International Scholarly Research Network
ISRN Discrete Mathematics
Volume 2012, Article ID 347430, 11 pages
doi:10.5402/2012/347430

Research Article

Spider Covers and Their Applications

Filomena De Santis,[1] Luisa Gargano,[1] Mikael Hammar,[2] Alberto Negro,[1] and Ugo Vaccaro[1]

[1] *Dipartimento di Informatica, Università di Salerno, 84084 Fisciano, Italy*
[2] *Research and Development, Apptus Technologies AB, Ideon, 223 70 Lund, Sweden*

Correspondence should be addressed to Ugo Vaccaro, uvaccaro@gmail.com

Received 27 September 2012; Accepted 19 October 2012

Academic Editors: G. Hahn and W. F. Klostermeyer

We introduce two new combinatorial optimization problems: the Maximum Spider Problem and the Spider Cover Problem; we study their approximability and illustrate their applications. In these problems we are given a directed graph $G = (V, E)$, a distinguished vertex s, and a family D of subsets of vertices. A *spider* centered at vertex s is a collection of arc-disjoint paths all starting at s but ending into pairwise distinct vertices. We say that a spider covers a subset of vertices X if at least one of the endpoints of the paths constituting the spider other than s belongs to X. In the Maximum Spider Problem the goal is to find a spider centered at s that covers the maximum number of elements of the family D. Conversely, the Spider Cover Problem consists of finding the minimum number of spiders centered at s that covers all subsets in D. We motivate the study of the Maximum Spider and Spider Cover Problems by pointing out a variety of applications. We show that a natural greedy algorithm gives a 2-approximation algorithm for the Maximum Spider Problem and a $(\log|\mathfrak{D}| + 1)$-approximation algorithm for the Spider Cover Problem.

1. Introduction

Given a digraph $G = (V, E)$ and a vertex $s \in V$, a *spider* centered at s is a subgraph S of G consisting of arc-disjoint paths sharing the initial vertex s and ending into pairwise distinct vertices. The vertex s is called *the center* of the spider. The endpoints of the paths composing the spider S—other than the center s—are called the *terminals* of the spider. In other words, a spider is a subdivision of $K_{1,m}$, where m is the number of terminals. Given a spider S, we say that S reaches a vertex $x \in V$ if x is a terminal of S; we say that the spider S covers a subset of vertices $D \subseteq V$ if S reaches at least a vertex in D.

In this paper we consider the approximability of the following problems.

Maximum Spider Problem (MSP)

We are given a digraph $G = (V, E)$, a distinguished node s, and a family $\mathfrak{D} \subseteq 2^{V\setminus\{s\}}$ of subsets of vertices. The objective is to find a spider S centered at s such that the number of subsets $D \in \mathfrak{D}$ covered by S is maximum among all possible spiders centered at s.

We also consider the related minimization problem, where one wants to cover all the elements of $\mathfrak{D}$.

Spider Cover Problem (SCP)

As before, we are given a digraph $G = (V, E)$, a distinguished vertex $s \in V$, and a family $\mathfrak{D} \in 2^{V\setminus\{s\}}$ of subsets of vertices. The goal is to find a minimum cardinality collection of spiders centered at s such that each subset $D \in \mathfrak{D}$ is covered by at least a spider in the collection.

1.1. Motivations

The Maximum Spider and the Spider Cover Problems are far reaching generalizations and unifications of several Maximum Coverage and Set Cover Problems which, in turn, are fundamental algorithmic and combinatorial problems that arise frequently in a variety of settings [3]. To start, recall that in the basic formulation of the Maximum Coverage Problem [3], one is given a ground set X, a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$, where each $S_j \subseteq X$, for $j = 1, \ldots, m$, and an integer k. The goal is to find $\ell \leq k$ sets $S_{i_1}, \ldots, S_{i_\ell}$ such that the cardinality $|\cup_{j=1}^{\ell} S_{i_j}|$ of their union is maximum. To see that the Maximum Coverage Problem is a very particular case of the Maximum Spider Problem, let us consider the digraph $G = (V, E)$ of Figure 1, with node set $V = \{s, x_1, \ldots, x_k, S_1, \ldots, S_m\}$. The vertex s is connected to each of the nodes $x_1, \ldots, x_k$, and each x_i is connected to every S_j, for $i = 1, \ldots, k$ and $j = 1, \ldots, m$. The family $\mathfrak{D} \subseteq 2^{V-\{s\}}$ is defined as $\mathfrak{D} = \{D_u : u \in X\}$, where $D_u = \{S_i : u \in S_i\}$. One can see that the Maximum Spider Problem in G is equivalent to the Maximum Coverage Problem on the original instance X, $\mathcal{S}$, and k. To that purpose, let us proceed as follows. Let S be a spider in G that covers a maximum number μ of subsets $D \in \mathfrak{D}$. Let $D_{u_1}, \ldots, D_{u_\mu}$ be these subsets. By our definition of spider cover, the (at most k) terminals of S in G correspond to some $S_{i_1}, \ldots, S_{i_\ell}$, $\ell \leq k$, such that for any $D_{u_t} \in \{D_{u_1}, \ldots, D_{u_\mu}\}$ there exists $S_{i_j} \in \{S_{i_1}, \ldots, S_{i_\ell}\}$ for which $S_{i_j} \in D_{u_t}$. This implies that for any $u_t \in \{u_1, \ldots, u_\mu\}$ there exists S_{i_j} such that $u_t \in S_{i_j}$, consequently $\cup_{j=1}^{\ell} S_{i_j} \supseteq \{u_1, \ldots, u_\mu\}$ and $|\cup_{j=1}^{\ell} S_{i_j}| \geq \mu$. Conversely, let $S_{i_1}, \ldots, S_{i_\ell}$, $\ell \leq k$, be a solution to the Maximum Coverage Problem on the original instance X, $\mathcal{S}$, and k. Let $\cup_{j=1}^{\ell} S_{i_j} = \{u_1, \ldots, u_\mu\}$. Consider now the spider s in G starting at s and having terminal nodes equal to $S_{i_1}, \ldots, S_{i_\ell}$. By definition, spider S covers at least the μ subsets $D_{u_1}, \ldots, D_{u_\mu}$.

