Algorithmica (2025) 87:148–165
https://doi.org/10.1007/s00453-024-01278-5

Better Hardness Results for the Minimum Spanning Tree Congestion Problem

Huong Luu[1] · **Marek Chrobak**[1]

Received: 31 May 2023 / Accepted: 8 October 2024 / Published online: 26 October 2024

Abstract
In the spanning tree congestion problem, given a connected graph G, the objective is to compute a spanning tree T in G that minimizes its maximum edge congestion, where the congestion of an edge e of T is the number of edges in G for which the unique path in T between their endpoints traverses e. The problem is known to be $\mathbb{NP}$-hard, but its approximability is still poorly understood, and it is not even known whether the optimum solution can be efficiently approximated with ratio $o(n)$. In the decision version of this problem, denoted $\boldsymbol{K}-\mathsf{STC}$, we need to determine if G has a spanning tree with congestion at most K. It is known that $\boldsymbol{K}-\mathsf{STC}$ is $\mathbb{NP}$-complete for $K \geq 8$, and this implies a lower bound of 1.125 on the approximation ratio of minimizing congestion. On the other hand, $\mathbf{3}-\mathsf{STC}$ can be solved in polynomial time, with the complexity status of this problem for $K \in \{4, 5, 6, 7\}$ remaining an open problem. We substantially improve the earlier hardness results by proving that $\boldsymbol{K}-\mathsf{STC}$ is $\mathbb{NP}$-complete for $K \geq 5$. This leaves only the case $K = 4$ open, and improves the lower bound on the approximation ratio to 1.2. Motivated by evidence that minimizing congestion is hard even for graphs of small constant radius, we also consider $\boldsymbol{K}-\mathsf{STC}$ restricted to graphs of radius 2, and we prove that this variant is $\mathbb{NP}$-complete for all $K \geq 6$.

Keywords Combinatorial optimization · Spanning trees · Congestion

1 Introduction

Problems involving constructing a spanning tree that satisfies certain requirements are among the most fundamental tasks in graph theory and algorithmics. One such problem is the *spanning tree congestion problem*, STC for short, that has been studied extensively for many years. In this problem we seek a spanning tree T of a given graph G that roughly approximates the connectivity structure of G, in the following sense: Embed G into T by replacing each edge (u, v) of G by the unique u-to-v path in T.

✉ Huong Luu
hluu008@ucr.edu

[1] University of California at Riverside, Riverside, USA

Define the *congestion of an edge e of T* as the number of such paths that traverse e. The objective of STC is to find a spanning tree T in which the maximum edge congestion is minimized.

The general concept of edge congestion was first introduced in 1986, under the name of *load factor*, as a measure of quality of an embedding of one graph into another [1] (see also the survey in [2]). The problem of computing trees with low congestion was studied by Khuller et al. [3] in the context of solving commodities network routing problems. The trees considered there were not required to be spanning subtrees, but the variant involving spanning trees was also mentioned. In 2003, Ostrovskii provided independently a formal definition of STC and established some fundamental properties of spanning trees with low congestion [4]. Since then, many combinatorial and algorithmic results about this problem have been reported in the literature — we refer the readers to the survey paper by Otachi [5] for more information, most of which is still up-to-date.

As established by Löwenstein [6], STC is $\mathbb{NP}$-hard. As usual, this is proved by showing $\mathbb{NP}$-completeness of its decision version, where we are given a graph G and an integer K, and we need to determine if G has a spanning tree with congestion at most K. Otachi et al. [7] strengthened this by proving that the problem remains $\mathbb{NP}$-hard even for planar graphs. In [8], STC is proven to be $\mathbb{NP}$-hard for chain graphs and split graphs. On the other hand, computing optimal solutions for STC can be achieved in polynomial time for some special classes of graphs: complete k-partite graphs, two-dimensional tori [9], outerplanar graphs [10], and two-dimensional Hamming graphs [11].

In our paper, we focus on the decision version of STC where the bound K on congestion is a fixed constant. We denote this variant by $\boldsymbol{K}-$ STC. Several results on the complexity of $\boldsymbol{K}-$ STC were reported in [7]. For example, the authors of [7] show that $\boldsymbol{K}-$ STC is decidable in linear time for planar graphs, graphs of bounded treewidth, graphs of bounded degree, and for all graphs when $K = 1, 2, 3$. On the other hand, they show that the problem is $\mathbb{NP}$-complete for any fixed $K \geq 10$. In [12], Bodlaender et al. proved that $\boldsymbol{K}-$ STC is linear-time solvable for graphs in apex-minor-free families and chordal graphs. They also show an improved hardness result of $\boldsymbol{K}-$ STC, namely that it is $\mathbb{NP}$-complete for $K \geq 8$, even in the special case of apex graphs that only have one unbounded degree vertex. As stated in [5], the complexity status of $\boldsymbol{K}-$ STC for $K \in \{4, 5, 6, 7\}$ remains an open problem.

Very little is known about the approximability of STC. The trivial upper bound for the approximation ratio is $n/2$ — this ratio is achieved in fact by *any* spanning tree [5]. As a direct consequence of the $\mathbb{NP}$-completeness of $\mathbf{8}-$ STC, there is no polynomial-time algorithm to approximate the optimum spanning tree congestion with a ratio better than 1.125 (unless $\mathbb{P} = \mathbb{NP}$).

Our contributions In this paper, addressing an open question in [5], we provide an improved hardness result for $\boldsymbol{K}-$ STC:

Theorem 1 *For any fixed integer $K \geq 5$, $\boldsymbol{K}-$ STC is $\mathbb{NP}$-complete.*

The proof of this theorem is given in Sect. 3. Combined with the results in [7], Theorem 1 leaves only the status of $\mathbf{4}-$ STC open. Furthermore, it also immediately improves the lower bound on the approximation ratio for STC:

Corollary 1 *For $c < 1.2$ there is no polynomial-time c-approximation algorithm for* STC*, unless* $\mathbb{P} = \mathbb{NP}$.

We remark that this hardness result remains valid even if an additive constant is allowed in the approximation bound. This follows by an argument in [12]. (In essence, the reason is that assigning a positive integer weight β to each edge increases its congestion by a factor β.)

A common feature of the hardness proofs for STC, including ours, is that they all use graphs of small constant radius (or, equivalently, diameter). Another property of STC that makes its approximation challenging is that the minimum congestion value is not monotone with respect to adding edges. The example graph in [4] showing this non-monotonicity is also of small radius (in fact, only 2). These observations indicate that a key to further progress may be in better understanding of STC in small-radius graphs.

This motivates our additional hardness result presented in Sect. 4, where we focus on graphs of radius 2. (For radius 1 the problem is trivial.) We prove there that $\boldsymbol{K}$ − STC remains $\mathbb{NP}$-complete for this class of graphs, for any fixed integer $K \geq 6$. In fact, this holds even if we further restrict such graphs to be bipartite and have only one vertex of non-constant degree.

Other related work The spanning tree congestion problem is closely related to the tree spanner problem, in which the objective is to find a spanning tree T of G that minimizes the stretch factor, defined as the maximum ratio, over all vertex pairs, between the length of the path in T and the length of the shortest path in G connecting these vertices. In fact, for any planar graph, its spanning tree congestion is equal to its dual's minimum stretch factor plus one [7, 13]. This direction of research has been extensively explored, see [14–16]. As an aside, we remark that the complexity of the tree 3-spanner problem has been open since its first introduction in 1995 [14].

STC is also intimately related to problems involving cycle bases in graphs. As each spanning tree induces a fundamental cycle basis of the given graph, a spanning tree with low congestion yields a cycle basis for which the edge-cycle incidence matrix is sparse. Sparsity of such matrices is desirable in linear-algebraic approaches to solving some graph optimization problems, for example analyses of distribution networks such as pipe flow systems [17].

STC can be thought of as an extreme case of the graph sparsification problem, where, given a graph G, the objective is to compute a sparse graph H that captures connectivity properties of G. See [18–20] (and the references therein) for some approaches to graph sparsification. Sparsification makes working with large graphs more efficient. The compressed graph H can be used instead of G to reduce computational costs, particularly when analyzing large-scale networks.

STC arises naturally in some applications in networking, specifically in network design and routing problems (see [3, 21], for example). In this context, the weight of an edge (u, v) represents, say, expected traffic volume between u and v, and the goal is to design a spanning tree that can handle this traffic efficiently, that is without overloading any of its links.

