Becoming Gatekeepers Together with Allies: Collaborative Brokerage over Social Networks

Yang Chen and Jiamou Liu
School of Computer Science, The University of Auckland
Auckland, New Zealand
{yang.chen, jiamou.liu}@auckland.ac.nz

***Abstract*—Information brokers control information flow and hold dominating positions in a social network. We study how a team of individuals with heterogeneous influencing power may gain such advantageous position through establishing new links. In particular, a collaborative brokerage problem aims to find the smallest set of nodes for a team of individuals with different influencing power to cover the entire network. We phrase this problem as an extension to the classical graph domination problem and thus this problem is NP-hard. We show that a polynomial-time solution exists for directed trees. We then develop efficient algorithms over arbitrary directed networks. To evaluate the algorithms, we run experiments over networks generated using well-known random graph models and real-world datasets. Experimental results show that our algorithms produce relatively good solutions with faster speed.**

***Index Terms*—collaborative brokerage, directed trees, social networks, dominating set**

I. Introduction

Social network integration refers to a process where ties are created between disjoint groups of individuals. The integration process links organizations or teams of social actors as they forge collaborations, bridge gaps, and build collective social capital. During the integration process, it is vital to establish key relationships, i.e., ties that are of strategic importance, which allow information to flow effectively between the two sides. Imagine a group of individuals who aims to embed themselves into an organization. Assuming that members of the group may purposely establish ties with others in the organization through activities such as meetings, collaborations, team building, etc. A question naturally arises in this context for these individuals: *Who should they target to establish relations with in order to integrate into the organization with the maximum effectiveness?* The answer to the question depends on the existing social structures in the organization. To integrate effectively, the social group must pick those individuals that allow them to reach diverse parts of the organization and exercise control over information flow.

ASONAM '19, August 27-30, 2019, Vancouver, Canada
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6868-1/19/08...$15.00
https://doi.org/10.1145/3341161.3342874

One instance of such a problem, referred to as the *network building problem*, was studied in [1]–[3]. There, the social network is seen as a graph whose nodes correspond to individuals and edges are interpersonal relationships. The problem takes as input a graph G and aims to establish edges between a *broker*, i.e., a node that is not a member of the network, and a selection of nodes in G such that all nodes in the combined network are within a uniform distance away from the broker. One obvious limitation here is that the "group" that we described in the early scenario degenerates to a singleton (i.e., the broker) in the network building problem. When the social group contains more than one member, there is a need to differentiate the influencing power between the brokers: Say, e.g., that in an organizational network, a senior manager can control the management hierarchy at a greater depth than a junior staff. The problem arises when the brokers with stronger influencing power are insufficient to cover the entire network, where a combination of strong/weak brokers must be involved.

This paper extends network building problem by assuming (1) the group consists of two brokers[1], and (2) these brokers may have different influencing power. More specifically, the *collaborative brokerage problem* involves a social network G and two nodes u, v who are not in G. These two nodes have different influence power, with u higher than v. The goal is to find a smallest *collaborative broker team*, which consists of a set of nodes in G to establish edges with u and v so that any node in G is within the influence coverage of u or v.

In this paper, we focus on directed graphs which are ubiquitous in both society and nature [4]. This means that the information flow over a tie is unidirectional. For example, the "following" relationship on an online social network is asymmetric. Another example refers to organizational networks: Most organizations such as governments or corporations resemble a directed structure: the director sits at the root and staffs connect to their subordinates through formal reporting roles [5]. By analyzing the structure, the director may enhance the awareness of information flow within the organization and hence exercise control; A cost-effective approach for the director is to access organizational information through key personnel in a group of information gatekeepers [6].

[1]This may be generalized naturally to more than two individuals by the framework that is to be proposed. Yet in this paper we restrict us to the two-person case for simplicity.

