Algorithms for node-weighted Steiner tree and maximum-weight connected subgraph

Austin Buchanan*[1], Yiming Wang[2], and Sergiy Butenko[2]

[1]Industrial Engineering & Management, Oklahoma State University
[2]Industrial and Systems Engineering, Texas A&M University

October 20, 2017

Abstract

This paper considers the node-weighted Steiner tree (NWST) problem and the maximum-weight connected subgraph (MWCS) problem, which have applications in the design of telecommunication networks and the analysis of biological networks. Exact algorithms with provable worst-case runtimes are provided. The first algorithm for NWST runs in time $O(n^3)$ for n-vertex instances when the number of terminals is bounded. It is based on dynamic programming and generalizes a Steiner tree algorithm of Dreyfus and Wagner. When used alongside Hakimi's spanning tree enumeration algorithm, it implies a time $O(1.5296^n)$ algorithm for NWST. It is also shown that Hakimi's 46-year-old algorithm for Steiner tree is essentially best-possible under the strong exponential time hypothesis (SETH). Then two algorithms for MWCS are provided. Their runtimes are polynomial in the number of vertices of the graph, but exponential in the number of vertices that have positive (or negative) weight. The latter is shown to be essentially best-possible under SETH. Together, they imply that MWCS can be solved in time $O(1.5875^n)$. To the best of the authors' knowledge, these are the first improvements over exhaustive search in the literature.

Keywords: Steiner tree; node-weighted Steiner tree; maximum weight connected subgraph; fixed-parameter tractable; strong exponential time hypothesis;

1 Introduction

The Steiner tree problem is well-studied in the literature and has applications in the design of telecommunication and transportation networks, see [12, 25] and references therein. It and its variants have experienced a surge of interest lately as the subject of the 11th DIMACS Implementation Challenge [1].

*Corresponding author: buchanan@okstate.edu

Problem: Steiner Tree.
Input: a graph $G = (V, E)$, a set $T \subseteq V$ of terminal vertices, a nonnegative weight $w(e)$ for each edge $e \in E$.
Output: a minimum weight subset of edges that connects the terminals.

The Steiner tree problem is well-known to be NP-hard [15], and this hardness persists even for planar instances [9]. However, it is fixed-parameter tractable (fpt) [5] with respect to the number k of terminals, as demonstrated by the Dreyfus-Wagner algorithm [6], which runs in time $O(3^k n + 2^k n^2 + n^3)$. Here, and throughout the paper, n and m denote the number of nodes and edges, respectively, unless stated otherwise.

If one allows for the vertices to also have costs associated with them, this results in the *node-weighted* Steiner tree (NWST) problem [21].

Problem: Node-Weighted Steiner Tree (NWST).
Input: a graph $G = (V, E)$, a set $T \subseteq V$ of terminal vertices, a nonnegative weight $w(i)$ for each vertex/edge $i \in E \cup V$.
Output: a minimum weight subgraph that connects the terminals.

There are considerably fewer algorithms in the literature for NWST. Indeed, we can find no previous algorithm for NWST with a nontrivial worst-case runtime. In Section 2, we generalize the Dreyfus-Wagner algorithm to handle both edge and vertex weights at no extra cost in runtime. So, if the number k of terminals is small compared to the number n of nodes, then this fpt algorithm will run more quickly than algorithms whose running time is exponential in n. When our generalization of the Dreyfus-Wagner algorithm is used alongside Hakimi's spanning tree enumeration algorithm [11], it implies a time $O(1.5296^n)$ algorithm for NWST, which is considerably faster than exhaustive search.

Interestingly, under the *strong exponential time hypothesis* (SETH) [13], Hakimi's algorithm (and its generalization for solving NWST) cannot be improved. SETH is an unproven complexity assumption that is stronger than P$\neq$NP. It asserts that for every $\epsilon > 0$, there is a k such that k-CNF-SAT cannot be solved in time $O((2-\epsilon)^n)$, where n denotes the number of variables. If true, this would imply that CNF-SAT cannot be solved in time $(2-\epsilon)^n(n+m)^{O(1)}$, where m denotes the number of clauses. While some doubt the verity of SETH (e.g., Williams [24]), it is consistent with the fastest known algorithms for SAT, and we argue that designing a faster algorithm than Hakimi's algorithm is as hard as finding a faster algorithm for SAT, cf. [20]. The reader is referred to the survey of Lokshtanov et al. [18] for more information on lower bounds conditional on SETH and its weaker variant, the *exponential time hypothesis* (ETH).

We also consider the related maximum-weight connected subgraph (MWCS) problem, which has applications in cluster detection in bioinformatics [4] and many other areas [2]. This problem is also NP-hard, even when restricted to planar graphs of maximum degree three with all weights either $+1$ or -1, see [14, 22]. The problem has no set of terminal vertices, and the problem's difficulty arises due to the possibility for negative-weight vertices.

Problem: Maximum-Weight Connected Subgraph (MWCS).
Input: a graph $G = (V, E)$, a weight $w(v)$ for each vertex $v \in V$.
Output: a maximum-weight subset S of vertices such that $G[S]$ is connected.