The extended version of this paper [22] considers also the problem of computing minimum-congestion spanning trees of depth 2. (Note that this is a different optimiza-

tion problem, whose optimal solution could be larger than for STC.) The results in [22] include the proof of $\mathbb{NP}$-hardness and polynomial-time algorithms for some special cases.

2 Preliminaries

Basic graph terminology Let G be a simple graph with vertex set V and edge set E. We use notation $N_G(v)$ for the neighborhood of a vertex $v \in V$ and $\deg_G(v)$ for its degree. For a vertex $v \in V$, its *eccentricity* $\mathrm{ecc}_G(v)$ is defined as the maximum distance from v to any other vertex. The *radius* of G is $\mathrm{rad}(G) = \min_{v \in V} \mathrm{ecc}_G(v)$.

Consider a spanning tree $T \subseteq E$ of G. If $e = (u, v) \in T$, removing e from T splits T into two subtrees. We denote by $T_{u,v}$ the subtree that contains u and by $T_{v,u}$ the subtree that contains v. Let the *cut-set* of e, denoted $\partial_{G,T}(e)$, be the set of edges in E that have one endpoint in $T_{u,v}$ and the other in $T_{v,u}$. In other words, $\partial_{G,T}(e)$ consists of the edges $(u', v') \in E$ for which the unique (simple) path in T from u' to v' goes through e. Note that $e \in \partial_{G,T}(e)$. The *congestion of* e, denoted by $\mathrm{cng}_{G,T}(e)$, is the cardinality of $\partial_{G,T}(e)$. The *congestion of tree* T is $\mathrm{cng}_G(T) = \max_{e \in T} \mathrm{cng}_{G,T}(e)$. Finally, the *spanning tree congestion of graph* G, denoted by $\mathrm{stc}(G)$, is defined as the minimum value of $\mathrm{cng}_G(T)$ over all spanning trees T of G.

Weighted edges The concept of the spanning tree congestion extends naturally to edge-weighted graphs. An edge $e = (u, v)$ with integer weight $\omega \geq 1$ contributes ω to the congestion of any edge f for which $e \in \partial_{G,T}(f)$. One can think of e as representing ω parallel edges between u and v. We refer to these parallel edges as a *non-weighted realization* of a weighed edge e. Indeed, replacing e by this realization does not affect the minimum congestion value, because in a multigraph only one edge between any two given vertices can be in a spanning tree, but all of them belong to the cut-set $\partial_{G,T}(f)$ of any edge $f \in T$ whose removal separates these vertices in T (and thus all contribute to $\mathrm{cng}_{G,T}(f)$).

We can also realize a weighted graph using a simple graph (without multiple edges).

As observed in [7, Lemma 7.2], edge subdivision does not affect the spanning tree congestion of a graph, so instead of using parallel edges we can realize an edge of weight ω using ω parallel disjoint paths.[1]

(See Fig. 1 for illustration.) We state our results in terms of simple graphs, but we use weighted graphs in our proofs with the understanding that they actually represent simple graphs. As all weights used in the paper are constant, the computational complexity of $\boldsymbol{K}$ − STC is not affected. The proof in Sect. 3 does not depend on what realization of weighted edges we use, while the proof in Sect. 4 uses a specific realization that we refer to as *spintop*: an edge (u, v) of weight ω is realized using $\omega - 1$ length-three u-to-v paths in addition to a non-weighted edge (u, v) itself (see Fig. 1b).

Double weights In fact, it is convenient to generalize this further by introducing edges with *double weights*. A double weight of an edge e is denoted $\omega : \omega'$, where ω

[1] The proof of Lemma 7.2 is omitted in [7], but the lemma is quite intuitive and can be shown by a simple modification of the optimal tree: if the original edge is in the spanning tree, the modified tree traverses all segments of the subdivided edge. If it isn't, the modified tree visits all but one segment, chosen arbitrarily. In both cases, the maximum edge congestion is not affected.

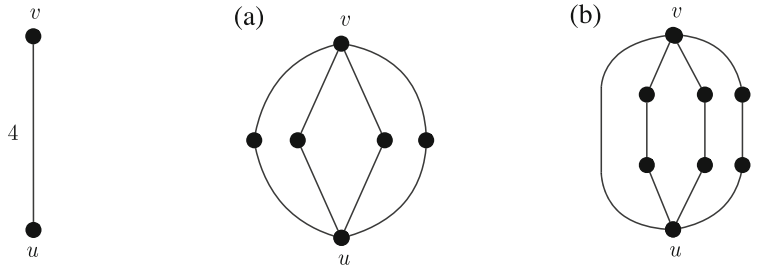

Fig. 1 Two different realizations of an edge (u, v) of multiplicity 4. **a** A basic realization using paths of length 2. **b** The spintop realization used in Sect. 4

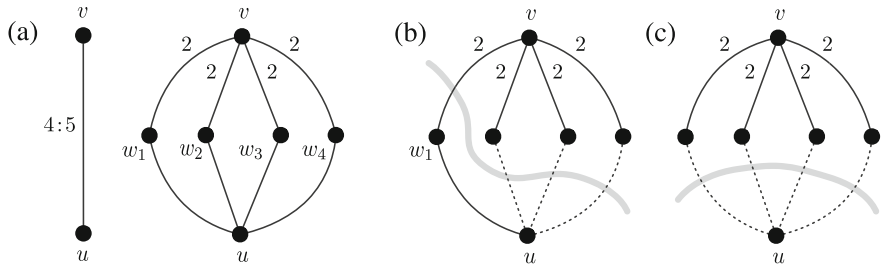

Fig. 2 **a** On the left, an edge (u, v) with double weight $4:5$ in G. On the right, the realization of (u, v) in G'. If one applies the spintop realization of the edges from v to w_i's, as in Fig. 1b, then the subgraph on the right realizing (u, v) is bipartite and all its nodes are within distance 2 from v. Figures (b) and (c) illustrate the proof of Lemma 1: **b** the traversal of T' and the cut of (u, v) when $(u, v) \in T$, **c** the traversal of T' and the cut containing (u, v) when $(u, v) \notin T$. Solid lines are tree edges and dotted lines are non-tree edges

and ω' are positive integers such that $\omega \leq \omega' \leq K - 1$, and its interpretation in the context of $\boldsymbol{K} - \mathsf{STC}$ is as follows: Given a spanning tree T,

- if $e \in E \setminus T$ then e contributes ω to the congestion $\mathrm{cng}_{G,T}(f)$ of any edge $f \in T$ for which $e \in \partial_{G,T}(f)$, and
- if $e \in T$ then e contributes ω' to its own congestion, $\mathrm{cng}_{G,T}(e)$.

The lemma below provides a simple-graph realization of double-weighted edges. It implies that including such edges does not affect the computational complexity of $\boldsymbol{K} - \mathsf{STC}$, allowing us to formulate our proofs in terms of graphs where some edges have double weights.

Lemma 1 *Let (u, v) be an edge in G with double weight $\omega : \omega'$, where $\omega \leq \omega' \leq K - 1$. Consider another graph G' obtained from G by removing (u, v), and for each $i = 1, 2, \cdots, \omega$ adding a new vertex w_i with two edges: edge (u, w_i) of weight 1 and edge (w_i, v) of weight $\omega' - \omega + 1$ (see Fig. 2a for an example). Then, $stc(G) \leq K$ if and only if $stc(G') \leq K$.*

Proof Denote by $W = \{w_1, w_2, ..., w_\omega\}$ the set of new vertices, and by $W_u = \{(u, w_i) \mid w_i \in W\}$ and $W_v = \{(w_i, v) \mid w_i \in W\}$ the sets of new edges added to G'.

($\Rightarrow$) Suppose that G has a spanning tree T with $\text{cng}_G(T) \leq K$. We will show that there exists a spanning tree T' of G' with $\text{cng}_{G'}(T') \leq K$. We break the proof into two cases, in both cases showing that $\text{cng}_{G',T'}(e) \leq K$ for each edge $e \in T'$.
Case 1: $(u, v) \in T$.

