Approximation Algorithms for Connected Maximum Coverage

Gianlorenzo D'Angelo
Gran Sasso Science Institute (GSSI)
L'Aquila, Italy
gianlorenzo.dangelo@gssi.it

Esmaeil Delfaraz
University of L'Aquila
L'Aquila, Italy
esmaeil.delfarazpahlevanloo@univaq.it

ABSTRACT

The *Connected Budgeted maximum Coverage* problem (**CBC**) is a combinatorial optimization problem that finds applications in path planning, wireless sensor networks, logistics, and bioinformatics. In **CBC**, we are given a collection of subsets $\mathcal{S}$, defined over a ground set X, and an undirected graph $G = (V, E)$, where each node is associated with a set of $\mathcal{S}$. Each set in $\mathcal{S}$ has a different cost and each element of X gives a different prize. The goal is to find a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that $\mathcal{S}'$ induces a connected subgraph in G, the total cost of the sets in $\mathcal{S}'$ does not exceed a budget B, and the total prize of the elements covered by $\mathcal{S}'$ (i.e., $\bigcup_{S \in \mathcal{S}'} S$) is maximized. The *Directed rooted Connected Budgeted maximum Coverage* problem (**DCBC**) is a generalization of **CBC** where the underlying graph G is directed and in the subgraph induced by $\mathcal{S}'$ in G there must be a path from a specific node called root to any other node.

These NP-hard problems have been widely studied from the approximation point of view. Still, the current best algorithms achieve approximation ratios that are linear in the size of the underlying graph or depend on B. In this paper, we provide two algorithms for **CBC** and **DCBC** that guarantee approximation ratios of $O\left(\frac{\log(|V|+|X|)\log|X|}{\epsilon^2}\right)$ and $O\left(\frac{\sqrt{|V|}\log^2|X|}{\epsilon^2}\right)$, respectively, at the cost of a violation in the budget constraint of a factor $1+\epsilon$, where $\epsilon \in (0, 1]$. We also improve the approximation factor for the *directed budgeted rooted out-tree maximization problem*, a particular case of **DCBC** where the prize function is additive, from $O\left(\frac{1}{\epsilon^2}|V|^{2/3}\log|V|\right)$ to $O\left(\frac{1}{\epsilon^2}|V|^{1/2}\log^2|V|\right)$, for any $\epsilon \in (0, 1]$.

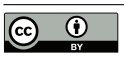
CCS CONCEPTS

• **Theory of computation → Routing and network design problems**; **Routing and network design problems**; • **Mathematics of computing → Graph algorithms**.

KEYWORDS

Combinatorial Optimization; Path Planning; Prize Collecting Steiner Tree; Maximum Coverage; Approximation Algorithms

ACM Reference Format:
Gianlorenzo D'Angelo and Esmaeil Delfaraz. 2025. Approximation Algorithms for Connected Maximum Coverage. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

In the *budgeted maximum coverage* problem, we are given a ground set X of elements with associated prizes, a collection $\mathcal{S}$ of subsets of X with associated costs, and a budget B. The aim is to find a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that the total cost of the sets in $\mathcal{S}'$ does not exceed B and the total prize of the elements covered by $\mathcal{S}'$ (i.e., $\bigcup_{S \in \mathcal{S}'} S$) is maximized [16]. The *Connected Budgeted maximum Coverage* problem (**CBC**) is a generalization of the budgeted maximum coverage problem in which the sets in $\mathcal{S}$ are associated with the nodes of a graph $G = (V, A)$ and the subcollection $\mathcal{S}'$ must induce a connected subgraph T in G.

The **CBC** problem is motivated by several applications in multi-agent path planning, wireless sensor networks, and bioinformatics. Consider, for example, the exploration of an area through an Unmanned Aerial Vehicle that can take pictures of an area close to its current location. In this scenario, the areas to be explored correspond to the elements in X, and the nodes of the graph correspond to the locations that the vehicle can reach. Finding a connected maximum coverage corresponds to finding a small set of connected locations that allow the exploration of the largest possible area [4, 30]. Another application is the deployment of wireless sensor networks in a scenario where each sensor is able to detect a set of target points in its sensing range, and one wants to find a bounded set of connected sensors that detects the largest number of target points [31]. Here, the sensor network corresponds to the graph G, and the target points to the elements X. Vandin et al. [29] studied **CBC** motivated by the detection of driver mutations in protein-to-protein interaction networks. In these networks, a node represents a protein, and an edge represents an interaction between two proteins. Each protein is associated with a gene mutation and a set of cancer patients who are affected by such mutation. It is widely believed that cancer is associated with a connected series of mutations in these networks, called pathways [14]. Therefore, finding a connected set of B nodes with maximum coverage corresponds to finding the B connected mutations that affect the largest number of cancer patients. Variants of **CBC** find application in network surveillance [23] and recovery of power networks [13].

The *Directed rooted Connected Budgeted maximum Coverage* problem (**DCBC**) is a generalization of **CBC** to directed graphs, where the aim is to find a rooted out-tree maximizing the prizes of covered elements and respecting a budget constraint. Besides the same applications as **CBC** in the cases where the underlying network is directed, the **DCBC** problem has specific motivating application scenarios in facility location, epidemiology, and computational social choice. Consider a large warehouse from which goods are delivered through a road network to smaller warehouses or retail shops that serve a set of customers. Here, the graph models the (directed) road network; the ground set represents the set of customers; and shops, represented as nodes in the graph, are associated with the set of

customers that they may serve. In order to maximize the number of customers that can be served at a given delivery cost, one needs to compute an out-tree, rooted at the node representing the warehouse, that maximizes the overall number of covered customers and satisfies the budget constraint. The connectivity is required by how the goods are distributed from the main warehouse to the opened shops through a directed road network. Vehicles departing from the warehouse (root) reach the retail shops (selected nodes) by means of directed paths. Other applications require computing rooted out-trees in directed graphs in order to reconstruct epidemic outbreaks [24, 28] or to maximize the voting power of a voter in liquid democracy [7].

Problems **CBC** and **DCBC** have already been studied under the lens of approximation algorithms. However, the state-of-the-art algorithms achieve approximation ratios that are, in the worst case, linear in $|V|$ [15, 26, 29] or depend on the budget B [6, 15], and, in some cases, only work under specific assumptions [15, 26, 29]. In this paper, we improve our knowledge on the approximabilty of **CBC** and **DCBC** by providing the first polynomial-time bicriteria approximation algorithm with sublinear approximation ratios. We now give more details on the state-of-the-art approximation algorithms for **CBC** and **DCBC** and then specify our results, comparing them with previous work. Vandin et al. [29] considered the special case of **CBC** where the cost is equal for all nodes and provided a $c\rho$-approximation, where $c = (2e-1)/(e-1)$ and ρ is the radius of the connected subgraph induced by an optimal solution. Hochbaum and Rao [15] improved this bound to $\min\{((1-1/e)(1/\rho - 1/B))^{-1}, B\}$. This latter result holds also in the more general case in which the prize function is a monotone submodular function of the set of nodes in the graph. Still, the cost function is assumed to be constant for all nodes. Ran et al. [26] considered **CBC** without restrictions on the cost function but under the assumption that if two sets in $\mathcal{S}$ have a non-empty intersection, then the corresponding nodes in G must be adjacent. In this setting, they provided an $O(\Delta \log |V|)$-approximation algorithm, where Δ is the maximum degree of G. D'Angelo et al. [6] improved this result to $O(\log |V|)$ under the same assumption. Moreover, they gave an $O(\sqrt{B})$-approximation algorithm for **DCBC**. In the worst case, $\rho = \Omega(|V|)$ and $\Delta = \Omega(|V|)$, and thus the approximation ratio of algorithms in [15, 26, 29] are linear in $|V|$. Moreover, B can be exponential in $|V|$ for general cost functions.

