminus \tilde{R}$.

Let $(Q, \tilde{Q})$ be an arbitrary T-cut for $R$, that is, a partition of $V_H$, such that $|Q \cap R|$ has odd size. We need to find a hyperedge of $M_{l+1}$ crossing the T-cut. First, we switch $Q$ and $\tilde{Q}$ if necessary such that $r \in Q$. If $R' \cap Q$ is odd, $M_l$ has a hyperedge crossing $(Q, \tilde{Q})$ and therefore $M_{l+1}$ also has a hyperedge crossing $(Q, \tilde{Q})$. So, from now on we assume $|R' \cap Q|$ is even (and so is $|R' \cap \tilde{Q}|$).

We have, unfortunately, 16 cases based on whether $x \in R$ or not, $r \in R$ or not, $|\tilde{R}|$ even or not, and $Y$ contains $xr$ or not. One could combine cases, but for checking correctness needs to be included in all cases. In all cases, we find a hyperedge of $M_{l+1}$ that crosses $(Q, \tilde{Q})$: either $\tilde{S}$ or an edge of $\tilde{Y}$. To do so, it is enough to find an edge $e$ of $Y$, other than $xr$, crossing $(Q \cap V(Z), \tilde{Q} \cap V(Z))$. Indeed, $\tilde{Y}$ is obtained from $Y$ by removing the edges incident to $x$ (if any), and all such edges other than $xr$ are contained in the hyperedge $\tilde{S}$; so if $e$ is incident to $x$, $\tilde{S}$ also crosses $(Q, \tilde{Q})$. To find $e$, one reduces the 32 cases to one of the following three arguments:

**Argument I.** If $Y$ does not contain $xr$ and $Q \cap R$ is odd-sized, then $Y$, being a T-join for $\tilde{R}$ in $Z$, has an edge $e$ of $Y$ crossing in $Z$ the cut $(Q \cap V(Z), \tilde{Q} \cap V(Z))$; note that $e \neq xr$ as $xr \notin Y$, and we are done.
Argument II. If $Y$ contains $xr$, $x \in Q$, and $|Q \cap \hat{R}|$ odd, then $Y$, being a T-join for $\hat{R}$ in $Z$, has an edge $e$ of $Y$ crossing in $Z$ the cut $(Q \cap V(Z), \hat{Q} \cap V(Z))$; note that $e \neq xr$ since both $x$ and $r$ are in $Q$, and we are done.

Argument III. If $Y$ contains $xr$, $x \not\in Q$, and $|Q \cap \hat{R}|$ even, then we argue as follows. Recall that $Y$ is a minimal T-join in the graph $Z$ for $\hat{R}$. Let $D$ be the connected component of $(V(Z), Y)$ containing both $x$ and $r$, and split $D$ in components $D_r$ and $D_x$ by removing the edge $rx$, which belongs to $Y$. Then both $|D_r \cap \hat{R}|$ and $|D_x \cap \hat{R}|$ are odd (or else, $Y \setminus \{xr\}$ would have an even number of elements of $\hat{R}$ in each connected component, and thus would also be a T-join for $\hat{R}$, contradicting the minimality of $Y$).

If $D_r \not\in Q$, using $r \in Q \cap D_r$, we get that an edge of $Y$ other than $xr$ crosses $(Q \cap V(Z), \hat{Q} \cap V(Z))$, since $D_r$ is connected and contains only edges of $Y \setminus \{xr\}$. Now assume that $Q$ contains $D_r$. Using $|D_r \cap \hat{R}|$ is odd, we get that $\hat{R} \cap ((Q \cap V(Z)) \setminus D_r)$ is an odd-sized subset of $\hat{R}$, and thus $Y$, being a T-join for $\hat{R}$ in $Z$, has an edge $e$ crossing from $((Q \cap V(Z)) \setminus D_r)$. $e$ cannot have $x$ and $r$ as endpoints, as neither of $x, r$ is in $((Q \cap V(Z)) \setminus D_r)$ (recall that $x \not\in Q$ and $r \in D_r$). The endpoint of $e$ not in $((Q \cap V(Z)) \setminus D_r)$ cannot be in $D_r$ by the maximality of the connected component $D_r$; indeed the only edge of $Y$ crossing $D_r$ is $xr$ and we ruled out $e = xr$. Therefore $Y \setminus \{xr\}$ has the edge $e$ crossing $(Q \cap V(Z), \hat{Q} \cap V(Z))$, and we are done.