Consider the spanning tree $T' = T \setminus \{(u, v)\} \cup W_v \cup \{(u, w_1)\}$ of G' (see Fig. 2b). For every edge $(x, y) \in E \setminus \{(u, v)\}$, the x-to-y paths in T and T' are the same, except that if the x-to-y path in T traverses edge (u, v) then the x-to-y path in T' traverses $(u, w_1), (w_1, v)$ instead. Therefore,

- If $e \in T' \setminus (W_v \cup \{(w_1, u)\})$, then $\partial_{G',T'}(e) = \partial_{G,T}(e)$. So $\text{cng}_{G',T'}(e) = \text{cng}_{G,T}(e) \leq K$.
- If $e = (u, w_1)$, then $\partial_{G',T'}(e) = \partial_{G,T}(u, v) \setminus \{(u, v)\} \cup W_u$. By the definition of double weights, (u, v) contributes ω' to $\text{cng}_{G,T}(u, v)$ while each edge in W_u contributes 1 to $\text{cng}_{G',T'}(e)$. Hence, $\text{cng}_{G',T'}(e) = \text{cng}_{G,T}(u, v) - \omega' + \omega \leq \text{cng}_{G,T}(u, v) \leq K$.
- If $e = (w_1, v)$, then $\partial_{G',T'}(e) = \partial_{G,T}(u, v) \setminus \{(u, v)\} \cup \{e\} \cup (W_u \setminus \{(w_1, u)\})$. Since e contributes $\omega' - \omega + 1$ to its own congestion, we have: $\text{cng}_{G',T'}(e) = \text{cng}_{G,T}(u, v) - \omega' + (\omega' - \omega + 1) + (\omega - 1) = \text{cng}_{G,T}(u, v) \leq K$.
- Lastly, if $e \in W_v \setminus \{(w_1, v)\}$, then it is a leaf edge, and we have $\text{cng}_{G',T'}(e) = \omega' - \omega + 2 \leq \omega' + 1 \leq K$.

Case 2: $(u, v) \notin T$.

Let $T' = T \cup W_v$, which is a spanning tree of G' (see Fig. 2c). We consider the following sub-cases:

- If $e \in T' \setminus W_v$ and e is not on the u-to-v path in T', then $\partial_{G',T'}(e) = \partial_{G,T}(e)$. So $\text{cng}_{G',T'}(e) = \text{cng}_{G,T}(e) \leq K$.
- If $e \in T' \setminus W_v$ and e is on the u-to-v path in T', then $\partial_{G',T'}(e) = \partial_{G,T}(e) \setminus \{(u, v)\} \cup W_u$. Since (u, v) contributes ω to $\text{cng}_{G,T}(e)$ and W_u also contributes ω to $\text{cng}_{G',T'}(e)$, we have $\partial_{G',T'}(e) = \partial_{G,T}(e) \leq K$.
- If $e \in W_v$, then e is a leaf edge and we have $\text{cng}_{G',T'}(e) = \omega' - \omega + 2 \leq \omega' + 1 \leq K$.

We have shown that $\text{cng}_{G'}(T') \leq K$ in all cases, which completes the proof for the forward implication. We now proceed to the proof of the converse implication.
($\Leftarrow$) Let T' be the spanning tree of G' with congestion $\text{cng}_{G'}(T') \leq K$. We will show that there exists a spanning tree T of G with $\text{cng}_G(T) \leq K$. Note that, for any $w_i \in W$, T' traverses at least one of the two edges (u, w_i) and (w_i, v). Furthermore, at most one vertex in W is a non-leaf. We consider three cases. In the first two cases the arguments follow the same pattern as in the proof for the ($\Rightarrow$) implication, in essence reversing the modification of the spanning tree. Then the third case reduces to the second case.
Case 1: Exactly one vertex in W is a non-leaf in T'.

Without loss of generality, we can assume w_1 is a non-leaf vertex (that is, both (u, w_1) and (w_1, v) are in T) and $W \setminus \{w_1\}$ are leaves. We construct T by adding (u, v) to T' and removing all vertices of W and their incident edges from T. By the construction, T is a spanning tree of G. We have:

- If $e \in T \setminus \{(u, v)\}$, then $\text{cng}_{G,T}(e) = \text{cng}_{G',T'}(e) \leq K$.
- If $e = (u, v)$, then $\text{cng}_{G,T}(e) = \text{cng}_{G',T'}(v, w_1) \leq K$.

Case 2: All vertices in W are leaves and T' traverse all edges in W_v.

Let $T = T' \setminus W_v$, which is a spanning tree of G. Then

- If $e \in T$ and e is not on the u-to-v path in T, then $\text{cng}_{G,T}(e) = \text{cng}_{G',T'}(e) \leq K$.
- If $e \in T$ and e is on the u-to-v path in T, then (u, v) and W_u contribute the same amount ω to the congestion of e in T and T', respectively, implying that $\text{cng}_{G,T}(e) = \text{cng}_{G',T'}(e) \leq K$.

Case 3: All vertices in W are leaves and T' traverses at least one edge in W_u.

In this case, we consider another spanning tree T'' of G' that traverses all edges in W_v and does not use any edge in W_u. It is sufficient to show that $\text{cng}_{G'}(T'') \leq \text{cng}_{G'}(T')$, since it implies that $\text{cng}_{G'}(T'') \leq K$, and then we can apply Case 2 to T''. We examine the congestion values of each edge $e \in T''$:

- If $e \in T'' \setminus W_v$ and e is not on the u-to-v path in T'', then $e \in T'$ and $\partial_{G',T''}(e) = \partial_{G',T'}(e)$, implying $\text{cng}_{G',T''}(e) = \text{cng}_{G',T'}(e)$.
- If $e \in T'' \setminus W_v$ and e is on the u-to-v path in T'', then for each vertex $w_i \in W$ either (u, w_i) contributes 1 or (w_i, v) contributes $\omega' - \omega + 1 \geq 1$ to $\text{cng}_{G',T'}(e)$. On the other hand, in T'', all edges in W_u are in $\partial_{G',T''}(e)$ and contribute a total of ω to $\text{cng}_{G',T''}(e)$. Thus, $\text{cng}_{G',T''}(e) \leq \text{cng}_{G',T'}(e)$.
- If $e \in W_v$, then $\text{cng}_{G',T''}(e) = \omega' - \omega + 2 \leq \omega' + 1 \leq K$.

In all cases, we have proved that there is a spanning tree T of G that has congestion at most K establishing the validity of the backward implication.

As explained earlier, in Sect. 4 we will use the spintop realization for weighted edges. Specifically, the realization of an edge $e = (u, v)$ with double weight $\omega : \omega'$ uses the spintop realization for the edges of weight $\omega' - \omega + 1$ between v and the w_i's. The crucial property of this realization, as discussed Sect. 4, is that it is bipartite and all its nodes are within distance 2 from v.

Remark Some readers may have noticed that there is a simpler way to realize an edge (u, v) with a double weight $\omega : \omega'$: replace it by a length-2 path (u, w), (w, v), where w is a new vertex, edge (u, w) has weight ω, and edge (w, v) has weight ω'. This indeed works, but can be used only when $\omega + \omega' \leq K$. This is because, in this construction, if w is a leaf of a spanning tree, the congestion of the tree edge from w will be $\omega + \omega'$, and this congestion value cannot exceed K. This realization of double-weighted edges would suffice for our proof in Sect. 3, but not the one in Sect. 4. (It may also be useful for establishing other hardness results for STC.)

3 $\mathbb{NP}$-Completeness Proof of K − STC for $K \geq 5$

In this section we prove our main result, the $\mathbb{NP}$-completeness of K − STC. Our proof uses an $\mathbb{NP}$-complete variant of the satisfiability problem called (2P1N)-SAT [23, 24]. An instance of (2P1N)-SAT is a boolean expression ϕ in conjunctive normal form, where each variable occurs exactly three times, twice positively and once negatively, and each clause contains exactly two or three literals of different variables. The objective is to decide if ϕ is satisfiable, that is if there is a satisfying assignment that makes ϕ true.

For each constant K, $\boldsymbol{K}$ − STC is clearly in $\mathbb{NP}$. We will present a polynomial-time reduction from (2P1N)-SAT. In this reduction, given an instance ϕ of (2P1N)-SAT, we construct a graph G with the following property:

(∗) ϕ has a satisfying truth assignment if and only if $\mathrm{stc}(G) \le K$.

Throughout the proof, the three literals of x_i in ϕ will be denoted by x_i, x_i', and $\bar{x}_i$, where x_i, x_i' are the two positive occurrences of x_i and $\bar{x}_i$ is the negative occurrence of x_i. We will also use notation $\tilde{x}_i$ to refer to an unspecified literal of x_i, that is, $\tilde{x}_i \in \{x_i, x_i', \bar{x}_i\}$.