Contribution. The main contributions of the paper are summarized below: **(1)** The main contribution of the paper involves the proposal of collaborative brokerage as a conceptual framework of vital node selection problem, which may facilitate applications in e.g. network monitoring or influence maximization over complex networks [7]. We put forward the collaborative brokerage problem as an extension to the network building problem. (See Sec. II.) **(2)** To solve the collaborative brokerage problem on directed trees, we give a polynomial-time algorithm (DP) with guaranteed optimal output. (See Sec. III.) **(3)** For arbitrary directed networks, we propose four approximation algorithms, including a spanning tree-based algorithm, a greedy algorithm, and two replacement-based algorithm. **(4)** We run our algorithms on both synthetic networks and real-world social networks. The experimental results demonstrate that the approximation algorithms strike a nice balance between accuracy and efficiency. (See Sec. V.)

Related work. Algorithmic studies of social network integration has been discussed in [8]–[11]. A solution to the network building problem finds a distance-$(r-1)$ dominating set of the given network G, where r is the radius of G [1]. When extending to collaborative brokerage, we propose a notion that resembles dominating set, namely, we are interested in a pair of node sets S_1, S_2 such that every node is either within distance d_1 away from S_1 or within distance d_2 away from S_2, for given d_1 & d_2. Dominating sets are of great significance in the recent program of network science due to its connection with the notion of key players [12], [13]. Finding minimal (distance-k) dominating sets has been a classical NP-complete problem [14], [15]. Many variants of the problem exist which naturally inherit the high complexity; we briefly overview these notions below: **(1)** ConnectedDominatingSet, proposed in [16], aims to find a minimum dominating set whose nodes form a connected induced subgraph. This problem is equivalent to finding a spanning tree with the maximum number of leaves, which can be used for routing in ad-hoc wireless networks [17]. **(2)** TotalDominatingSet seeks for a set of nodes such that all nodes in the graph (which may or may not be in the found set) have a neighbor in the dominating set and have attracted much investigation [18]. **(3)** kTupleDominatingSet looks for a set of nodes such that each node has at least k neighbors in the set [19]. **(4)** MixedDominatingSet takes not only the nodes but also edges into account and aims to find a set of nodes and edges with the minimum size such that every node/edge not in it is adjacent to one of its elements [20], [21].

II. Problem Setup

We view social networks as (directed) graphs of the form $G=(V,E)$ where V is a set of nodes and $E \subseteq V^2 \setminus \{(u,u) \mid u \in V\}$ is a set of (directed) edges. For convenience we denote an edge (u,v) by uv, where u is called the *predecessor* of v and v is the *successor* of u. The *distance* from u to v, $\mathrm{dist}(u,v)$, is the minimum length of a (directed) path that goes from u to v. In particular, $\mathrm{dist}(u,u)=0$, and $\mathrm{dist}(u,v)=\infty$ in case there is no path that goes from u to v.

Definition 1. *Fix a positive integer ρ. A set of nodes $D \subseteq V$ is a* distance-ρ dominating (dom-ρ) set *if for all nodes $u \in V$, there is some $v \in D$ such that $\mathrm{dist}(v,u) \leq \rho$. Let $\delta_\rho(G)$ denote the size of a minimum* dom-ρ *set for G.*

The parameter ρ can be viewed as the influence capacity of a node in a dom-ρ set. A natural generalization is to differentiate the influence capacity between nodes in the broker set to two values $\rho_1 \leq \rho_2$.

Definition 2. *For a directed graph $G=(V,E)$, a* distance-(ρ_1,ρ_2) dominating (dom-(ρ_1,ρ_2)) team *consists of a pair (D_1,D_2) of node sets where $D_1 \cap D_2 = \varnothing$, and for any node $u \in V$, either $\mathrm{dist}(v,u) \leq \rho_1$ for some $v \in D_1$ or $\mathrm{dist}(v,u) \leq \rho_2$ for some $v \in D_2$.*

A dom-(ρ_1,ρ_2) team represents a way to influence G by two brokers p,q where the *strong* broker p has power to cover nodes up to distance ρ_2+1 while the *weak* one q up to ρ_1+1. For a set of new edges $E' \subseteq \{p,q\} \times V$ and $u \in \{p,q\}$, let $E'_u = \{v \in V \mid uv \in E'\}$. Our objective is to make sure that p,q can collaboratively cover the entire network while respecting their influence power. This is equivalent to adding edges E' so that the corresponding sets (E'_p, E'_q) form a dom-(ρ_1,ρ_2) team. Therefore we refer to dom-(ρ_1,ρ_2) teams also as *collaborative broker teams*. The *size* of (D_1,D_2) is $|D_1 \cup D_2|$. We first state a simple but useful observation:

Lemma 1. *Let G be a directed graph and $d,k \in \mathbb{N}$. Suppose $|D_1| \geq k$ for all* dom-(ρ_1,ρ_2) *teams (D_1,D_2) with $|D_2|=d$. Then the size of any* dom-(ρ_1,ρ_2) *team with $|D_2| \leq d$ is at least $d+k$.*

Proof. Suppose that $|D_1| > k$ for all dom-(ρ_1,ρ_2) teams (D_1,D_2) with $|D_2|=d$. We need to show that no dom-(ρ_1,ρ_2) team (D_1,D_2) exists with $|D_2|<d$ and $|D_1|+|D_2|<d+k$. Suppose $|D_2|<d$ and $|D_1|<d+k-|D_2|$. Take any subset $S \subseteq D_1$ with exactly $d-|D_2|$ nodes. Set $D'_1 = D_1 \setminus S$ and $D'_2 = D_2 \cup S$. Then (D'_1,D'_2) also forms a dom-(ρ_1,ρ_2) team with $|D'_2|=d$ and $|D'_1| = |D_1| - d + |D_2| < k$, contradicting our assumption. ☐

Intuitively, Lemma 1 stratifies the class of all dom-(ρ_1,ρ_2) teams using the size of D_2: To find the smallest collaborative broker team (D_1,D_2) (with bounds on D_2), it makes sense to "fill in" as many nodes in D_2 as possible. The smallest dom-(ρ_1,ρ_2) team is one where $|D_1| = \varnothing$. Consequently, we focus on the following:

Definition 3. *An integer d is an* order *of a* dom-(ρ_1,ρ_2) *team (D_1,D_2) if $|D_2| \leq d$. The* order-d dom-(ρ_1,ρ_2) index*, $\delta_{\rho_1,\rho_2}(G:d)$, is the size of a smallest order-d* dom-(ρ_1,ρ_2) *team (D_1,D_2).*

The following result naturally follows Lemma 1.

Theorem 1. *For any directed graph G, we have*

1. $\delta_{\rho_1,\rho_2}(G:d) = \delta_{\rho_2}(G)$ *for all $d \geq \delta_{\rho_2}(G)$;*
2. $\delta_{\rho_1,\rho_2}(G:d) \leq \delta_{\rho_1,\rho_2}(G;d')$ *for any $d' < d$;*
3. $\delta_{\rho_1,\rho_2}(G:0) = \delta_{\rho_1}(G)$.

We are now ready to give the formal definition of the collaborative brokerage problem:

$$\textsc{CollabBroker} = \{\langle G, d, \rho_1, \rho_2, k\rangle \mid G \text{ is a directed graph}, \rho_1, \rho_2, d, k \in \mathbb{N}, \rho_1 \leq \rho_2, \delta_{\rho_1,\rho_2}(G : d) \leq k\}.$$

Namely, given a graph G and $d \in \mathbb{N}$, the objective of the CollabBroker problem is to compute the size of a smallest order-d dom-(ρ_1, ρ_2) team.

Theorem 2 shows the complexity of the problem (treating $\rho_1 = \rho_2 = 1$) and follows from [22] (using the well-known equivalence between SetCover and DominatingSet problem).

Theorem 2. *[22] It is NP-hard to approximate $\delta_1(G)$ on input graph of size N to within $(1 - \alpha) \ln N$.*