1. $x \in R$, $r \in R$, $|\hat{R}|$ even, $Y$ contains $xr$ (so $x \neq r$). Then $\hat{R} = \hat{R}$, we are in Subcase 3, and $R' = R \setminus V(Z)$. If $x \in Q$, then in order to have $|Q \cap R|$ odd, we must have $|Q \cap \hat{R}|$ odd (as we assumed $|R' \cap Q|$ is even, and $x, r \in R \cap Q$). Argument II applies. If $x \not\in Q$, then in order to have $|Q \cap R|$ odd, we must have $|Q \cap \hat{R}|$ even (as we assumed $|R' \cap Q|$ is even, and $r \in Q$, $x \not\in Q$). Argument III applies.

2. $x \in R$, $r \in R$, $|\hat{R}|$ even, $Y$ does not contain $xr$. It does not matter below whether $x \in Q$ ($x = r$ is possible) or $x \not\in Q$ (so $x \neq r$). Then $\hat{R} = \hat{R}$, we are in Subcase 4, and $R' = R \setminus \hat{R}$. In order to have $|Q \cap R|$ odd, we must have $|Q \cap \hat{R}|$ odd (as we assumed $|R' \cap Q|$ is even). Argument I applies.

3. $x \in R$, $r \in R$, $|\hat{R}|$ odd, $Y$ contains $xr$ (so $x \neq r$). Then $\hat{R} = \hat{R} \cup \{r\}$, we are in Subcase 1, and $R' = (R \setminus V(Z)) \cup \{r\}$. If $x \in Q$, then in order to have $|Q \cap R|$ odd, we must have $|Q \cap \hat{R}|$ even (as we assumed $|R' \cap Q|$ is even, and $x \in Q$), and therefore $|Q \cap \hat{R}|$ odd. Argument II applies. If $x \not\in Q$, then in order to have $|Q \cap R|$ odd, we must have $|Q \cap \hat{R}|$ odd (as we assumed $|R' \cap Q|$ is even, and $x \not\in Q$); thus $|Q \cap \hat{R}|$ is even. Argument III applies.

4. $x \in R$, $r \in R$, $|\hat{R}|$ odd, $Y$ does not contain $xr$. It does not matter below whether $x \in Q$ ($x = r$ is possible) or $x \not\in Q$ (so $x \neq r$). Then $\hat{R} = \hat{R} \cup \{r\}$, we are in Subcase 2, and $R' = R \setminus \hat{R}$. In order to have $|Q \cap R|$ odd, we must have $|Q \cap \hat{R}|$ odd (as we assumed $|R' \cap Q|$ is even). Argument I applies.

5. $x \in R$, $r \not\in R$ (so $x \neq r$), $|\hat{R}|$ even, $Y$ contains $xr$. Then $\hat{R} = \hat{R}$, we are in Subcase 1, and $R' = (R \setminus V(Z)) \cup \{r\}$. If $x \in Q$, then in order to have $|Q \cap R|$ odd, we must have $|Q \cap \hat{R}|$ odd (as we assumed $|R' \cap Q|$ is even, and using $x \in Q \cap R$ and $r \in (Q \cap R') \setminus \hat{R}$). Argument II
applies. If \( x \not\in Q \), then in order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) even (as we assumed \( |R' \cap Q| \) is even, and using \( x \not\in Q \) and \( r \in (Q \cap R') \setminus R \)). Argument III applies.

6. \( x \in R, r \not\in R \) (so \( x \neq r \)), \( |\hat{R}| \) even, \( Y \) does not contains \( xr \). It does not matter below whether \( x \in Q \) or \( x \not\in Q \). Then \( \hat{R} = \hat{R} \cup \{ r \} \), we are in Subcase 2, and \( R' = R \setminus \hat{R} \). In order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) odd (as we assumed \( |R' \cap Q| \) is even). Argument I applies.