Thus, the Maximum Coverage Problem corresponds to the Maximum Spider Problem in a very simple digraph G. By allowing more flexibility in the structure of G, one can describe many more combinatorial optimization problems in this framework. For instance, Chekuri and Kumar in [4] considered the following generalization of Maximum Coverage.

Maximum Coverage with Group Budget Constraints (MCG) (see [4])

We are given a ground set X and a collection $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of X. We are also given sets $\mathcal{G}_1, \ldots, \mathcal{G}_\ell$, each $\mathcal{G}_i \subseteq \mathcal{S} = \{S_1, \ldots, S_m\}$, with $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ for $i \neq j$, and integer bounds

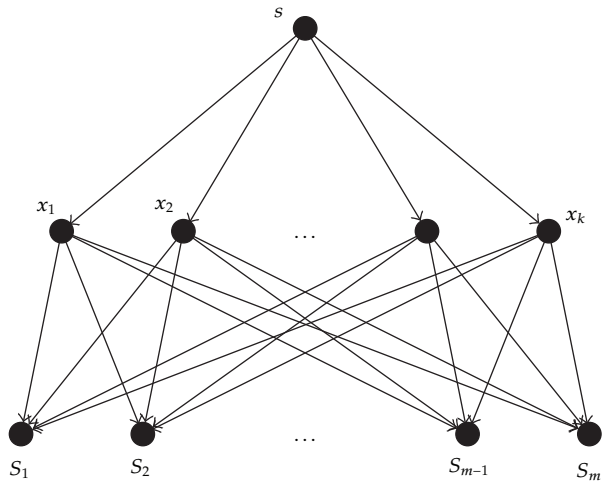

Figure 1: Maximum Coverage as a Maximum Spider Problem.

$k, k_1, \ldots, k_\ell$. A solution is a subset $\mathcal{H} \subseteq \{S_1, \ldots, S_m\}$, such that $|\mathcal{H}| \leq k$ and $|\mathcal{H} \cap \mathcal{G}_i| \leq k_i$, for $i = 1, \ldots, \ell$. The goal is to find a solution $\mathcal{H}$ such that $|\bigcup_{H \in \mathcal{H}} H|$ is maximized.

Before showing how MGC easily fits into our scenario, let us mention that the MGC problem itself was introduced and studied in [4] since it represents a useful generalization of several combinatorial optimization problems, like the multiple depot k-traveling repairmen problem with covering constraints [5] and the orienteering problem with time windows [6–8].

Given an instance $\langle X, \mathcal{G}_1, \ldots, \mathcal{G}_\ell, \{S_1, \ldots, S_m\}, k, k_1, \ldots, k_\ell \rangle$ of MCG, consider the digraph $G = (V, E)$ with vertex set

$$V = \left\{ s, x_1, \ldots, x_k, y_1^1, \ldots y_1^{k_1}, \ldots, y_\ell^1, \ldots, y_\ell^{k_\ell}, S_1, \ldots, S_m \right\}. \tag{1.1}$$

There is an edge from s to each x_i, $i = 1, \ldots, k$. Moreover, there is a complete bipartite graph between $\{x_1, \ldots, x_k\}$ and $\{y_1^1, \ldots y_1^{k_1}, \ldots, y_\ell^1, \ldots, y_\ell^{k_\ell}\}$ (with orientation of the edges going from the x's to the y's). Finally, there is a complete bipartite graph between the set $\mathcal{Y}_i = \{y_i^1, \ldots, y_i^{k_i}\}$ and the set $\mathcal{G}_i \subseteq \{S_1, \ldots, S_m\}$, for $i = 1, \ldots, \ell$ and, in case $\sum_{i=1}^{\ell} k_i < k$, there is a complete bipartite graph between $\{x_1, \ldots, x_k\}$ and $\mathcal{S} \setminus \bigcup_{i=1}^{\ell} \mathcal{G}_i$. As before, the family $\mathfrak{D}$ is defined as consisting of subsets of vertices $D_u = \{S_i : u \in S_i\}$, for each $u \in X$. Figure 2 below depicts the situation.

