Wu *et al. Algorithms Mol Biol* (2019) 14:5
https://doi.org/10.1186/s13015-019-0141-z

Algorithms for Molecular Biology

RESEARCH Open Access

Connectivity problems on heterogeneous graphs

Jimmy Wu[1], Alex Khodaverdian[2], Benjamin Weitz[2] and Nir Yosef[2*]

Abstract

Background: Network connectivity problems are abundant in computational biology research, where graphs are used to represent a range of phenomena: from physical interactions between molecules to more abstract relationships such as gene co-expression. One common challenge in studying biological networks is the need to extract meaningful, small subgraphs out of large databases of potential interactions. A useful abstraction for this task turned out to be the Steiner Network problems: given a reference "database" graph, find a parsimonious subgraph that satisfies a given set of connectivity demands. While this formulation proved useful in a number of instances, the next challenge is to account for the fact that the reference graph may not be static. This can happen for instance, when studying protein measurements in single cells or at different time points, whereby different subsets of conditions can have different protein milieu.

Results and discussion: We introduce the *condition* Steiner Network problem in which we concomitantly consider a set of distinct biological conditions. Each condition is associated with a set of connectivity demands, as well as a set of edges that are assumed to be present in that condition. The goal of this problem is to find a minimal subgraph that satisfies all the demands through paths that are present in the respective condition. We show that introducing multiple conditions as an additional factor makes this problem much harder to approximate. Specifically, we prove that for C conditions, this new problem is NP-hard to approximate to a factor of $C - \epsilon$, for every $C \geq 2$ and $\epsilon > 0$, and that this bound is tight. Moving beyond the worst case, we explore a special set of instances where the reference graph grows *monotonically* between conditions, and show that this problem admits substantially improved approximation algorithms. We also developed an integer linear programming solver for the general problem and demonstrate its ability to reach optimality with instances from the human protein interaction network.

Conclusion: Our results demonstrate that in contrast to most connectivity problems studied in computational biology, accounting for multiplicity of biological conditions adds considerable complexity, which we propose to address with a new solver. Importantly, our results extend to several network connectivity problems that are commonly used in computational biology, such as Prize-Collecting Steiner Tree, and provide insight into the theoretical guarantees for their applications in a multiple condition setting.

Keywords: Steiner Network, NP hard, Approximation algorithm, Protein–protein interaction

*Correspondence: niryosef@eecs.berkeley.edu
[2] Department of Electrical Engineering and Computer Science, UC Berkeley, Berkeley, CA, USA
Full list of author information is available at the end of the article

Background

In molecular biology applications, networks are routinely defined over a wide range of basic entities such as proteins, genes, metabolites, or drugs, which serve as nodes. The edges in these networks can have different meanings, depending on the particular context. For instance, in protein–protein interaction (PPI) networks, edges represent physical contact between proteins, either within stable multi-subunit complexes or through transient causal interactions (i.e., an edge (x, y) means that protein x can cause a change to the molecular structure of protein y and thereby alter its activity). The body of knowledge encapsulated within the human PPI network (tens of thousands of nodes and hundreds of thousands of edges in current databases, curated from thousands of studies [1]) is routinely used by computational biologists to generate hypotheses of how various signals are transduced in eukaryotic cells [2–6]. The basic premise is that a process that starts with a change to the activity of protein u and ends with the activity of protein v must be propagated through a chain of interactions between u and v. The natural extension regards a process with a certain collection of protein pairs $\{(u_1, v_1), \ldots, (u_k, v_k)\}$, where we are looking for a chain of interactions between each u_i and v_i [7]. In another set of applications, the notion of directionality is not directly assumed and instead, one is looking for a parsimonious subgraph that connects together a set S of proteins that are postulated to be active [8, 9].

In most applications, the identity of the so called terminal nodes (i.e., (u_i, v_i) pairs or the set S) is assumed to be known (or inferred from experimental data such as ChIP-seq [5, 8, 9]), while the identity of the intermediate nodes and interactions is unknown. The goal therefore becomes to complete the gap and find a probable subgraph of the PPI network that simultaneously satisfies all the connectivity demands, thereby explaining the overall biological activity. Since the edges in the PPI network can be assigned a probability value (reflecting the credibility of their experimental evidence), by taking the negative log of these values as edge weights, the task becomes minimizing the total edge weight, leading to an instance of the Steiner Network problem. We have previously used this approach to study the propagation of a stabilizing signal in pro-inflammatory T cells, leading to the identification of a new molecular pathway (represented by a sub-graph of the PPI network) that is critical for mounting an autoimmune response, as validated experimentally by perturbation assays and disease models in mice [5]. Tuncbag et al. [9] have utilized the undirected approach using the Prize-Collecting Steiner Tree model, where the input is a network G along with a penalty function, $p(v)$ for each protein (node) in the network (based on their importance; e.g., fold-change across conditions). The goal in this case is to find a probable subtree which contains the majority of the high cost proteins in G, while accounting for penalties paid by both edge usage and missing proteins, in order to capture the biological activity represented in such a network [8, 9].

While these studies contributed to our understanding of signal transduction pathways in living cells, they do not account for a critical aspect of the underlying biological complexity. In reality, proteins (nodes) can become activated or inactivated at different conditions, thereby giving rise to a different set of potential PPIs that might take place [10]. Here, the term *condition* can refer to different points in time [11], different treatments [12], or, more recently, different cells [13]. Indeed, advances in experimental proteomics provide a way to estimate these changes at high throughput, e.g., measuring phosphorylation levels or overall protein abundance, proteome-wide for a limited number of samples [12]. A complementary line work provides a way to evaluate the abundance of smaller numbers of proteins (typically dozens of them) in hundreds of thousands of single cells [13].

The next challenge is therefore to study connectivity problems that take into account not only the endpoints of each demand, but also the condition in which these demands should be satisfied. This added complication was tackled by Mazza et al. [14], who introduced the "Minimum k-Labeling (MKL)" problem. In this setting, each connectivity demand comes with a label, which represents a certain experimental condition or time point. The task is to label edges in the PPI network so as to satisfy each demand using its respective label, while minimizing the number of edges in the resulting sub-graph and the number of labels used to annotate these edges. While MKL was an important first step, namely introducing the notion of different demands for each condition, the more difficult challenge still remains that of considering variability in the reference graph, namely different sets of proteins that may be active and available for use in each condition. To this effect, we note the existence of multi-layer networks in the data-mining space. In this context, studies have focused on networks which have edges that span across specified dimensions, or conditions [15, 16]. However, we could not find studies that tackle the problem of parsimonious connectivity in this domain.

Summary of main contributions

To address this open challenge, here we introduce the Condition Steiner Network (CSN) problem. In this setting, we are given a weighted undirected graph G, a set of C conditions and a set of $k \geq C$ demands, at least one per condition (note that we also cover the case of directed graphs, with similar results). The conditions are specified

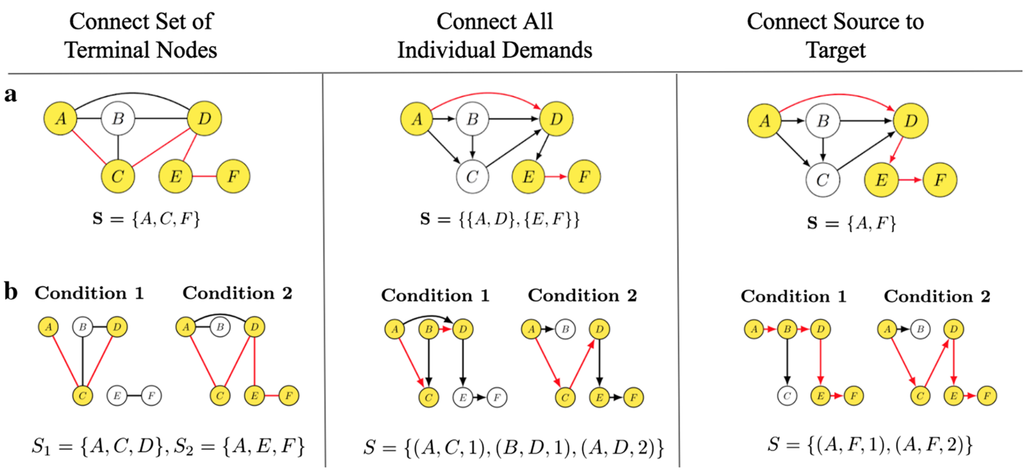

Connect Set of Terminal Nodes
Connect All Individual Demands
Connect Source to Target
a
b
$\mathbf{S} = \{A, C, F\}$
$\mathbf{S} = \{\{A, D\}, \{E, F\}\}$
$\mathbf{S} = \{A, F\}$
Condition 1
Condition 2
$S_1 = \{A, C, D\}, S_2 = \{A, E, F\}$
$S = \{(A, C, 1), (B, D, 1), (A, D, 2)\}$
$S = \{(A, F, 1), (A, F, 2)\}$

Fig. 1 Examples of well studied network problems (**a**), and their corresponding extension with multiple conditions (**b**). The problems shown are: Undirected Steiner Tree, Directed Steiner Network, and Shortest Path, respectively. Yellow nodes and red edges correspond to nodes and edges used in the optimal solutions for the corresponding instances

Table 1 Approximation bounds for the various Steiner Network Problems in their classic setting and condition setting

Problems	Classic		Condition	
	Lower bound	Upper bound	Lower bound(s)	Upper bound(s)
Steiner Forest	1.01 [19]	2 [18]	$C - \epsilon, k - \epsilon$	$2C, k$
Directed Steiner Network	$k^{1/4-o(1)}$ [27]	$k^{1/2+\epsilon}$ [21, 22]	$C - \epsilon, k - \epsilon$	$C \cdot k^{1/2+\epsilon}, k$
Undirected Shortest Path	N/A	1	$C - \epsilon$	C
Directed Shortest Path	N/A	1	$C - \epsilon$	C
Steiner Tree	1.01 [19]	1.39 [17]	$C - \epsilon$	$1.39C$
Prize-Collecting Steiner Tree	1.01 [19]	1.97 [26]	$C - \epsilon$	$1.97C$

For the classic problems, we have indicated the papers in which the bounds are shown. For the condition problems, all the lower bounds are developed in the present work; all the upper bounds are the naive bounds obtained from the "union of shortest paths" heuristic, or from applying the best known approximation algorithm for the appropriate classic Steiner problem to each condition, then taking the union of those solutions

over a sequence of graphs G_c defined over each condition, where vertices remain the same, but edges are allowed to change across conditions (notably, our results also hold when G_c is defined with changing vertices rather than edges). Furthermore, demands are in the form of "connect node u to node v through a path of nodes that are present in condition c". The goal is to find a minimum-weight subgraph of G that satisfies all the demands (Fig. 1).