We are aware of no previously published MWCS algorithms with nontrivial worst-case runtimes, but we show that the problem can be solved in time $O(1.5875^n)$. Our algorithm relies upon two different fixed-parameter tractable algorithms. The first solves MWCS in time $O(2^q(m+n))$, where q denotes the number of negative-weight vertices. Using a similar reduction as before, we show that this is essentially best-possible under SETH. The second uses the NWST algorithm as a subroutine and runs in time $O(4^p n^3)$ for MWCS instances with p positive-weight vertices.

Recently, we have shown that MWCS can be solved in polynomial time whenever the independence number $\alpha(G)$ of the graph G is at most two. Indeed, in this case, we have shown that the following linear programming relaxation for MWCS has integral extreme points [23].

$$\max \sum_{i \in V} w(i) x_i \tag{1}$$

$$x_a + x_b - \sum_{i \in C} x_i \leq 1 \qquad \forall a,b\text{-separator } C \subseteq V \setminus \{a,b\},\ \forall \{a,b\} \in \binom{V}{2} \setminus E \tag{2}$$

$$0 \leq x_i \leq 1 \qquad \forall i \in V. \tag{3}$$

Thus, by the ellipsoid method [10], we can solve MWCS in polynomial time[1] when $\alpha(G) \leq 2$. This observation naturally leads to the question of whether MWCS admits efficient algorithms when $\alpha(G)$ is sufficiently small. In Section 3.2, we provide the first such algorithm, showing that MWCS is solvable in time $O(n^3)$ for classes of graphs G in which $\alpha(G)$ is bounded. Indeed, the runtime is $O(4^{\alpha(G)} n^3)$.

2 Node-Weighted Steiner Tree

In this section, we provide algorithms for NWST. The first generalizes the Drefyus-Wagner algorithm and runs in time $O(n^3)$ when the number of terminals is bounded. Using (essentially) Hakimi's spanning tree enumeration algorithm [11], this implies that NWST can be solved in time $O(1.5296^n)$.

2.1 NWST with few terminals

Perhaps the most well-known algorithm to solve the Steiner tree problem is a dynamic programming algorithm due to Dreyfus and Wagner [6]. It runs in time $O(3^k n +$

[1] Note that there can be exponentially many separator inequalities (2), but they can be separated in polynomial time by reduction to max flow, see [7]. Alternatively, there is an equivalent polynomial-size multicommodity flow formulation [23], so one can use any polynomial-time LP algorithm.

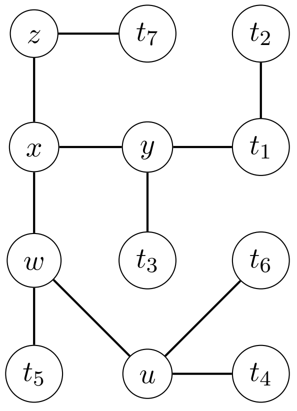

Figure 1: Illustration of the notations. This tree $\mathcal{T}$ has 12 vertices. Let the terminal set be $T = \{t_1, \ldots, t_7\}$. Here, $T_y(x) = \{t_1, t_2, t_3\}$, $T_z(x) = \{t_7\}$, and $T_{\{y,z\}}[x] = \{x, t_1, t_2, t_3, t_7\}$. The subtree $\mathcal{T}^{\{w,y\}}(x)$ is $\mathcal{T} - z - t_7$.

$2^k n^2 + n^3$), where k denotes the number of terminals. We generalize this algorithm to solve NWST within the same time bound.

To prove the correctness of the NWST algorithm, we will need some notation and an optimal substructure lemma. Consider a subtree $\mathcal{T}$ connecting a set $T \subseteq V$ of terminals. The neighborhood of vertex v in graph G is denoted $N_G(v) := \{u \in V \mid \{u, v\} \in E\}$. For a vertex $x \in V(\mathcal{T})$ and one of its neighbors $y \in N_{\mathcal{T}}(x)$ in $\mathcal{T}$, denote by $T_y(x) \subseteq T$ the set of terminal vertices reachable from x via a (simple) path in $\mathcal{T}$ that first crosses y. Further, for $Y \subseteq N_{\mathcal{T}}(x)$, define $T_Y(x) := \cup_{y \in Y} T_y(x)$ and $T_Y[x] := T_Y(x) \cup \{x\}$. Finally, denote by $\mathcal{T}^Y(x)$ the (smallest) subtree of $\mathcal{T}$ that connects $T_Y[x]$. Note that Y is superscript for subtree $\mathcal{T}^Y(x)$, but is subscript for terminal subset $T_Y(x)$. These notations are illustrated in Figure 1.

Lemma 1 (Optimal substructure)**.** *Let $\mathcal{T}$ be a minimum NWST connecting terminals $T \subseteq V$. Consider a vertex $x \in V(\mathcal{T})$ and a subset $Y \subseteq N_{\mathcal{T}}(x)$ of its neighbors in $\mathcal{T}$. Then the subtree $\mathcal{T}^Y(x)$ is a minimum NWST connecting $T_Y[x]$.*