Again, it is not hard to see that MGC is equivalent to the Maximum Spider Problem in the graph G. At this point it should be clear that by variating the structure of the graph between the vertex s and the family of subsets $\{S_1, \ldots, S_m\}$, one can describe many more covering problems.

Just as the Maximum Spider Problem encompasses a variety of coverage problems formulated in term of maximization of the objective function, the related Spider Cover minimization problem includes particular cases variants and extensions of the well-known Set Cover Problem. One of such an extension was considered in [4, 9, 10].

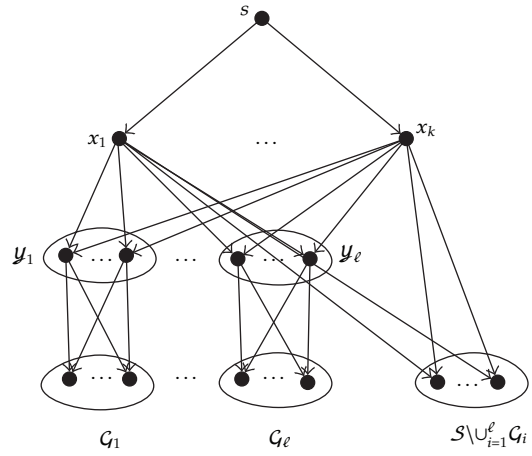

Figure 2: MGC as a Maximum Spider Problem.

Set Cover with Group Budget (SCG)

We are given a ground set X and a family $\mathcal{S} = \{S_1, \ldots, S_m\}$ of subsets of X. The family $\mathcal{S}$ is partitioned into subfamilies $\mathcal{G}_1, \ldots, \mathcal{G}_\ell$. The goal is to find an $\mathcal{H} \subseteq \mathcal{S}$ such that all elements of X are covered by sets in $\mathcal{H}$, and $\max_{i=1,\ldots,\ell} |\mathcal{H} \cap \mathcal{G}_i|$ is minimized.

Elkin and Kortsarz [9] studied the SCG problem as a preliminary tool for their multicasting algorithm in synchronous directed networks. Gargano et al. [10] studied the SCG problem in the context of multicasting in optical networks. Interestingly, Gargano et al. [10] also noticed that SCG naturally arises in airline scheduling problems [11]. We trust that the experienced reader can now appreciate the flexibility of our approach by checking that the SCG is equivalent to the Spider Cover problem in the graph shown in Figure 3. The family $\mathfrak{D}$ to cover is $\mathfrak{D} = \{D_u : u \in X\}$, where for each $u \in X$ we have $D_u = \{S \in \mathcal{S} : u \in S\}$.

In general, we expect that the capability of our approach to easily describe and deal with diverse requirements in covering problems to be quite useful. In any case, it seems to provide a nice and unified view of many different questions.

1.2. Our Results in Comparison with Previous Work

To the best of our knowledge, the Maximum Spider and the Spider Cover Problems have not been considered before, apart from the different special cases mentioned in the previous section. Our results are the following.

(1) We show that the greedy approach yields a 2-approximation algorithm for the Maximum Spider Problem. (In this paper approximation ratios for both maximization problems and minimization problems will be greater than 1). It is remarkable that we achieve the same approximation ratio obtained in [4] for the Maximum Coverage with Group Budget Constraints, although our Maximum Spider Problem is much more general. Since the Maximum Spider Problem contains the classical Maximum Coverage Problem as particular case, from results of [12] it follows that it is hard to approximate within a factor of $e/(e-1) - o(1)$, unless NP$\subset$ $D\text{TIME}(n^{\log\log n})$. In the paper [4] it is additionally proved that the approximation

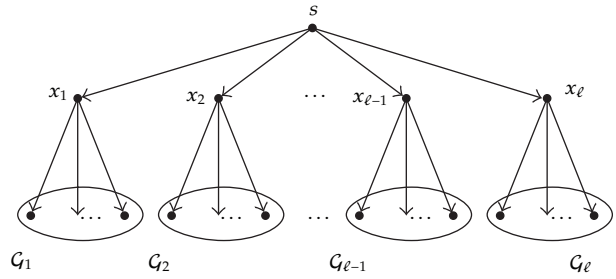

Figure 3: SCG as a Spider Cover Problem.

factor 2 is tight for their problem in the *oracle* model. Obviously, this tightness of analysis transfers also to our Maximum Spider Problem.