III. CollabBroker over Directed Trees

Theorem 2 indicates hardness of CollabBroker on arbitrary graphs. Over directed trees, however, the problem have a more efficient solution. A *(directed) tree* is a finite directed acyclic graph where there is a unique node, called the *root*, that has indegree 0 and every other node has exactly one incoming neighbor. We denote a directed tree as $T = (V, E, r)$ which designates a node $r \in V$ as the root. The outgoing neighbors of a node v are called the *children* of v; the only incoming neighbor of a non-root node v is called the *parent* of v. We use Ch_v to denote the set of children of v and pa_v to denote the parent of v. A node v is called a *leaf* if $\mathsf{Ch}_v = \varnothing$; otherwise, v is an *internal node*. If a path exists from v to u, then this path is unique; In this case, v is called an *ancestor* of u and u is called a *descendent* of v. The set $V_v \subseteq V$ of all descendants of v form a *subtree* rooted at v, which is denoted by T_v; Thus $T_r = T$. The distance $\text{dist}(r, v)$ is also called the *depth* of v, denoted by $\mathsf{dp}(v)$. The maximum level of T is also called its *height*. Trees form an important class of graphs for which CollabBroker problem has polynomial time solutions [23]. Applications of selecting key nodes in a tree network appear in the context of organizational management [5], ecological control of river networks [24], [25], and topic segmentation of websites [26]. The next theorem generalizes a linear-time algorithm in [27] to compute $\delta_\rho(T)$ of a directed tree T.

Theorem 3. *There is an $O(n)$ time algorithm that computes a minimum dom-ρ set of a tree T with n nodes.*

Proof. Alg. 1 computes a dom-ρ set D on a tree $T = (V, E, r)$. The procedure runs in a bottom-up manner, starting from the leaves of T and moves in a reversed edge direction until reaching the root. Along the process, the procedure iteratively adds nodes into D while computing two pieces of information for each node v: the first is a set S_v of covered nodes below any node v (i.e., descendants of v that are reachable within ρ steps); the second is the maximum depth b_v of an uncovered descendant of v (i.e. $b = \max\{\mathsf{dp}(x) \mid x \in V_v \backslash S\}$), or the depth of v's parent if all nodes in V_v are covered. When processing a node v, the procedure puts v into D if the distance from v to the furthest uncovered descendant is ρ, or when v is the root of T.

The algorithm runs in linear time as each node is processed once. Procedure. 1 describes a recursive implementation and utilizes $\mathsf{dp}(v)$ for every node $v \in V$, which can clearly be computed in time $O(n)$.

To verify that the output D is indeed a dom-ρ set, it is easy to show that all nodes are covered, i.e.,

$$\forall v \in V : S_v = \{u \in V_v \mid \text{dist}(x, u) \leq \rho \text{ for some } x \in D\}.$$

To verify the optimality of the set D, it is sufficient to prove two claims, which are easy to show:
(1) For any node $v \in V$ whose maximum distance between v and a leaf in T_v is exactly ρ, there is a minimum dom-ρ set D of T that contains v and no node from $V_v \setminus \{v\}$; and
(2) Suppose D is a minimum dom-ρ set of T, and a non-root node $v \in D$. Then the set of nodes $D' = D \setminus V_v$ is a minimum dom-ρ set of the tree T' obtained from T by pruning T_v away. □

Algorithm 1: ComputeDom(T, v, ρ): Compute dom-ρ set for tree T rooted at v

```
Input: Tree $T$, node $v$ in $T$, integer $\rho > 0$
Output: $(D, b_v)$ where $D \subseteq V$
Initialize: $D \leftarrow \varnothing$, $b_v \leftarrow \mathsf{dp}(v)$;
if $v$ is not a leaf then
    for each child $u \in \mathsf{Ch}_v$ do
        $(D_u, b_u) \leftarrow$ ComputeDom$(T, u, \rho)$;
        $D \leftarrow D \cup D_u$, $b_v \leftarrow \max\{b_v, b_u\}$;
if $b_v - \mathsf{dp}(v) = \rho$ or $v$ is the root of $T$ then
    $D \leftarrow D \cup \{v\}$;
    $b_v \leftarrow \mathsf{dp}(v) - 1$;
return $(D, b_v)$
```

The main result of this section generalizes Theorem 3 to the CollabBroker problem and proposes an algorithm DP that produces optimal dom-(ρ_1, ρ_2) teams for trees. To explain the procedure of DP, we need the following auxiliary definitions. A dom-(ρ_1, ρ_2) team classifies nodes in V into three mutually exclusive sets, D_1, D_2 and $V \setminus (D_1 \cup D_2)$. Hence, we define the *class* of nodes.