We first show that it is NP-hard to find a solution that achieves a nontrivial approximation factor (by the "trivial" approximation, we mean the one obtained by solving the problem independently for each condition). This result extends to several types of connectivity problems and provides a theoretical lower bounds to the best-possible approximation guarantee that can be achieved in a multiple condition setting (Table 1). For instance, we can conclude that concomitantly solving the shortest path problem for a set of conditions is hard to approximate, and that the trivial solution (i.e., solving the problem to optimality in each condition) is, theoretically, the best that one can do. Another example, commonly used in PPI analysis, is the Prize-Collecting Steiner Tree problem [8, 9]. Here, our results indicate that given a fixed input for this problem (i.e., a penalty function $p(v)$ for each vertex), it is NP-hard to solve it concomitantly in C conditions, such that the weight of the obtained solution is less than C times that of the optimal solution. Interestingly, a theoretical guarantee of $C \cdot (2 - \frac{2}{|V|})$[1] can be obtained by solving the problem independently for each time point

While these results provide a somewhat pessimistic view, they rely on the assumption that the network frames G_c are arbitrary. In the last part of this paper, we show that for the specific case where the conditions can be ordered such that each condition is a subset of the

[1] V is the set of nodes in the reference graph G.

next (namely, $G_c \subseteq G_{c'}$ for $c \leq c'$) then the CSN problem can be reduced to a standard connectivity problem with a single condition, leading to substantially better theoretical guarantees. Finally, we develop an integer linear program for the general CSN problem, and show that provided with real-world input (namely, the human PPI) it is capable of reaching an optimal solution in a reasonable amount of time.

Introduction to Steiner problems

The Steiner Tree problem, along with its many variants and generalizations, form a core family of NP-hard combinatorial optimization problems. Traditionally, the input to one of these problems is a single (usually weighted) graph, along with requirements about which nodes need to be connected in some way; the goal is to pick a minimum-weight subgraph satisfying the connectivity demands.

In this paper, we offer a *multi-condition* perspective; in our setting, multiple graphs over the same vertex set (which one can think of as an initial graph changing over a set of discrete conditions), are all given as input, and the goal is to pick a subgraph satisfying condition-sensitive connectivity requirements. Our study of this problem draws motivation and techniques from several lines of research, which we briefly summarize.

Classic Steiner problems

A basic problem in graph theory is finding the *shortest path* between two nodes; this problem is efficiently solved using, for example, Dijkstra's algorithm.

A natural extension of this is the Steiner Tree problem: given a weighted undirected graph $G = (V, E)$ and a set of terminals $T \subseteq V$, find a minimum-weight subtree that connects all the nodes in T. A further generalization is Steiner Forest: given $G = (V, E)$ and a set of demand pairs $D \subseteq V \times V$, find a subgraph that connects each pair in D. Currently the best known approximation algorithms give a ratio of 1.39 for Steiner Tree [17] and 2 for Steiner Forest [18]. These problems are known to be NP-hard to approximate to within some small constant [19].

For directed graphs, we have the Directed Steiner Network (DSN) problem, in which we are given a weighted directed graph $G = (V, E)$ and k demands $(a_1, b_1), \ldots, (a_k, b_k) \in V \times V$, and must find a minimum-weight sub-graph in which each a_i has a path to b_i. When k is fixed, DSN admits a polynomial-time exact algorithm [20]. For general k, the best known approximation algorithms have ratio $O(k^{1/2+\epsilon})$ for any fixed $\epsilon > 0$ [21, 22]. On the complexity side, Dodis and Khanna [23] ruled out a polynomial-time $O(2^{\log^{1-\epsilon} n})$-approximation for this problem unless NP has quasipolynomial-time algorithms.[2] An important special case of DSN is Directed Steiner Tree, in which all demands have the form (r, b_i) for some root node r. This problem has an $O(k^{\epsilon})$-approximation scheme [24] and a lower bound of $\Omega(\log^{2-\epsilon} n)$ [25].

Finally, a Steiner variant that has found extensive use in computational biology is the Prize-Collecting Steiner Tree problem, in which the input contains a weighted undirected graph $G = (V, E)$ and penalty function $p : V \to \mathbb{R}_{\geq 0}$; the goal is to find a subtree which simultaneously minimizes the weights of the edges in the tree and the penalties paid for nodes not included within the tree, i.e. $\text{cost}(T) := \sum_{e \in T} w(e) + \sum_{v \notin T} p(v)$. For this problem, an approximation algorithm with ratio 1.967 is known [26].

Condition Steiner problems

In this paper, we generalize the Shortest Path, Steiner Tree, Steiner Forest, Directed Steiner Network, and Prize-Collecting Steiner Tree problems to the multi-condition setting. In this setting, we have a set of *conditions* $[C] := \{1, \ldots, C\}$, and are given a graph for each condition.

Our main object of study is the natural generalization of Steiner Forest (in the undirected case) and Directed Steiner Network (in the directed case), which we call Condition Steiner Network:

Definition 1 (*Condition Steiner Network (CSN)*) We are given the following inputs:

1. A sequence of undirected graphs $G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_C = (V, E_C)$, one for each *condition* $c \in [C]$. Each edge e in the underlying edge set $E := \bigcup_c E_c$ has a weight $w(e) \geq 0$.
2. A set of k *connectivity demands* $\mathcal{D} \subseteq V \times V \times [C]$. We assume that for every $c \in C$ there exists at least one demand and therefore that $k \geq |C|$.

We call $G = (V, E)$ the *underlying graph*. We say a subgraph $H \subseteq G$ *satisfies* demand $(a, b, c) \in \mathcal{D}$ if H contains an a-b path P along which all edges exist in G_c. The goal is to output a minimum-weight subgraph $H \subseteq G$ that satisfies every demand in $\mathcal{D}$.

Definition 2 (*Directed Condition Steiner Network (DCSN)*) This is the same as CSN except that all the edges

[2] Throughout this paper, $n := |V|$ denotes the number of nodes in the relevant graph.

are directed, and a demand (a, b, c) must be satisfied by a directed path from a to b in G_c.

We can also define the analogous generalizations of Shortest Path, (undirected) Steiner Tree, and Prize-Collecting Steiner Tree. We give hardness results and algorithms for these problems by demonstrating reductions to and from CSN and DCSN.

Definition 3 (*Condition Shortest Path (CSP), Directed Condition Shortest Path (DCSP)*) These are the special cases of CSN and DCSN in which the demands are precisely $(a, b, 1), \dots, (a, b, C)$ where $a, b \in V$ are common source and target nodes.

Definition 4 (*Condition Steiner Tree (CST)*) We are given a sequence of undirected graphs $G_1 = (V, E_1), \dots, G_C = (V, E_C)$, a weight $w(e) \geq 0$ on each $e \in E$, and sets of *terminal nodes* $X_1, \dots, X_C \subseteq V$. We say a subgraph $H \subseteq (V, \bigcup_c E_c)$ *satisfies* the terminal set X_c if the nodes in X_c are mutually reachable using edges in H that exist at condition c. The goal is to find a minimum-weight subgraph H that satisfies X_c for every $c \in [C]$.

Definition 5 (*Condition Prize-Collecting Steiner Tree (CPCST)*) We are given a sequence of undirected graph $G_1 = (V, E_1), \dots, G_C = (V, E_C)$, a weight $w(e) \geq 0$ on each $e \in E$, and a penalty $p(v, c) \geq 0$ for each $v \in V, c \in [C]$. The goal is to find a subtree T that minimizes $\sum_{e \in T} w(e) + \sum_{v \notin T, c \in [C]} p(v, c)$.

Finally, in molecular biology applications, it is often the case that all the demands originate from a common root node. To capture this, we define the following special case of DCSN:

Definition 6 (*Single-Source DCSN*) This is the special case of DCSN in which the demands are precisely $(a, b_1, c_1), (a, b_2, c_2), \dots, (a, b_k, c_k)$, for some *root* $a \in V$. We can assume that $c_1 \leq c_2 \leq \dots \leq c_k$.

It is also natural to consider variants of these problems in which nodes (rather than edges) vary across the conditions, or in which both nodes and edges vary. In Problem variants, we show that all three variants are in fact equivalent; thus we focus on the edge-based formulations.

Our results

In this work, we perform a systematic study of the condition Steiner problems defined above, from the standpoint of approximation algorithms—that is, algorithms that return subgraphs whose total weights are not much greater than that of the optimal subgraph—as well as integer linear programming (ILP). Since all of the condition Steiner problems listed in the previous section turn out to be NP-hard (and in fact all of them except Shortest Path are hard even in the classic single-condition setting) we cannot hope for algorithms that find optimal solutions and run in polynomial time.

First, in Hardness of condition Steiner problems, we show a series of strong negative results, starting with (directed and undirected) Condition Steiner Network:

Theorem 1 (Main Theorem) *CSN and DCSN are NP-hard to approximate to a factor of $C - \epsilon$ as well as $k - \epsilon$ for every fixed $k \geq 2$ and every constant $\epsilon > 0$. For DCSN, this holds even when the underlying graph is acyclic.*

Thus the best approximation ratio one can hope for is C or k; the latter upper bound is easily achieved by the trivial "union of shortest paths" algorithm: for each demand (a, b, c), compute the shortest a-b path at condition c; then take the union of these k paths. This contrasts with the classic Steiner Network problems, which have nontrivial approximation algorithms and efficient fixed-parameter algorithms.

Next, we show similar hardness results for the other three condition Steiner problems. This is achieved by a series of simple reductions from CSN and DCSN.

Theorem 2 *Condition Shortest Path, Directed Condition Shortest Path, Condition Steiner Tree, and Condition Prize-Collecting Steiner Tree are all NP-hard to approximate to a factor of $C - \epsilon$ for every fixed $C \geq 2$ and $\epsilon > 0$.*

Note that each of these condition Steiner problems can be naively approximated by applying the best known algorithm for the classic version of that problem in each graph in the input, then taking the union of all those subgraphs. If the corresponding classic Steiner problem can be approximated to a factor of α, then this process gives an $\alpha \cdot C$-approximation for the condition version. Thus using known constant-factor approximation algorithms, each of the condition problems in Theorem 2 has an $O(C)$-approximation algorithm. Our result shows that in the worst case, one cannot do much better.

While these results provide a somewhat pessimistic view, the proofs rely on the assumption that the edge sets in the input networks (that is, $E_1, \dots, E_C$) do not necessarily bear any relationship to one another. In Monotonic special cases, we move beyond this worst-case assumption by studying a broad class of special cases in which the conditions are *monotonic*: if an edge e exists in some graph G_c, then it exists in all the subsequent graphs $G_{c'}, c' \geq c$. In other words, each graph in the input is a subgraph of the next. For these problems, we prove the following two theorems:

Theorem 3 *Monotonic CSN has a polynomial-time $O(\log k)$-approximation algorithm. It has no $\Omega(\log\log n)$-approximation algorithm unless* $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log\log\log n})$.

In the directed case, for monotonic DCSN with a single source (that is, every demand is of the form (*r*, *b*, *c*) for a common root node *r*), we show the following:

Theorem 4 *Monotonic Single-Source DCSN has a polynomial-time $O(k^\epsilon)$-approximation algorithm for every $\epsilon > 0$. It has no $\Omega(\log^{2-\epsilon} n)$-approximation algorithm unless* $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathsf{polylog}(n)})$.

These bounds are proved via approximation-preserving reductions to and from classic Steiner problems, namely Priority Steiner Tree and Directed Steiner Tree. Conceptually, this demonstrates that imposing the monotonicity requirement makes the condition Steiner problems much closer to their classic counterparts, allowing us to obtain algorithms with substantially better approximation guarantees.