The results of this paper are summarized in the following:

- For **DCBC**, we provide a bicriteria approximation algorithm that, for any $\epsilon \in (0, 1]$, guarantees an approximation ratio of $O\left(\frac{\sqrt{|V|}\log^2 |X|}{\epsilon^2}\right)$ at the cost of a violation in the budget constraint of a factor at most $1+\epsilon$. We observe that the approximation ratio of our algorithm is sublinear in $|V|$, while the previous bound [6] depends on the budget B, which might be exponential in $|V|$. However, the algorithm in [6] works for any monotone submodular prize function.
- For **CBC**, we observe that our algorithm for **DCBC** can also be used for the undirected case, achieving the same bound. However, we show that a slight modification of it significantly improves the approximation to $O\left(\frac{\log(|V|+|X|)\log|X|}{\epsilon^2}\right)$, again with a budget violation of $1+\epsilon$, for any $\epsilon \in (0, 1]$. Compared to previous work, we achieve an approximation ratio which depends logarithmic on $|V|$, whereas the previous bounds are linear in $|V|$ (see [15, 26, 29]) or depend on the budget B (see [6, 15]), or only work under specific assumptions on the input (papers [15, 29] consider constant cost functions, while papers [6, 26] assumes a connection between adjacent nodes and intersecting sets, see discussion above).
- Our algorithm for **DCBC** uses, as a subroutine, an algorithm for the node-weighted Steiner tree problem in directed graphs (**DST**). In particular, our algorithm requires that such a subroutine computes a tree whose cost is within a bounded factor from the optimum of the *standard flow-based linear programming relaxation* of **DST**. Previous algorithms for **DST** only focus on the approximation factor with respect to the *optimum* of **DST** but do not ensure a bounded factor over the optimum of its relaxation [3]. Therefore, we introduce a new algorithm for **DST** that guarantees this factor to be $O(\sqrt{|V|}\log|V|)$. Besides its usage as a subroutine in the algorithm for **DCBC**, we believe that this result is interesting on its own.
- As a consequence of our result for **DCBC**, we improve the approximation ratio for the *Directed Budgeted rooted Out-tree Maximization* problem (**DBOM**), a particular case of **DCBC** in which both costs and prizes are associated with the nodes of a directed graph and the goal is to find an out-tree rooted at a specific node that satisfies a budget constraint on the sum of costs and maximizes the sum of prizes of the nodes. For **DBOM**, D'Angelo and Delfaraz [5] gave an $O\left(\frac{1}{\epsilon^2}|V|^{2/3}\log|V|\right)$-approximation algorithm which violates the budget constraint by a factor of at most $1+\epsilon$, for any $\epsilon \in (0, 1]$. Our algorithm for **DCBC** improves this bound to $O\left(\frac{1}{\epsilon^2}|V|^{1/2}\log^2|V|\right)$, with the same budget violation.
- The *Budgeted Node-weighted Steiner* problem (**BNS**) is the **DBOM** problem restricted to undirected graphs and can be seen as a particular case of **CBC** with additive prize function. Our algorithm for **CBC** can be used also for **BNS** and achieves an approximation factor of $O\left(\frac{\log^2|V|}{\epsilon^2}\right)$ with a budget violation of $1+\epsilon$, for any $\epsilon \in (0, 1]$. Our algorithm almost matches (up to an $O(\log|V|)$ factor) the current best algorithm for this problem, which achieved an $O(\frac{1}{\epsilon^2}\log|V|)$ approximation factor with a budget violation of $1+\epsilon$, for any $\epsilon \in (0, 1]$, see [1].

Related work. The generalization of **CBC** in which the prize function is a monotone submodular function on the set of nodes in the graph and the cost function on nodes is defined over positive integers has been studied by Kuo et al. [20], who gave an $O(\Delta\sqrt{B})$-approximation algorithm. For the same problem, D'Angelo et al. [6] gave an $O(\sqrt{B})$-approximation algorithm, which also applies to the directed case with the same bound. They also considered the rooted variant of the same problem in which a specific root node is required to belong to the solution and provided an $O(\frac{1}{\epsilon^3}\sqrt{B})$-approximation algorithm, if a budget violation of a factor $1+\epsilon$, for some $\epsilon \in (0, 1]$, is allowed. Ghuge and Nagarajan [11] provided a

tight quasi-polynomial time $O(\frac{\log n'}{\log\log n'})$-approximation algorithm for the directed case, where n' is the number of nodes in an optimal solution.

The **BNS** problem can be seen as a particular case of **CBC** in which both costs and prizes are associated with the nodes of an undirected graph, and the goal is to find a tree that satisfies a budget constraint on the sum of costs and maximizes the sum of prizes of the nodes. For this problem, Guha et al. [13] gave a polynomial time $O(\log^2|V|)$-approximation algorithm that violates the budget constraint by a factor of at most 2. Moss and Rabani [25] improved the approximation factor to $O(\log|V|)$, with the same budget violation. Later, Bateni et al. [1] proposed an $O(\frac{1}{\epsilon^2}\log|V|)$-approximation algorithm which requires a budget violation of only $1+\epsilon$, for any $\epsilon \in (0,1]$. Kortsarz and Nutov [19] showed that this problem admits no $o(\log\log|V|)$-approximation algorithm, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$, even if the algorithm is allowed to violate the budget constraint by a factor equal to a universal constant. Bateni et al. [1] showed that the integrality gap of the standard flow-based LP relaxation for the budgeted node-weighted Steiner is unbounded if no budget violation is allowed. **CBC** also generalizes the well-known *budgeted connected dominating set* problem [17] for which there exists a constant factor approximation algorithm [21].

2 NOTATION AND PROBLEM STATEMENT

For an integer i, let $[i] := \{1,\dots,i\}$. Let $G=(V,A)$ be a directed graph and $c: V \to \mathbb{R}^{\geq 0}$ be a nonnegative cost function on nodes.

A *path* is a directed graph made of a sequence of distinct nodes $(v_1,\dots,v_s)$ and a sequence of directed arcs (v_i,v_{i+1}), $i \in [s-1]$. An *out-tree* (a.k.a. out-arborescence) is a directed graph in which there is exactly one directed path from a specific node r, called *root*, to each other node. If a subgraph T of a directed graph G is an out-tree, then we say that T is an out-tree of G. For simplicity of reading, we will refer to out-trees simply as trees when it is clear that we are in the context of directed graphs. Given two nodes $u,v \in V$, the cost of a path from u to v in G is the sum of the cost of its nodes. A path from u to v with the minimum cost is called a *shortest path* and its cost, denoted by $dist(u,v)$, is called the *distance* from u to v in G. A graph G is called *B-proper* for the node r if $dist(r,v) \leq B$ for any v in V. For any subgraph G' of G, we denote by $V(G')$ and $A(G')$ the set of nodes and arcs in G', respectively. Given a subset of nodes $V' \subseteq V$, $G[V']$ denotes the subgraph of G induced by nodes V', i.e., $V(G[V']) = V'$ and $A(G[V']) = \{(u,v) \in A : u,v \in V'\}$).

Let X be a ground set of elements, $\mathcal{S} \subseteq 2^X$ be a collection of subsets of X, and $p: X \to \mathbb{R}^{\geq 0}$ be a prize function over the elements of X. In the *Directed rooted Connected Budgeted maximum Coverage* (**DCBC**), each node v of a directed graph G is associated with a set S_v of $\mathcal{S}$ and the goal is to find a rooted out-tree T of G with bounded cost that maximizes the overall prize of the union of the sets associated with the nodes in T. Formally, in **DCBC** we are given as input a ground set X, a collection $\mathcal{S} \subseteq 2^X$ of subsets of X, a directed graph $G=(V,A)$, where each node $v \in V$ is associated with a set S_v of $\mathcal{S}$, a root node $r \in V$, a cost function $c: V \to \mathbb{R}^{\geq 0}$ on the nodes of G, a prize function $p: X \to \mathbb{R}^{\geq 0}$ on the ground set X, and a budget $B \in \mathbb{R}^+$. The goal is to find an out-tree T of G rooted at r, such that $c(T) = \sum_{v\in V(T)} c(v) \leq B$ and $p(T) = \sum_{x\in X_T} p(x)$ is maximum, where $X_T = \bigcup_{v\in V(T)} S_v$.

The **CBC** problem is a restriction of **DCBC** to undirected graphs. We consider a *rooted* version of **CBC**, which is more general from an approximation point of view because one can guess a node in an optimal solution of **CBC** and use it as a root. The *Undirected rooted Connected Budgeted maximum Coverage* (**UCBC**) is defined as follows. As input, we are given a ground set X, a collection $\mathcal{S} \subseteq 2^X$ of subsets of X, an *undirected* graph $G=(V,A)$, where each node $v \in V$ is associated with a set S_v of $\mathcal{S}$, a root node $r \in V$, a cost function $c: V \to \mathbb{R}^{\geq 0}$ on the nodes of G, a prize function $p: X \to \mathbb{R}^{\geq 0}$ on the ground set X, and a budget $B \in \mathbb{R}^+$. The goal is to find a *tree* T of G such that $r \in V(T)$, $c(T) = \sum_{v\in V(T)} c(v) \leq B$, and $p(T) = \sum_{x\in X_T} p(x)$ is maximum, where $X_T = \bigcup_{v\in V(T)} S_v$.