Proof. Let $\mathcal{T}$ be a NWST connecting terminals T, x be a vertex in $\mathcal{T}$, and Y be a subset of neighbors of x in $\mathcal{T}$. We show that if the subtree $\mathcal{T}^Y(x)$ is not a minimum NWST connecting $T_Y[x]$, then $\mathcal{T}$ is not a minimum NWST connecting T. Let $\mathcal{T}_2$ be the subgraph of $\mathcal{T}$ such that $\mathcal{T}_2 \cup \mathcal{T}^Y(x) = \mathcal{T}$ and $\mathcal{T}_2 \cap \mathcal{T}^Y(x) = (\{x\}, \emptyset)$, Then,

$$\begin{aligned}
&\sum_{v \in V(\mathcal{T})} w(v) + \sum_{e \in E(\mathcal{T})} w(e) \\
= &\sum_{v \in V(\mathcal{T}^Y(x))} w(v) + \sum_{e \in E(\mathcal{T}^Y(x))} w(e) + \sum_{v \in V(\mathcal{T}_2)} w(v) + \sum_{e \in E(\mathcal{T}_2)} w(e) - w(x)
\end{aligned}$$

If $\mathcal{T}^Y(x)$ is not a minimum NWST connecting $T_Y[x]$, then there is a connected subgraph $\mathcal{T}_1$ that connects $T_Y[x]$ and

$$\sum_{v\in V(\mathcal{T}_1)} w(v) + \sum_{e\in E(\mathcal{T}_1)} w(e) < \sum_{v\in V(\mathcal{T}^Y(x))} w(v) + \sum_{e\in E(\mathcal{T}^Y(x))} w(e).$$

Then, consider $\mathcal{T}' = \mathcal{T}_1 \cup \mathcal{T}_2$, which is connected and has weight

$$\begin{aligned} & \sum_{v\in V(\mathcal{T}')} w(v) + \sum_{e\in E(\mathcal{T}')} w(e) \\ \leq \quad & \sum_{v\in V(\mathcal{T}_1)} w(v) + \sum_{e\in E(\mathcal{T}_1)} w(e) + \sum_{v\in V(\mathcal{T}_2)} w(v) + \sum_{e\in E(\mathcal{T}_2)} w(e) - w(x) \\ < \quad & \sum_{v\in V(\mathcal{T})} w(v) + \sum_{e\in E(\mathcal{T})} w(e) \end{aligned}$$

This shows that if $\mathcal{T}^Y(x)$ is not a minimum NWST connecting $T_Y[x]$, then $w(\mathcal{T}') < w(\mathcal{T})$, implying that $\mathcal{T}$ cannot be a minimum NWST. □

Algorithm 1, which solves NWST using dynamic programming, is described in a bottom-up fashion—much like the Dreyfus-Wagner algorithm—but needs adjustments to account for the vertex weights. It computes the minimum cost $q(v, D)$ of connecting vertex v to the (subset of) terminals D. Before computing the value of $q(v, D)$, the algorithm computes the minimum cost $p(v, D)$ of connecting v to D in a solution in which v is an interior (non-leaf) node. This cost is $p(v, D) = q(v, A) + q(v, D \setminus A) - w(v)$ for some nonempty subset $A \subsetneq D$. The adjustment $-w(v)$ avoids double-counting the cost of v. Now, if v is indeed an interior node in a minimum cost NWST that connects $D \cup \{v\}$, then the algorithm ultimately sets $q(v, D) = d_{vv} + p(v, D) - w(v) = p(v, D)$, as desired. Otherwise, v is a leaf node in all minimum cost NWSTs that connect $D \cup \{v\}$, in which case the cost is $q(v, D) = d_{vu} + p(u, D) - w(u)$, where u is the first vertex of degree $\neq 2$ that is encountered in the tree by starting a path from v. Again, the term $-w(u)$ avoids double-counting the weight of u.

Theorem 1. *Algorithm 1 is correct and solves the NWST problem in time $O(3^k n + 2^k n^2 + n^3)$, where k denotes the number of terminals.*

Proof. The proof of the algorithm's correctness is based on the claim that, for each vertex v and each nonempty vertex subset D, the term $q(v, D)$ is the weight of a minimum NWST connecting $D \cup \{v\}$. Thus, the returned value $q(s, \tilde{T})$ will be the weight of a minimum NWST connecting $T = \tilde{T} \cup \{s\}$. The claim about $q(\cdot, \cdot)$ is proven by induction on $|D|$.

If $|D| = 1$, then this is a shortest path problem and $q(v, D) = d_{vu}$ for some u, so the statement is true. Now suppose the statement is true for $|D| < i$. When $|D| = i$, consider a minimum NWST $\mathcal{T}_0$ connecting v and D. Denote the weight of a subgraph H by $w(H)$. We consider three cases regarding the neighborhood of vertex v in $\mathcal{T}_0$.

Algorithm 1 An algorithm for NWST with terminal set T, $k = |T|$

```
1: If terminals are not reachable from each other, return $\infty$;
2: Compute length $d_{uv}$ of shortest path between all $u, v \in V$, where we define
   $d_{uu} = w(u)$. A path's length is the sum of weights of all of its edges and vertices
   (including endpoint vertices);
3: For every pair $u, v \in V$ of vertices, let $q(u, \{v\}) = d_{uv}$;
4: Arbitrarily pick $s \in T$ and let $\tilde{T} = T \setminus \{s\}$;
5: for $i = 2, 3, \ldots, k-1$ do
6:   for all $D \subseteq \tilde{T}$ with $|D| = i$ do
7:     for all $v \in V$ do
8:       $p(v, D) = \min_{A: 0<|A|<|D|} \{q(v, A) + q(v, D \setminus A) - w(v)\}$;
9:     end for
10:    for all $v \in V$ do
11:      $q(v, D) = \min_{u \in V} \{d_{vu} + p(u, D) - w(u)\}$;
12:    end for
13:  end for
14: end for
15: return $q(s, \tilde{T})$;
```