We now describe the reduction. Set $k_i = K - i$ for $i = 1, 2, 3, 4$. (In particular, for $K = 5$, we have $k_1 = 4$, $k_2 = 3$, $k_3 = 2$, $k_4 = 1$). G will consist of gadgets corresponding to variables, with the gadget corresponding to x_i having three vertices x_i, x_i', and $\bar{x}_i$, that represent its three occurrences in the clauses. G will also have vertices representing clauses and edges connecting literals with the clauses where they occur (see Fig. 3b for an example). As explained in Sect. 2, without any loss of generality we can allow edges in G to have constant-valued weights, single or double. Specifically, starting with G empty, the construction of G proceeds as follows:

- Add a *root vertex* r.
- For each variable x_i, construct the *x_i-gadget* (see Fig. 3a). This gadget has three vertices corresponding to the literals: a *negative literal vertex* $\bar{x}_i$ and two *positive literal vertices* x_i, x_i', and two auxiliary vertices y_i and z_i. Its edges and their weights are given in the table below:

edge	$(\bar{x}_i, z_i)$	(z_i, x_i)	(x_i, x_i')	(r, x_i')	(r, y_i)	(y_i, z_i)	$(y_i, \bar{x}_i)$
weight	$1:k_3$	$1:k_3$	$1:k_2$	k_3	k_4	k_4	$1:k_2$

  the corresponding *clause-to-literal edge* $(c, \tilde{x}_i)$ of weight $1:k_2$. Importantly, as all literals in c correspond to different variables, these edges will go to different variable gadgets.
- For each two-literal clause c, add a *root-to-clause* edge (r, c) of weight $1:k_1$.

Note that the edges with double weights satisfy the assumption of Lemma 1 in Sect. 2; that is, each such weight $1:\omega'$ satisfies $1 \le \omega' \le K - 1$.

We now show that G has the required property (∗), proving the two implications separately.

(⇒) Suppose that ϕ has a satisfying assignment. Using this assignment, we construct a spanning tree T of G as follows:

- For every x_i-gadget, include in T edges (r, x_i'), (r, y_i), and (y_i, z_i). If $x_i = 0$, include in T edges $(\bar{x}_i, z_i)$ and (x_i, x_i'), otherwise include in T edges $(y_i, \bar{x}_i)$ and (z_i, x_i).
- For each clause c, include in T one clause-to-literal edge that is incident to any literal vertex that satisfies c in our chosen truth assignment for ϕ.

By routine inspection, T is indeed a spanning tree of G: each x_i-gadget is traversed from r without cycles, and all clause vertices are leaves of T. Figures 4 and 5 show how T traverses an x_i-gadget in different cases, depending on whether $x_i = 0$ or $x_i = 1$ in the truth assignment for ϕ, and on which literals are chosen to satisfy each clause.

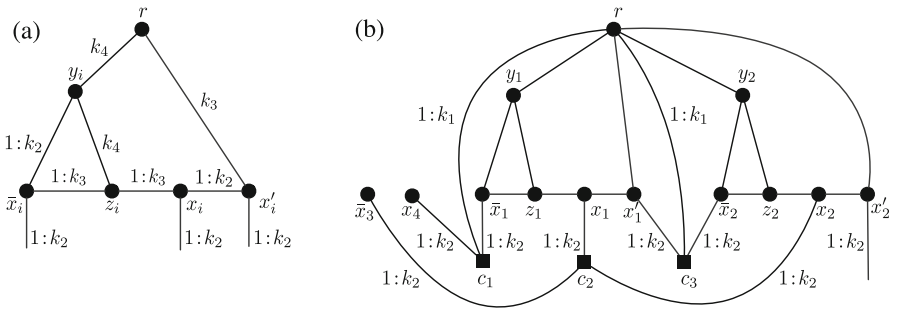

Fig. 3 **a**The x_i-gadget. **b** An example of a partial graph G for the boolean expression $\phi = c_1 \wedge c_2 \wedge c_3 \wedge \cdots$ where $c_1 = \bar{x}_1 \vee x_4$, $c_2 = x_1 \vee x_2 \vee \bar{x}_3$, and $c_3 = x_1 \vee \bar{x}_2$. (The weights of edges inside the variable gadgets are not shown.)

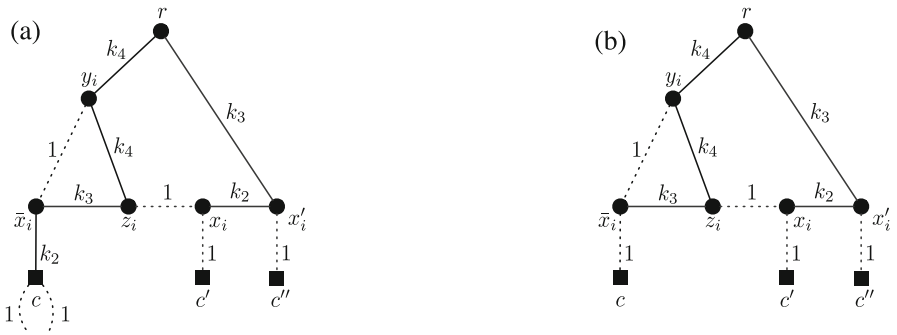

Fig. 4 The traversal of the x_i-gadget by T when $x_i = 0$. Solid lines are tree edges, dotted lines are non-tree edges. **a** $\bar{x}_i$ is chosen by clause c. **b** $\bar{x}_i$ is not chosen by clause c

We need to verify that each edge in T has congestion at most K. All the clause vertices are leaves in T, thus the congestion of each clause-to-literal edge is $k_2+2 = K$ (this holds for both three-literal and two-literal clauses). To analyze the congestion of the edges inside an x_i-gadget, we consider two cases, depending on the value of x_i in our truth assignment.

When $x_i = 0$, we have two sub-cases (a) and (b) as shown in Fig. 4. The congestions of the edges in the x_i-gadget are as follows:

- In both cases, $\text{cng}_{G,T}(r, x'_i) = k_3 + 3$.
- In case (a), $\text{cng}_{G,T}(r, y_i) = k_4 + 3$. In case (b), it is $k_4 + 2$.
- In case (a), $\text{cng}_{G,T}(y_i, z_i) = k_4 + 4$. In case (b), it is $k_4 + 3$.
- In case (a), $\text{cng}_{G,T}(\bar{x}_i, z_i) = k_3 + 3$. In case (b), it is $k_3 + 2$.
- In both cases, $\text{cng}_{G,T}(x_i, x'_i) = k_2 + 2$.

On the other hand, when $x_i = 1$, we have four sub-cases. Figure 4 illustrates cases (a)–(c). In case (d) (not shown in Fig. 4), none of the positive literal vertices x_i, x'_i is chosen to satisfy their corresponding clauses. The congestions of the edges in the x_i-gadget are as follows:

- In cases (a) and (b), $\text{cng}_{G,T}(r, x'_i) = k_3 + 3$. In cases (c) and (d), it is $k_3 + 2$.

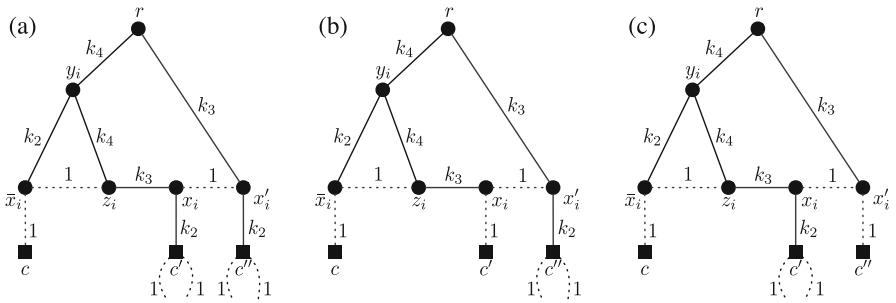

Fig. 5 The traversal of the x_i-gadget by T when $x_i = 1$. By c, c' and c'' we denote the clauses that contain literals $\bar{x}_i$, x_i and x'_i, respectively. **a** x_i and x'_i are chosen by clauses c' and c''. **b** x'_i is chosen by clause c''. **c** x_i is chosen by clause c'

- In cases (a) and (c), $\text{cng}_{G,T}(r, y_i) = k_4 + 4$. In cases (b) and (d), it is $k_4 + 3$.
- In cases (a) and (c), $\text{cng}_{G,T}(y_i, z_i) = k_4 + 4$. In cases (b) and (d), it is $k_4 + 3$.
- In cases (a) and (c), $\text{cng}_{G,T}(z_i, x_i) = k_3 + 3$. In cases (b) and (d), it is $k_3 + 2$.
- In all cases, $\text{cng}_{G,T}(y_i, \bar{x}_i) = k_2 + 2$.