(2) We give a greedy algorithm for the Spider Cover Problem with approximation ratio $\log|\mathfrak{D}|+1$. Again, we match the results of [4, 9, 10], who obtained the same result in case the graph G is the simple tree of Figure 3. Since the Maximum Spider Problems include the Set Cover problem as a particular case, from [12] one gets a $(1-\epsilon)\ln|\mathfrak{D}|)$ factor for the hardness of its approximation, for any $\epsilon > 0$. We also observe that our algorithm for the Spider Cover Problem provides a $O(\log|\mathfrak{D}|)$-approximation algorithm for the Multicasting-to-Groups Problems considered in [10], extending the main result of the same paper from trees to general networks. The problem considered therein was to find a set of paths from a source node to at least one node in each subset of a set of groups $\mathfrak{D}$ and assignments of wavelengths to paths so that paths sharing a same physical link of the network are assigned different wavelengths. The goal is to minimize the number of wavelengths. It can be seen that the paths constituting the spiders covering the family $\mathfrak{D}$, and an assignment of different wavelengths to paths in different spiders, give an admissible solution to the Multicasting-to-Group problem in general optical networks.

2. A Greedy Algorithm for the Maximum Spider Problem

In this section we will present a 2-approximation greedy algorithm for the Maximum Spider Problem (MSP).

Given an instance $\langle G, s, \mathfrak{D}\rangle$ of the MSP, where $G = (V, E)$ is a digraph, s is a designated vertex in V, and $\mathfrak{D}$ is a family of subsets of $V \setminus \{s\}$, we say that the subsets of vertices $X \subseteq V$ are *reachable* if there exists a spider in G, with center in s, such that each node $v \in X$ is reached by such a spider. In other words, X is reachable if there is a spider in G whose set of terminals includes X. For any set $X \subseteq V$—not necessarily reachable—we define $C(X)$ as the number of elements in $\mathfrak{D}$ covered by X, that is,

$$C(X) = |\{D \in \mathfrak{D} : D \cap X \neq \emptyset\}|. \tag{2.1}$$

In terms of the function $C(\cdot)$, our original objective is essentially that of finding a reachable set X of maximum value $C(X)$.

For any $X, Y \subseteq V$, we define the *covering improvement* $C(Y \mid X)$ of Y over X as

$$C(Y \mid X) = C(X \cup Y) - C(X) = |\{D \in \mathfrak{D} : D \cap Y \neq \emptyset,\ D \cap X = \emptyset\}|. \tag{2.2}$$

Definition 2.1. Given a reachable set X we say that:

(1) a node $x \in V$ *improves* on X if $X \cup \{x\}$ is reachable;

(2) a node $x \in V$ *maximally improves* on X if $C(X \cup \{x\}) = \max_y C(X \cup \{y\})$, where the maximum is taken on all nodes y that improve on X;

(3) the set X is *maximal* if no node $x \in V \setminus X$ improves on X.

We can now describe the skeleton of our 2-approximation algorithm. (We point out that the algorithm could also stop as soon as it finds a first node maximally improving on X with the property that $C(\{x\} \mid X) = 0$. However, we let MAX_SP generate a maximal set X to make the analysis cleaner).

In the rest of this section we will show how to efficiently implement step 2. Of the above greedy algorithm and how to compute a spider centered at s and with set of terminals X, and we will also show that the number of sets in $\mathfrak{D}$ covered by the terminals in X is at least half of the optimum number.

Let us first check that the algorithm is polynomial.

Lemma 2.2. *The algorithm* MAX_SP$(G, s, \mathfrak{D})$ *is polynomial.*

Proof. In order to compute the node $x \in V \setminus X$ that maximally improves on X we proceed as follows. First, for each $y \in V \setminus X$ we check whether $X \cup \{y\}$ is reachable, that is, whether there is a spider centered at s and with set of terminals equal to $X \cup \{y\}$. This can be done by constructing a flow network (For undefined terminology about flows in networks, see for example [13]) from G, assigning the source node at s, connecting all nodes in $X \cup \{y\}$ to a sink node t, setting all flow capacities equal to 1, and by verifying whether or not in this flow network there exists a flow of value $|X| + 1$. This entire procedure can be performed clearly in polynomial time. Subsequently, among all y's for which $X \cup \{y\}$ is reachable, we compute the one that maximally improves on X by using the identity $C(X \cup \{y\}) = C(X) + C(\{y\} \mid X)$. Finally, the spider that reaches the set X,—output of the algorithm MAX_SP—is computed from the executions of the maximum flow algorithm, and it consists of all the flow paths from s to X with assigned flow value equal to 1. □

In order to show that Algorithm MAX_SP$(G, s, \mathfrak{D})$ is a 2-approximation algorithm for the Maximum Spider Problem, we first need the following technical result.