Finally in application to protein–protein interaction networks, we show how to model various condition Steiner problems as integer linear programs (ILPs). In experiments on real-world inputs derived from the human PPI network, we find that these ILPs are capable of reaching optimal solutions in a reasonable amount of time.

Table 1 summarizes our results, emphasizing how the known upper and lower bounds change when going from the classic Steiner setting to the condition Steiner setting.

Preliminaries

Note that the formulations of CSN and DCSN in the introduction involved a fixed vertex set; only the edges change over the conditions. It is also natural to formulate the Condition Steiner Network problem with nodes changing over condition, or both nodes and edges. However by the following proposition, it is no loss of generality to discuss only the edge-condition variant.

Proposition 1 The edge, node, and node-and-edge variants of CSN are mutually polynomial-time reducible via strict reductions (i.e. preserving the approximation ratio exactly). Similarly all three variants of DCSN are mutually strictly reducible.

We defer the precise definitions of the other two variants, as well as the proof of this proposition, to Problem variants.

In this edge-condition setting, it makes sense to define certain set operations on graphs, which will be of use in our proofs. To that end, let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two graphs on the same vertex set. Their union, $G_1 \cup G_2$, is defined as $(V, E_1 \cup E_2)$. Their intersection, $G_1 \cap G_2$, is defined as $(V, E_1 \cap E_2)$. Subset relations are defined analogously; for example, if $E_1 \subseteq E_2$, then we say that $G_1 \subseteq G_2$.

Next we state the Label Cover problem, which is the starting point of one of our reductions to CSN.

Definition 7 (*Label Cover (LC)*) An instance of this problem consists of a bipartite graph $G = (U, V, E)$ and a set of possible labels Σ. The input also includes, for each edge $(u, v) \in E$, *projection functions* $\pi_u^{(u,v)} : \Sigma \to C$ and $\pi_v^{(u,v)} : \Sigma \to C$, where C is a common set of colors; $\Pi = \{\pi_v^e : e \in E, v \in e\}$ is the set of all such functions. A labeling of G is a function $\phi : U \cup V \to \Sigma$ assigning each node a label. We say a labeling ϕ *satisfies* an edge $(u, v) \in E$, or (u, v) is *consistent* under ϕ, if $\pi_u^{(u,v)}(\phi(u)) = \pi_v^{(u,v)}(\phi(v))$. The task is to find a labeling that satisfies as many edges as possible.

This problem was first defined in [28]. It has the following gap hardness, as shown by Arora et al. [29] and Raz [30].

Theorem 5 *For every $\epsilon > 0$, there is a constant $|\Sigma|$ such that the following promise problem is NP-hard: Given a Label Cover instance (G, Σ, Π), distinguish between the following cases*:

- *(YES instance) There exists a* total labeling *of G; i.e. a labeling that satisfies every edge.*
- *(NO instance) There does not exist a labeling of G that satisfies more than $\epsilon|E|$ edges.*

In Hardness of condition Steiner problems, we use Label Cover to show $(2 - \epsilon)$-hardness for 2-CSN and 2-DCSN; that is, when there are only two demands. To prove our main result however, we will actually need a generalization of Label Cover to partite hypergraphs, called *k*-Partite Hypergraph Label Cover. Out of space considerations we defer the statement of this problem and its gap hardness to Proof of inapproximability for general *C* and *k*, where the $(2 - \epsilon)$-hardness result is generalized to show $(C - \epsilon)$-hardness and $(k - \epsilon)$-hardness for general number of conditions *C* and demands *k*.

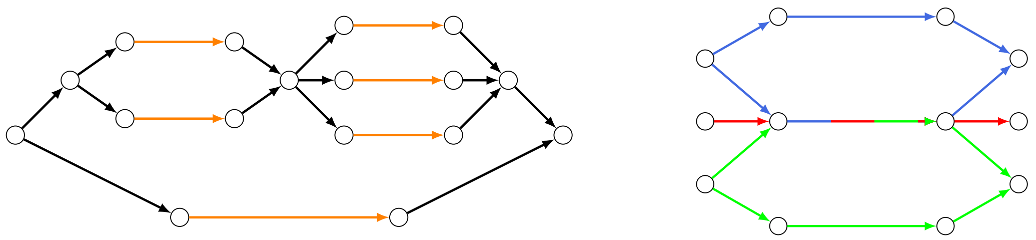


Fig. 2 (Left) A bundle whose upper strand is a chain of two bundles; the lower strand is a simple strand. Contact edges are orange. (Right) Three bundles (blue, green, red indicate different conditions), with one strand from each merged together

Hardness of condition Steiner problems

Overview of the reduction

Here we outline our strategy for reducing Label Cover to the condition Steiner problems. First, we reduce to the CSN problem restricted to having only $C = 2$ conditions and $k = 2$ demands; we call this problem 2-CSN. The directed problem 2-DCSN is defined analogously. Later, we obtain similar hardness for CSN with more conditions or demands by using the same ideas, but reducing from k-Partite Hypergraph Label Cover.

Consider the nodes $u_1, \ldots, u_{|U|}$ on the "left" side of the LC instance. We build, for each u_i, a gadget (which is a small sub-graph in the Steiner instance) consisting of multiple parallel directed paths from a source to a sink—one path for each possible label for u_i. We then chain together these gadgets, so that the sink of u_1's gadget is the source of u_2's gadget, and so forth. Finally we create a connectivity demand from the source of u_1's gadget to the sink of $u_{|U|}$'s gadget, so that a solution to the Steiner instance must have a path from u_1's gadget, through all the other gadgets, and finally ending at $u_{|U|}$'s gadget. This path, depending on which of the parallel paths it takes through each gadget, induces a labeling of the left side of the Label Cover instance. We build an analogous chain of gadgets for the nodes on the right side of the Label Cover instance.

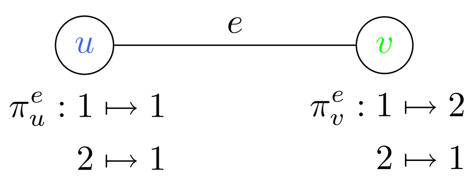
The last piece of the construction is to ensure that the Steiner instance has a low-cost solution if and only if the Label Cover instance has a consistent labeling. This is accomplished by setting all the u_i gadgets to exist only at condition 1 (i.e. in frame G_1), setting the v_j gadgets to exist only in G_2, and then merging certain edges from the u_i-gadgets with edges from the v_j -gadgets, replacing them with a single, shared edge that exists in both frames. Intuitively, the edges we merge are from paths that correspond to labels that satisfy the Label Cover edge constraints. The result is that a YES instance of Label Cover (i.e. one with a total labeling) will enable a high degree of overlap between paths in the Steiner instance, so that there is a very low-cost solution. On the other hand, a NO instance of LC will not result in much overlap between the Steiner gadgets, so every solution will be costly.

Let us define some of the building blocks of the reduction we just sketched:

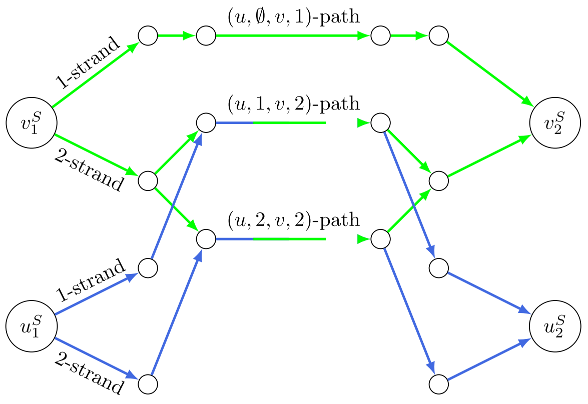
- A *simple strand* is a directed path of the form $b_1 \to c_1 \to c_2 \to b_2$.
- In a simple strand, we say that (c_1, c_2) is the *contact edge*. Contact edges have weight 1; all other edges in our construction have zero weight.
- A *bundle* is a graph gadget consisting of a source node b_1, sink node b_2, and parallel, disjoint *strands* from b_1 to b_2.
- A *chain* of bundles is a sequence of bundles, with the sink of one bundle serving as the source of another.
- More generally, a *strand* can be made more complicated, by replacing a contact edge with another bundle (or even a chain of them). In this way, bundles can be nested, as shown in Fig. 2.
- We can *merge* two or more simple strands from different bundles by setting their contact edges to be the same edge, and making that edge existent at the union of all conditions when the original edges existed (Fig. 2).

Before formally giving the reduction, we illustrate a simple example of its construction.

Example 1 Consider a toy Label Cover instance whose bipartite graph is a single edge, label set is $\Sigma = \{1, 2\}$, color set is $C = \{1, 2\}$, and projection functions are shown:



Our reduction outputs this corresponding 2-CSN instance:



G_1 comprises the set of blue edges; G_2 is green. The demands are $(u_1^S, u_2^S, 1)$ and $(v_1^S, v_2^S, 2)$. For the Label Cover node u, G_1 (the blue sub-graph) consists of two strands, one for each possible label. For the Label Cover node v, G_2 (green sub-graph) consists of one simple strand for the label '1', and a bundle for label '2', which branches out into two simple strands, one for each agreeing labeling of u. Finally, strands (more precisely, their contact edges) whose labels map to the same color are merged.

The input is a YES instance of Label Cover whose optimal labelings (u gets either label 1 or 2, v gets label 2) correspond to 2-CSN solutions of cost 1 (both G_1 and G_2 contain the $(u, 1, v, 2)$-path, and both contain the $(u, 2, v, 2)$-path). If this were a NO instance and edge e could not be satisfied, then the resulting 2-CSN subgraphs G_1 and G_2 would have no overlap.

Inapproximability for two demands

We now formalize the reduction in the case of two conditions and two demands; later, we extend this to general C and k.

Theorem 6 *2-CSN and 2-DCSN are NP-hard to approximate to within a factor of $2 - \epsilon$ for every constant $\epsilon > 0$. For 2-DCSN, this holds even when the underlying graph is acyclic.*

Proof Fix any desired $\epsilon > 0$. We describe a reduction from Label Cover (LC) with any parameter $\varepsilon < \epsilon$ (that is, in the case of a NO instance, no labeling satisfies more than an ε-fraction of edges) to 2-DCSN with an acyclic graph. Given the LC instance $(G = (U, V, E), \Sigma, \Pi)$, construct a 2-DCSN instance $(\mathcal{G} = (G_1, G_2)$, along with two connectivity demands) as follows. Create nodes $u_1^S, \ldots, u_{|U|+1}^S$ and $v_1^S, \ldots, v_{|V|+1}^S$. Let there be a bundle from each u_i^S to u_{i+1}^S; we call this the *u_i-bundle*, since a choice of path from u_i^S to u_{i+1}^S in $\mathcal{G}$ will indicate a labeling of u_i in G.