1. $|N_{\mathcal{T}_0}(v)| \geq 2$, then there exist nonempty subsets $Q \subsetneq N_{\mathcal{T}_0}(v)$ and $Q' := N_{\mathcal{T}_0}(v) \setminus Q$. Since $\mathcal{T}_0$ is a minimum NWST connecting D, we can assume, without loss of generality, that every leaf of $\mathcal{T}_0$ belongs to D. This implies that[2] $|D_Q(v)| \geq 1$ and $|D_{Q'}(v)| \geq 1$. Also $D_Q(v) \cap D_{Q'}(v) = \emptyset$ and $D_Q(v) \cup D_{Q'}(v) = D$. By Lemma 1, $\mathcal{T}^Q(v)$ is a minimum NWST connecting $D_Q[v]$, and $\mathcal{T}^{Q'}(v)$ is a minimum NWST connecting $D_{Q'}[v]$. Since $|D_Q(v)| < |D|$ and $|D_{Q'}(v)| < |D|$ (and by the induction assumption), the following holds:

$$\begin{aligned} w(\mathcal{T}_0) &= w(\mathcal{T}^Q(v)) + w(\mathcal{T}^{Q'}(v)) - w(v) \\ &= q(v, D_Q(v)) + q(v, D_{Q'}(v)) - w(v) \\ &\geq p(v, D) \\ &= p(v, D) + d_{vv} - w(v) \\ &\geq q(v, D). \end{aligned}$$

Now we show that a contradiction arises if $w(\mathcal{T}_0) > q(v, D)$, thus showing that $w(\mathcal{T}_0) = q(v, D)$. By the algorithm, there exists a vertex u and a nonempty subset $A \subsetneq D$ of vertices such that

$$\begin{aligned} p(u, D) &= q(u, A) + q(u, D \setminus A) - w(u) \\ q(v, D) &= d_{vu} + p(u, D) - w(u). \end{aligned}$$

[2]For the definition of $D_Q(v)$, recall the notation $T_Y(x)$ introduced prior to Lemma 1.

Let $H = \mathcal{T}_1 \cup \mathcal{T}_2 \cup P_{vu}$ where $\mathcal{T}_1$ is a minimum NWST connecting u and A, $\mathcal{T}_2$ is a minimum NWST connecting u and $D \setminus A$, and P_{vu} is a shortest path from v to u. This results in the contradiction that

$$\begin{aligned} q(v,D) &= d_{vu} + p(u,D) - w(u) \\ &= d_{vu} + q(u,A) + q(u, D \setminus A) - w(u) - w(u) \\ &= w(H) \\ &\geq w(\mathcal{T}_0) \\ &> q(v,D). \end{aligned}$$

2. $|N_{\mathcal{T}_0}(v)| = 1$ and the branch of $\mathcal{T}_0$ touching v 'divides' before touching another terminal. Let the dividing vertex be u. Then no terminal is an interior vertex on the path P_{vu} from v to u in $\mathcal{T}_0$. Let $\mathcal{T}_1 = \mathcal{T}_0 \setminus P_{vu} \cup \{u\}$, then $|N_{\mathcal{T}_1}(u)| \geq 2$. By Lemma 1, $\mathcal{T}_1$ is a minimum NWST connecting u and $D \setminus \{v\}$, and P_{vu} is the shortest path from v to u, so $w(\mathcal{T}_1) = q(u, D \setminus \{v\}) = p(u, D \setminus \{v\})$ as it is in Case 1. Now,

$$\begin{aligned} w(\mathcal{T}_0) &= d_{vu} + w(\mathcal{T}_1) - w(u) \\ &= d_{vu} + q(u, D \setminus \{v\}) - w(u) \\ &= d_{vu} + p(u, D \setminus \{v\}) - w(u) \\ &\geq q(v, D \setminus \{v\}) \\ &= q(v,D). \end{aligned}$$

   Then, $w(\mathcal{T}_0) = q(v,D)$ holds by the same analysis as in Case 1.

3. $|N_{\mathcal{T}_0}(v)| = 1$ and the branch of $\mathcal{T}_0$ touching v does not divide before touching another terminal. Denote by u the first terminal reachable from v in $\mathcal{T}_0$. Again, let $\mathcal{T}_1 = \mathcal{T}_0 \setminus P_{vu} \cup \{u\}$. Then, by Lemma 1, $\mathcal{T}_1$ is a minimum NWST connecting u and $D \setminus \{u, v\}$. By the induction assumption, $w(\mathcal{T}_1) = q(u, D \setminus \{u,v\})$, so

$$\begin{aligned} w(\mathcal{T}_0) &= d_{vu} + w(\mathcal{T}_1) - w(u) \\ &= d_{vu} + q(u, D \setminus \{u,v\}) - w(u) \\ &= d_{vu} + (q(u,\{u\}) + q(u, D \setminus \{u,v\}) - w(u)) - w(u) \\ &\geq d_{vu} + p(u, D \setminus \{v\}) - w(u) \\ &\geq q(v, D \setminus \{v\}) \\ &= q(v,D). \end{aligned}$$

   Then, $w(\mathcal{T}_0) = q(v,D)$ holds by the same analysis as in Case 1.