Lemma 2.3. *Let* $\langle G = (V, E), s, \mathfrak{D}\rangle$ *be an instance of the Maximum Spider Problem, and let* $\mathcal{R} = \{X \mid X \subseteq V, X$ *is reachable*$\}$ *denote the family of reachable subsets of* V*. For any* $X, Y \in \mathcal{R}$ *with* $|X| > |Y|$ *there exists* $x \in X$ *such that the set* $Y \cup \{x\} \in \mathcal{R}$*.*

Proof. Consider two arbitrary sets $X, Y \in \mathcal{R}$, such that $|X| > |Y|$. Let $S(X)$ denote a spider reaching X, and let $S(Y)$ be a spider reaching Y. We will show that there exists a new spider $S(W)$, with terminals $W = Y \cup \{x\}$, where $x \in X \setminus Y$. Hence, we will get that $W \in \mathcal{R}$.

Starting from G, let us construct the flow network $G' = (V', E')$ with

$$V' = V \cup \{t\}, \qquad E' = E \cup \{(v,t) \mid v \in X \cup Y\}, \tag{2.3}$$

where s is the source of the flow network, t is the sink, and each arc has capacity 1.

The existence of the spider $S(Y)$ in G centered in s and reaching all nodes in Y implies the existence of a flow f in G' such that

$$f(u,v) = \begin{cases} 1 & \text{if } (u,v) \text{ is an arc of } S(Y) \text{ or } u \in Y,\ v = t, \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

The value of f is $|Y|$.

In the same way, the existence of spider $S(X)$ in G implies the existence of a flow of value $|X|$ in G'. Since $|X| > |Y|$, we know that the maximum flow in G' is at least $|Y|+1$. Hence, the flow f given in (2.4) can be augmented. Consider then the residual graph D_f obtained starting from the initial flow f; D_f must contain an augmenting path P from s to t. Moreover, the path P must use the arc (x,t) for some $x \in X \setminus Y$ (since D_f only contains the arcs (t,y) for any $y \in Y$). Consider then the augmented flow g implied by f and P. Since it modifies the values of f only on arcs on P, we get that g induces a set of arc disjoint paths in G from s to the nodes in $Y \cup \{x\}$. This gives the desired spider $S(W)$ covering $W = Y \cup \{x\}$. □

We notice that the family $\mathcal{R}$ is hereditary, that is, any subset of a reachable set is reachable. This fact and Lemma 2.3 tell us that

Lemma 2.4. *The pair $(V, \mathcal{R})$ forms a matroid.*

However, the set system associated to our optimization problem is not $(V, \mathcal{R})$, but it is $(\mathfrak{D}, \mathcal{G})$, where $\mathcal{G} = \{\mathfrak{D}' \subseteq \mathfrak{D} : \text{ all subsets in } \mathfrak{D}' \text{ are covered by a spider in } G \text{ centered at } s\}$; which is hereditary but not a matroid.

Nonetheless, the fact that $(V, \mathcal{R})$ is a matroid represents a useful fact for us. Indeed our coverage function is submodular, for example for any $X, Y \subseteq V$ it holds

$$C(X \cup Y) + C(X \cup Y) \leq C(X) + C(Y). \tag{2.5}$$

Hence the Maximum Spider Problem corresponds to the maximization of the submodular function $C(\cdot)$ on the independent sets of the matroid $(V, \mathcal{R})$. By a well-known result of Nemhauser et al. [14] we have that the greedy algorithm MAX_SP given in Algorithm 1 returns a set X such that

$$C(X^*) \leq 2C(X), \tag{2.6}$$

where X^* represents an optimal solution to the problem. Hence, we have proved the desired approximation result.

Theorem 2.5. *The Algorithm* MAX_SP$(G, s, \mathfrak{D})$ *is a 2-approximation algorithm for the Maximum Spider Problem.*

Algorithm MAX_SP$(G, s, \mathfrak{D})$
(1) Set $X \leftarrow \emptyset$
(2) **while** X is not maximal
(3) Let $x \in V \setminus X$ be the node that maximally improves on X
(4) Set $X \leftarrow X \cup \{x\}$.
(5) **Output** X, $C(X)$, and the spider with set of terminals X.

Algorithm 1: The algorithm for the Maximum Spider Problem on G, s, and $\mathfrak{D}$.