The u_i-bundle has a strand for each possible label $\ell \in \Sigma$. Each of these *ℓ-strands* consists of a chain of bundles—one for each edge $(u_i, v) \in E$. Finally, each such *(u_i, ℓ, v)-bundle* has a simple strand for each label $r \in \Sigma$ such that $\pi_{u_i}^{(u_i,v)}(\ell) = \pi_v^{(u_i,v)}(r)$; call this the *$(u_i, \ell, v, r)$-path*. In other words, there is ultimately a simple strand for each possible labeling of u_i's neighbor v such that the two nodes are in agreement under their mutual edge constraint. If there are no such consistent labels r, then the (u_i, ℓ, v)-bundle consists of just one simple strand, which is not associated with any r. Note that every minimal $u_1^S \to u_{|U|+1}^S$ path (that is, one that proceeds from one bundle to the next) has total weight exactly $|E|$.

Similarly, create a v_j-bundle from each v_j^S to v_{j+1}^S, whose r-strands (for $r \in \Sigma$) are each a chain of bundles, one for each $(u, v_j) \in E$. Each (u, r, v_j)-bundle has a (u, ℓ, v_j, r)-path for each agreeing labeling ℓ of the neighbor u, or a simple strand if there are no such labelings.

Set all the edges in the u_i-bundles to exist in G_1 only. Similarly the v_j-bundles exist solely in G_2. Now, for each (u, ℓ, v, r)-path in G_1, merge it with the (u, ℓ, v, r)-path in G_2, if it exists. The demands are $\mathcal{D} = \left\{ \left(u_1^S, u_{|U|+1}^S, 1\right), \left(v_1^S, v_{|V|+1}^S, 2\right) \right\}$.

We now analyze the reduction. The main idea is that any $u_i^S \to u_{i+1}^S$ path induces a labeling of u_i; thus the demand $\left(u_1^S, u_{|U|+1}^S, 1\right)$ ensures that any 2-DCSN solution indicates a labeling of all of U. Similarly, $\left(v_1^S, v_{|V|+1}^S, 2\right)$ forces an induced labeling of V. In the case of a YES instance of Label Cover, these two connectivity demands can be satisfied by taking two paths with a large amount of overlap, resulting in a low-cost 2-DCSN solution. In contrast when we start with a NO instance of Label Cover, any two paths we can choose to satisfy the 2-DCSN demands will be almost completely disjoint, resulting in a costly solution. We now fill in the details.

Suppose the Label Cover instance is a YES instance, so that there exists a labeling ℓ_u^* to each $u \in U$, and r_v^* to each $v \in V$, such that for all edges $(u, v) \in E$, $\pi_u^{(u,v)}(\ell_u^*) = \pi_v^{(u,v)}(r_v^*)$. The following is an optimal solution $\mathcal{H}^*$ to the constructed 2-DCSN instance:

- To satisfy the demand at condition 1, for each u-bundle, take a path through the ℓ_u^*-strand. In particular for each (u, ℓ_u^*, v)-bundle in that strand, traverse the (u, ℓ_u^*, v, r_v^*)-path.

- To satisfy the demand at condition 2, for each v-bundle, take a path through the r_v^*-strand. In particular for each (u, r_v^*, v)-bundle in that strand, traverse the (u, ℓ_u^*, v, r_v^*)-path.

In tallying the total edge cost, $\mathcal{H}^* \cap G_1$ (i.e. the subgraph at condition 1) incurs a cost of $|E|$, since one contact edge in $\mathcal{G}$ is encountered for each edge in G. $\mathcal{H}^* \cap G_2$ accounts for no additional cost, since all contact edges correspond to a label which agrees with some neighbor's label, and hence were merged with the agreeing contact edge in $\mathcal{H}^* \cap G_1$. Clearly a solution of cost $|E|$ is the best possible, since every $u_1^S \to u_{|U|+1}^S$ path in G_1 (and every $v_1^S \to v_{|V|+1}^S$ path in G_2) contains at least $|E|$ contact edges.

Conversely suppose we started with a NO instance of Label Cover, so that for any labeling ℓ_u^* to u and r_v^* to v, for at least $(1-\varepsilon)|E|$ of the edges $(u,v) \in E$, we have $\pi_u^{(u,v)}(\ell_u^*) \neq \pi_v^{(u,v)}(r_v^*)$. By definition, any solution to the constructed 2-DCSN instance contains a simple $u_1^S \to u_{|U|+1}^S$ path $P_1 \in G_1$ and a simple $v_1^S \to v_{|V|+1}^S$ path $P_2 \in G_2$. P_1 alone incurs a cost of exactly $|E|$, since one contact edge in $\mathcal{G}$ is traversed for each edge in G. However, P_1 and P_2 share at most $\varepsilon|E|$ contact edges (otherwise, by the merging process, this implies that more than $\varepsilon|E|$ edges could be consistently labeled, which is a contradiction). Thus the solution has a total cost of at least $(2-\varepsilon)|E|$.

It is thus NP-hard to distinguish between an instance with a solution of cost $|E|$, and an instance for which every solution has cost at least $(2-\varepsilon)|E|$. Thus a polynomial-time algorithm for 2-DCSN with approximation ratio $2-\epsilon$ can be used to decide Label Cover (with parameter ε) by running it on the output of the aforementioned reduction. If the estimated objective value is at most $(2-\varepsilon)|E|$ (and thus strictly less than $(2-\epsilon)|E|$) output YES; otherwise output NO. In other words, 2-DCSN is NP-hard to approximate to within a factor of $2-\epsilon$.

To complete the proof, observe that the underlying directed graph we constructed is acyclic, as every edge points "to the right" as in Example 1. Hence 2-DCSN is NP-hard to approximate to within a factor of $2-\epsilon$ for every $\epsilon > 0$, even on acyclic graphs. Finally, note that the same analysis holds for 2-CSN, by simply making every edge undirected; however in this case the graph is clearly not acyclic. □

Inapproximability for general *C* and *k*

Theorem 1 (Main Theorem) *CSN and DCSN are NP-hard to approximate to a factor of $C-\epsilon$ as well as $k-\epsilon$ for every fixed $k \geq 2$ and every constant $\epsilon > 0$. For DCSN, this holds even when the underlying graph is acyclic.*

Proof We perform a reduction from k-Partite Hypergraph Label Cover, a generalization of Label Cover to hypergraphs, to CSN, or DCSN with an acyclic graph. Using the same ideas as in the $C = k = 2$ case, we design k demands composed of parallel paths corresponding to labelings, and merge edges so that a good global labeling corresponds to a large overlap between those paths. The full proof is left to Proof of inapproximability for general C and k. □

Note that a k-approximation algorithm is to simply choose $\mathcal{H} = \bigcup_{c_i} \tilde{P_{c_i}}$, where $\tilde{P_{c_i}}$ is the shortest $a_{c_i} \to b_{c_i}$ path in G_{c_i} for demands $\mathcal{D} = \{(a,b,c_i) : c_i \in [C]\}$. Thus by Theorem 1, essentially no better approximation is possible in terms of k alone. In contrast, most classic Steiner problems have good approximation algorithms [21, 22, 24, 25], or are even exactly solvable for constant k [20].

Inapproximability for Steiner variants

We take advantage of our previous hardness of approximation results in Theorem 1 and show, via a series of reductions, that CSP, CSN, and CPCST are also hard to approximate.

Theorem 2 *Condition Shortest Path, Directed Condition Shortest Path, Condition Steiner Tree, and Condition Prize-Collecting Steiner Tree are all NP-hard to approximate to a factor of $C-\epsilon$ for every fixed $C \geq 2$ and $\epsilon > 0$.*

Proof We first reduce from CSN to CSP (and DCSN to DCSP). Suppose we are given an instance of CSN with graph sequence $\mathcal{G} = (G_1, \ldots, G_C)$, underlying graph $G = (V,E)$, and demands $\mathcal{D} = \{(a_i, b_i, c_i) : i \in [k]\}$. We build a new instance $\left(\mathcal{G}' = (G'_1, \ldots, G'_k), G' = (V', E'), \mathcal{D}'\right)$ as follows.

Initialize G' to G. Add to G' the new nodes a and b, which exist at all conditions G'_i. For all $e \in E$ and $i \in [k]$, if $e \in G_{c_i}$, then let e exist in G'_i as well. For each $(a_i, b_i, c_i) \in \mathcal{D}$,

1. Create new nodes x_i, y_i. Create zero-weight edges (a, x_i), (x_i, a_i), (b_i, y_i), and (y_i, b).
2. Let (a, x_i), (x_i, a_i), (b_i, y_i), and (y_i, b) exist only in frame G'_i.

Lastly, the demands are $\mathcal{D}' = \{(a, b, i) : i \in [k]\}$.

Given a solution $H' \subseteq G'$ containing an $a \to b$ path at every condition $i \in [k]$, we can simply exclude nodes a, b, $\{x_i\}$, and $\{y_i\}$ to obtain a solution $H \subseteq G$ to the original instance, which contains an $a_i \to b_i$ path in G_{c_i} for all $i \in [k]$, and has the same cost. The converse is also true by including these nodes.

Observe that essentially the same procedure shows that DCSN reduces to DCSP; simply ensure that the edges added by the reduction are directed rather than undirected.

Next, we reduce CSP to CST. Suppose we are given an instance of CSP with graph sequence $\mathcal{G} = (G_1, \ldots, G_C)$, underlying graph $G = (V, E)$, and demands $\mathcal{D} = \{(a, b, i) : i \in [C]\}$. We build a new instance of CST as follows: $\left(\mathcal{G}' = (G'_1, \ldots, G'_C), G' = (V', E'), \mathcal{X} = (X_1, \ldots, X_C)\right)$. Set $\mathcal{G}'$ to $\mathcal{G}$, and G' to G. Take the set of terminals in each condition to be $X_i = \{a, b\}$. We note that a solution $H' \subseteq G'$ to the CST instance is trivially a solution the CSP instance with the same cost, and vice-versa.

Finally, we reduce CST to CPCST. We do this by making an appropriate assignment of the penalties $p(v, c)$. Suppose we are given an instance of CST with graph sequence $\mathcal{G} = (G_1, \ldots, G_C)$, underlying graph $G = (V, E)$, and terminal sets $\mathcal{X} = (X_1, \ldots, X_C)$. We build a new instance of CPCST, $\left(\mathcal{G}' = (G'_1, \ldots, G'_C), G' = (V', E'), p(v, c)\right)$. In particular, set $\mathcal{G}'$ to $\mathcal{G}$, and G' to G. Set $p(v, c)$ as follows:

$$p(v, c) = \begin{cases} \infty, & v \in X_c \\ 0, & \text{otherwise} \end{cases}$$

Consider any solution $H \subseteq G$ to the original CST instance. Since H spans the terminals $X_1, \ldots, X_c$ (thus avoiding any infinite penalties), and since the non-terminal vertices have zero cost, the overall cost of H remains the same cost in the constructed CPCST instance. Conversely, suppose we are given a solution $H' \subseteq G'$ to the constructed CPCST instance. If the cost of H' is ∞, then H' does not span all the X_c's simultaneously, and thus H' is not a possible solution for the CST instance. On the other hand if H' has finite cost, then H' is also a solution for the CST instance, with the same cost.