7. \( x \in R, r \not\in R \) (so \( x \neq r \)), \( |\hat{R}| \) odd, \( Y \) contains \( xr \). Then \( \hat{R} = \hat{R} \cup \{ r \} \), we are in Subcase 3, and \( R' = (R \setminus V(Z)) \cup \{ r \} \). If \( x \in Q \), then in order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) even (as we assumed \( |R' \cap Q| \) is even, and also \( x \in Q \cap R \) and \( r \not\in (R \cup R') \)). With \( r \in Q \), we get \( |Q \cap \hat{R}| \) odd and Argument II applies. If \( x \not\in Q \), in order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) odd (as we assumed \( |R' \cap Q| \) is even, and using \( x \not\in Q \) and \( r \not\in (R \cup R') \)). Then \( |Q \cap \hat{R}| \) is even, and Argument III applies.

8. \( x \in R, r \not\in R \) (so \( x \neq r \)), \( |\hat{R}| \) odd, \( Y \) does not contains \( xr \). It does not matter below whether \( x \in Q \) or \( x \not\in Q \). Then \( \hat{R} = \hat{R} \cup \{ r \} \), we are in Subcase 4, and \( R' = (R \setminus \hat{R}) \cup \{ r \} \). In order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) even (as we assumed \( |R' \cap Q| \) is even, and using \( r \in R \setminus R' \) and \( x \in (R' \setminus Q) \setminus R \)). Therefore \( |Q \cap \hat{R}| \) is odd and Argument I applies.

9. \( x \not\in R, r \in R \) (so \( x \neq r \)), \( |\hat{R}| \) even, \( Y \) contains \( xr \). Then \( \hat{R} = \hat{R} \cup \{ r \} \), we are in Subcase 3, and \( R' = (R \setminus V(Z)) \cup \{ r \} \). If \( x \in Q \), then in order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) even (as we assumed \( |R' \cap Q| \) is even, and using \( r \in R \setminus R' \) and \( x \in (R' \setminus Q) \setminus R \)). Therefore \( |Q \cap \hat{R}| \) is even, and \( x \not\in Q \), and Argument III applies.

10. \( x \not\in R, r \in R \) (so \( x \neq r \)), \( |\hat{R}| \) even, \( Y \) does not contains \( xr \). It does not matter below whether \( x \in Q \) or \( x \not\in Q \). Then \( \hat{R} = \hat{R} \cup \{ r \} \), we are in Subcase 4, and \( R' = (R \setminus V(Z)) \cup \{ r \} = R \setminus \hat{R} \). In order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) odd (as we assumed \( |R' \cap Q| \) is even). With \( Q \cap \hat{R} = Q \cap R \), Argument I applies.

11. \( x \not\in R, r \in R \) (so \( x \neq r \)), \( |\hat{R}| \) odd \( Y \) contains \( xr \). Then \( \hat{R} = \hat{R} \cup \{ r \} \), we are in Subcase 1, and \( R' = (R \setminus V(Z)) \cup \{ r, x \} \). If \( x \in Q \), then in order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) even (as we assumed \( |R' \cap Q| \) is even, and using \( x \in Q \) and thus \( |Q \cap \hat{R}| \) odd. Argument II applies. If \( x \not\in Q \), then in order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) odd (as we assumed \( |R' \cap Q| \) is even, and using \( x \not\in Q \)), and therefore \( |Q \cap \hat{R}| \) even. Argument III applies.

12. \( x \not\in R, r \in R \) (so \( x \neq r \)), \( |\hat{R}| \) odd \( Y \) does not contains \( xr \). It does not matter below whether \( x \in Q \) or \( x \not\in Q \). Then \( \hat{R} = \hat{R} \cup \{ r \} \), we are in Subcase 2, and \( R' = R \setminus \hat{R} \). In order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) odd (as we assumed \( |R' \cap Q| \) is even). Argument I applies.