3. The Spider Cover Problem

In this section we will build up on the results for the Maximum Spider Problem in order to design a $O(\log|\mathfrak{D}|)$-approximation algorithm for the Spider Cover Problem. Recall that in this latter problem we are given digraph G, a vertex s, a family $\mathfrak{D} \subseteq 2^{V\setminus\{s\}}$, and the goal is to cover all elements in $\mathfrak{D}$ by using the minimum number of spiders centered at s. Our first step will be to introduce a parametrized family of digraphs $\{H_t\}_{t\geq 1}$ and reduce the problem of determining the minimum number of spiders in G necessary to cover all elements of $\mathfrak{D}$ to the problem of determining the minimum value of t for which H_t contains a single spider covering *all* vertices in a designated subset of vertices of H_t. Subsequently, using iteratively the approximation algorithm MAX_SP on certain H_t's, plus some additional constructions, will allow us to construct an approximation algorithm for the Spider Cover Problem.

3.1. Constructing the Digraph H_t

Let $\langle G = (V, E), s, \mathfrak{D}\rangle$ be an instance of the Spider Cover Problem, and let $t \geq 1$ be an integer. We first construct t graphs $G_1 = (V_1, E_1), \ldots, G_t = (V_t, E_t)$ as follows: for any $v \in V$ the vertex set V_i of the ith digraph G_i contains a corresponding vertex v_i, for $i = 1, \ldots, t$. Vertex v_i will be called the ith copy of v in the final digraph H_t. If the designated vertex s is connected to k vertices $v^1, \ldots, v^k$ in G, then each V_i contains k copies of s, let $s_i^1, \ldots, s_i^k$ be such copies, for $i = 1, \ldots, t$.

Now for the arcs in the G_i's. For each arc $(u, v) \in E$, $u \neq s \neq v$, we insert a corresponding arc (u_i, v_i) in E_i. We also insert in E_i the arcs $(s_i^1, v_i^1), \ldots, (s_i^k, v_i^k)$, where, we recall, $(s, v^1), \ldots, (s, v^k) \in E$.

For the final construction of H_t we introduce new nodes $n_t(v)$, for each $v \in \cup_{D\in\mathfrak{D}} D$, and a special node z. There are arcs between z and each s_i^j, and for each $v \in \cup_{D\in\mathfrak{D}} D$ there is an arc $(v_i, n_t(v))$ from v_i to $n_t(v)$, for each $i = 1, \ldots, t$.

Formally, $H_t = (U_t, A_t)$ is a directed graph where

$$U_t = \left(\bigcup_{i=1}^{t} V_i\right) \cup \{z\} \cup \left\{s_i^j : i = 1, \ldots, t,\ j = 1, \ldots, k\right\} \cup \left\{n_t(v) : v \in \bigcup_{D\in\mathfrak{D}} D\right\},$$

$$A_t = \left\{\left(z, s_i^j\right) : i = 1, \ldots, t,\ j = 1, \ldots, k\right\} \cup \left(\bigcup_{i=1}^{t} E_i\right) \cup \left\{(v_i, n_t(v)) : v \in \bigcup_{D\in\mathfrak{D}} D,\ i = 1, \ldots, t\right\}. \tag{3.1}$$

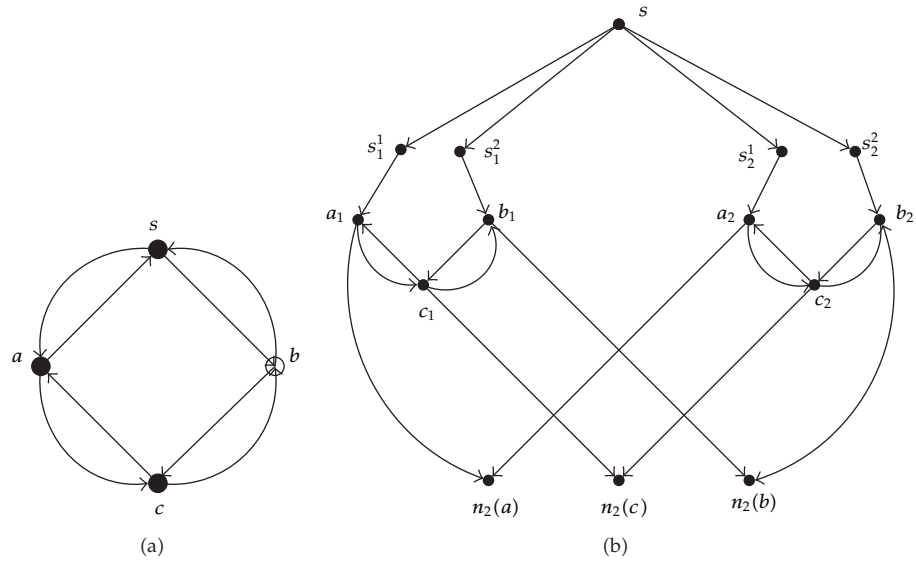

Figure 4: (a) A digraph G; (b) its corresponding graph H_2 when $\mathfrak{D}$ consists of $D_1 = \{a, c\}$ and $D_2 = \{b, c\}$, and the designated node is s.