To summarize: in the first reduction from CSN to CSP, the number of demands, k, in the CSN instance is the same as the number of the conditions, C, in the CSP instance; we conclude that CSP is NP-hard to approximate to a factor of $C - \epsilon$ for every fixed $C \geq 2$ and $\epsilon > 0$. Since C remains the same in the two subsequent reductions, we also have that CST and CPCST are NP-hard to approximate to a factor of $C - \epsilon$. □

Monotonic special cases

In light of the strong lower bounds in the previous theorems, in this section we consider more tractable special cases of the condition Steiner problems. A natural restriction is that the changes over conditions are *monotonic*:

Definition 8 (*Monotonic {CSN, DCSN, CSP, DCSP, CST, CPCST}*) In this special case (of any of the condition Steiner problems), we have that for each $e \in E$ and $c \in [C]$, if $e \in G_c$, then $e \in G_{c'}$ for all $c' \geq c$.

We now examine the effect of monotonicity on the complexity of the condition Steiner problems.

Monotonicity in the undirected case

In the undirected case, we show that monotonicity has a simple effect: it makes CSN equivalent to the following well-studied problem:

Definition 9 (*Priority Steiner Tree* [31]) The input is a weighted undirected multigraph $G = (V, E, w)$, a *priority level* $p(e)$ for each $e \in E$, and a set of k demands (a_i, b_i), each with priority $p(a_i, b_i)$. The output is a minimum-weight forest $F \subseteq G$ that contains, between each a_i and b_i, a path in which every edge e has priority $p(e) \leq p(a_i, b_i)$.

Priority Steiner Tree was introduced by Charikar, Naor, and Schieber [31], who gave a $O(\log k)$ approximation algorithm. Moreover, it cannot be approximated to within a factor of $\Omega(\log \log n)$ assuming NP $\notin \mathsf{DTIME}(n^{\log \log \log n})$ [32]. We now show that the same bounds apply to Monotonic CSN, by showing that the two problems are essentially equivalent from an approximation standpoint.

Lemma 1 *Fix any function $f : \mathbb{Z}_{>0} \to \mathbb{R}_{>0}$. If either Priority Steiner Tree or Monotonic CSN can be approximated to a factor of f(k) in polynomial time, then so can the other.*

Proof We transform an instance of Priority Steiner Tree into an instance of Monotonic CSN as follows: the set of priorities becomes the set of conditions; if an edge e has priority $p(e)$, it now exists at all conditions $t \geq p(e)$; if a demand (a_i, b_i) has priority $p(a_i, b_i)$, it now becomes $(a_i, b_i, p(a_i, b_i))$. If there are parallel multiedges, break up each such edge into two edges of half the original weight, joined by a new node. Given a solution $H \subseteq G$ to this CSN instance, contracting any edges that were originally multiedges gives a Priority Steiner Tree solution of the same cost. This reduction also works in the opposite direction (in this case there are no multiedges), which shows the equivalence. □

Furthermore, the $O(\log k)$ upper bound applies to CST (We note that Monotonic CSP admits a trivial algorithm, namely take the subgraph induced by running Djikstra's Algorithm on G_1).

Lemma 2 *If Monotonic CSN can be approximated to a factor of f(k) for some function f in polynomial time, then Monotonic CST can also be approximated to within f(k) in polynomial time.*

Proof We now show a reduction from CST to CSN. Suppose we are given a CST instance on graphs $\mathcal{G} = (G_1, \ldots, G_C)$ and terminal sets $\mathcal{X} = (X_1, \ldots, X_C)$. Our CSN instance has precisely the same graphs, and has the following demands: for each terminal set X_c, pick any terminal $a \in X_c$ and create a demand (a, b, c) for each $b \neq a \in X_c$. A solution to the original CST instance is a solution to the constructed CSN instance with the same cost, and vice-versa; moreover, if the CST instance is monotonic, then so is the constructed CSN instance. Observe that if the total number of CST terminals is k, then the number of constructed demands is $k - C$, and therefore an $f(k)$-approximation for CSN implies an $f(k - C) \leq f(k)$-approximation for CST, as required. □

Monotonicity in the directed case

In the directed case, we give an approximation-preserving reduction from a single-source special case of DCSN to the Directed Steiner Tree (DST) problem (in fact, we show that they are essentially equivalent from an approximation standpoint), then apply a known algorithm for DST. Recall the definition of Single-Source DCSN:

Definition 6 (*Single-Source DCSN*) This is the special case of DCSN in which the demands are precisely $(a, b_1, c_1), (a, b_2, c_2), \ldots, (a, b_k, c_k)$, for some *root* $a \in V$. We can assume that $c_1 \leq c_2 \leq \cdots \leq c_k$.

Lemma 3 *Fix any function $f : \mathbb{Z}_{>0} \to \mathbb{R}_{>0}$. If either Monotonic Single-Source DCSN or Directed Steiner Tree can be approximated to a factor of f(k) in polynomial time, then so can the other.*

For the remainder of this section, we refer to Monotonic Single-Source DCSN as simply DCSN. Towards proving the theorem, we now describe a reduction from DCSN to DST. Given a DCSN instance $(G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_C = (V, E_C), \mathcal{D})$ with underlying graph $\mathcal{G} = (V, E)$, we construct a DST instance $(G' = (V', E'), D')$ as follows:

- G' contains a vertex v^i for each $v \in V$ and each $i \in [c_k]$. It contains an edge (u^i, v^i) with weight $w(u, v)$ for each $(u, v) \in E_i$. Additionally, it contains a zero-weight edge (v^i, v^{i+1}) for each $v \in V$ and each $i \in [c_k]$.
- D' contains a demand $(a^1, b_i^{c_i})$ for each $(a, b_i, c_i) \in \mathcal{D}$.

Now consider the DST instance (G', D').

Lemma 4 *If the DCSN instance $(G_1, \ldots, G_C, \mathcal{D})$ has a solution of cost C^*, then the constructed DST instance (G', D') has a solution of cost at most C^*.*

Proof Let $\mathcal{H} \subseteq \mathcal{G}$ be a DCSN solution having cost C^*. For any edge $(u, v) \in E(\mathcal{H})$, define the *earliest necessary condition* of (u, v) to be the minimum c_i such that removing (u, v) would cause $\mathcal{H}$ not to satisfy demand (a, b_i, c_i). □

Claim 1 *There exists a solution $\mathcal{C} \subseteq \mathcal{H}$ that is a directed tree and has cost at most C^*. Moreover for every path P_i in $\mathcal{C}$ from the root a to some target b_i, as we traverse P_i from a to b_i, the earliest necessary conditions of the edges are non-decreasing.*

Proof of Claim 1 Consider a partition of $\mathcal{H}$ into edge-disjoint sub-graphs $\mathcal{H}_1, \ldots, \mathcal{H}_k$, where $\mathcal{H}_i$ is the sub-graph whose edges have earliest necessary condition c_i.

If there is a directed cycle or parallel paths in the first sub-graph $\mathcal{H}_1$, then there is an edge $e \in E(\mathcal{H}_1)$ whose removal does not cause $\mathcal{H}_1$ to satisfy fewer demands at condition c_1. Moreover by monotonicity, removing e also does not cause $\mathcal{H}$ to satisfy fewer demands at any future conditions. Hence there exists a directed tree $\mathcal{T}_1 \subseteq \mathcal{H}_1$ such that $\mathcal{T}_1 \cup \left(\bigcup_{i=2}^{k} \mathcal{H}_i\right)$ has cost at most C^* and still satisfies $\mathcal{T}$.

Now suppose by induction that for some $j \in [k-1]$, $\bigcup_{i=1}^{j} \mathcal{T}_i$ is a tree such that $\left(\bigcup_{i=1}^{j} \mathcal{T}_i\right) \cup \left(\bigcup_{i=j+1}^{k} \mathcal{H}_i\right)$ has cost at most C^* and satisfies $\mathcal{D}$. Consider the partial solution $\left(\bigcup_{i=1}^{j} \mathcal{T}_i\right) \cup \mathcal{H}_{j+1}$; if this sub-graph is not a directed tree, then there must be an edge $(u, v) \in E(\mathcal{H}_{j+1})$ such that v has another in-edge in the sub-graph. However by monotonicity, (u, v) does not help satisfy any new demands, as v is already reached by some other path from the root. Hence by removing all such redundant edges, we have $\mathcal{T}_{j+1} \subseteq \mathcal{H}_{j+1}$ such that $\left(\bigcup_{i=1}^{j+1} \mathcal{T}_i\right) \cup \left(\bigcup_{i=j+2}^{k} \mathcal{H}_i\right)$ has cost at most C^* and satisfies $\mathcal{D}$, which completes the inductive step.

We conclude that $\mathcal{T} := \bigcup_{i=1}^{k} \mathcal{T}_i \subseteq \mathcal{H}$ is a tree of cost at most C^* satisfying $\mathcal{D}$. Observe also that by construction, as $\mathcal{T}$ is a tree that is iteratively constructed by $\mathcal{T}_i \subseteq \mathcal{H}_i$, $\mathcal{T}$ has the property that if we traverse any $a \to b_i$ path, the earliest necessary conditions of the edges never decrease. □

Now let $\mathcal{T}$ be the DCSN solution guaranteed to exist by Claim 1. Consider the sub-graph $H' \subseteq G'$ formed by adding, for each $(u, v) \in E(\mathcal{T})$, the edge

$(u^c, v^c) \in E'$ where c is the earliest necessary condition of (u, v) in $E(\mathcal{H})$. In addition, for all vertices $v^i \in H'$ where $v^{i+1} \in H'$, add the free edge (v^i, v^{i+1}). Since $w(u^c, v^c) = w(u, v)$ by construction, $\text{cost}(H') \leq \text{cost}(\mathcal{T}) \leq C^*$.

To see that H' is a valid solution, consider any demand $(a^1, b_i^{c_i})$. Recall that $\mathcal{T}$ has a unique $a \to b_i$ path P_i along which the earliest necessary conditions are nondecreasing. We added to H' each of these edges at the level corresponding to its earliest necessary condition; moreover, whenever there are adjacent edges $(u, v), (v, x) \in P_i$ with earliest necessary conditions c and $c' \geq c$ respectively, there exist in H' free edges $(v^t, v^{c+1}), \ldots, (v^{c'-1}, v^{c'})$. Thus H' contains an $a^1 \to b_i^{c_i}$ path, which completes the proof. □

Lemma 5 *If the constructed DST instance (G', D') has a solution of cost C^*, then the original DCSN instance $(G_1, \ldots, G_C, \mathcal{D})$ has a solution of cost at most C^*.*

Proof First note that any DST solution ought to be a tree; let $T' \subseteq G'$ be such a solution of cost C. For each $(u, v) \in G$, T' might as well use at most one edge of the form (u^i, v^i), since if it uses more, it can be improved by using only the one with minimum i, then taking the free edges (v^i, v^{i+1}) as needed. We create a DCSN solution $\mathcal{T} \subseteq \mathcal{G}$ as follows: for each $(u^i, v^i) \in E(T')$, add (u, v) to $\mathcal{T}$. Since $w(u, v) = w(u^i, v^i)$ by design, we have $\text{cost}(\mathcal{T}) \leq \text{cost}(T') \leq C$. Finally, since each $a^1 \to b_i^{t_i}$ path in G' has a corresponding path in $\mathcal{G}$ by construction, $\mathcal{T}$ satisfies all the demands. □

Lemma 3 follows from Lemma 4 and Lemma 5. Finally we can obtain the main result of this subsection:

Theorem 4 *Monotonic Single-Source DCSN has a polynomial-time $O(k^\epsilon)$-approximation algorithm for every $\epsilon > 0$. It has no $\Omega(\log^{2-\epsilon} n)$-approximation algorithm unless* $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathsf{polylog}(n)})$.