13. \( x \not\in R, r \not\in R \), \( |\hat{R}| \) even, \( Y \) contains \( xr \) (so \( x \neq r \)). Then \( \hat{R} = \hat{R} \), we are in Subcase 1, and \( R' = (R \setminus V(Z)) \cup \{ r, x \} \). If \( x \in Q \), then in order to have \( |Q \cap R| \) odd, we must have \( |Q \cap \hat{R}| \) odd (as we assumed \( |R' \cap Q| \) is even, and using \( \{ r, x \} \subseteq ((R' \setminus R) \cap Q) \)). With \( \hat{R} = \hat{R} \) and
Lemma 6 in [14] by changing what quantities represent and some parameters. which is true since in this (third) case

Proof. First, if \( c(T) \leq \alpha_{\text{opt}} \), then before any improvement we have a solution of cost at most \( 2\alpha_{\text{opt}} \) and \( 2\alpha \leq \beta \). Thus in the following we assume \( \text{opt} \geq c(T) > \alpha_{\text{opt}} \) (the first inequality is due from \( T \) being a minimum spanning tree).
Note that at the end of the algorithm, \( M \) contains exactly one of the two antiparallel arcs for each edge of \( T \). Then, for the final collection of stars \( A \), the output \( H \) satisfies
\[
p(H) \leq c(T) + w(A)
\] (9)
as it follows by summation over \( u \in V \) from
\[
p_H(u) = \max_{uv \in H} c(uv) \leq \sum_{uv \in M} c(uv) + \sum_{S \in A} c(S)
\]
which holds for every vertex \( u \) (recall that \( E(H) = \cup S \in A E(S) \cup M \)).

Let \( S_1, S_2, \ldots, S_q \) be the stars picked by our algorithm and let \( A_i \), for \( 1 \leq i \leq q \) be the collection of the first \( i \) stars; also let for convenience \( A_0 \) be the empty collection. For \( 1 \leq i \leq q \), let \( p_i = p(S_i) = r_i \), where \( r_i \) comes from \( S_i = S(u_i, r_i) \), and let \( f_i = f_{A_{i-1}}(S_i) \).

The greedy choice of the algorithm and the submodularity of the set function \( f_{A_{i-1}} \) (see Equation 1) give:
\[
p_i f_i \leq \sum_{B_j \in B} p(B_j) \leq \frac{w(B)}{f_{A_{i-1}}(B)} = \frac{w(B)}{c(T) - f(A_{i-1})} = \frac{w(B)}{c(T) - \sum_{j=1}^{i-1} f_j}.
\]

Rewriting and replacing \( w(B) \) with \( \alpha opt \), we obtain
\[
p_i \leq f_i \frac{\alpha opt}{c(T) - \sum_{j=1}^{i-1} f_j}.
\] (10)

Define the function \( g : [0..c(T)] \rightarrow [0..1] \) by \( g(x) = \alpha opt / (c(T) - x) \) for \( x \leq c(T) - \alpha opt \), and \( g(x) = 1 \) for \( x > c(T) - \alpha opt \). Then from Equation 10 and the observation (made right after the algorithm) that \( \frac{p_i}{f_i} \leq 1 \), we obtain: (see Figure 8):

Figure 8: The function \( g(x) \) is given by the solid curve. \( \sum_{i=1}^{q} p_i \) is the shaded area, in rectangles of width \( f_i \) and height \( \alpha opt / (c(T) - \sum_{j=1}^{i-1} f_j) \). In this particular picture, \( \alpha opt = (2/3) c(T) \) and therefore the integral is circa 0.94 \( c(T) \) (as mentioned after the lemma).
\[ \sum_{i=1}^{q} p_i \leq \int_{0}^{c(T)} g(x)dx = \int_{0}^{c(T) - \alpha \text{opt}} \frac{\alpha \text{opt}}{c(T) - x}dx + \int_{c(T) - \alpha \text{opt}}^{c(T)} 1dx \]

\[ = (-\alpha \text{opt}) \ln(c(T) - x) \big|_{0}^{c(T) - \alpha \text{opt}} + (c(T) - (c(T) - \alpha \text{opt})) \]

\[ = (-\alpha \text{opt}) (\ln(c(T) - (c(T) - \alpha \text{opt})) - \ln(c(T))) + \alpha \text{opt} = \alpha \text{opt} \left( 1 + \ln \frac{c(T)}{\alpha \text{opt}} \right) \]

Using this and \( c(T) \leq \text{opt} \) and Equation 9 (recall that \( w(A) = \sum_{i=1}^{q} p_i \)), we obtain that the power of the output is at most

\[ c(T) + \alpha \text{opt} \left( 1 + \ln \frac{c(T)}{\alpha \text{opt}} \right) \leq \text{opt} (1 + \alpha + \alpha \ln(1/\alpha)) \]

finishing the proof. ■

Based on Lemmas 2 and 3, Theorem 1 follows immediately from the fact that \( \alpha < 1 \) implies \( \beta < 2 \), which follows from \( \alpha (1 + \ln(1/\alpha)) < 1 \), which is equivalent to \( \ln(1/\alpha) < 1/\alpha - 1 \), a fact that holds for all \( \alpha < 1 \). For \( \alpha = 7/8 \), we obtain \( \beta \leq 1.992 \). For \( \alpha = 4/5 \) (see next section), we obtain \( \beta \leq 1.98 \).