Problems **DCBC** and **UCBC** generalize several well-known *NP*-hard problems, including the budgeted maximum coverage problem [16], which is the particular case where the input graph is a (bidirected) clique; the *Directed Budgeted rooted Out-tree Maximization* (**DBOM**) problem [1, 5], which is the particular case in which each node of the graph is associated with a distinct singleton set and hence $|X| = |V|$; and the *Budgeted Node-weighted Steiner* problem (**BNS**), which is the undirected version of **DBOM**. Therefore, both **DCBC** and **UCBC** problems are *NP*-hard to approximate within a factor $1-1/e$ like the budgeted maximum coverage problem [9]. Moreover, like **BNS**, they admit no $o(\log\log|V|)$-approximation algorithm, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$, even if the algorithm is allowed to violate the budget constraint by a factor equal to a universal constant [19].

In order to provide a bicriteria approximation algorithm for **DCBC**, we will use as a subroutine a polynomial time approximation algorithm for the *node-weighted Directed Steiner tree problem* (**DST**), defined as follows. We are given as input a directed graph $G=(V,A)$, a root node $r \in V$, a set of terminal nodes $R \subseteq V$, and a cost function $c: V \to \mathbb{R}^{\geq 0}$ defined on the nodes of G. The goal is to find an out-tree of G rooted at r and spanning all nodes in R, i.e., $R \subseteq V(T)$, such that $c(T) = \sum_{v\in V(T)} c(v)$ is minimum.

Our algorithms will provide a solution with a bounded approximation ratio and a bounded violation of the budget constraint. A polynomial time algorithm is a bicriteria (β,α)-approximation algorithm if it achieves an approximation ratio of $\alpha > 1$ and a budget violation factor of at most $\beta > 1$, that is, for any instance I of **DCBC**, it returns a solution T such that $p(T) \geq \frac{p(T_B^*)}{\alpha}$ and $c(T) \leq \beta B$, where $p(T_B^*)$ is the optimum for I and B is the budget in I.

3 APPROXIMATION ALGORITHMS FOR CBC, DCBC, AND DBOM

In this section, we introduce our approximation algorithms. We start with the polynomial-time bicriteria $\left(1+\epsilon, O\left(\frac{\sqrt{|V|}\log^2|X|}{\epsilon^2}\right)\right)$-approximation algorithm for **DCBC**, where ϵ is an arbitrary number in $(0,1]$. We then observe that this algorithm provides a bicriteria $\left(1+\epsilon, O\left(\frac{1}{\epsilon^2}|V|^{1/2}\log^2|V|\right)\right)$ for **DBOM**, for $\epsilon \in (0,1]$. Finally, we show how to modify the algorithm and its analysis in the particular case of undirected graphs to achieve a bicriteria $\left(1+\epsilon, O\left(\frac{\log(|V|+|X|)\log|X|}{\epsilon^2}\right)\right)$-approximation for **UCBC**, for any $\epsilon \in (0,1]$.

Let $I =< X, \mathcal{S}, G = (V, A), c, p, r, B >$ be an instance of **DCBC**. We denote by T_B^* an optimal solution for I.

Our algorithm for **DCBC** can be summarized in the following three steps:

(1) We first define a linear program, denoted as (LP-DCBC), whose optimum OPT is an upper bound on the optimum prize $p(T_B^*)$ of I.
(2) We give a polynomial-time algorithm that, starting from an optimal solution for (LP-DCBC), computes a tree T for which $p(T) = \Omega(OPT)$ and the ratio between prize and cost is $\gamma = \frac{p(T)}{c(T)} = \Omega\left(\frac{OPT}{B\sqrt{|V|\log^2|X|}}\right)$.
(3) The cost of T can exceed the budget B but, since the prize-to-cost ratio of T is bounded, we can apply to it a variant of the trimming process given in [1] to obtain another tree $\hat{T}$ with cost $\frac{\epsilon}{2}B \le c(\hat{T}) \le (1+\epsilon)B$, for any $\epsilon \in (0, 1]$, and prize-to-cost ratio $\frac{p(\hat{T})}{c(\hat{T})} \ge \frac{\epsilon\gamma}{4}$. Therefore, the prize accrued by $\hat{T}$ is $p(\hat{T}) \ge \frac{\epsilon\gamma}{4}c(\hat{T}) = \Omega\left(\frac{\epsilon^2}{\sqrt{|V|\log^2|X|}}OPT\right)$, which implies an approximation ratio of $O\left(\frac{\sqrt{|V|\log^2|X|}}{\epsilon^2}\right)$ with a budget violation factor of at most $1+\epsilon$.

Recall that a directed graph $G = (V, A)$ is B-proper for a node r if, for every $v \in V$, it holds $dist(r, v) \le B$. Initially, we remove from the input graph all the nodes v having a distance more than B from r, making G a B-proper graph for r.

Upper Bound on the Optimal Prize

We now provide a linear program whose optimum OPT is an upper bound to the optimum $p(T_B^*)$ of I, i.e., $p(T_B^*) \le OPT$.

We create a directed graph G' in which each node is associated with a cost function $c' : V \to \mathbb{R}^{\ge 0}$ and a prize function $p' : V \to \mathbb{R}^{\ge 0}$. Graph G' is created from G by adding, for each element x of X, a node w_x with cost 0 and prize $p(x)$ and, for each node $v \in V$ and each element x that belongs to S_v, a directed arc from v to w_x. Formally, we let $G' = (V', A')$, where $V' = V \cup W$ with $W = \{w_x : x \in X\}$ and $A' = A \cup \{(v, w_x) : v \in V, x \in S_v\}$. For each $v \in V$, we let $c'(v) = c(v)$ and $p'(v) = 0$, and, for each $w_x \in W$, we let $c'(w_x) = 0$ and $p'(w_x) = p(x)$. For each $v \in V'$, we use shortcuts $c_v = c'(v)$ and $p_v = p'(v)$.

For every $v \in V'$, we let $\mathcal{P}_v$ be the set of simple paths in G' from r to v. Our linear program (LP-DCBC) is defined as follows.

$$\begin{aligned}
\text{maximize} \quad & \textstyle\sum_{v\in V'} y_v p_v && && \text{(LP-DCBC)}\\
\text{subject to} \quad & \textstyle\sum_{v\in V'} y_v c_v \le B && && (1)\\
& \textstyle\sum_{P\in\mathcal{P}_v} f_P^v = y_v, && \forall v \in V' \setminus \{r\} && (2)\\
& \textstyle\sum_{P\in\mathcal{P}_v : z\in P} f_P^v \le y_z, && \forall z, v \in V' \setminus \{r\} && (3)\\
& 0 \le y_v \le 1, && \forall v \in V'\\
& 0 \le f_P^v \le 1, && \forall v \in V', P \in \mathcal{P}_v.
\end{aligned}$$

We use variables f_P^v and y_v, for each $v \in V'$ and $P \in \mathcal{P}_v$, where f_P^v is the amount of flow sent from r to v using path P and y_v is the capacity of node v and the overall amount of flow sent from r to v. Variables y_v, for $v \in V'$, are called *capacity variables*, while variables f_P^v for $v \in V'$ and $P \in \mathcal{P}_v$ are called *flow variables*.

The constraints in (LP-DCBC) are as follows. Constraint (1) ensures that the (fractional) solution to the LP costs at most B. Constraints (2) and (3) formulate the connectivity constraint through a standard flow encoding, that is they ensure that the nodes v with $y_v > 0$ induce a subgraph in which all nodes are reachable from r. In particular, Constraint (2) ensures that the amount of flow that is sent from r to a node v is equal to y_v and Constraint (3) ensures that the total flow from r to v passing through a node z does not exceed y_z.

Note that the number of flow variables is exponential in the size of the input. However, (LP-DCBC) can be solved in polynomial time since, given an assignment of capacity variables, we need to find, independently for any $v \in V' \setminus \{r\}$, a flow from r to v of overall value y_v that satisfies the capacities of nodes $z \in V' \setminus \{r, v\}$ (see e.g. [12]).

We now show that the optimum OPT of (LP-DCBC) is an upper bound to the optimum of I. In particular, the next lemma shows that, for any feasible solution T_B for I, we can compute a feasible solution $\{y_v\}_{v\in V'}$ for (LP-DCBC) such that $p(T_B) = \sum_{v\in V'} y_v p_v$.