In summary, the congestion of each edge of T is at most K. Thus $\text{cng}_G(T) \leq K$; in turn, $\text{stc}(G) \leq K$, as claimed.

($\Leftarrow$) We now prove the other implication in ($*$). We assume that G has a spanning tree T with $\text{cng}_G(T) \leq K$. We will show how to convert T into a satisfying truth assignment for ϕ. The proof consists of a sequence of claims showing that T must have a special form that will allow us to define this truth assignment.

Claim 1 *Each x_i-gadget satisfies the following property: for each literal vertex $\tilde{x}_i$, if some edge e of T (not necessarily in the x_i-gadget) is on the r-to-$\tilde{x}_i$ path in T, then $\partial_{G,T}(e)$ contains at least two distinct edges from this gadget other than (y_i, z_i).*

This claim is straightforward: it follows directly from the fact that there are two edge-disjoint paths from r to any literal vertex $\tilde{x}_i \in \{\bar{x}_i, x_i, x'_i\}$ that do not use edge (y_i, z_i).

Claim 2 *For each two-literal clause c, edge (r, c) is not in T.*

For both literals $\tilde{x}_i$ of clause c, there is an r-to-c path via the x_i-gadget, so, together with edge (r, c), G has three disjoint r-to-c paths. Thus, if (r, c) were in T, its congestion would be at least $k_1 + 2 > K$, proving Claim 2.

Claim 3 *All clause vertices are leaves in T.*

To prove Claim 3, suppose there is a clause c that is not a leaf. Then, by Claim 2, c has at least two clause-to-literal edges in T, say $(c, \tilde{x}_i)$ and $(c, \tilde{x}_j)$. We can assume that the last edge on the r-to-c path in T is $e = (c, \tilde{x}_i)$. Clearly, $r \in T_{\tilde{x}_i, c}$ and $\tilde{x}_j \in T_{c, \tilde{x}_i}$. By Claim 1, at least two edges of the x_j-gadget are in $\partial_{G,T}(e)$, and they contribute at least 2 to $\text{cng}_{G,T}(e)$. We now have some cases to consider.

If c is a two-literal clause, its root-to-clause edge (r, c) is also in $\partial_{G,T}(e)$, by Claim 2. Thus, $\text{cng}_{G,T}(e) \geq k_2 + 3 > K$ (see Fig. 6a). So assume now that c is a

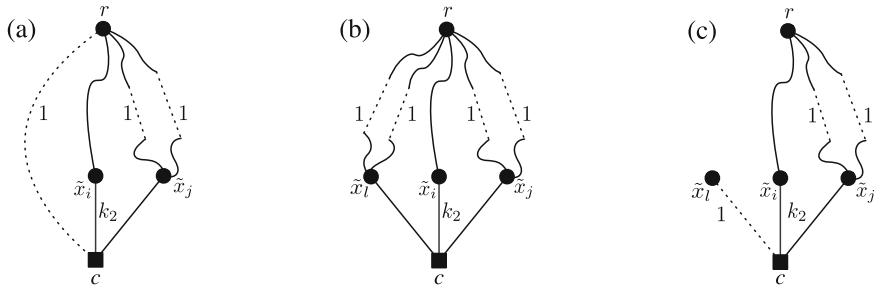

Fig. 6 Illustration of the proof of Claim 3. In **a** c is a two-literal clause; in Vb and **c**, c is a three-literal clause

three-literal clause, and let $\tilde{x}_l \neq \tilde{x}_i, \tilde{x}_j$ be the third literal of c. If T contains $(c, \tilde{x}_l)$, the x_l-gadget would also contribute at least 2 to $\text{cng}_{G,T}(e)$, so $\text{cng}_{G,T}(e) \geq k_2 + 4 > K$ (see Fig. 6b). Otherwise, $(c, \tilde{x}_l) \notin T$, and $(c, \tilde{x}_l)$ itself contributes 1 to $\text{cng}_{G,T}(e)$, so $\text{cng}_{G,T}(e) \geq k_2 + 3 > K$ (see Fig. 6c).

We have shown that if a clause vertex c is not a leaf in T, then in all cases the congestion of T would exceed K, completing the proof of Claim 3.

Claim 4 *For each x_i-gadget, edge (r, x'_i) is in T.*

Towards contradiction, suppose that (r, x'_i) is not in T. Let (x'_i, c) be the clause-to-literal edge of x'_i. If only one of the two edges (x'_i, x_i), (x'_i, c) is in T, making x'_i a leaf, then the congestion of that edge is $k_3 + k_2 + 1 > K$. Otherwise, both (x'_i, x_i), (x'_i, c) are in T. Because c is a leaf in T by Claim 3, $e = (x_i, x'_i)$ is the last edge on the r-to-x'_i path in T. As shown in Fig. 7a, $\text{cng}_{G,T}(e) \geq k_3 + k_2 + 2 > K$. This proves Claim 4.

Claim 5 *For each x_i-gadget, edge (r, y_i) is in T.*

To prove this claim, suppose (r, y_i) is not in T. We consider the congestion of the first edge e on the r-to-y_i path in T. By Claims 3 and 4, we have $e = (r, x'_i)$, all vertices of the x_i-gadget have to be in $T_{x'_i,r}$, and $T_{x'_i,r}$ does not contain literal vertices of another variable $x_j \neq x_i$. For each literal $\tilde{x}_i$ of x_i, if a clause-to-literal edge $(c, \tilde{x}_i)$ is in T, then the two other edges of c contribute 2 to $\text{cng}_{G,T}(e)$, otherwise $(c, \tilde{x}_i)$ contributes 1 to $\text{cng}_{G,T}(e)$. Then, $\text{cng}_{G,T}(e) \geq k_4 + k_3 + 3 > K$ (see Fig. 7b), proving Claim 5.

Claim 6 *For each x_i-gadget, exactly one of edges (z_i, x_i) and (x_i, x'_i) is in T.*

By Claims 4 and 5, edges (r, y_i) and (r, x'_i) are in T. Since the clause neighbor c' of x_i is a leaf of T, by Claim 3, if none of (z_i, x_i), (x_i, x'_i) were in T, x_i would not be reachable from r in T. Thus, at least one of them is in T. Now, assume both (z_i, x_i) and (x_i, x'_i) are in T (see Fig. 8a). Then, edge (y_i, z_i) is not in T, as otherwise we would create a cycle. Let us consider the congestion of edge $e = (r, x'_i)$. Clearly, x_i and x'_i are in $T_{x'_i,r}$. The edges of the two clause neighbors c' and c'' of x_i and x'_i contribute at least 2 to $\text{cng}_{G,T}(e)$, by Claim 3. In addition, by Claim 1, besides e and (y_i, z_i), $\partial_{G,T}(e)$ contains another edge of the x_i-gadget which contributes at least another 1 to $\text{cng}_{G,T}(e)$. Thus, $\text{cng}_{G,T}(e) \geq k_4 + k_3 + 3 > K$ — a contradiction. This proves Claim 6.