If one were to prove \( \alpha = 1/2 \) or \( \alpha = 2/3 \) in Lemma 3, the resulting approximation ratio would be less than 1.85 or 1.94 respectively.

### 3.3 Improved T-join Ratio

With Lemma 3 so arduous, we decided to only sketch a better ratio.

**Lemma 7** There exists a collection of stars \( B \) with \( f(B) = c(T) \) and \( w(B) \leq (4/5) \text{opt} \), where \( \text{opt} \) is the power of the optimum solution.

**Proof sketch.** The proof is as in Lemma 3 before the charging/accounting scheme. However, we allow debt \( \text{debt}_i(v) \leq (1/5)c_i(e_v) \) instead of \( (1/8)c_i(e_v) \), and we recourse in a similar but more complicated way.

The base case needs to pay \( (1/2)c_1(H_1) + (1/5)c_1(H_1) \) for the T-join \( J_1 \) and retiring the debt of all vertices, using cash of \( (4/5)p_1(H_1) \), which is enough.

For the recursion, as before, we have \( H_i \), and follow the third case of the proof of Lemma 3. We construct \( Z_{i-1}^j \) as there, but then instead of settling for one \( Y_{i-1}^j \) of weight at most \( \frac{1}{2}w(Z_{i-1}^j) \), find (next paragraph) two T-joins \( \hat{Y}_{i-1}^j \) and \( \bar{Y}_{i-1}^j \) such that \( \hat{Y}_{i-1}^j \) contains \( r_{i-1}^j x_{i-1} \) (assuming this edge exists, i.e. \( r_{i-1}^j \neq x_{i-1} \)) and \( \bar{Y}_{i-1}^j \) does not contain \( r_{i-1}^j x_{i-1} \) and such that \( w(\hat{Y}_{i-1}^j) + w(\bar{Y}_{i-1}^j) \leq w(Z_{i-1}^j) \); this is indeed possible as argued below.

If the edge \( r_{i-1}^j x_{i-1} \) does not exist (that is, if \( r_{i-1}^j = x_{i-1} \)) then set \( \hat{Y}_{i-1}^j = \bar{Y}_{i-1}^j = Y_{i-1}^j \), where \( Y_{i-1}^j \) comes from the proof of Lemma 3. Otherwise, do an ear decomposition of \( Z_{i-1}^j \) with the first cycle containing the edge \( r_{i-1}^j x_{i-1} \). For every ear other than the first cycle, traverse it changing sides each time you meet a vertex of \( R_{i-1}^j \) - then pick the cheapest of the two edge sets. Set up recursion \( R - \) like in the Lemma 3, but simpler; we pay half of the cost reduction when we recourse. Finally, in the last cycle, partition it into two sets of edges as in the base case of Lemma 3, making sure the edge \( r_{i-1}^j x_{i-1} \) is in \( \hat{Y}_{i-1}^j \) and not \( \bar{Y}_{i-1}^j \).
Let $B' = \bigcup \tilde{Y}^j_{i-1}$ and $B'' = \tilde{Y}^j_{i-1}$; thus $w(B') + w(B'') \leq c_i(B_i)$. Edges of $B'$ and $B''$ come from either arcs of $B_{i-1}$ or arcs of $\tilde{S}_{i-1}$ or are of type $r^j_{i-1}x_{i-1}$ for some $j$, with all edges of this later type in $B''$. Let $B'$ be the arcs of $B_{i-1}$ which give rise to edges of $B'$, and $B''$ be the arcs of $B_{i-1}$ which give rise to edges of $B''$. We do not have that $B''$ and $B'$ are disjoint but we do have

$$c_i(B') + c_i(B'') \leq c_i(B_{i-1}).$$

(11)