Lemma 3.1. *Given an instance $I =< X, \mathcal{S}, G = (V, A), c, p, r, B >$ of **DCBC**, for any feasible solution T_B for I there exists a feasible solution $\{y_v, f_P^v\}_{v\in V', P\in\mathcal{P}_v}$ for* (LP-DCBC) *such that $p(T_B) = \sum_{v\in V'} y_v p_v$.*

Proof. Let $X_{T_B} = \bigcup_{v\in V(T_B)} S_v$. We define a solution to the linear program (LP-DCBC) in which for all $v \in V(T_B)$ and $x \in X_{T_B}$, we set $y_v = 1$ and $y_{w_x} = 1$, while we set $y_u = 0$ for any other node u of V'. Since $c_w = 0$, for all $w \in W$, and $c(T_B) = \sum_{v\in V(T_B)} c_v \le B$, then $\sum_{v\in V'} y_v c_v = \sum_{v\in V} y_v c_v + \sum_{w\in W} y_w c_w = \sum_{v\in V(T_B)} y_v c_v \le B$ and the budget Constraint (1) is satisfied.

Since T_B is an out-tree, then there exists exactly one path from r to v in T_B, for each $v \in V(T_B)$. Let us denote this path by P_v. For each $x \in X_{T_B}$, let us select an arbitrary $v \in V(T_B)$ such that $x \in S_v$ and let P_x be the path $P_v \cup \{(v, w_x)\}$. For each $v \in V(T_B)$ and $x \in X_{T_B}$, we set $f_{P_v}^v = 1$ and $f_{P_x}^{w_x} = 1$, while any other flow variable is set to 0. Then, Constraints (2) and (3) are satisfied.

Given the definition of y and since $p_v = 0$, for all $v \in V$, then $\sum_{v\in V'} y_v p_v = \sum_{v\in V(T_B)} y_v p_v + \sum_{x\in X_{T_B}} y_{w_x} p_{w_x} = \sum_{x\in X_{T_B}} p_{w_x} = p(T_B)$. This concludes the proof. □

A Tree with a Good Ratio between Prize and Cost

Here, we give a polynomial time algorithm that computes an out-tree T of G' rooted at r, whose prize is $\Omega(OPT)$ and whose ratio between prize and cost is $\Omega\left(\frac{OPT}{B\sqrt{|V|\log^2|X|}}\right)$. Note, however, that the cost of T can exceed the budget B by an unbounded factor. We will show in the next section how to trim T in order to bound its cost and, at the same time, retain a good prize. Here we show the following theorem.

Theorem 3.2. *There exists a polynomial time algorithm that computes an out-tree T of G' rooted at r such that $p'(T) = \sum_{v\in V(T)} p'(v) = \Omega(OPT)$ and the ratio between prize and cost of T is*

$$\frac{p'(T)}{c'(T)} = \Omega\left(\frac{OPT}{B\sqrt{|V|\log^2|X|}}\right),$$

where OPT is the optimum of (LP-DCBC).

To prove the theorem, we start by introducing a polynomial time algorithm that computes an out-tree spanning a given set of nodes, called terminals. The cost of this out-tree is bounded by a function of a lower bound on the amount of flow received by each terminal in an optimal solution for (LP-DCBC). Formally, we prove the next lemma. The algorithm in Theorem 3.2, carefully chooses suitable terminal sets that guarantee a lower bound on the obtained prize and on the received flow.

Lemma 3.3. *Let $\{y_v, f_P^v\}_{v\in V', P\in\mathcal{P}_v}$ be an optimal solution for linear program* (LP-DCBC), *$\delta \geq 1$ be a real number, and $R \subseteq W$ be a set of nodes such that $y_w \geq 1/\delta$, for each $w \in R$. Then there exists a polynomial time algorithm that computes an out-tree T of G' rooted at r that spans all the nodes in R and costs $c'(T) = O(\delta B\sqrt{|V|}\log|R|)$.*

Proof. The proof is summarized as follows. We consider the set of nodes in R as the set of terminals in an instance of the node-weighted Directed Steiner tree problem (**DST**) where r is the root node. By using solution $\{y_v, f_P^v\}_{v\in V', P\in\mathcal{P}_v}$ and the definition of R, we show that the optimum for a fractional relaxation of this instance of **DST** is at most δB. Then, we apply the approximation algorithm for **DST** that we will give in Section 4, which computes a tree whose cost is a factor $O(\sqrt{|V|}\log|R|)$ from the optimum of the same fractional relaxation. Therefore, we obtain an out-tree that is rooted at r, spans all nodes in R, and costs $O(\delta B\sqrt{|V|}\log|R|)$, proving the theorem.

We now give the details of the proof. We first problem **DST** and its linear relaxation. In **DST**, we are given a directed graph $G'' = (V'', A'')$ with nonnegative costs assigned to its nodes and a set of terminals $R \subseteq V''$, and the goal is to find an out-tree of G'' rooted at the given root node spanning R such that the total cost on its nodes is minimum. We consider the standard flow-based linear programming relaxation of **DST** (called **FDST**) in which we need to assign capacities to nodes in such a way that the total flow sent from the root node to any terminal is 1 and the sum of node capacities multiplied by their cost is minimized. Formally, given a directed graph $G'' = (V'', A'')$, a root node $r \in V''$, a nonnegative node-cost function $c'' : V'' \to \mathbb{R}^{\geq 0}$, and a set of terminals $R \subseteq V''$, **FDST** requires to solve the following linear program.

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{v\in V''} x_v c_v && && \text{(LP-DST)}\\
\text{subject to} \quad & \textstyle\sum_{P\in\mathcal{P}_t} g_P^t = 1, && \forall t \in R && (4)\\
& \textstyle\sum_{P\in\mathcal{P}_t : v\in P} g_P^t \leq x_v, && \forall v \in V'', t \in R && (5)\\
& 0 \leq x_v \leq 1, && \forall v \in V''\\
& 0 \leq g_P^t \leq 1, && \forall t \in R, P \in \mathcal{P}_t,
\end{aligned}$$

where $\mathcal{P}_t$ is the set of all simple paths from r to t in G'', for each $t \in R$, and $c_v = c''(v)$, for each $v \in V''$. Similarly to (LP-DCBC), we use variables x_v and g_P^t as capacity and flow variables, respectively, for each $v \in V''$, $t \in R$, and $P \in \mathcal{P}_t$. As for (LP-DCBC), Constraints (4) and (5) ensure connectivity, but, differently from (LP-DCBC), we require that all terminals receive an amount of flow from r equal to 1, while the other nodes do not need to receive a predefined amount of flow.

From $G' = (V', A')$ and R, we define an instance I_{DST} of **DST** as follows. We create a directed graph $G'' = (V'', A'')$ as the subgraph of G' induced by $V'' = V \cup R$. The set of terminals in I_{DST} is R, the root node is r and the node costs are defined as c', i.e., $c''(v) = c'(v)$, for each $v \in V''$. Let I_{FDST} be the instance of **FDST** induced by I_{DST} as in (LP-DST) and let OPT_{FDST} be the optimum for I_{FDST}.

We now argue that the optimum OPT_{FDST} for I_{FDST} is at most δB. Starting from the solution $\{y_v, f_P^v\}_{v\in V', P\in\mathcal{P}_v}$ for (LP-DCBC), we define a solution $\{x_v, g_P^t\}_{v\in V'', t\in R, P\in\mathcal{P}_t}$ for (LP-DST) as follows: $x_t = 1$, for each $t \in R$; $g_P^t = f_P^t/y_t$, for each $t \in R$ and $P \in \mathcal{P}_t$; and $x_v = \max_{t\in R}\{\sum_{P\in\mathcal{P}_t : v\in P} g_P^t\}$, for each $v \in V'' \setminus R$. We show that the defined solution is feasible for (LP-DST) and its cost is at most δB, which implies that $OPT_{FDST} \leq \delta B$. Constraint (4) is satisfied as, by Constraint (2) of (LP-DCBC), we have that, for each $t \in R$, $\sum_{P\in\mathcal{P}_t} f_P^t = y_t$ and hence $\sum_{P\in\mathcal{P}_t} g_P^t = \sum_{P\in\mathcal{P}_t} f_P^t/y_t = 1$. Constraint (5) is satisfied, as by definition of x_v, it holds $x_v \geq \sum_{P\in\mathcal{P}_t : v\in P} g_P^t$, for each $v \in V''$ and $t \in R$. The last two constraints are satisfied by definition of $\{x_v, g_P^t\}_{v\in V'', t\in R, P\in\mathcal{P}_t}$ and by Constraint (4). The cost of $\{x_v\}_{v\in V''}$ is equal to $\sum_{v\in V''} x_v c_v$. For each $v \in V'' \setminus R$, let t_v be the terminal that attains the maximum in the definition of x_v, i.e., $t_v := \arg\max_{t\in R}\{\sum_{P\in\mathcal{P}_t : v\in P} g_P^t\}$, then

$$x_v = \sum_{P\in\mathcal{P}_{t_v} : v\in P} g_P^{t_v} = \sum_{P\in\mathcal{P}_{t_v} : v\in P} f_P^{t_v}/y_{t_v} \leq y_v/y_{t_v} \leq \delta y_v,$$

where the first inequality is due to Constraint (3) of (LP-DCBC) and the last inequality is due to $y_t \geq 1/\delta$ for each node $t \in R$. Moreover, $c_t = 0$ for each $t \in R$, because $R \subseteq W$. It follows that $\sum_{v\in V''} x_v c_v = \sum_{v\in V''\setminus R} x_v c_v \leq \delta \sum_{v\in V''\setminus R} y_v c_v \leq \delta \sum_{v\in V'\setminus R} y_v c_v \leq \delta B$, by Constraint (1) of (LP-DCBC).