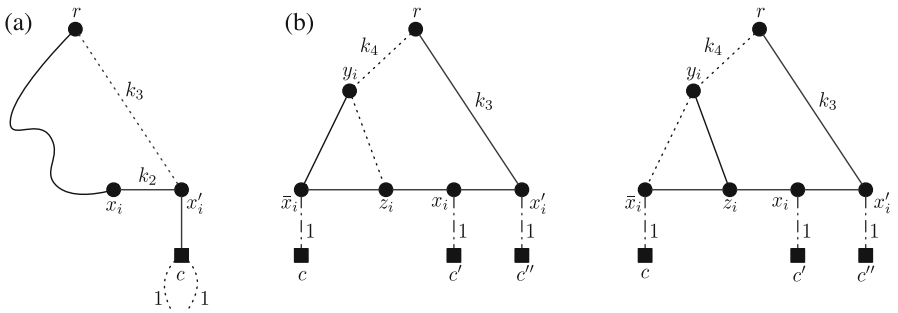

Fig. 7 **a** Illustration of the proof of Claim 4. **b** Illustration of the proof of Claim 5. Dot-dashed lines are edges that may or may not be in T

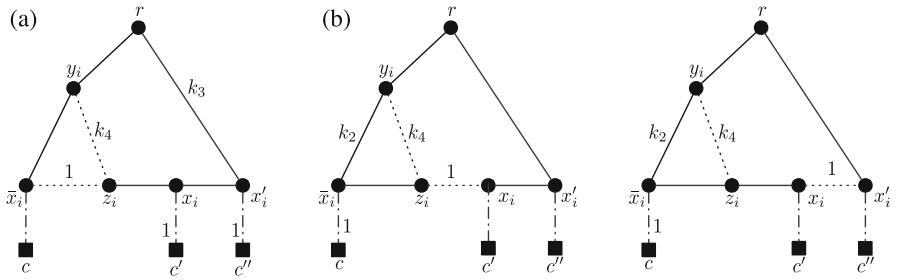

Fig. 8 **a** Illustration of the proof of Claim 6. **b** Illustration of the proof of Claim 7

Claim 7 *For each x_i-gadget, edge (y_i, z_i) is in T.*

By Claims 4 and 5, the two edges (r, x'_i) and (r, y_i) are in T. Now assume, towards contradiction, that (y_i, z_i) is not in T (see Fig. 8b). By Claim 6, only one of (z_i, x_i) and (x_i, x'_i) is in T. Furthermore, the clause neighbor c' of x_i is a leaf of T, by Claim 3. As a result, (z_i, x_i) cannot be on the y_i-to-z_i path in T. To reach z_i from y_i, the two edges $(y_i, \bar{x}_i)$, $(\bar{x}_i, z_i)$ have to be in T. Let us consider the congestion of $e = (y_i, \bar{x}_i)$. The edges of the clause neighbor c of $\bar{x}_i$ contribute at least 1 to the congestion of e, by Claim 3. Also, by Claim 1, besides e and (y_i, z_i), $\partial_{G,T}(e)$ contains another edge of the x_i-gadget which contributes at least 1 to $\text{cng}_{G,T}(e)$. In total, $\text{cng}_{G,T}(e) \geq k_4 + k_2 + 2 > K$, reaching a contradiction and completing the proof of Claim 7.

Claim 8 *For each x_i-gadget, if its clause-to-literal edge $(\bar{x}_i, c)$ is in T, then its other two clause-to-literal edges (x_i, c') and (x'_i, c'') are not in T.*

Assume the clause-to-literal edge $(\bar{x}_i, c)$ of the x_i-gadget is in T. By Claim 5 and 7, edges (r, y_i) and (y_i, z_i) are in T. If $(y_i, \bar{x}_i)$ is also in T, edge $(\bar{x}_i, z_i)$ cannot be in T, and it contributes 1 to $\text{cng}_{G,T}(y_i, \bar{x}_i)$. As shown in Fig. 9a, $\text{cng}_{G,T}(y_i, \bar{x}_i) = k_2 + 3 > K$. Thus, $(y_i, \bar{x}_i)$ cannot be in T. Since c is a leaf of T, edge $(\bar{x}_i, z_i)$ has to be in T, for otherwise $\bar{x}_i$ would not be reachable from r. By Claim 6, one of edges (z_i, x_i) and (x_i, x'_i) is in T. If (z_i, x_i) is in T (see Fig. 9b), $\text{cng}_{G,T}(y_i, z_i) \geq k_4 + 5 > K$. Hence, (z_i, x_i) is not in T, which implies that (x_i, x'_i) is in T.

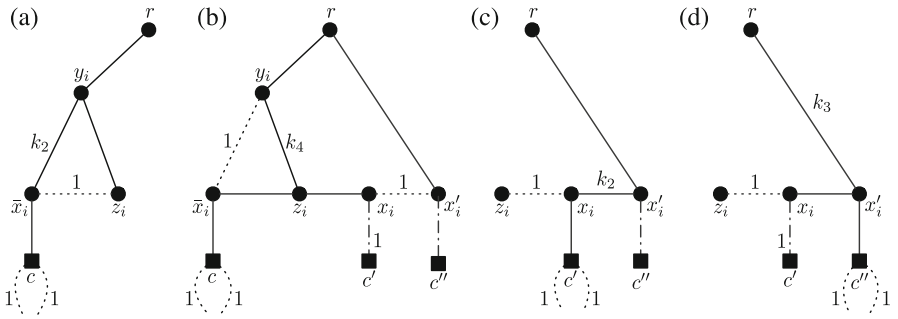

Fig. 9 Illustration of the proof of Claim 8

Now, we proceed by contradiction assuming that at least one other clause-to-literal edge of the x_i-gadget is in T. If edge (x_i, c') is in T, $\text{cng}_{G,T}(x_i, x'_i) \geq k_2 + 3 > K$, as shown in Fig. 9c. Similarly, if (x'_i, c'') is in T, $\text{cng}_{G,T}(r, x'_i) \geq k_3 + 4 > K$ (see Fig. 9d). So we reach a contradiction in both cases, thus proving Claim 8.

We are now ready to complete the proof of the ($\Leftarrow$) implication in the equivalence ($*$). We use our spanning tree T of congestion at most K to create a truth assignment for ϕ by setting $x_i = 0$ if the clause-to-literal edge of $\bar{x}_i$ is in T, otherwise $x_i = 1$. By Claim 8, this truth assignment is well-defined. Each clause has one clause-to-literal edge in T which ensures that all clauses are indeed satisfied.

4 $\mathbb{NP}$-Completeness Proof of K − STC for Bipartite Graphs of Radius 2 and $K \geq 6$

In this section we establish the following result:

Theorem 2 *For any fixed integer $K \geq 6$, K − STC is $\mathbb{NP}$-complete for bipartite graphs of radius 2, even if they have only one vertex of degree greater than* $\max(6, K - 2)$.

First, we introduce a restricted variant of the satisfiability problem, which we name (M2P1N)-SAT, that is used in the reduction. An instance of (M2P1N)-SAT is a boolean expression in conjunctive normal form with the following properties:

- Each clause either contains three positive literals (a 3P-clause), or two positive literals (a 2P-clause), or two negative literals (a 2N-clause). Also, literals in the same clause are of different variables.
- Each variable appears exactly three times: once in a 3P-clause, once in a 2P-clause and once in a 2N-clause.
- Two clauses share at most one variable.

Lemma 2 *(M2P1N)-SAT is $\mathbb{NP}$-complete.*

Proof It is clear that (M2P1N)-SAT belongs to $\mathbb{NP}$. To demonstrate $\mathbb{NP}$-completeness, we show a polynomial-time reduction from the $\mathbb{NP}$-complete problem called BALANCED-3SAT [25]. BALANCED-3SAT is a restriction of the satisfiability problem to boolean expressions in conjunctive normal form where, for each variable x,

the positive literal x appears the same number of times as the negative literal $\bar{x}$. We can further assume that every variable appears at least four times, and that, for each clause, all variables that appear in this clause are different.

Given an instance ψ of BALANCED-3SAT, we construct an instance ϕ of (M2P1N)-SAT as follows:

- For each variable x in ψ, if x appears $2t$ times (for some integer $t \geq 2$), create $2t$ new variables $x_0, x_1, \ldots, x_{2t-1}$.
- Replace the t positive occurrences of x by even-indexed variables $x_0, x_2, \ldots, x_{2t-2}$, and replace its t negative occurrences by odd-indexed variables $x_1, x_3, \ldots, x_{2t-1}$.
- Add t clauses of the form $(x_i \vee x_{i+1})$ for $i = 0, 2, \ldots, 2t-2$, and t clauses of the form $(\bar{x}_i \vee \bar{x}_{(i+1) \bmod 2t})$ for $i = 1, 3, \ldots, 2t-1$.

By the construction, ϕ is a correct instance of (M2P1N)-SAT. For each variable x of ψ, its corresponding "cycle" of the newly added two-literal clauses in ϕ ensures that $x_0 = \bar{x}_1 = x_2 = \bar{x}_3 = \ldots = x_{2t-2} = \bar{x}_{2t-1}$. Thus, a truth assignment that satisfies ψ can be converted into a truth assignment that satisfies ϕ by setting the even-indexed variables to the truth value of the original variable in ψ, and the odd-indexed variables to the opposite value. Conversely, a truth assignment that satisfies ϕ can be converted into a truth assignment that satisfies ψ by reversing this process. This shows that ψ is satisfiable if and only if ϕ is satisfiable, completing the proof of the lemma. □

In order to prove Theorem 2, we show a polynomial-time reduction from (M2P1N)-SAT. Given an instance ϕ of (M2P1N)-SAT, we construct a graph G such that

$(*)$ ϕ has a satisfying truth assignment if and only if $\mathrm{stc}(G) \leq K$.