So, $w(\mathcal{T}_0) = q(v,D)$ holds in all cases. This shows $q(v,D)$ is the weight of a minimum NWST connecting v and D and the statement is true for $|D| = i$. So, the statement is true in general, and thus the returned value $q(s, \tilde{T})$ is the weight of a minimum NWST connecting T.

We can compute the pairwise distances d_{uv} in time $O(n^3)$ by node-splitting and applying a standard all-pairs shortest paths algorithm. So, the algorithm's runtime is the same as that of Dreyfus and Wagner, i.e., $O(3^k n + 2^k n^2 + n^3)$. $\square$

2.2 Spanning tree enumeration and consequences

Another well-known Steiner tree algorithm is the *spanning tree enumeration* (STE) algorithm due to Hakimi [11], cf. [16]. In it, one enumerates every superset $T' \supseteq T$ of the terminal set T and computes a minimum spanning tree on the subgraph induced by T'. It then returns a cheapest such spanning tree. A straightforward generalization of this idea leads to the following lemma.

Lemma 2. *NWST with k terminals can be solved in time $O(2^{n-k} n^2)$.*

We can combine the STE algorithm with Algorithm 1 as follows. Letting k denote the number of terminals, the algorithm solves NWST as follows.

1. If $k \leq n \log_6 2$, run Algorithm 1;
2. Otherwise, run STE.

Corollary 1. *NWST can be solved in time $O(1.5296^n)$.*

Proof. The algorithm is clearly correct, and its runtime is bounded by
$O(3^{n \log_6 2} n^3) + O(2^{n - n \log_6 2} n^2) = O(2^{n \log_6 3} n^3 + 2^{n \log_6 3} n^2) = O(1.5296^n)$. $\square$

Finally, we show a negative result: Hakimi's algorithm for Steiner tree is essentially best-possible under SETH. This suggests that it will be difficult to improve the base of the exponential term of Lemma 2. The lower bound is based on the hardness of Hitting Set. Recall that Hitting Set is defined as follows.

Problem: Hitting Set.
Input: a family $\mathcal{F} \subseteq 2^U$ of subsets of U and an integer t.
Question: does there exist $X \subseteq U$ such that $|X| \leq t$ and X has nonempty intersection with each $F \in \mathcal{F}$?

Theorem 2 (Theorem 1.1 of Cygan et al. [3])**.** *For every constant $\epsilon > 0$,* Hitting Set *cannot be solved in time $(2-\epsilon)^{|U|}(|U| + |\mathcal{F}|)^{O(1)}$, unless SETH fails.*

We note that Cygan et al. [3] prove a hardness result concerning Steiner Tree, but it is conditional on the hardness of Set Cover (not on SETH), and they parameterize with respect to the solution size[3]. Our hardness result is shown with respect to the number $n-k$ of non-terminals and is conditional on SETH.

Proposition 1 (Hakimi's STE algorithm is hard to beat)**.** *For every constant $\epsilon > 0$,* Steiner Tree *cannot be solved in time $(2-\epsilon)^{n-k}n^{O(1)}$, unless SETH fails.*

Proof. We construct a reduction from Hitting Set and rely on Theorem 2. Let the instance of Hitting Set be defined by a family $\mathcal{F} \subseteq 2^U$ of subsets of U with target size t. We can assume that $t \leq |U|$, since otherwise it is a trivial "yes" instance.

We construct an instance of Steiner Tree (the decision version) on a graph $G = (V, E)$, where $V = U \cup \mathcal{F}$. Construct the edge set E so that $G[U]$ is complete, and for every pair of vertices $u \in U$ and $F \in \mathcal{F}$ such that $u \in F$, add the edge $\{u, F\}$. Let the set of terminals be $\mathcal{F}$. Note that $n = |V|$ and $k = |\mathcal{F}|$, so $n - k = |U|$. Give each edge that crosses from U to $\mathcal{F}$ weight $|U|$. The edges within $E(G[U])$ should be given weight one. We claim that there is a Steiner tree of weight (at most) $(t-1) + |U| \cdot |\mathcal{F}|$ if and only if the instance of Hitting Set is a "yes" instance.

Suppose there is a hitting set $X \subseteq U$ of size (at most) t, $t \leq |U|$. We construct a Steiner tree $G' = (V', E')$ as follows. Let $V' = \mathcal{F} \cup X$. For each $F \in \mathcal{F}$ pick one edge $\{u, F\}$ (where $u \in X$) and add it to E'. Further pick some $|X| - 1$ edges from $E(G[U])$ that induce a tree and add them to E'. The construction ensures that G' is connected, and the weight of E' is $|\mathcal{F}| \cdot |U| + (|X| - 1) \leq |\mathcal{F}| \cdot |U| + (t-1)$.

Now suppose there is a Steiner tree $G' = (V', E')$ such that E' has weight at most $(t-1) + |U| \cdot |\mathcal{F}|$. To connect a terminal vertex $F \in \mathcal{F}$ to the tree we must select at least one of its incident edges. They are all of weight $|U|$. If some terminal is incident to two edges of G', then the weight of E' will be at least $|\mathcal{F}| \cdot |U| + |U| \geq |\mathcal{F}| \cdot |U| + t$. Thus there are exactly $|\mathcal{F}|$ edges crossing from $\mathcal{F}$ to U and each $F \in \mathcal{F}$ is a leaf in G'. Since the vertices of $\mathcal{F}$ are leaves, they can be removed and G' remains connected, i.e., $X' = V' \cap U$ induces a tree in G'. And, by the weighting, this tree has at most $t - 1$ edges, so $|X'| \leq t$. So, by the construction, X' is a hitting set of size at most t.