Finally, we apply the algorithm in Section 4. This algorithm is a polynomial time $O(\sqrt{|V''\setminus R|}\log|R|)$-approximation algorithm for **DST** that, starting from an optimal solution to (LP-DST), computes a tree T_{DST} rooted at r spanning all the terminals. Moreover, the cost of T_{DST} is at most a factor $O(\sqrt{|V''\setminus R|}\log|R|)$ from the fractional optimum OPT_{FDST}, that is

$$c''(T_{DST}) = \sum_{v\in V(T_{DST})} c''(v) = O(\sqrt{|V''\setminus R|}\log|R|)OPT_{FDST},$$

see Theorem 4.1.[1] By applying this algorithm to our instance I_{DST} of **DST**, we obtain a tree T_{DST} that is rooted at r and spans all the nodes in R. The costs of T_{DST} is

$$\begin{aligned}
c''(T_{DST}) &= O(\sqrt{|V''\setminus R|}\log|R|)OPT_{FDST}\\
&= O(\sqrt{|V|}\log|R|)OPT_{FDST}\\
&= O(\delta B\sqrt{|V|}\log|R|),
\end{aligned}$$

as $V'' \setminus R = V$ and $OPT_{FDST} \leq \delta B$. This concludes the proof. □

We now prove Theorem 3.2.

Proof of Theorem 3.2. We first compute an optimal solution $\{y_v, f_P^v\}_{v\in V', P\in\mathcal{P}_v}$ for the Linear Program (LP-DCBC). Let $Z \subseteq W$ be the set of nodes in W that in solution $\{y_v, f_P^v\}_{v\in V', P\in\mathcal{P}_v}$ receive at least $\frac{1}{|X|^2}$ amount of flow from r, i.e., for any $w \in Z$, $y_w \geq \frac{1}{|X|^2}$.

[1] Here we ignore the term $F = \max_{v\in V} dist(r, v)$ because G is B-proper and hence $F \leq B$.

The overall prize accrued by all nodes in Z is

$$\sum_{w\in Z} p_w \geq \sum_{w\in Z} y_w p_w = OPT - \sum_{w\in W\setminus Z} y_w p_w$$
$$\geq \left(1 - \sum_{w\in W\setminus Z} y_w\right) OPT \geq \left(1 - |X|\cdot\frac{1}{|X|^2}\right) OPT$$
$$= \left(1 - \frac{1}{|X|}\right) OPT,$$

where the first inequality holds as $y_w \leq 1$, for each $w \in Z$, the second inequality holds as the prize of each node is no more than OPT and the third inequality holds because each node $w \in W \setminus Z$ has $y_w < \frac{1}{|X|^2}$ and $|W \setminus Z| \leq |W| = |X|$.

From now on we only consider the prize accrued by nodes in Z, which results in losing a factor of at most $1 - \frac{1}{|X|} = \Theta(1)$ with respect to the optimum of (LP-DCBC). To simplify the reading, we ignore this constant factor and assume that $\sum_{w\in Z} y_w p_w = OPT$.

We partition the nodes of Z into k disjoint sets $Z_1, \ldots, Z_k$ defined as $Z_i = \left\{w \in Z : y_w \in \left(\frac{1}{2^i}, \frac{1}{2^{i-1}}\right]\right\}$, for each $i \in [k]$. It is easy to see that $k = O(\log |X|)$ such sets are enough to cover all nodes of Z. In fact, if the smallest value of y_w for a node $w \in Z$ is in the interval $\left(\frac{1}{2^k}, \frac{1}{2^{k-1}}\right]$, then, since $y_w \geq \frac{1}{|X|^2}$, we have $\frac{1}{2^{k-1}} \geq \frac{1}{|X|^2}$, and hence $2^{k-1} \leq |X|^2$ and $k \leq 2\log|X| + 1$.

We distinguish between two cases by dividing Z into two parts $Z_A = \bigcup_{i=1}^{\lfloor \log\log|X|\rfloor} Z_i$ and $Z_B = Z \setminus Z_A = \bigcup_{i=\lfloor\log\log|X|\rfloor+1}^{k} Z_i$. Since $\sum_{w\in Z} y_w p_w = OPT$, we must have $\sum_{w\in Z_A} y_w p_w \geq \frac{OPT}{2}$ or $\sum_{w\in Z_B} y_w p_w \geq \frac{OPT}{2}$.

(1) $\sum_{w\in Z_A} y_w p_w \geq \frac{OPT}{2}$. In this case, we consider the set of nodes in Z_A as the set of terminals R in Lemma 3.3. Since $y_w \geq 1/2^{\lfloor\log\log|X|\rfloor} \geq 1/2^{\log\log|X|} = 1/\log|X|$, for each $w \in Z_A$, in Lemma 3.3 we can set $\delta = \log|X|$. Therefore, by applying the algorithm in Lemma 3.3, we obtain a tree T rooted at r that spans all the nodes in Z_A and costs $c'(T) = O(B\sqrt{|V|}\log^2|X|)$. Moreover, as T spans all the nodes in Z_A, its prize is at least

$$p'(T) = \sum_{v\in V(T)} p'(v) \geq \sum_{w\in Z_A} p_w \geq \sum_{w\in Z_A} y_w p_w \geq \frac{OPT}{2},$$

by the case assumption and monotonicity of the prize function. Therefore, the ratio between prize and cost of T is $\frac{p'(T)}{c'(T)} = \Omega\left(\frac{OPT}{B\sqrt{|V|}\log^2|X|}\right)$.

(2) $\sum_{w\in Z_B} y_w p_w \geq \frac{OPT}{2}$. Since $k \leq 2\log|X| + 1$, there must be an index i between $\lfloor\log\log|X|\rfloor + 1$ and $2\log|X| + 1$ such that

$$\sum_{w\in Z_i} y_w p_w \geq \frac{OPT/2}{2\log|X| - \lfloor\log\log|X|\rfloor + 1} \geq \frac{OPT}{4\log|X|},$$

for $|X|$ sufficiently large. Let $p'(Z_i)$ be the sum of prizes of all the nodes in Z_i. Then,

$$p'(Z_i) = \sum_{w\in Z_i} p_w \geq 2^{i-1} \sum_{w\in Z_i} y_w p_w \geq 2^{i-1}\frac{OPT}{4\log|X|}, \quad (6)$$

since $y_w \in \left(\frac{1}{2^i}, \frac{1}{2^{i-1}}\right]$, for each $w \in Z_i$. Moreover, since $i \geq \lfloor\log\log|X|\rfloor + 1 \geq \log\log|X|$, then

$$p'(Z_i) \geq 2^{i-1}\frac{OPT}{4\log|X|} \geq 2^{\log\log|X|-1}\frac{OPT}{4\log|X|}$$
$$= \frac{\log|X|}{2}\frac{OPT}{4\log|X|} = \Omega(OPT). \quad (7)$$

Similarly to the previous case, we apply the algorithm in Lemma 3.3, considering Z_i as set of terminals and $\delta = 2^i$, since $y_w \geq 1/2^i$, for each $w \in Z_i$. The tree T computed by the algorithm in the lemma has cost $c'(T) = O(2^i B\sqrt{|V|}\log|X|)$ and, since it spans all the nodes in Z_i, has prize $p'(T) \geq p'(Z_i) \geq 2^{i-1}\frac{OPT}{4\log|X|}$, by Inequality (6). Therefore, the prize-to-cost ratio of T is $\frac{p'(T)}{c'(T)} = \Omega\left(\frac{OPT}{B\sqrt{|V|}\log^2|X|}\right)$. Moreover, by Inequality (7), $p'(T) \geq p'(Z_i) = \Omega(OPT)$. □

Trimming Process

In the previous step, we computed an out-tree T of G' rooted at r whose prize is $\Omega(OPT)$. If the cost of T satisfies the budget constraint, this gives a constant approximation factor. However, the cost of T can exceed the budget B. In this case, we can exploit the fact that the ratio between prize and cost of T is bounded by $\gamma = \frac{p'(T)}{c'(T)} = \Omega\left(\frac{OPT}{B\sqrt{|V|}\log^2|X|}\right)$. In fact, this property allows us to use the trimming process introduced in the following lemma by Bateni et al. [1] for the node-weighted budgeted problem in undirected graphs.