An example of digraph G and associated graph H_2 is presented in Figure 4. The relevance of digraph H_t to our questions is explained by the following two evident results.

Lemma 3.1. *Let $\langle G, s, \mathfrak{D}\rangle$ be an instance of the Spider Cover Problem. There are t spiders centered at s in G that altogether reach a set of nodes $X \subseteq \cup_{D\in\mathfrak{D}} D$ if and only if there exists a spider centered at z in the digraph H_t reaching the corresponding set of nodes $\{n_t(x) : x \in X\}$.*

Notice that the t spiders in G can also be easily constructed from the "big" spider in H_t and vice versa.

Given an instance $\langle G, s, \mathfrak{D}\rangle$ of the Spider Cover Problem, let $n_t(\mathfrak{D})$ be the family of subsets of nodes of digraph H_t consisting of all subsets $n_t(D) = \{n_t(v) : v \in D\}$, for any $D \in \mathfrak{D}$.

Theorem 3.2. *An instance $\langle G, s, \mathfrak{D}\rangle$ of the Spider Cover Problem admits an optimal solution with t^* spiders if and only if t^* is the minimum integer for which an optimal solution of the Maximum Spider Problem on the instance $\langle H_{t^*}, z, n_{t^*}(\mathfrak{D})\rangle$ consists in a spider covering all elements in the family of subsets $n_{t^*}(\mathfrak{D})$.*

3.2. The Spider Cover Algorithm

Our spider cover algorithm SP_COV$(G, s, \mathfrak{D})$ is presented in next box Algorithm 2. The algorithm consists of successive iterations, based on the Algorithm MAX_SP. At each iteration a certain set of spiders is constructed in order to cover as many subsets in $\mathfrak{D}$ as possible. Namely, at each iteration, if $\Delta \subseteq \mathfrak{D}$ is the subfamily of subsets not covered yet, the algorithm seeks for the minimum number w for which the algorithm MAX_SP$(H_w, z, n_w(\Delta))$ returns a spider centered in z that covers at least half of the subsets in $n_w(\Delta)$. The minimum number w

Algorithm SP_COV$(G, s, \mathfrak{D})$
Set $\Delta = \mathfrak{D}$ [*Family of groups that need to be covered*]
Set $S = \emptyset$, $\overline{w} = 0$
Repeat
(i) Compute the minimum integer w (with $1 \leq w \leq |\Delta|$) such that the algorithm MAX_SP $(H_w, z, n_w(\Delta))$ outputs a spider $\overline{S}$ in H_w reaching a set X for which $C(X) = |\{D \in n_w(\Delta) : D \cap X \neq \emptyset\}| \geq |n_w(\Delta)|/2 = |\Delta|/2$
(ii) From the spider $\overline{S}$ in H_w obtain (via Lemma 4) w new spiders in G that cover at least $|\Delta|/2$ elements of Δ
(iii) Let Δ be the new family of uncovered subsets, put in $\mathcal{S}$ the new w spiders, set $\overline{w} = \overline{w} + w$.
Until $\Delta = \emptyset$.
Output: $\mathcal{S}$ and $\overline{w}$.

Algorithm 2: The algorithm for the Spider Cover Problem on G, s, and $\mathfrak{D}$.

can be obtained by applying the algorithm MAX_SP$(H_w, z, n_w(\Delta))$ in a binary search fashion, with w in the range $[1, |\Delta|]$. Thereafter, via Lemma 3.1, one obtains w spiders in G from the "big" spider in H_w.

The total number of used spiders $\overline{w}$ will be the sum of the number of spiders used at each iteration.

We show now that the number of spiders returned by the algorithm SP_COV$(G, s, \mathfrak{D})$ is at most $\log_2|\mathfrak{D}| + 1$ times the optimal number of spiders necessary for the given instance $\langle G, s, \mathfrak{D}\rangle$ of the Spider Cover Problem.

Theorem 3.3. *The number of spiders returned by the algorithm* SP_COV$(G, s, \mathfrak{D})$ *is* $\overline{w} \leq w^*(\log_2|\mathfrak{D}| + 1)$, *where* w^* *is the number of spiders in an optimal solution for the given instance* $\langle G, s, \mathfrak{D}\rangle$ *of the problem.*

Proof. Consider any iteration of the cycle. The algorithm computes the minimum integer w such that MAX_SP$(H_w, z, n_w(\Delta))$ outputs a spider covering at least $|\Delta|/2$ elements of the family $n_w(\Delta)$. This means that the current size of the family of yet uncovered groups is decreased of at least $1/2$ of its value during each iteration. Hence, the algorithm SP_COV$(G, s, \mathfrak{D})$ consists of at most $\log|\mathfrak{D}| + 1$ iterations.