Now suppose there is an $\epsilon > 0$ such that Steiner Tree can be solved in time $(2-\epsilon)^{n-k}n^{O(1)}$. Then, by the reduction described herein, Hitting Set can be solved in time $(2-\epsilon)^{|U|}(|U| + |\mathcal{F}|)^{O(1)}$, thus disproving SETH (by Theorem 2). $\square$

3 Maximum-Weight Connected Subgraph

In this section, we describe two algorithms for MWCS. The runtime of the first is parameterized by the number of negative-weight vertices, and the runtime of the

[3]Specifically, in their Theorem 1.2, Cygan et al. show the following. Suppose that, for all $\epsilon < 1$, there exists a k such that k-Set Cover cannot be solved in time $2^{\epsilon n}m^{O(1)}$. (This is the set cover problem for set systems over $[n]$ with m sets of size at most k.) Then, for all $\epsilon < 1$, t-Steiner Tree cannot be solved in time $2^{\epsilon t}n^{O(1)}$. (This is the problem of determining the existence of a Steiner tree with t vertices, including terminals.) The exact relationship between SETH and this hardness assumption for Set Cover is murky; Cygan et al. state that they would like to base their result on the hardness of SETH instead of on Set Cover, but were "unable to find a suitable reduction."

second is parameterized by the number of positive-weight vertices. These two algorithms imply a third algorithm that solves MWCS in time $O(4^{n/3}n^3) = O(1.5875^n)$. They also imply that MWCS can be solved in time $O(4^{\alpha(G)}n^3)$.

3.1 MWCS with few negative-weight vertices

We describe a simple algorithm for the maximum-weight connected subgraph problem. It is analogous to Hakimi's spanning tree enumeration algorithm, but it does not require one to compute a minimum spanning tree (since the edges have no weight).

Algorithm 2 An algorithm for MWCS with few negative-weight vertices

```
1: Let $\bar{S}$ be the set of negative-weight vertices;
2: $z \leftarrow 0$ (the weight of the empty vertex set);
3: for all $D \subseteq \bar{S}$ do
4:    Find the connected components of $H_D := G - D$;
5:    Let $H' = (V', E')$ be a component of $H_D$ of maximum $w(V')$.
6:    $z \leftarrow \max\{z, w(V')\}$;
7: end for
8: return $z$;
```

For the following theorem, denote by q the number of negative-weight vertices, n the total number of vertices, and m the number of edges.

Theorem 3. *MWCS can be solved in time $O(2^q(m+n))$. For every constant $\epsilon > 0$, MWCS cannot be solved in time $(2-\epsilon)^q n^{O(1)}$, unless SETH fails.*

Proof. Algorithm 2 proves the first claim. Its runtime and correctness are easy to see. For the lower bound, we construct a reduction from Hitting Set and rely on Theorem 2. Let the instance of Hitting Set be defined by a family $\mathcal{F} \subseteq 2^U$ of subsets of U with target size t. We construct an instance of MWCS on a graph $G = (V, E)$, where $V = U \cup \mathcal{F}$. Construct the edge set E so that $G[U]$ is complete, and for every pair of vertices $u \in U$ and $F \in \mathcal{F}$ such that $u \in F$, add the edge $\{u, F\}$. Give each vertex from U weight -1 and each vertex from $\mathcal{F}$ weight $t+1$.

First we claim that G has a connected subgraph of weight $W := |\mathcal{F}|(t+1) - t$ if and only if the instance of Hitting Set is a "yes" instance. If the instance of Hitting Set has a solution $X \subseteq U$ with $|X| \leq t$, then $\mathcal{F} \cup X$ induces a connected subgraph of weight at least $|\mathcal{F}|(t+1) - t = W$. For the other direction, suppose that G has a connected subgraph $G[S']$ of weight at least W. Recognize that all vertices from $\mathcal{F}$ must belong to S' since otherwise the weight of S' is at most

$$(|\mathcal{F}| - 1)(t+1) = |\mathcal{F}|t + |\mathcal{F}| - t - 1 < |\mathcal{F}|t + |\mathcal{F}| - t = W. \quad (4)$$

Then, by construction of the edge set of G, the set $X' = S' \setminus \mathcal{F}$ provides a solution to the instance of Hitting Set. The inequality $|X'| \leq t$ holds by a weight argument.

Now suppose there is an $\epsilon > 0$ such that MWCS can be solved in time $(2-\epsilon)^q n^{O(1)}$. Then, by the reduction described herein, Hitting Set can be solved in time $(2-\epsilon)^{|U|}(|U|+|\mathcal{F}|)^{O(1)}$, thus disproving SETH (by Theorem 2). $\square$

3.2 MWCS with few positive-weight vertices

We now describe an algorithm for instances of MWCS that have few positive-weight vertices. This leads to an efficient algorithm for solving instances with small independence number. The most important insights in this section are described below.

1. **Preprocessing**. For a pair of adjacent, nonnegative-weight vertices, the edge between them can be contracted without altering the MWCS optimal objective.