Lemma 3.4 (Lemma 3 in [1]). *Let T be a tree rooted at r with prize-to-cost ratio $\gamma = \frac{p(T)}{c(T)}$. Suppose the underlying graph is B-proper for r and for $\epsilon \in (0, 1]$ the cost of the tree is at least $\frac{\epsilon B}{2}$. One can find a tree $\hat{T}$ containing r with prize-to-cost ratio at least $\frac{\epsilon\gamma}{4}$ such that $\epsilon B/2 \leq c(\hat{T}) \leq (1+\epsilon)B$.*

Note that the above lemma has been introduced for (undirected) rooted trees, but it is easy to see that it can be extended to rooted out-trees, see e.g. [5]. If $c'(T) > B$, we apply to T the trimming process of Lemma 3.4 and obtain another out-tree $\hat{T}$ of G' with cost between $\frac{\epsilon B}{2}$ and $(1+\epsilon)B$ and prize-to-cost ratio $\frac{p'(\hat{T})}{c'(\hat{T})} \geq \frac{\epsilon\gamma}{4}$, for any $\epsilon \in (0, 1]$. Tree $\hat{T}$ violates the budget at most by a factor $1 + \epsilon$. Moreover, the prize of $\hat{T}$ is $p'(\hat{T}) \geq \frac{\epsilon\gamma}{4}c'(\hat{T}) = \Omega\left(\frac{\epsilon OPT}{B\sqrt{|V|}\log^2|X|}c'(\hat{T})\right)$. Since $c'(\hat{T}) \geq \epsilon B/2$ and $OPT \geq p(T_B^*)$, then $p'(\hat{T}) = \Omega\left(\frac{\epsilon^2 p(T_B^*)}{\sqrt{|V|}\log^2|X|}\right)$.

It remains to turn the tree $\hat{T}$ of G' into a tree of G with the same prize and cost by taking the maximal subtree of $\hat{T}$ containing only nodes in V. This results in the following theorem.

Theorem 3.5. *Problem **DCBC** admits a polynomial time bicriteria $\left(1+\epsilon, O\left(\frac{\sqrt{|V|}\log^2|X|}{\epsilon^2}\right)\right)$-approximation algorithm, for any $\epsilon \in (0, 1]$.*

The following corollary follows since we can reduce any instance of the directed budgeted rooted out-tree maximization problem

(**DBOM**) to an instance of **DCBC** where each node of the graph is associated with a distinct singleton set and hence $|X| = |V|$.

Corollary 3.6. *Problem **DBOM** admits a polynomial time bi-criteria* $\left(1+\epsilon, O\left(\frac{\sqrt{|V|}\log^2|V|}{\epsilon^2}\right)\right)$*-approximation algorithm, for any* $\epsilon \in (0,1]$.

The Case of Undirected Graphs

As **UCBC** is a special case of **DCBC**, we can use our algorithm in Theorem 3.5 to obtain a bicriteria $\left(1+\epsilon, O\left(\frac{\sqrt{|V|}\log^2|X|}{\epsilon^2}\right)\right)$-approximation for **UCBC**. We can show that a small modification of the same algorithm actually yields an improved approximation of $O\left(\frac{\log(|V|+|X|)\log|X|}{\epsilon^2}\right)$, with a budget violation of $1+\epsilon$, for any $\epsilon \in (0,1]$. The main difference consists in using, in the algorithm of Lemma 3.3, the $O(\log(|V''|))$-approximation algorithm by Klein and Ravi [18] for the node-weighted Steiner tree problem in *undirected* graphs instead of our $O(\sqrt{|V''\setminus R|}\log|R|)$-approximation algorithm for the same problem on *directed* graphs, where V'' and R are the set of nodes and terminals in the instance of a node-weighted Steiner tree problem instance. The approximation ratio can be shown using the same analysis used for **DCBC**, with this difference. (see full paper for more details).

Also in the undirected case, we can reduce an instance of **BNS** to an instance of **UCBC** where $|X| = |V|$. Therefore, our algorithm for **UCBC** is a bicriteria $\left(1+\epsilon, O\left(\frac{\log(|V|+|X|)\log|X|}{\epsilon^2}\right)\right)$-approximation for **BNS**, for any $\epsilon \in (0,1]$.

4 THE NODE-WEIGHTED STEINER TREE PROBLEM IN DIRECTED GRAPHS

In this section, we present a polynomial time approximation algorithm for **DST** with approximation ratio $O(\sqrt{|V|}\log|V|)$, where V is the set of nodes in the graph. More precisely, the cost of the tree computed by our algorithm is a factor $O(\sqrt{|V\setminus R|}\log|R|)$ far from the optimum of its standard flow-based linear programming relaxation given in (LP-DST) plus the maximum distance from the root to a node, where R is the set of terminals. The algorithm is used as a subroutine in the previous section but might be of its own interest. Formally, we show the following theorem.

Theorem 4.1. *Problem **DST** admits a* $O\left((1+\epsilon)\sqrt{|V\setminus R|}\log|R|\right)$*-approximation algorithm whose running time is polynomial in the input size and in* $1/\epsilon$*, for any* $\epsilon > 0$*. Moreover, the cost of the tree computed by the algorithm is* $O\left((OPT+F)\sqrt{|V\setminus R|}\log|R|\right)$*, where* OPT *is the optimum of* (LP-DST) *and* $F = \max_{v\in V} dist(r,v)$.

We prove Theorem 4.1 in what follows. Let T^* be an optimal solution to **DST**. We use the standard flow-based linear programming relaxation for **DST** given in (LP-DST) in Section 3. For the sake of completeness, we report the linear program below.[2]

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{v\in V} x_v c_v && && \text{(LP-DST)}\\
\text{subject to} \quad & \textstyle\sum_{P\in\mathcal{P}_t} g_P^t = 1, && \forall t\in R && (8)\\
& \textstyle\sum_{P\in\mathcal{P}_t: v\in P} g_P^t \le x_v, && \forall v\in V, t\in R && (9)\\
& 0 \le x_v \le 1, && \forall v\in V\\
& 0 \le g_P^t \le 1, && \forall t\in R, P\in\mathcal{P}_t.
\end{aligned}$$

It is easy to see that OPT, the optimum for (LP-DST), provides a lower bound to $c(T^*)$. In fact, the solution to (LP-DST) in which x_v is set to 1 if $v \in V(T^*)$ and 0 otherwise, and g_P^t is set to 1 if P is the unique path from r to t in T^* and to 0 otherwise, is feasible for (LP-DST) and has value $\sum_{v\in V} x_v c_v = c(T^*)$.

Let $\{x_v\}_{v\in V}$ be an optimal solution for (LP-DST) and let $S \subseteq V$ be the set of all nodes v with $x_v > 0$. Let $U \subseteq S$ be the set of all nodes with $x_v \ge \frac{1}{\sqrt{|V\setminus R|}}$ for any $v \in U$. Note that nodes in R and r belong to U since we need to send one unit of flow from r to any terminal by Constraint (8). We call a terminal $t \in R$ a *cheap terminal* if there exists a path from r to t in $G[U]$. We call a terminal $t \in R$ an *expensive terminal* otherwise. Let CH and EX be the set of all cheap and expensive terminals in R, respectively.

We now show that we can compute in polynomial time two trees spanning CH and EX, resp., and then we show how to merge the two trees into a single tree with cost $O\left((OPT+F)\sqrt{|V\setminus R|}\log|R|\right)$. We first show how to compute a tree T^{CH} rooted at r spanning all the cheap terminals CH with cost $c(T^{CH}) \le \sqrt{|V\setminus R|}\cdot OPT$. The proof of the next lemma is given in the full version of the paper.

Lemma 4.2. *There exists a polynomial time algorithm that finds a tree* T^{CH} *rooted at* r *spanning all the cheap terminals* CH *with cost* $c(T^{CH}) \le \sqrt{|V\setminus R|}\cdot OPT$.