Proof The upper bound follows by composing the reduction (from Monotonic Single-Source DCSN to Directed Steiner Tree) with the algorithm of Charikar et al. [24] for Directed Steiner Tree, which achieves ratio $O(k^\epsilon)$ for every $\epsilon > 0$. More precisely they give an $i^2(i-1)k^{1/i}$ -approximation for any integer $i \geq 1$, in time $O(n^i k^{2i})$. The lower bound follows by composing the reduction (in the opposite direction) with a hardness result of Halperin and Krauthgamer [25], who show the same bound for Directed Steiner Tree. A quick note regarding the reduction in the opposite direction: Directed Steiner Tree is a precisely a Monotonic Single-Source DCSN instance with exactly one condition. □

In Explicit algorithm for Monotonic Single-Source DCSN, we show how to modify the algorithm of Charikar et al. to arrive at a simple, explicit algorithm for Monotonic Single-Source DCSN achieving the same guarantee.

Application to protein–protein interaction networks

Methods such as Directed Condition Steiner Network can be key in identifying underlying structure in biological processes. As a result, it is important to assess the runtime feasibility of solving for an optimal solution. We show via simulation on human protein–protein interaction networks, that our algorithm on single-source instances is able to quickly and accurately infer maximum likelihood subgraphs for a certain biological process.

Building the protein–protein interaction network

We represent the human PPI network as a weighted directed graph, where proteins serve as nodes, and interactions serve as edges. The network was formed by aggregating information from four sources of interaction data, including Netpath [33], Phosphosite [34], HPRD [35], and InWeb [36], altogether, covering 16222 nodes and 437888 edges. Edge directions are assigned where these annotations were available (primarily in Phopshosite and NetPath). The remaining edges are represented by two directed edges between the proteins involved. Edge weights were assigned by taking the negative logarithm of the associated confidence score, indicating that finding the optimal Steiner Network would be the same as finding the most confident solution (assuming independence between edges). Confidence data was available for the largest of the data sets (InWeb). For HPRD edges that are not in InWeb, we used the minimum nonzero confidence value by default. For the smaller and highly curated datasets, Phopshosite and NetPath, we used the maximal confidence level.

Solving DCSN to optimality

Definition 6 (*Single-Source DCSN*) This is the special case of DCSN in which the demands are precisely $(a, b_1, c_1), (a, b_2, c_2), \ldots, (a, b_k, c_k)$, for some *root* $a \in V$. We can assume that $c_1 \leq c_2 \leq \cdots \leq c_k$.

We can derive a natural integer linear program for the Single-Source Directed Condition Steiner Network in terms of network flows, with each demand being met by a flow from source to target:

$$\begin{aligned}
&\text{minimize} && \sum_{(u,v)\in E} d_{uv}\cdot w(u,v) \\
&\text{subject to} && \\
&& k_c\cdot d_{uv} &\geq d_{uvc} && \forall c,\ (u,v)\in E_c \\
&& \sum_{(v,w)\in E_c} d_{vwc} &= \sum_{(u,v)\in E_c} d_{uvc}+\delta_{vc} && \forall c,\ v\in V \\
&& d_{uvc} &\in \{0,1,\ldots,k_c\} && \forall c,\ (u,v)\in E_c \\
&& d_{uv} &\in \{0,1\} && \forall (u,v)\in E
\end{aligned}$$

Fig. 3 Integer linear program for Single-Source Condition Steiner Network. $\delta_{vc}=1$ for v at condition c if v is a target at condition c, $-k_c$ for v at condition c if v is the source node at condition c, 0 otherwise

Each variable d_{uvc} denotes the flow through edge (u, v) at condition c, if it exists; each variable d_{uv} denotes whether (u, v) is ultimately in the chosen solution subgraph; k_c denotes the number of demands at condition c . The first constraint ensures that if an edge is used at any condition, it is chosen as part of the solution. The second constraint enforces flow conservation, and hence that the demands are satisfied, at all nodes and all conditions.

We note that DCSN easily reduces DCSP, as outlined in Theorem 2. However, DCSP is a special case of Single-Source DCSN. Therefore, the integer linear program defined above can be applied to any DCSN instance with a transformation of the instance to DCSP (Fig. 3).

Performance analysis of integer linear programming

Given the protein–protein interaction network G, we sample an instance of the node-variant Single-Source DCSN as so[3]:

- Instantiate a source node a.
- Independently sample β nodes reachable from a, for each of the C conditions, giving us $\{b_{1,1},\ldots,b_{\beta,C}\}$.
- For each node $v\in V$, include $v\in V_c$ if v lies on the shortest path from a to one of $\{b_{1,c},..,b_{\beta,c}\}$
- For all other nodes $v\in V$ for all c, include $v\in V_c$ with probability p.

Using a workstation running an Intel Xeon E5-2690 processor and 250 GB of RAM, *optimal solutions* to instances of modest size (generated using the procedure just described) were within reach (Table 2):

We notice that our primary runtime constraint comes from C, the number of conditions. In practice, the number of conditions does not exceed 100.

In addition, we decided to test our DCSN ILP formulation against a simple algorithm of optimizing over each demand independently via shortest path. Theoretically, the shortest path method can perform up to k times worse than DCSN. We note that having zero weight edges complicates the comparison of algorithms' performance on real data. The reason is that we can have the same weight for a large and small networks. Instead, we wanted to also take into account the size of the returned networks. To do that we added a constant weight for every edge. Testing over a sample set of instances generated with parameters $\beta=100$, $C=10$, $p=0.25$, we found that the shortest path method returns a solution on average 1.07 times more costly.

Therefore, we present a model showing preliminary promises of translating and finding **optimal solutions** to real world biological problems with practical runtime.

Table 2 ILP solve times for some random instances generated by our random model using the Gurobi Python Solver package [37]

β	C	p	Time to solve
100	1	1.0	45 s ± 5 s
10	1	0.25	1 m ± 10 s
100	1	0.25	1 m ± 10 s
10	1	0.75	1 m ± 10 s
100	1	0.75	1 m ± 10 s
100	10	1.0	7 m ± 30 s
10	10	0.25	9 m ± 30 s
10	10	0.75	11 m ± 30 s
100	10	0.25	12 m ± 30 s
100	10	0.75	17 m ± 2 m
100	100	1.0	1 h 40 m ± 15 m
10	100	0.75	2 h 30 m ± 12 m
100	100	0.75	4 h ± 40 m

Conclusion and discussion

In this paper we introduced the Condition Steiner Network (CSN) problem and its directed variant, in which the goal is to find a minimal subgraph satisfying a set of k condition-sensitive connectivity demands. We show, in contrast to known results for traditional Steiner problems, that this problem is NP-hard to approximate to a factor of $C-\epsilon$, as well as $k-\epsilon$, for every $C,k\geq 2$ and $\epsilon>0$. We then explored a special case, in which the conditions/graphs satisfy a *monotonicity* property. For such instances we proposed algorithms significantly beating the pessimistic lower bound for the general problem; this was accomplished by reducing the problem to certain traditional Steiner problems. Lastly, we developed and

[3] As previously mentioned, this variant reduces to the edge variant via reduction, and vice versa

applied an integer programming-based exact algorithm on simulated instances built over the human protein–protein interaction network, and reported feasible runtimes for real-world problem instances.

Importantly, along the way we showed implications of these results for CSN on other network connectivity problems that are commonly used in PPI analysis—such as Shortest Path, Steiner Tree, Prize-Collecting Steiner Tree—when conditions are added. We showed that for each of these problems, we cannot guarantee (in polynomial time) a solution with a value below $C - \epsilon$ times the optimal value. These lower bounds are quite strict, in the sense that naively approximating the problem separately in every condition, and taking the union of those solutions, already gives an approximation ratio of $O(C)$. At the same time, by relating the various condition Steiner problems to one another, we also obtained some positive results: the condition versions of Shortest Path and Steiner Tree admit good approximations when the conditions are monotonic. Moreover, all of the condition problems (with the exception of Prize-Collecting Steiner Tree) can be solved using a natural integer programming framework that works well in practice.

Proofs of main theorems

Problem variants

There are several natural ways to formulate the condition Steiner Network problem, depending on whether the edges are changing over condition, or the nodes, or both.

Definition 10 (*Condition Steiner Network (edge variant)*) This is the formulation described in the introduction: the inputs are $G_1 = (V, E_1), \ldots, G_C = (V, E_C)$, $w(\cdot)$, and $\mathcal{D} = \{(a_i, b_i, c_i)\}$. The task is to find a minimum-weight sub-graph $\mathcal{H} \subseteq \mathcal{G}$ that satisfies all of the demands.

Definition 11 (*Condition Steiner Network (node variant)*) Let the underlying graph be $\mathcal{G} = (V, E)$. The inputs are $G_1 = (V_1, E(V_1)), \ldots, G_C = (V_C, E(V_C))$, $w(\cdot)$, and $\mathcal{D}$. Here, $E(V_c) \subseteq E$ denotes the edges induced by $V_c \subseteq V$. A path satisfies a demand at condition t if all edges along that path exist in G_c.

Definition 12 (*Condition Steiner Network (node and edge variant)*) The inputs are precisely $G_1 = (V_1, E_1), \ldots, G_C = (V_C, E_C)$, $w(\cdot)$, and $\mathcal{D}$. This is the same as the node variant except that each E_c can be any subset of $E(V_c)$.

Similarly, define the corresponding directed problem Directed Condition Steiner Network (DCSN) with the same three variants. The only difference is that the edges are directed, and a demand (a, b, c) must be satisfied by a directed $a \to b$ path in G_c.

The following observation enables all our results to apply to all problem variants.

Proposition 2 *The edge, node, and node-and-edge variants of CSN are mutually polynomial-time reducible via strict reductions (i.e. preserving the approximation ratio exactly). Similarly all three variants of DCSN are mutually strictly reducible.*

Proof The following statements shall hold for both undirected and directed versions. Clearly the node-and-edge variant generalizes the other two. It suffices to show two more directions:

(Node-and-edge reduces to node) Let (u, v) be an edge existent at a set of conditions $\tau(u, v)$, whose endpoints exist at conditions $\tau(u)$ and $\tau(v)$. To make this a node-condition instance, create an intermediate node $x_{(u,v)}$ existent at conditions $\tau(u, v)$, an edge $(u, x_{(u,v)})$ with the original weight $w(u, v)$, and an edge $(x_{(u,v)}, v)$ with zero weight. A solution of cost W in the node-and-edge instance corresponds to a node-condition solution of cost W, and vice-versa.