In a first case, $p_i(\tilde{S}_{i-1}) \leq c_i(B'')$. Then we proceed as in the third case of Lemma 3. The cash in hand is $(4/5)\left(p_i(\tilde{S}_{i-1}) + c_i(B_{i-1})\right)$. We use it to pay $p_i(\tilde{S}_{i-1}) + \min\left(c_i(B'), c_i(B'')\right)$, the cost of upgrading $J_{i-1}$ to $J_i$, and another $(1/5)c_i(B_{i-1})$ to pay for retiring the debt of vertices in $V(H_i) \setminus v(H_{i-1})$. Thus to maintain the credit invariant it will be enough if

$$\frac{3}{5}c_i(B_{i-1}) \geq \frac{1}{5}c_i(B'') + \min\left(c_i(B'), c_i(B'')\right),$$

(12)

where we used $p_i(\tilde{S}_{i-1}) \leq c_i(B'')$. Then, if $c_i(B'') \leq c_i(B')$, then the inequality above becomes $\frac{3}{5}c_i(B_{i-1}) \geq \frac{1}{5}c_i(B'') + c_i(B')$, which is indeed true in this subcase, using Inequation 11. If $c_i(B'') > c_i(B')$, then Inequation 12 becomes $\frac{3}{5}c_i(B_{i-1}) \geq \frac{1}{5}c_i(B'') + c_i(B')$, which is, using Inequation 11, true in this second subcase.

So from now we assume $p_i(\tilde{S}_{i-1}) \geq c_i(B'')$. Also, if

$$\frac{3}{5}c_i(B_{i-1}) \geq \frac{1}{5}p_i(\tilde{S}_{i-1}) + c_i(B''),$$

(13)

then we proceed as above, and the credit invariant is maintained.

So from now on, Inequality 13 does not hold, and $p_i(\tilde{S}_{i-1}) > c_i(B'')$. Set $c_{i-1}(e_{i-1}) = c_i(e_{i-1}) - c_i(\tilde{B}'')$; recall that $c_i(e_{i-1}) = p_i(\tilde{S}_{i-1})$. Set $R_{i-1}$ as in the third case in the proof of Lemma 3 using for each $j$, $\tilde{Y}^j_{i-1}$ instead of $Y^j_{i-1}$. We are either in Subcase 2 (with $R^j_{i-1}$ even-sized) or Subcase 4 (when on can check that $r^j_{i-1}$ is in the final $R_t$). It is important to observe that $R_t$ is the same as in the second case of the proof of Lemma 3. We recurse in $H_{i-1}$ with cost $c_{i-1}$, obtaining T-join $J_{i-1}$ in $K_{i-1}$ for $R_{i-1}$. If $J_{i-1}$ does not contain $e_{i-1}$, we set $J_i = J_{i-1} \cup B_{i-1}$, which is indeed a T-join in $K_i$ for $R_t$ as argued in the second case of the proof of Lemma 3. Otherwise, $J_{i-1}$ contains $e_{i-1}$, we set $J_i = J_{i-1} \setminus \lbrace e_{i-1} \rbrace \cup \lbrace \tilde{S}_{i-1} \rbrace \cup B'$, which is indeed a T-join in $K_i$ for $R_t$ as argued in the third case (Subcases 2 and 4, see Claim 5) of the proof of Lemma 3. Note that in the end, all the arcs selected at artificial (reduced by the procedure) cost are removed and replaced by a bigger star/hyperedge, with its original cost.

In both subcases, we have:

$$w_i(J_i) - w_{i-1}(J_{i-1}) \leq c_i(B_{i-1}),$$

(14)

using in the second subcase that $c_{i-1}(e_{i-1}) = c_i(e_{i-1}) - c_i(\tilde{B}'') = p_i(\tilde{S}_{i-1}) - c_i(\tilde{B}'')$ and Inequality 11.