We next show how to compute in polynomial time a tree T^{EX} rooted at r spanning all the expensive terminals EX with cost $c(T^{EX}) = O\left((OPT+F)\sqrt{|V\setminus R|}\log|R|\right)$. The algorithm to build T^{EX} can be summarized as follows. We first compute, for each $t \in EX$, the set X_t of nodes w in $S\setminus U$ for which there exists a path P from w to t that uses only nodes in $U\cup\{w\}$, i.e., $V(P)\setminus\{w\} \subseteq U$. Then, we compute a small-size hitting set X' of all X_t. Finally, we connect r to the nodes of X' and the nodes of X' to those in EX in such a way that each node t in EX is reached from one of the nodes in X' that hits X_t. The bound on the cost of T^{EX} follows from the size of X' and from the cost of nodes in U.

Lemma 4.3. *There exists a polynomial time algorithm that finds a tree* T^{EX} *rooted at* r *spanning all the expensive terminals* EX *with cost* $c(T^{EX}) \le (OPT+F)\sqrt{|V\setminus R|}\log|R|$.

Proof. Let $U' \subseteq S$ be the set of all nodes v with $0 < x_v < \frac{1}{\sqrt{|V\setminus R|}}$, i.e., $U' = S\setminus U$. Recall that for any expensive terminal $t \in EX$, we define X_t as the set of nodes w in U' such that there exists a path from w to t in $G[U\cup\{w\}]$.

We first show a lower bound on the size of sets X_t, for each $t \in EX$, which will allow us to compute a small hitting set of all

[2]Note that here the graph is denoted as $G = (V, A)$ instead of $G'' = (V'', A'')$.

such sets. The proof of the next claim is given in the full version of the paper.

Claim 4.4. *$|X_t| \geq \sqrt{|V \setminus R|}$, for each $t \in EX$.*

We use the following well-known result (see, e.g., Lemma 3.3 in [2]) to find a small set of nodes that hits all sets X_t, for all $t \in EX$.

Claim 4.5. *Let V' be a set of M elements and $\sum = (X'_1, \ldots, X'_N)$ be a collection of subsets of V' such that $|X'_i| \geq L$, for each $i \in [N]$. There is a deterministic algorithm that runs in polynomial time in N and M and finds a subset $X' \subseteq V'$ with $|X'| \leq (M/L) \ln N$ and $X' \cap X'_i \neq \emptyset$ for all $i \in [N]$.*

Thanks to Claim 4.4, we can use the algorithm of Claim 4.5 to find a set $X' \subseteq \bigcup_{t \in EX} X_t$ such that $X' \cap X_t \neq \emptyset$, for all $t \in EX$, whose size is at most $|X'| \leq \frac{|V \setminus R| \log |R|}{\sqrt{|V \setminus R|}} = \sqrt{|V \setminus R|} \log |R|$, where the parameters of Claim 4.5 are $L = \sqrt{|V \setminus R|}$, $N = |EX| \leq |R|$, and $M = \left| \bigcup_{t \in EX} X_t \right| \leq |V \setminus R|$, since $x_t = 1$, for each $t \in R$, and hence no node in R can belong to $\bigcup_{t \in EX} X_t \subseteq U'$.

Since for any $t \in EX$ and any $w \in X_t$ there exists a path from w to t in $G[U \cup \{w\}]$ and $X' \cap X_t \neq \emptyset$, then there exists at least a node $w \in X'$ for which there is a path from w to t in $G[U \cup \{w\}]$.

Now, for each $w \in X'$, we find a shortest path from r to w in G. Let $\mathcal{P}_1$ be the set of all these shortest paths. We also select, for each $t \in EX$, an arbitrary node w in $X' \cap X_t$ and compute a shortest path from w to t in $G[U \cup \{w\}]$. Let $\mathcal{P}_2$ be the set of all these shortest paths. Let $V(\mathcal{P}_1)$ and $V(\mathcal{P}_2)$ denote the union of all nodes of the paths in $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, and let G^{EX} be the graph induced by all the nodes in $V(\mathcal{P}_1) \cup V(\mathcal{P}_2)$. We compute a tree T^{EX} rooted at r spanning G^{EX}. Note that such a tree exists as in G^{EX} we have for each $w \in X'$ a path from r to w and, for each terminal $t \in EX$, at least a path from one of the nodes in X' to t.

We next move to bound the cost of T^{EX}; Indeed, we bound the cost of all nodes in G^{EX}. Since $|X'| \leq \sqrt{|V \setminus R|} \log |R|$ and $dist(r, v) \leq F$ for any node v, then $c(V(\mathcal{P}_1)) \leq F\sqrt{|V \setminus R|} \log |R|$. Since $x_v \geq \frac{1}{\sqrt{|V \setminus R|}}$ for any $v \in U$, and $\sum_{v \in U} x_v c_v \leq \sum_{v \in S} x_v c_v \leq OPT$, then $c(U) = \sum_{v \in U} c_v \leq \sqrt{|V \setminus R|} \cdot OPT$. Therefore, since $V(\mathcal{P}_2) \setminus X' \subseteq U$, then $c(V(\mathcal{P}_2) \setminus X') \leq c(U) \leq \sqrt{|V \setminus R|} \cdot OPT$. Overall, G^{EX} costs at most $(OPT + F)\sqrt{|V \setminus R|} \log |R|$. This finishes the proof of Lemma 4.3. □

We can now prove Theorem 4.1.

Proof of Theorem 4.1. Since both T^{EX} and T^{CH} are rooted at r, we can find a tree T rooted at r spanning all nodes $V(T^{EX}) \cup V(T^{CH})$.

By Lemmas 4.2 and 4.3 we have that the cost of T is $c(T) = O\left((OPT + F)\sqrt{|V \setminus R|} \log |R|\right)$. This shows the second part of the statement.

To show the bound on the approximation ratio, we observe that $OPT \leq c(T^*)$. Moreover, we can assume that $F \leq (1+\epsilon)c(T^*)$ since we can remove from the graph all the nodes v such that $dist(r, v) > (1+\epsilon)c(T^*)$ by estimating the value of $c(T^*)$ using a binary search. Therefore, the cost of T is $c(T) = O\left(\sqrt{|V \setminus R|} \log |R|\right) c(T^*)$. □

5 DISCUSSION AND FUTURE RESEARCH

DCBC and **CBC** are basic combinatorial optimization problems with many applications in diverse areas such as logistics, wireless sensor networks, and bioinformatics. Besides their relevance, their approximation properties still need to be better understood. In this paper, we make an important step forward, providing the first algorithms for **DCBC** and **CBC** with sublinear approximation ratios that significantly improve over the current best algorithms. Our results also imply an improved approximation for the particular case of additive prize function, **DBOM**.

The most interesting but very ambitious research question is whether there is a polynomial lower bound on the approximability of **DCBC**. In other words, whether it is hard to compute in polynomial time a solution that is asymptotically better than a polynomial factor from the optimum. The same question for the directed Steiner tree problem has been open for a long time. However, it is known that the integrality gap of the standard flow-based LP relaxation for **DCBC** is unbounded if no budget violation is allowed [1] and has a polynomial lower bound for the directed Steiner tree problem [22]. This suggests that we cannot significantly improve our approximation factors for **DCBC** by using the linear relaxation (LP-DCBC). Using LP-hierarchies [10, 27] could be a promising research direction to improve our approximation factors. For the Directed Steiner Network, it is known that the integrality gap of the Lasserre Hierarchy has a polynomial lower bound [8]. An even harder research question is to find a lower bound on the approximation of **CBC**.

The techniques introduced in this paper might be useful to approximate other more general network design problems. One interesting example is the case when the prize function is a monotone submodular set function of the nodes. In this case, the best algorithm is the one in [6] that achieves an approximation factor of $O(\frac{1}{\epsilon^3}\sqrt{B})$-approximation algorithm with a budget violation of a factor $1 + \epsilon$, for any $\epsilon \in (0, 1]$. Our algorithms cannot directly be applied to this case because the linear program (LP-DCBC) does not give an upper bound to the optimum. Therefore, the first step in using our techniques should be to find a suitable linear relaxation.

ACKNOWLEDGMENTS

This work was partially supported by: PNRR MUR project GAMING "Graph Algorithms and MinINg for Green agents" (PE0000013, CUP D13C24000430001); the European Union - NextGenerationEU under the Italian Ministry of University and Research National Innovation Ecosystem grant ECS00000041 - VITALITY - CUP: D13C21000430001; the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP J33C22002880001, partnership on "Telecommunications of the Future"(PE00000001 - program "RESTART"), project MoVeOver/SCHEDULE ("Smart interseCtions witH connEcteD and aUtonomous vehicLEs", CUP J33C22002880001); RASTA project - ARS01_00540, funded by the Italian Ministry of Research PNR 2015-2020; The Italian Institute for Advanced Mathematics and the Italian National Group for Scientific Computation (GNCS-INdAM). The first author acknowledges the support of the MUR (Italy) Department of Excellence 2023–2027.