Graph G will be bipartite, of radius 2, and will have only one vertex of degree larger than $\max(6, K-2)$. We will describe G using some double-weighted edges, that we refer to as *fat edges*. As previously discussed in Sect. 2, here we need a specific realization of these double weighted edges, in which weights are realized using the spintop graph. (See Figs. 1 and 2.) For $i \in \{1, 2, 3, 4, 5\}$, let $k_i = K - i$. We start with an empty graph G and proceed as follows:

- Add a *root vertex* r.
- For each variable x of ϕ, add a *variable vertex* x and a *root-to-vertex* edge (r, x).
- For each clause c, add a clause vertex c, and add edges from c to the vertices representing variables whose literals (positive or negative) appear in c. If clause c contains all positive literals, we call its clause-to-variable edges *positive-clause edges*, otherwise its clause-to-variable edges are *negative-clause edges*.
- For each 2P-clause vertex c, add a fat edge (r, c) of double weight $k_5 : k_1$.
- For each 2N-clause vertex c, add a fat edge (r, c) of double weight $k_4 : k_1$.

It is important to note that all edges with double-weights $\omega : K-1$ satisfy $1 \leq \omega \leq K-1$, as mentioned in Sect. 2. See Fig. 10a for an example of a partial graph constructed using the above rules. By routine inspection, taking into account the spintop realizations of weighted edges, G is bipartite, all vertices are at distance at most 2 from r, and r is the only vertex of degree larger than $\max(6, K-2)$. We now proceed to show that G satisfies property $(*)$.

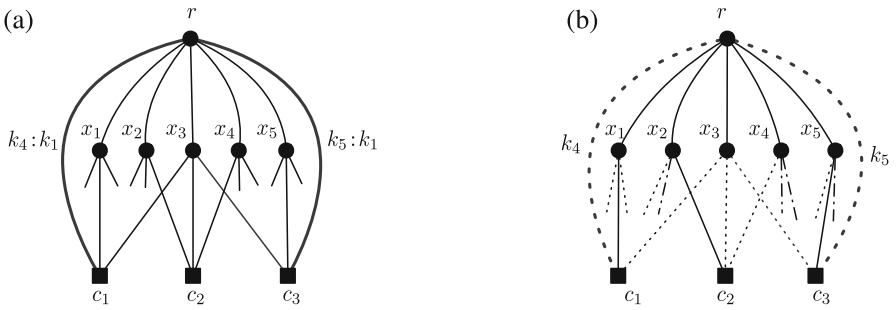

Fig. 10 **a** An example of a partial graph G for $\phi = c_1 \wedge c_2 \wedge c_3 \wedge \cdots$ where $c_1 = (\bar{x}_1 \vee \bar{x}_3), c_2 = (x_2 \vee x_3 \vee x_4), c_3 = (x_3 \vee x_5)$. Bold lines represent fat edges with given double weights. **b** An example of a partial tree T of G where x_1 is chosen by c_1, x_2 by c_2, x_5 is by c_3. Solid lines are tree edges, dotted lines are non-tree edges, and dot-dashed lines are edges that may or may not be in T. Non-tree double-weighted edges contribute the indicated weights to edge congestion

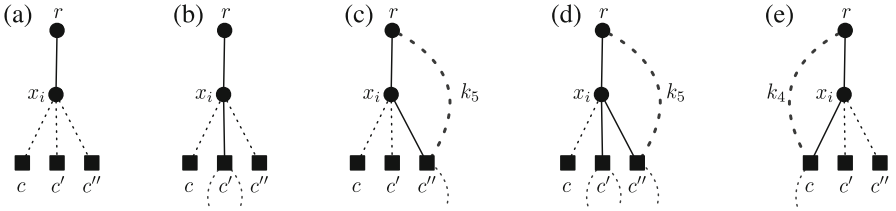

Fig. 11 By c, c', c'', we denote the 2N-clause, 3P-clause and 2P-clause of x_i respectively. In **a**, x_i is not chosen by any clause, it is chosen by c' in **b**, by c'' in **c**, by both c' and c'' in **d**, and by c in **e**

($\Rightarrow$) Assume that ϕ has a satisfying truth assignment. From this assignment we construct a spanning tree T of G by adding all root-to-vertex edges, and, for each clause c, adding to T an edge from c to any variable vertex whose literal satisfies c (see Fig. 10b). By the construction, T is a spanning tree of G. Note that all clause vertices in T are leaves and all fat edges are non-tree edges.

Now, we proceed to verify that all tree edges of T have congestion at most K. We start with leaf edges of T. The congestion of the leaf edge of a 3P-clause vertex is 3. For a 2P-clause vertex, the congestion of its leaf edge is $K - 3$, because its fat edge contributes $k_5 = K - 5$. For a 2N-clause vertex, the congestion of its leaf edge is $K - 2$, because its fat edge contributes $k_4 = K - 4$.

Next, consider the root-to-vertex edge of a variable x_i. If x_i is not chosen to satisfy any clauses, then $\mathrm{cng}_{G,T}(r, x_i) = 4$ (see Fig. 11a). If it is chosen to satisfy only its 3P-clause, then $\mathrm{cng}_{G,T}(r, x_i) = 5$ (see Fig. 11b). If it is chosen to satisfy only its 2P-clause, then $\mathrm{cng}_{G,T}(r, x_i) = k_5 + 4 = K - 1$ (see Fig. 11c). If it is chosen to satisfy both its 3P-clause and its 2P-clause, then $\mathrm{cng}_{G,T}(r, x_i) = k_5 + 5 = K$ (see Fig. 11d). Finally, if it is chosen to satisfy its 2N-clause, then $\mathrm{cng}_{G,T}(r, x_i) = k_4 + 4 = K$ (see Fig. 11e).

There are also edges inside the realizations of fat edges, but their congestion does not exceed K, by Lemma 1. We have thus shown that the congestions of all edges in T are at most K; that is, $\mathrm{stc}(G) \leq K$.

($\Leftarrow$) Assume T is a spanning tree of G with $\text{cng}_G(T) \leq K$. From T, we will construct a satisfying truth assignment for ϕ. The argument here, while much shorter, has a subtle aspect that was not present in the proof in Sect. 3, namely now it is not necessarily true that all clause vertices in T are leaves.[2]

We present two claims showing that T must have a special form that will allow us to define the truth assignment for ϕ.

Claim 9 *For each two-literal clause vertex c, its fat edge (r, c) is not in T.*

For each literal of c, there is an r-to-c path via the variable vertex of this literal. So, together with edge (r, c), G has three disjoint r-to-c paths. Thus, if (r, c) were in T, its congestion would be at least $k_1 + 2 > K$, proving Claim 9.

Claim 10 *For each variable vertex x_i, if its negative-clause edge is in T then its two positive-clause edges are not in T.*

Denote by c, c', c'' the 2N, 3P, 2P-clause vertices of x_i respectively. Since c, c', c'' all contain variable x_i, they cannot share any other variables (by the definition of (M2P1N)-SAT). Therefore, the four literals in c, c', c'' other than x_i and $\bar{x}_i$ must all involve different variables.

Toward contradiction, suppose (x_i, c) and at least one of (x_i, c'), (x_i, c'') are in T. We will estimate the congestion of the first edge $e = (r, v)$ on the r-to-c path in T.

By Claim 9, fat edge (r, c) contributes k_4 to $\text{cng}_{G,T}(e)$. The rest of the argument is based on the following two observations: (i) If a clause $\tilde{c} \in \{c, c', c''\}$ is in $T_{v,r}$, and some variable x is in $\tilde{c}$, then either (r, x) or $(x, \tilde{c})$ is in $\partial_{G,T}(e)$; that is, this x contributes 1 to $\text{cng}_{G,T}(e)$. (This is true whether or not $v = x$. And if $x = x_i$ and $\tilde{c} = c$, then (r, x_i) is the edge that contributes to $\text{cng}_{G,T}(e)$.) On the other hand, (ii) if a clause $\tilde{c} \in \{c', c''\}$ is not in $T_{v,r}$, then $(x_i, \tilde{c})$ contributes 1 to $\text{cng}_{G,T}(e)$.