Moreover, at each iteration the minimum integer w computed by the algorithm is upperbounded by w^*. In fact, it is certain that in H_{w^*} there exists a spider reaching $|\Delta|$ elements of $n_{w^*}(\mathfrak{D})$, for any $\Delta \subseteq \mathfrak{D}$, and the algorithm MAX_SP$(H_{w^*}, z, n_{w^*}(\Delta))$ is guaranteed to find a spider that covers at least $|\Delta|/2$ elements of $n_{w*}(\Delta)$.

We can then conclude that the total number of spiders $\overline{w}$ used by SP_COV$(G, s, \mathfrak{D})$, which is the sum of all the values obtained at the various iterations, is upperbounded by $w^*(\log_2|\mathfrak{D}| + 1)$. □

4. Final Comments

We have provided a general framework for covering problems and shown that several seemingly different problems naturally fit in our scenario. We have given approximation algorithms with best possible approximation ratios, under widely believed computational

complexity assumptions. We would like to point out that we can easily extend our results to undirected graphs or to spiders defined as a collection of vertex disjoint paths sharing only a common vertex, using standard tricks.

In case the graph $G = (V, E)$ is undirected, we can consider the corresponding directed symmetric graph $G' = (V, E')$ where E' contains the pair of arcs (x, y) and (y, x) if and only if x and y are neighbors in G. One must only be careful in the case in which one could get a spider containing both the opposite arcs, say (x, y) and (y, x), corresponding to one edge of G. However, if two branches of a spider are of the form P_1, x, y, P_2 and Q_1, y, x, Q_2, one can modify the spider so to contain P_1, x, Q_2 and Q_1, y, P_2. This implies that we can always get spiders in G with edge disjoint branches. We can then apply the result of the present paper to the directed graph $G' = (V, E')$.

In case we are interested in spiders made of vertex disjoint paths sharing a single vertex, we can obtain the same results as for arc-disjoint spiders by substituting in G each node v with a pair of nodes v' and v'', connected by the arc (v', v''). Moreover, each arc entering v in G now enters v', and each arc leaving v in G now leaves v''.

References

[1] P. Klein and R. Ravi, "A nearly best-possible approximation algorithm for node-weighted Steiner trees," *Journal of Algorithms*, vol. 19, no. 1, pp. 104–115, 1995.

[2] L. Gargano, M. Hammar, P. Hell, L. Stacho, and U. Vaccaro, "Spanning spiders and light-splitting switches," *Discrete Mathematics*, vol. 285, no. 1–3, pp. 83–95, 2004.

[3] D. S. Hochbaum, "Approximating covering and packing problems: set cover, vertex cover, indipendent set, and related problems," in *Approximation Algorithms For NP-Hard Problems*, pp. 94–143, PWS Publishing, Boston, Mass, USA, 1997.

[4] C. Chekuri and A. Kumar, "Maximum coverage problem with group budget constraints and applications," in *Proceedings of 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, (APPROX '04)*, vol. 3122 of *Lecture Notes in Computer Science*, pp. 72–83, 2004.

[5] J. Fakcharoenphol, C. Harrelson, and S. Rao, "The k-traveling repairmen problem," *ACM Transactions on Algorithms*, vol. 3, no. 4, article 40, 2007.

[6] J. N. Tsitsiklis, "Special cases of traveling salesman and repairman problems with time windows," *Networks*, vol. 22, no. 3, pp. 263–282, 1992.

[7] R. Bar-Yehuda, G. Even, and S. Shahar, "On approximating a geometric prize-collecting traveling salesman problem with time windows," *Journal of Algorithms*, vol. 55, no. 1, pp. 76–92, 2005.

[8] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, "Approximation algorithms for deadline-TSP and vehicle routing with time-windows," in *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC '04)*, pp. 166–174, ACM, New York, NY, USA, 2004.

[9] M. Elkin and G. Kortsarz, "An approximation algorithm for the directed telephone multicast problem," *Algorithmica*, vol. 45, no. 4, pp. 569–583, 2006.

[10] L. Gargano, A. Rescigno, and U. Vaccaro, "Multicasting to groups in optical networks and related combinatorial optimization problems," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS '03)*, p. 8, 2003.

[11] J. Desrosiers, Y. Dumas, M. M. Solomon, and F. Soumis, "Time constrained routing and scheduling," in *Network Routing*, vol. 8 of *Handbooks in Operations Research and Management Science*, pp. 35–139, North-Holland, Amsterdam, The Netherlands, 1995.

[12] U. Feige, "A threshold of $ln n$ for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, 1998.

[13] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, NJ, USA, 1993.

[14] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions. I," *Mathematical Programming*, vol. 14, no. 3, pp. 265–294, 1978.