REFERENCES

[1] Mohammad Hossein Bateni, Mohammad Taghi Hajiaghayi, and Vahid Liaghat. 2018. Improved Approximation Algorithms for (Budgeted) Node-weighted Steiner Problems. *SIAM J. Comput.* 47, 4 (2018), 1275–1293.

[2] Timothy M. Chan. 2007. More Algorithms for All-Pairs Shortest Paths in Weighted Graphs. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing (STOC '07)*. Association for Computing Machinery, 590–598. https://doi.org/10.1145/1250790.1250877

[3] Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. 1999. Approximation Algorithms for Directed Steiner Problems. *J. Algorithms* 33, 1 (1999), 73–91.

[4] Xuefeng Chen, Xin Cao, Yifeng Zeng, Yixiang Fang, and Bin Yao. 2020. Optimal Region Search with Submodular Maximization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI*, Christian Bessiere (Ed.). 1216–1222.

[5] Gianlorenzo D'Angelo and Esmaeil Delfaraz. 2024. Approximation Algorithms for Node-Weighted Directed Steiner Problems. In *Proceedings of the 35th International Workshop on Combinatorial Algorithms (IWOCA2024) (Lecture Notes in Computer Science, Vol. 14764)*, Adele Anna Rescigno and Ugo Vaccaro (Eds.). Springer, 273–286. https://doi.org/10.1007/978-3-031-63021-7_21

[6] Gianlorenzo D'Angelo, Esmaeil Delfaraz, and Hugo Gilbert. 2022. Budgeted Out-Tree Maximization with Submodular Prizes. In *Proceedings of the 33rd International Symposium on Algorithms and Computation, ISAAC 2022, December 19-21, 2022, Seoul, Korea (LIPIcs, Vol. 248)*, Sang Won Bae and Heejin Park (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 9:1–9:19.

[7] Gianlorenzo D'Angelo, Esmaeil Delfaraz, and Hugo Gilbert. 2022. Computation and Bribery of Voting Power in Delegative Simple Games. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, May 9-13, 2022*, Piotr Faliszewski, Viviana Mascardi, Catherine Pelachaud, and Matthew E. Taylor (Eds.). International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 336–344.

[8] Michael Dinitz, Yasamin Nazari, and Zeyu Zhang. 2020. Lasserre Integrality Gaps for Graph Spanners and Related Problems. In *Approximation and Online Algorithms - 18th International Workshop, WAOA 2020 (Lecture Notes in Computer Science, Vol. 12806)*, Christos Kaklamanis and Asaf Levin (Eds.). Springer, 97–112.

[9] Uriel Feige. 1998. A Threshold of ln *n* for Approximating Set Cover. *Journal of ACM* 45, 4 (1998), 634–652. https://doi.org/10.1145/237814.237977

[10] Zachary Friggstad, Jochen Könemann, Young Kun-Ko, Anand Louis, Mohammad Shadravan, and Madhur Tulsiani. 2014. Linear Programming Hierarchies Suffice for Directed Steiner Tree. In *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8494)*, Jon Lee and Jens Vygen (Eds.). Springer, 285–296.

[11] Rohan Ghuge and Viswanath Nagarajan. 2020. Quasi-Polynomial Algorithms for Submodular Tree Orienteering and Other Directed Network Design Problems. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, Shuchi Chawla (Ed.). SIAM, 1039–1048.

[12] Michel X. Goemans and Young-Soo Myung. 1993. A catalog of steiner tree formulations. *Networks* 23 (1993), 19–28. Issue 1.

[13] Sudipto Guha, Anna Moss, Joseph Naor, and Baruch Schieber. 1999. Efficient Recovery from Power Outage (Extended Abstract). In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*. ACM, 574–582.

[14] William C. Hahn and Robert A. Weinberg. 2002. Modelling the molecular circuitry of cancer. *Nat Rev Cancer* 2 (2002), 331–341. https://doi.org/10.1038/nrc795

[15] Dorit S. Hochbaum and Xu Rao. 2020. Approximation algorithms for connected maximum coverage problem for the discovery of mutated driver pathways in cancer. *Inf. Process. Lett.* 158 (2020), 105940.

[16] Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. 1999. The budgeted maximum coverage problem. *Inform. Process. Lett.* 70, 1 (1999), 39–45. https://doi.org/10.1016/S0020-0190(99)00031-9

[17] Samir Khuller, Manish Purohit, and Kanthi K. Sarpatwar. 2020. Analyzing the Optimal Neighborhood: Algorithms for Partial and Budgeted Connected Dominating Set Problems. *SIAM J. Discret. Math.* 34, 1 (2020), 251–270.

[18] Philip N. Klein and R. Ravi. 1995. A Nearly Best-Possible Approximation Algorithm for Node-Weighted Steiner Trees. *J. Algorithms* 19, 1 (1995), 104–115.

[19] Guy Kortsarz and Zeev Nutov. 2011. Approximating some network design problems with node costs. *Theor. Comput. Sci.* 412, 35 (2011), 4482–4492.

[20] Tung-Wei Kuo, Kate Ching-Ju Lin, and Ming-Jer Tsai. 2015. Maximizing Submodular Set Function With Connectivity Constraint: Theory and Application to Networks. *IEEE/ACM Trans. Netw.* 23, 2 (2015), 533–546.

[21] Ioannis Lamprou, Ioannis Sigalas, and Vassilis Zissimopoulos. 2021. Improved Budgeted Connected Domination and Budgeted Edge-Vertex Domination. *Theor. Comput. Sci.* 858 (2021), 1–12.

[22] Shi Li and Bundit Laekhanukit. 2022. Polynomial Integrality Gap of Flow LP for Directed Steiner Tree. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*, Joseph (Seffi) Naor and Niv Buchbinder (Eds.). SIAM, 3230–3236.

[23] Yaakov Livne, Dor Atzmon, Shawn Skyler, Eli Boyarski, Amir Shapiro, and Ariel Felner. 2023. Optimally Solving the Multiple Watchman Route Problem with Heuristic Search. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, Noa Agmon, Bo An, Alessandro Ricci, and William Yeoh (Eds.). ACM, 905–913. https://doi.org/10.5555/3545946.3598728

[24] Ritwick Mishra, Jack Heavey, Gursharn Kaur, Abhijin Adiga, and Anil Vullikanti. 2023. Reconstructing an Epidemic Outbreak Using Steiner Connectivity. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 10 (Jun. 2023), 11613–11620. https://doi.org/10.1609/aaai.v37i10.26372

[25] Anna Moss and Yuval Rabani. 2007. Approximation Algorithms for Constrained Node Weighted Steiner Tree Problems. *SIAM J. Comput.* 37, 2 (2007), 460–481.

[26] Yingli Ran, Zhao Zhang, Ker-I Ko, and Jun Liang. 2016. An approximation algorithm for maximum weight budgeted connected set cover. *J. Comb. Optim.* 31, 4 (2016), 1505–1517.

[27] Thomas Rothvoß. 2011. Directed Steiner Tree and the Lasserre Hierarchy. *CoRR* abs/1111.5473 (2011). http://arxiv.org/abs/1111.5473

[28] Polina Rozenshtein, Aristides Gionis, B. Aditya Prakash, and Jilles Vreeken. 2016. Reconstructing an Epidemic Over Time. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. 1835–1844. https://doi.org/10.1145/2939672.2939865

[29] Fabio Vandin, Eli Upfal, and Benjamin J. Raphael. 2011. Algorithms for Detecting Significantly Mutated Pathways in Cancer. *J. Comput. Biol.* 18, 3 (2011), 507–522.

[30] Wenzheng Xu, Yueying Sun, Rui Zou, Weifa Liang, Qiufen Xia, Feng Shan, Tian Wang, Xiaohua Jia, and Zheng Li. 2022. Throughput Maximization of UAV Networks. *IEEE/ACM Transactions on Networking* 30, 2 (2022), 881–895. https://doi.org/10.1109/TNET.2021.3125982

[31] Nan Yu, Haipeng Dai, Guihai Chen, Alex X. Liu, Bingchuan Tian, and Tian He. 2021. Connectivity-Constrained Placement of Wireless Chargers. *IEEE Trans. Mob. Comput.* 20, 3 (2021), 909–927.