Thus we need to pay at most $(6/5)c_i(B_{i-1})$ for the operation, including retiring the debt of the vertices of $V(H_i) \setminus V(H_{i-1})$. The cash in hand is $(4/5)\left(p_i(H_i) - p_{i-1}(H_{i-1})\right) = (4/5)\left(c_i(B_{i-1}) + c_i(\tilde{B}'')\right)$. In addition, we put a debt of $(1/5)c_{i-1}(e_{i-1})$ on $x_{i-1}$ (previously, debt-free). Thus, to maintain the credit invariant, it is enough that

$$\frac{6}{5}c_i(B_{i-1}) \leq \frac{4}{5} \left(c_i(B_{i-1}) + c_i(\tilde{B}'')\right) + \frac{1}{5} \left(p_i(\tilde{S}_{i-1}) - c_i(\tilde{B}'')\right).$$

(15)
This is equivalent to
\[ \frac{2}{5} c_i(B_{i-1}) \leq \frac{1}{5} p_i(\hat{S}_{i-1}) + \frac{3}{5} c_i(\bar{B}'''). \] (16)

Since Equation 13 does not hold in this subcase, we obtain:
\[
\frac{2}{5} c_i(B_{i-1}) = \frac{2}{3} \cdot \frac{3}{5} c_i(B_{i-1}) < \frac{2}{3} \left( \frac{1}{5} p_i(\hat{S}_{i-1}) + c_i(\bar{B}'') \right)
\]
\[
\leq \frac{2}{15} p_i(\hat{S}_{i-1}) + \frac{2}{3} c_i(B'')
\]
\[
< \frac{2}{15} p_i(\hat{S}_{i-1}) + \frac{2}{3} c_i(B'') + \frac{1}{15} p_i(\hat{S}_{i-1}) - \frac{1}{15} c_i(\bar{B}'')
\]
\[
= \frac{1}{5} p_i(\hat{S}_{i-1}) + \frac{3}{5} c_i(\bar{B}''),
\]
with the last inequality holding since we are in the case \( p_i(\hat{S}_{i-1}) \geq c_i(\bar{B}'') \). Thus there is enough cash to maintain the credit invariant and pay for the operation. \( \blacksquare \)

The sketch/proof above holds for T-joins in strongly connected, weighted, directed hypergraphs, with minor modifications (when doing the ear decomposition, one may have to start \( V_{i+1} - V_i \) with a hyperedge with only the tail in \( V_i \). In that case we cannot put a debt on this hyperedge, but it is possible to check that either \( B_i \), or this edge plus the smaller of \( B' \) and \( B'' \) can be used for the recursion).

For two-edge-connected hypergraphs, the T-ratio converges to 1 as the number of edges increases, as we see in the following series of examples. For integer \( k \) multiple of 8, have \( \binom{k}{2} \) vertices \( u_{ij} \), where \( 1 \leq i < j \leq k \). The \( k \) hyperedges are \( e_1, e_2, \ldots, e_k \) (all with weight 1), and \( e_i \) contains, for all \( j \) with \( 1 \leq j < i \), \( u_{ji} \), and for all \( j \) with \( i < j \leq k \), \( u_{ij} \). One can check this hypergraph is two-edge-connected. With \( R \) given by \( V \), missing any two hyperedges results in an isolated vertex and then the T-cut with this vertex on one side is not covered; thus any T-join has size/weight at least \( k-1 \). A T-ratio of 1 does not give any improvement in the approximation ratio, so we had to use strong connectivity instead of two-edge-connectivity.

### 4 Conclusions

Based on the discussion in the previous section, we also have: if we are looking for a minimum weight strongly connected spanning subgraph in a directed graph, and a bidirected tree exists of cost at most twice optimum, then Christofides’ method gives a 1.5 approximation. The same holds for Minimum Weight Two-Edge-Connected Spanning Subgraph if there always exists a “double tree” (that is, where each edge of the tree has a twin) of weight at most twice optimum. These facts were known (at least by the author since 2003), and are implicit in the work of Williamson [27], which shows a 1.5 approximation for Minimum Weight Two-Edge-Connected Spanning Subgraph with triangle inequality.

We do not see how the recent breakthrough by Byrka et. al [3] for Steiner Tree would give techniques that can be applied to any Power Assignment problem, since we do not see the equivalent of the concept of “loss” used explicitly by [23] and implicitly by [3].

We leave open the exact T-ratio for hypergraphs that admit strongly connected orientation.
5 Acknowledgments

We would like to thank Alexander Zelikovsky for a streamlined explanation of his algorithms.

References