Now we have some cases to consider. First, if $c' \in T_{v,r}$ and $c'' \notin T_{v,r}$, by the above observations, four different variables in c, c' contribute 4 to $\text{cng}_{G,T}(e)$ and (x_i, c'') contributes 1. In total, $\text{cng}_{G,T}(e) \geq k_4 + 4 + 1 > K$. On the other hand, when $c'' \in T_{v,r}$ and $c' \notin T_{v,r}$, the three different variables of c, c'' contribute 3 while (x_i, c') contributes 1 to $\text{cng}_{G,T}(e)$. Also, the fat edge (r, c'') contributes k_5, by Claim 9. Thus, $\text{cng}_{G,T}(e) \geq k_4 + k_5 + 3 + 1 > K$. Lastly, when both c', c'' are in $T_{v,r}$, the five different variables of c, c', c'' contribute to $\text{cng}_{G,T}(e)$, so $\text{cng}_{G,T}(e) \geq k_4 + 5 > K$. We have thus shown that the congestion of e exceeds K in all cases, completing the proof of Claim 10.

We are now ready to describe the truth assignment for ϕ using T. For each variable x_i, assign $x_i = 0$ if its negative clause edge is in T, otherwise, $x_i = 1$. By Claim 10, the truth assignment is well-defined. By Claim 9, each clause vertex has at least one edge to a variable vertex, which ensures all clauses are satisfied. This completes the proof of Theorem 2.

[2] For large K, it is possible that a single branch out of r may visit multiple variable gadgets via 3P-clause vertices. For example, a 3P-clause vertex c can have tree edges to two literal vertices, and the congestion of the first edge on the r-to-c path may be as small as 7.

5 Final Comments

The complexity status of $\boldsymbol{K}$ − STC for $K = 4$ remains open. Our efforts to extend our construction to $K = 4$ by refining the x_i-gadget in Fig. 3 were not successful. One can think of this variable gadget as a black-box subgraph, with edges connecting it to the rest of the graph, that satisfies certain properties. We formalized this concept and were able to prove that such a gadget does not in fact exist when $K = 4$. For this reason, we believe that the $\mathbb{NP}$-hardness proof, if at all possible, would require substantial new insights and approaches. It may be easier to use a reduction from a different $\mathbb{NP}$-hard problem, not necessarily a version of SAT.

At the same time, we also would not exclude the possibility that $\mathbf{4}$ − STC may be solvable in polynomial time. (We have also done some work in this direction.) Given our still limited understanding, we are not prepared to make a conjecture in favor or against $\mathbb{NP}$-hardness.

The complexity status of $\boldsymbol{K}$ − STC on graphs of radius 2 also remains open when $K \in \{4, 5\}$. Resolving this problem could give some insights into the complexity of $\mathbf{4}$ − STC.

Notes A preliminary version of this paper, containing only the results in Sect. 3, appeared in [26].

Author Contributions All authors wrote and reviewed the manuscript.

Funding The authors' work was partially supported by National Science Foundation grant CCF-2153723.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Bhatt, S., Chung, F., Leighton, T., Rosenberg, A.: Optimal simulations of tree machines. In: Proceedings of the 27th Annual Symposium on Foundations of Computer Science. SFCS '86, pp. 274–282. IEEE Computer Society, USA (1986)
2. Rosenberg, A.L.: Graph embeddings 1988: Recent breakthroughs, new directions. In: Reif, J.H. (ed.) VLSI Algorithms and Architectures, pp. 160–169. Springer, New York, NY (1988)
3. Khuller, S., Raghavachari, B., Young, N.: Designing multi-commodity flow trees. Inf. Process. Lett. **50**(1), 49–55 (1994)
4. Ostrovskii, M.I.: Minimal congestion trees. Discret. Math. **285**(1), 219–226 (2004)
5. Otachi, Yota: A survey on spanning tree congestion. In: Fomin, Fedor V., Kratsch, Stefan, van Leeuwen, Erik Jan (eds.) Treewidth, Kernels, and Algorithms: Essays Dedicated to Hans L. Bodlaender on the

Occasion of His 60th Birthday, pp. 165–172. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-42071-0_12
6. Löwenstein, C.: In the complement of a dominating set. PhD thesis, Technische Universität at Ilmenau (2010)
7. Otachi, Yota, Bodlaender, Hans L., van Leeuwen, Erik Jan: Complexity results for the spanning tree congestion problem. In: Graph Theoretic Concepts in Computer Science, pp. 3–14. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16926-7_3
8. Okamoto, Yoshio, Otachi, Yota, Uehara, Ryuhei, Uno, Takeaki: Hardness results and an exact exponential algorithm for the spanning tree congestion problem. In: Ogihara, Mitsunori, Tarui, Jun (eds.) Theory and Applications of Models of Computation, pp. 452–462. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20877-5_44
9. Kozawa, K., Otachi, Y., Yamazaki, K.: On spanning tree congestion of graphs. Discret. Math. **309**(13), 4215–4224 (2009)
10. Bodlaender, H.L., Kozawa, K., Matsushima, T., Otachi, Y.: Spanning tree congestion of k-outerplanar graphs. Discret. Math. **311**(12), 1040–1045 (2011)
11. Kozawa, K., Otachi, Y.: Spanning tree congestion of rook's graphs. Discussiones Mathematicae Graph Theory **31**(4), 753–761 (2011)
12. Bodlaender, H., Fomin, F., Golovach, P., Otachi, Y., Leeuwen, E.: Parameterized complexity of the spanning tree congestion problem. Algorithmica **64**, 1–27 (2012)
13. Fekete, Sándor. P., Kremer, Jana: Tree spanners in planar graphs. Discr. Appl. Math. **108**(1–2), 85–103 (2001). https://doi.org/10.1016/S0166-218X(00)00226-2
14. Cai, L., Corneil, D.G.: Tree spanners. SIAM J. Discret. Math. **8**(3), 359–387 (1995)
15. Dragan, F.F., Fomin, F.V., Golovach, P.A.: Spanners in sparse graphs. J. Comput. Syst. Sci. **77**(6), 1108–1119 (2011)
16. Emek, Y., Peleg, D.: Approximating minimum max-stretch spanning trees on unweighted graphs. SIAM J. Comput. **38**(5), 1761–1781 (2009)
17. Alvarruiz Bermejo, F., Martínez Alzamora, F., Vidal Maciá, A.M.: Improving the efficiency of the loop method for the simulation of water distribution networks. J. Water Resour. Plan. Manag. **141**(10), 1–10 (2015)
18. Benczúr, A.A., Karger, D.R.: Approximating $s-t$ minimum cuts in $\tilde{O}(n^2)$ time. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, pp. 47–55 (1996)
19. Fung, W.S., Hariharan, R., Harvey, N.J., Panigrahi, D.: A general framework for graph sparsification. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, pp. 71–80 (2011)
20. Spielman, D.A., Teng, S.-H.: Spectral sparsification of graphs. SIAM J. Comput. **40**(4), 981–1025 (2011)
21. Chandran, L.S., Cheung, Y.K., Issac, D.: Spanning Tree Congestion and Computation of Generalized Győri-Lovász Partition. In: 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), 107, 32–13214 (2018)
22. Luu, H., Chrobak, M.: Better hardness results for the minimum spanning tree congestion problem. CoRR (2022) https://doi.org/10.48550/arXiv.2209.08219https://arxiv.org/abs/2209.08219
23. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. SIAM J. Comput. **23**(4), 864–894 (1994)
24. Yoshinaka, Ryo: Higher-order matching in the linear lambda calculus in the absence of constants is NP-complete. In: Term Rewriting and Applications, pp. 235–249. Springer Berlin Heidelberg, Berlin, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32033-3_18
25. Hägele, Klemens, Ó Dúnlaing, Colm, Riis, Søren.: The complexity of scheduling TV commercials. Electron. Notes Theor. Comput. Sci. **40**, 162–185 (2001). https://doi.org/10.1016/S1571-0661(05)80043-X
26. Luu, H., Chrobak, M.: Better hardness results for the minimum spanning tree congestion problem. In: Proceedings of 17th International Conference and Workshop on Algorithms and Computation WALCOM'23, pp. 167–178 (2023)