(Node reduces to edge) Let (u, v) be an edge whose endpoints exist at conditions $\tau(u)$ and $\tau(v)$. To make this an edge-condition instance, let (u, v) exist at conditions $\tau(u, v) := \tau(u) \cap \tau(v)$. Let every node exist at all conditions; let the edges retain their original weights. A solution of cost W in the node-condition instance corresponds to an edge-condition solution of cost W, and vice-versa. □

Proof of inapproximability for general *C* and *k*

Here we prove our main theorem, showing optimal hardness for any number of demands. To do this, we introduce a generalization of Label Cover to partite hypergraphs:

Definition 13 (*k-Partite Hypergraph Label Cover* (*k-PHLC*)) An instance of this problem consists of a k-partite, k-regular hypergraph $G = (V_1, \ldots, V_k, E)$ (that is, each edge contains exactly one vertex from each of the k parts) and a set of possible labels Σ. The input also includes, for each hyperedge $e \in E$, a *projection function* $\pi_v^e : \Sigma \to C$ for each $v \in e$; Π is the set of all such functions. A labeling of G is a function $\phi : \bigcup_{i=1}^{k} V_i \to \Sigma$ assigning each node a label. There are two notions of edge satisfaction under a labeling ϕ:

- ϕ *strongly satisfies* a hyperedge $e = (v_1, \ldots, v_k)$ if the labels of all its vertices are mapped to the same color, i.e. $\pi_{v_i}^e(\phi(v_i)) = \pi_{v_j}^e(\phi(v_j))$ for all $i, j \in [k]$.

- ϕ *weakly satisfies* a hyperedge $e = (v_1, \ldots, v_k)$ if there exists some pair of vertices v_i, v_j whose labels are mapped to the same color, i.e. $\pi^e_{v_i}(\phi(v_i)) = \pi^e_{v_j}(\phi(v_j))$ for some $i \neq j \in [k]$.

The following gap hardness for this problem was shown by Feige [38]:

Theorem 7 *For every $\epsilon > 0$ and every fixed integer $k \geq 2$, there is a constant $|\Sigma|$ such that the following promise problem is* NP*-hard: Given a k-Partite Hypergraph Label Cover instance (G, Σ, Π), distinguish between the following cases*:

- *(YES instance) There exists a labeling of G that strongly satisfies every edge.*
- *(NO instance) Every labeling of G weakly satisfies at most $\epsilon|E|$ edges.*

The proof of $(C - \epsilon)$-hardness and $(k - \epsilon)$-hardness follows the same outline as the $C = k = 2$ case (Theorem 6).

Theorem 8 (Main Theorem) *CSN and DCSN are NP-hard to approximate to a factor of $C - \epsilon$ as well as $k - \epsilon$ for every fixed $k \geq 2$ and every constant $\epsilon > 0$. For DCSN, this holds even when the underlying graph is acyclic.*

Proof Given the k-PHLC instance in the form $(G = (V_1, \ldots, V_k, E), \Sigma, \Pi)$, and letting $v_{c,i}$ denote the i-th node in V_c, construct a DCSN instance $(\mathcal{G} = (G_1, \ldots, G_k)$, along with k demands) as follows. For every $c \in [k]$, create nodes $v^S_{c,1}, \ldots, v^S_{t,|V_c|+1}$. Create a $v_{c,i}$*-bundle* from each $v^S_{c,i}$ to $v^S_{c,i+1}$, whose ℓ-strands (for $\ell \in \Sigma$) are each a chain of bundles, one for each incident hyperedge $e = (v_{1,i_1}, \ldots, v_{c,i}, \ldots, v_{k,i_k}) \in E$. Each $(v_{1,i_1}, \ldots, v_{c,i}, \ldots, v_{k,i_k})$-bundle has a $(v_{1,i_1}, \ell_1, \ldots, v_{c,i}, \ell_c, \ldots, v_{k,i_k}, \ell_k)$-path for each agreeing combination of labels—that is, every k-tuple $(\ell_1, \ldots, \ell_c, \ldots, \ell_k)$ such that: $\pi^e_{v_{1,i_1}}(\ell_1) = \cdots = \pi^e_{v_{c,i}}(\ell_c) = \cdots = \pi^e_{v_{k,i_k}}(\ell_k)$, where e is the shared edge. If there are no such combinations, then the e-bundle is a single simple strand.

For $c \in [k]$, set all the edges in the $v_{c,i}$-bundles to exist in G_c only. Now, for each $(v_{1,i_1}, \ell_1, \ldots, v_{k,i_k}, \ell_k)$, merge together the $(v_{1,i_1}, \ell_1, \ldots, v_{k,i_k}, \ell_k)$-paths across all G_c that have such a strand. Finally, the connectivity demands are $\mathcal{D} = \left\{ \left(v^S_{c,1}, v^S_{c,|V_c|+1}, c \right) : c \in [k] \right\}$.

The analysis follows the $k = 2$ case. Suppose we have a YES instance of k-PHLC, with optimal labeling ℓ^*_v to each node $v \in \bigcup_{t=1}^k V_c$. Then an optimal solution $\mathcal{H}^*$ to the constructed DCSN instance is to traverse, at each condition c and for each $v_{c,i}$-bundle, the path through the $\ell^*_{v_{c,i}}$-strand. In particular for each $(v_{1,i_1}, \ldots, v_{k,i_k})$-bundle in that strand, traverse the $(v_{1,i_1}, \ell^*_1, \ldots, v_{k,i_k}, \ell^*_k)$-path.

In tallying the total edge cost, $\mathcal{H}^* \cap G_1$ (the sub-graph at condition 1) incurs a cost of $|E|$, one for each contact edge. The sub-graphs of $\mathcal{H}^*$ at conditions $2, \ldots, k$ account for no additional cost, since all contact edges correspond to a label which agrees with all its neighbors' labels, and hence were merged with the agreeing contact edges in the other sub-graphs.

Conversely suppose we have a NO instance of k-PHLC, so that for any labeling ℓ^*_v, for at least $(1 - \epsilon)|E|$ hyperedges e, the projection functions of all nodes in e disagree. By definition, any solution to the constructed DCSN instance contains a simple $v^S_{t,1} \rightarrow v^S_{t,|V_c|+1}$ path P_c at each condition c. As before, P_1 alone incurs a cost of exactly $|E|$. However, at least $(1 - \epsilon)|E|$ of the hyperedges in G cannot be weakly satisfied; for these hyperedges e, for every pair of neighbors $v_{c,i_c}, v_{c',i_{c'}} \in e$, there is no path through the e-bundle in v_{t,i_c}'s $\ell^*_{v_{c,i_c}}$-strand that is merged with any of the paths through the e-bundle in $v_{c',i_{c'}}$'s $\ell^*_{v_{c,i_{c'}}}$-strand (for otherwise, it would indicate a labeling that weakly satisfies e in the k-PHLC instance). Therefore paths $P_2, \ldots, P_k$ each contribute at least $(1 - \epsilon)|E|$ additional cost, so the solution has total cost at least $(1 - \epsilon)|E| \cdot k$.

It follows from the gap between the YES and NO cases that DCSN is NP-hard to approximate to within a factor of $k - \epsilon$ for every constant $\epsilon > 0$; and since $C = k$ in our construction, it is also NP-hard for $C - \epsilon$. Moreover since The directed condition graph we constructed is acyclic, this result holds even on DAGs. As before, the same analysis holds for the undirected problem CSN by undirecting the edges. □

Explicit algorithm for Monotonic Single-Source DCSN

We provide a modified version of the approximation algorithm presented in Charikar et al. [24] for Directed Steiner Tree (DST), which achieves the same approximation ratio for our problem Monotonic Single-Source DCSN.

We provide a similar explanation as of that presented in Charikar et al. Consider a trivial approximation algorithm, where we take the shortest path from the source to each individual target. Consider the example where there are edges of cost $C - \epsilon$ to each target, and a vertex v with distance C from the source, and with distance 0 to each target. In such a case, this trivial approximation algorithm will achieve only an $\Omega(k)$-approximation. Consider instead an algorithm which found, from the root, an intermediary vertex v, which was connected to all the targets via shortest path. In the case of the above example, this would find us the optimal sub-graph. The algorithm below generalizes this process, by progressively finding optimal substructures with good cost relative to the number of targets connected. We show that this algorithm provides a good approximation ratio.

Definition 14 (*Metric closure of a condition graph*) For a directed condition graph $\mathcal{G} = (G_1 = (V, E_1), G_2 = (V, E_2), \ldots, G_C = (V, E_C))$, define its *metric closure* to be $\tilde{G} = (V, E, \tilde{w})$ where $E = \bigcup_c E_c$ and $\tilde{w}(u, v, c)$ is the length of the shortest $u \to v$ path in G_c (note that in contrast with w, $\tilde{w}$ takes three arguments).

Definition 15 ($V(T)$) Let T be a tree with root r. We say a demand of the form (r, b, c) is *satisfied* by T if there is a path in T from r to b at condition c. $V(T)$ is then the set of demands satisfied by T.

Definition 16 ($D(T)$) The *density* of a tree T is $D(T) = \frac{cost(T)}{|V(T)|}$, where $cost(T)$ is the sum of edge weights of T.

```
function $A_i$(transitive closure $G = (V, E, w)$, $r$, $c$, $k$, $\mathcal{D} \subseteq V \times [C]$)
    if $(r, b_i, c_i)$ does not exist for least $k(b_i, c_i) \in \mathcal{D}, c_i \geq c$ then return NO SOLUTION
    $T \leftarrow \emptyset$
    while $k > 0$ do
        $T_{best} \leftarrow \emptyset$
        for all $(v, t') \in V \times [C], c' \geq c$ and $k', 1 \geq k' \geq k$ do
            $T' \leftarrow A_{i-1}(G, v, c', k', \mathcal{D}) \cup (r, v, c')$
            if $d(T_{BEST}) > d(T')$ then $T_{BEST} \leftarrow T'$    ▷ Demand $i$ satisfied only if edge to $b_i$ at $c_i$ (ie $(x, b_i, c_i)$ for some x)
        $T \leftarrow T \cup T_{BEST}$; $k \leftarrow k - |D \cap V(T_{BEST})|$; $X \leftarrow X - V(T_{BEST})$
    return $T$
```

The way we will prove the approximation ratio of this algorithm is to show that it behaves precisely as the algorithm of Charikar et al. does, when given as input the DST instance produced by our reduction from Monotonic Single Source DCSN (Lemma 3).

Proposition 3 *The algorithm above is equivalent to the algorithm of Charikar et al., when applied to the DST instance output by the reduction of Lemma 3.*

Proof To see this, note that in our reduced instance, we see a collection of vertices, $v^1, \ldots, v^{|C|}$. Therefore, the only equivalent modifications needed to the original algorithm are:

- In the input, rather than keeping track of the current root as some vertex v^i, keep track of v at the current condition instead, i.e. (v, i).
- The distance from some v^i to $x^j, j \geq i$ is simply the distance from v to x at condition j, i.e. $\tilde{w}(v, x, j)$.
- Instead of looping through all vertices in the form $v^1, \ldots, v^{|C|}$, we instead loop through all vertices, and all conditions.

Therefore this algorithm guarantees the same approximation ratio for Monotonic Single Source DCSN as the original algorithm achieved for DST. In particular for all $i > 1$, $A_i(G, a, 0, k, D)$ provides an $i^2(i-1)k^{1/i}$ approximation to DCSN, in time $O(n^i k^{2i})$ [24, 39][4]. □

Abbreviations
CPCST: Condition Prize-Collecting Steiner Tree; CSN: Condition Steiner Network; CST: Condition Steiner Tree; CSP: Condition Shortest Path; DSN: Directed Steiner Network; DST: Directed Steiner Tree; DCSN: Directed Condition Steiner Network; DCSP: Directed Condition Shortest Path; *k*-PHLC: *k*-Partite Hypergraph Label Cover; MKL: Minimum *k*-Labeling; PPI: protein–protein interaction.