2. **Use of NWST subroutine**. If one first decides which positive-weight vertices to select, then the problem of optimally connecting them with nonpositive-weight vertices is an instance of NWST.

Algorithm 3 An algorithm for MWCS with few positive-weight vertices

```
1: Let $\hat{S}$ be the set of positive-weight vertices in $G$;
2: $z \leftarrow \max\{w(u) \mid u \in \hat{S}\}$ or $z \leftarrow 0$ if $|\hat{S}| = 0$;
3: for $i = 2, 3, \ldots, |\hat{S}|$ do
4:     for all $D \subseteq \hat{S}$ with $|D| = i$ do
5:         Construct instance of NWST in $G[(V \setminus \hat{S}) \cup D]$ with terminal set $D$ and
           weight function $w''$, where the edges have zero weight, $w''(u) = 0$ for vertices
           $u \in D$, and $w''(v) = -w(v)$ for vertices $v \in V \setminus \hat{S}$;
6:         Solve NWST instance yielding optimal NWST cost $z_D$;
7:         $z \leftarrow \max\{z, w(D) - z_D\}$;
8:     end for
9: end for
10: return $z$;
```

For the following theorem, let p denote the number of positive-weight vertices, and let $\tau(n,k)$ denote the time to solve an n-vertex instance of NWST with k terminals.

Theorem 4. *Algorithm 3 correctly solves MWCS in time* $O\left(n + \sum_{k=2}^{p} \binom{p}{k} \tau(n,k)\right)$.

Proof. The proofs of correctness and runtime are straightforward. $\square$

Corollary 2. *MWCS can be solved in time $O(4^p n^3)$, where p denotes the number of positive-weight vertices.*

Proof. This follows directly from Theorem 4 using Algorithm 1 as the NWST subroutine, since $\sum_{k=2}^{p} \binom{p}{k} O(3^k n^3) = O(4^p n^3)$. $\square$

Corollary 3. *Instances of MWCS with independence number $\alpha(G)$ can be solved in time $O(4^{\alpha(G)}n^3)$.*

Proof. Given an instance of MWCS, perform a preprocessing procedure that iteratively contracts edges between positive-weight vertices. We show that the number of remaining positive-weight vertices is at most $\alpha(G)$. Actually, this entire procedure can be done in time $O(n^2)$ as follows.

Formally, denote by $\hat{S}$ the set of positive-weight vertices, and let the components of $G[\hat{S}]$ be $G[\hat{S}_i],\ i = 1, \dots, r$. Construct the smaller graph $G' = (V', E')$ as below, where vertex $c_i \in C := \{c_1, \dots, c_r\}$ represents component $G[\hat{S}_i]$ and

$$\begin{aligned} V' &= (V \setminus \hat{S}) \cup C \\ E' &= \Big\{\{u, v\} \in E \mid u, v \in V \setminus \hat{S}\Big\} \\ &\qquad \cup \{\{v, c_i\} \mid \exists s \in \hat{S}_i \text{ and } v \in V \setminus \hat{S} \text{ such that } \{v, s\} \in E\}. \end{aligned}$$

Then, create a new weight function $w' : V' \to \mathbb{R}$, where:

- for each $v \in V \setminus \hat{S}$, set $w'(v) := w(v)$, and
- for each $i = 1, \dots, r$, set $w'(c_i) := \sum_{s \in \hat{S}_i} w(s)$.

Now we show that the instance of MWCS defined by graph G' and weight function w' has at most $\alpha(G)$ positive-weight vertices. Indeed, for each component $G[\hat{S}_i]$ of $G[\hat{S}]$, choose a vertex $s_i \in \hat{S}_i$. And, $\{s_1, \dots, s_r\}$ is independent in G, so $r \leq \alpha(G)$.

Since the number of vertices v in G' that have positive weight $w'(v)$ is r, this new instance of MWCS on graph G' with weights w' can be solved in time $O(4^r|V'|^3)$ by Corollary 2. Since $r \leq \alpha(G)$ and $|V'| \leq |V|$, the original instance of MWCS can be solved in time $O(|V|^2 + 4^r|V'|^3) = O(4^{\alpha(G)}|V|^3)$. □

3.3 Combining the approaches

Now we can combine Algorithms 2 and 3 to beat the time $O(2^n(m+n))$ exhaustive-search algorithm. Letting p denote the number of positive-weight vertices, the algorithm solves MWCS as follows.

1. If $p \leq n/3$, run Algorithm 3;
2. Otherwise, run Algorithm 2.

Corollary 4. *MWCS can be solved in time $O(1.5875^n)$.*

Proof. The algorithm is clearly correct, and its runtime is bounded by
$O(4^{n/3}n^3) + O(2^{2n/3}n^2) = O(4^{n/3}n^3) = O(1.5875^n)$. □

4 Conclusion

We provide fixed-parameter tractable algorithms for the node-weighted Steiner tree (NWST) problem and the maximum-weight connected subgraph (MWCS) problem. The NWST algorithm is parameterized by the number of terminals, and (by a generalization of Hakimi's spanning tree enumeration algorithm [11]) implies a time $O(1.5296^n)$ NWST algorithm. We also show that Hakimi's algorithm is essentially best-possible under the strong exponential time hypothesis (SETH). The MWCS algorithms are parameterized by the number of positive-weight (or negative-weight) vertices and imply that MWCS can be solved in time $O(1.5875^n)$. Again, under SETH, the MWCS algorithm parameterized by the number of negative-weight vertices is essentially best-possible.