Authors' contributions
All authors conceived and designed the study. JW and BW derived the main hardness results. AK derived the monotonic hardness results and approximation algorithm. NY was the PI and oversaw the project. All authors read and approved the final manuscript.

Author details
[1] Department of Computer Science, Stanford University, Stanford, CA, USA.
[2] Department of Electrical Engineering and Computer Science, UC Berkeley, Berkeley, CA, USA.

Competing interests
The authors declare that they have no competing interests.

Availability of data and materials
Our solver for the general Condition Steiner Network problem is available at: https://github.com/YosefLab/condition_connectivity_problems.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Funding
This work was partially supported by the National Science Foundation Graduate Research Fellowship Program award DGE 1106400, NIH grants U01HG007910 and U01MH105979, and the U.S.-Israel Binational Science Foundation.

[4] The first paper [24] incorrectly claims a bound of $i(i-1)k^{1/i}$; this was corrected in [39].

Received: 3 April 2018 Accepted: 26 February 2019
Published online: 08 March 2019

References

1. Chatr-aryamontri A, Breitkreutz B-J, Oughtred R, Boucher L, Heinicke S, Chen D, Stark C, Breitkreutz A, Kolas N, O'Donnell L, Reguly T, Nixon J, Ramage L, Winter A, Sellam A, Chang C, Hirschman J, Theesfeld C, Rust J, Livstone MS, Dolinski K, Tyers M. The BioGRID interaction database: 2015 update. Nucleic Acids Res. 2015;43(Database issue):470–8. https://doi.org/10.1093/nar/gku1204.
2. Ben-Shitrit T, Yosef N, Shemesh K, Sharan R, Ruppin E, Kupiec M. Systematic identification of gene annotation errors in the widely used yeast mutation collections. Nat Methods. 2012;9(4):373–8.
3. Huang S-sC, Fraenkel E. Integration of proteomic, transcriptional, and interactome data reveals hidden signaling components. Sci Signal. 2009;2(81):40.
4. Scott MS, Perkins T, Bunnell S, Pepin F, Thomas DY, Hallett M. Identifying regulatory subnetworks for a set of genes. Mol Cell Proteom. 2005;4(5):683–92.
5. Wu C, Yosef N, Thalhamer T, Zhu C, Xiao S, Kishi Y, Regev A, Kuchroo VK. Induction of pathogenic TH17 cells by inducible salt-sensing kinase SGK1. Nature. 2013;496(7446):513–7.
6. Yosef N, Ungar L, Zalckvar E, Kimchi A, Kupiec M, Ruppin E, Sharan R. Toward accurate reconstruction of functional protein networks. Mol Syst Biol. 2009;5(1):248.
7. Garmaroudi FS, Marchant D, Si X, Khalili A, Bashashati A, Wong BW, Tabet A, Ng RT, Murphy K, Luo H, Janes KA, McManus BM. Pairwise network mechanisms in the host signaling response to coxsackievirus B3 infection. Proc Natl Acad Sci. 2010;107(39):17053–8. https://doi.org/10.1073/pnas.1006478107.
8. Huang SS, Fraenkel E. Integrating proteomic, transcriptional, and interactome data reveals hidden components of signaling and regulatory networks. Sci Signal. 2009;2(81):40. https://doi.org/10.1126/scisignal.2000350.
9. Tuncbag N, Gosline SJC, Kedaigle A, Soltis AR, Gitter A, Fraenkel E. Network-based interpretation of diverse high-throughput datasets through the omics integrator software package. PLOS Comput Biol. 2016;12(4):1–18. https://doi.org/10.1371/journal.pcbi.1004879.
10. Przytycka TM, Singh M, Slonim DK. Toward the dynamic interactome: it's about time. Brief Bioinform. 2010;11(1):15–29. https://doi.org/10.1093/bib/bbp057.
11. Mertins P, Przybylski D, Yosef N, Qiao J, Clauser K, Raychowdhury R, Eisenhaure TM, Maritzen T, Haucke V, Satoh T, Akira S, Carr SA, Regev A, Hacohen N, Chevrier N. An integrative framework reveals signaling-to-transcription events in toll-like receptor signaling. Cell Rep. 2017;19(13):2853–66. https://doi.org/10.1016/j.celrep.2017.06.016.
12. Kanshin E, Bergeron-Sandoval L-P, Isik SS, Thibault P, Michnick SW. A cell-signaling network temporally resolves specific versus promiscuous phosphorylation. Cell Rep. 2015;10(7):1202–14. https://doi.org/10.1016/j.celrep.2015.01.052.
13. Bendall SC, Simonds EF, Qiu P, Amir E-aD, Krutzik PO, Finck R, Bruggner RV, Melamed R, Trejo A, Ornatsky OI, Balderas RS, Plevritis SK, Sachs K, Pe'er D, Tanner SD, Nolan GP. Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. Science. 2011;332(6030):687–96. https://doi.org/10.1126/science.1198704 21551058[pmid].
14. Mazza A, Gat-Viks I, Farhan H, Sharan R. A minimum-labeling approach for reconstructing protein networks across multiple conditions. Algorithms Mol Biol. 2014;9(1):1.
15. Berlingerio M, Pinelli F, Calabrese F. ABACUS: frequent pAttern mining-BAsed Community discovery in mUltidimensional networkS. Data Min Knowl Discov. 2013;27:294–320.
16. Kivelä M, Arenas A, Barthelemy M, Gleeson JP, Moreno Y, Porter MA. Multilayer networks. J Complex Netw. 2014;2(3):203–71. https://doi.org/10.1093/comnet/cnu016.
17. Byrka J, Grandoni F, Rothvoß T, Sanità L. An improved LP-based approximation for Steiner tree. In: Proceedings of the forty-second ACM symposium on theory of computing. New York City: ACM; 2010. p. 583–92.
18. Agrawal A, Klein P, Ravi R. When trees collide: an approximation algorithm for the generalized Steiner problem on networks. SIAM J Comput. 1995;24(3):440–56.
19. Chlebík M, Chlebíková J. The Steiner tree problem on graphs: inapproximability results. Theor Comput Sci. 2008;406(3):207–14.
20. Feldman J, Ruhl M. The directed Steiner network problem is tractable for a constant number of terminals. In: 40th annual symposium on foundations of computer science, 1999. New York: IEEE; 1999. p. 299–308.
21. Feldman M, Kortsarz G, Nutov Z. Improved approximation algorithms for directed Steiner Forest. In: Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms. Philadelphia: Society for Industrial and Applied Mathematics; 2009. p. 922–31.
22. Chekuri C, Even G, Gupta A, Segev D. Set connectivity problems in undirected graphs and the directed Steiner network problem. ACM Trans Algorithm. 2011;7(2):18.
23. Dodis Y, Khanna S. Designing networks with bounded pairwise distance. In: Proceedings of the thirty-first annual ACM symposium on theory of computing. New York City: ACM; 1999. p. 750–9.
24. Charikar M, Chekuri C, Cheung T-Y, Dai Z, Goel A, Guha S, Li M. Approximation algorithms for directed Steiner problems. J Algorithms. 1999;33(1):73–91.
25. Halperin E, Krauthgamer R. Polylogarithmic inapproximability. In: Proceedings of the thirty-fifth annual ACM symposium on theory of computing. New York City: ACM; 2003. p. 585–94.
26. Archer A, Bateni M, Hajiaghayi M, Karloff H. Improved approximation algorithms for prize-collecting Steiner tree and TSP. SIAM J Comput. 2011;40(2):309–32.
27. Dinur I, Manurangsi P. Eth-hardness of approximating 2-CSPs and directed Steiner network. In: LIPIcs-Leibniz international proceedings in informatics, vol. 94. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2018.
28. Arora S, Babai L, Stern J, Sweedy Z. The hardness of approximate optima in lattices, codes, and systems of linear equations. In: Proceedings., 34th annual symposium on foundations of computer science, 1993. New York: IEEE; 1993. p. 724–33.
29. Arora S, Lund C, Motwani R, Sudan M, Szegedy M. Proof verification and the hardness of approximation problems. J ACM. 1998;45(3):501–55.
30. Raz R. A parallel repetition theorem. SIAM J Comput. 1998;27(3):763–803.
31. Charikar M, Naor J, Schieber B. Resource optimization in QoS multicast routing of real-time multimedia. IEEE/ACM Trans Netw. 2004;12(2):340–8.
32. Chuzhoy J, Gupta A, Naor JS, Sinha A. On the approximability of some network design problems. ACM Trans Algorithms. 2008;4(2):23.
33. Kandasamy K, Mohan SS, Raju R, Keerthikumar S, Kumar GSS, Venugopal AK, Telikicherla D, Navarro JD, Mathivanan S, Pecquet C, Gollapudi SK, Tattikota SG, Mohan S, Padhukasahasram H, Subbannayya Y, Goel R, Jacob HK, Zhong J, Sekhar R, Nanjappa V, Balakrishnan L, Subbaiah R, Ramachandra Y, Rahiman BA, Prasad TK, Lin J-X, Houtman JC, Desiderio S, Renauld J-C, Constantinescu SN, Ohara O, Hirano T, Kubo M, Singh S, Khatri P, Draghici S, Bader GD, Sander C, Leonard WJ, Pandey A. Netpath: a public resource of curated signal transduction pathways. Genome Biol. 2010;11(1):3. https://doi.org/10.1186/gb-2010-11-1-r3.
34. Hornbeck PV, Zhang B, Murray B, Kornhauser JM, Latham V, Skrzypek E. Phosphositeplus, 2014: mutations, PTMs and recalibrations. Nucleic Acids Res. 2015;43(Database issue):512–20. https://doi.org/10.1093/nar/gku1267.
35. Keshava Prasad TS, Goel R, Kandasamy K, Keerthikumar S, Kumar S, Mathivanan S, Telikicherla D, Raju R, Shafreen B, Venugopal A, Balakrishnan L, Marimuthu A, Banerjee S, Somanathan DS, Sebastian A, Rani S, Ray S, Harrys Kishore CJ, Kanth S, Ahmed M, Kashyap MK, Mohmood R, Ramachandra YL, Krishna V, Rahiman BA, Mohan S, Ranganathan P, Ramabadran S, Chaerkady R, Pandey A. Human protein reference database—2009 update. Nucleic Acids Res. 2009;37(suppl 1):767–72. https://doi.org/10.1093/nar/gkn892.
36. Li T, Wernersson R, Hansen RB, Horn H, Mercer J, Slodkowicz G, et al. A scored human protein–protein interaction network to catalyze genomic interpretation. Nat Methods. 2016;14:61.
37. Gurobi Optimization L. Gurobi optimizer reference manual. 2018. http://www.gurobi.com.
38. Feige U. A threshold of ln n for approximating set cover. J ACM. 1998;45(4):634–52.
39. Helvig CS, Robins G, Zelikovsky A. An improved approximation scheme for the group Steiner problem. Networks. 2001;37(1):8–20.