On the other hand, we strongly suspect that the other algorithms can be improved. As noted previously, the Steiner tree algorithm due to Dreyfus and Wagner [6] runs in time $O^*(3^k)$ for instances with k terminals (cf. an independent development by Levin [17]). A more recent paper [8] solves the Steiner tree problem in time $O^*((2+\delta)^k)$ where δ is any positive constant. (The $O^*(\cdot)$ notation suppresses polynomial factors in the input size.) We suspect that their algorithm can be generalized to solve the NWST problem in roughly the same time bound. However, the analysis is more involved, and the authors state that the constants in their algorithm's runtime "become very large even for moderate δ" [19]. For these reasons, we do not improve the dependence on k for NWST in this paper. This is a subject for future work which can improve our runtimes for solving both NWST and MWCS.

Another interesting question for future work is to determine whether the MWCS problem where both vertices and edges are weighted can be solved in time $O(\delta^n)$ for some constant $\delta < 2$. Note that this problem can be solved in time $O^*(2^n)$ by the following algorithm which finds, for each vertex subset $S \subseteq V$, a maximum-weight connected subgraph G_S that has S as its vertex set. It proceeds by first finding a maximum-weight spanning tree $\mathcal{T}_S$ of $G[S]$. Then, let $G_S := (S, E(\mathcal{T}_S) \cup E^+(S)\}$, where $E^+(S)$ is the set of positive-weight edges of G that have both endpoints in S. To get an optimal solution to this vertex-and-edge-weighted MWCS problem, pick a vertex subset $S \subseteq V$ whose associated graph G_S has the largest weight.

Acknowledgements

This material is based upon work supported by the AFRL Mathematical Modeling and Optimization Institute and by the National Science Foundation under Grant Nos. 1662757 and 1538493.

References

[1] 11th DIMACS Implementation Challenge in Collaboration with ICERM: Steiner tree problems. `http://dimacs11.zib.de/home.html`. Accessed: 2017-04-20.

[2] E. Álvarez-Miranda, I. Ljubić, and P. Mutzel. The maximum weight connected subgraph problem. In M. Jünger and G. Reinelt, editors, *Facets of Combinatorial Optimization*, pages 245–270. Springer, Berlin, 2013.

[3] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlstrom. On problems as hard as CNF-SAT. In *Computational Complexity (CCC), 2012 IEEE 27th Annual Conference on*, pages 74–84. IEEE, 2012.

[4] M.T. Dittrich, G.W. Klau, A. Rosenwald, T. Dandekar, and T. Müller. Identifying functional modules in protein–protein interaction networks: an integrated exact approach. *Bioinformatics*, 24(13):i223–i231, 2008.

[5] R.G. Downey and M.R. Fellows. *Parameterized complexity.* Springer, 1999.

[6] S.E. Dreyfus and R.A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[7] M. Fischetti, M. Leitner, I. Ljubic, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl. Thinning out Steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, 9(2):203–229, 2017.

[8] B. Fuchs, W. Kern, D. Molle, S. Richter, P. Rossmanith, and X. Wang. Dynamic programming for minimum Steiner trees. *Theory of Computing Systems*, 41(3):493–500, 2007.

[9] M.R. Garey and D.S. Johnson. The rectilinear Steiner tree problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834, 1977.

[10] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[11] S.L. Hakimi. Steiner's problem in graphs and its implications. *Networks*, 1(2):113–133, 1971.

[12] F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner tree problem.* Elsevier, 1992.

[13] R. Impagliazzo and R. Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62:367–375, 2001.

[14] D.S. Johnson. The NP-completeness column: an ongoing guide. *Journal of Algorithms*, 6(1):145–159, 1985.

[15] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations.* Plenum Press, 1972.

[16] E.L. Lawler. *Combinatorial optimization: networks and matroids.* Holt, Rinehart and Winston, 1976.

[17] A.Y. Levin. Algorithm for the shortest connection of a group of graph vertices. *Soviet Mathematics Doklady*, 12(5):1477–1481, 1971.

[18] D. Lokshtanov, D. Marx, S. Saurabh, et al. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, (105):41–72, 2011.

[19] D. Mölle, S. Richter, and P. Rossmanith. A faster algorithm for the Steiner tree problem. In B. Durand and W. Thomas, editors, *STACS 2006*, pages 561–570. Springer, 2006.

[20] M. Pătraşcu and R. Williams. On the possibility of faster SAT algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1065–1075. SIAM, 2010.

[21] A. Segev. The node-weighted Steiner tree problem. *Networks*, 17(1):1–17, 1987.

[22] A. Vergis. Manuscript, 1983.

[23] Y. Wang, A. Buchanan, and S. Butenko. On imposing connectivity constraints in integer programs. *Mathematical Programming*, 166(1):241–271, 2017.

[24] R. Williams. Algorithms for circuits and circuits for algorithms. In *Computational Complexity (CCC), 2014 IEEE 29th Conference on*, pages 248–261. IEEE, 2014.

[25] P. Winter. Steiner problem in networks: a survey. *Networks*, 17(2):129–167, 1987.