Definition 4. *Let cls_v denote the class of v. For a directed tree T with a dom-(ρ_1, ρ_2) team (D_1, D_2), define $\mathsf{cls}_v \in \{0, 1, 2\}$ for a node v as follows: $\mathsf{cls}_v = 1$ if $v \in D_1$; $\mathsf{cls}_v = 2$ if $v \in D_2$; otherwise $\mathsf{cls}_v = 0$.*

Intuitively, the *positional advantage* of a specific node depends on the distance to the node that covers it. To define positional advantage, for any $v \in V$, write

$$u_v^* = \underset{u \in D_1}{\arg\min}\, \text{dist}(v, u) \quad \text{and} \quad w_v^* = \underset{w \in D_2}{\arg\min}\, \text{dist}(v, w).$$

Definition 5. *Let lev_v denote the* positional advantage *of v. Define lev_v for a node v as follows: $\mathsf{lev}_v = 0$ if $v \in D_2$; $\mathsf{lev}_v = \min\{\rho_2 - \rho_1, \text{dist}(v, w_v^*)\}$ if $v \in D_1$; $\mathsf{lev}_v = \mathsf{lev}_x + \text{dist}(v, x)$, where $x = u_v^*$ if $\text{dist}(v, u_v^*) < \text{dist}(v, w_v^*)$, otherwise $x = w_v^*$.*

We say that v has cls-c and lev-ℓ if $\mathsf{cls}_v = c$, $\mathsf{lev}_v = \ell$. The following lemma easily follows from Def. 4 and Def. 5.

Lemma 2. *Let (D_1, D_2) be a* dom-(ρ_1, ρ_2) *team in a directed tree $T = (V, E)$, for any $v \in V$, we have: (a)* $\mathsf{cls}_v = 2 \Rightarrow \mathsf{lev}_v = 0$*; (b)* $\mathsf{cls}_v = 1 \Rightarrow 1 \le \mathsf{lev}_v \le \rho_2 - \rho_1$*; (c)* $\mathsf{cls}_v = 0 \Rightarrow 1 \le \mathsf{lev}_v \le \rho_2$*. Say a combination of* $(\mathsf{cls}_v, \mathsf{lev}_v)$ *is* legal *if either above condition is satisfied.*

Theorem 4. *There is an $O(d^2 n)$ algorithm that, given a tree T and value $d \in \mathbb{N}$, computes a minimum* dom-(ρ_1, ρ_2) *team (D_1, D_2) of T where $d = |D_2|$ and n is the size of T.*

proof sketch. We propose a dynamic programming (DP) algorithm for CollabBroker over trees that runs in polynomial time. The basic procedure of DP is to partition the entire problem to sub-problems, which are further broken down to subsub-problems. Given a tree $T = (V, E, r)$ and $d \in \mathbb{N}$, we parameterize the problem by introducing two new parameters: a node $v \in V$ and a vector $\Delta = \{0, \ldots, d\}$. For each $d' \in \Delta$, a *sub-problem* with parameters (v, d') asks for the value of $\delta_{\rho_1,\rho_2}(T_v : d') - d'$ for each legal combination of $(\mathsf{cls}_v, \mathsf{lev}_v)$. Namely, given $|D_2| = d'$, the desired output is the smallest size of any D_1 in a minimum dom-(ρ_1, ρ_2) team (D_1, D_2) in the subtree T_v. For each sub-problem, we derive a $(d+1)$-length vector

$$ {}^{c}_{\ell}\Omega^{*}_{v} = ({}^{c}_{\ell}\Omega^{*}_{v}[0], \cdots, {}^{c}_{\ell}\Omega^{*}_{v}[d]) $$

to record such desired values, where cls_v and lev_v are abbreviated as c and ℓ respectively. Note that when $v = r$ and $d' = d$, the sub-problem is consistent with the original problem.

A sub-problem is further partitioned into *subsub-problems*. Given a subtree T_v, we split T_v into *subsub-trees* as follows: Arrange the elements in Ch_v into a fixed order; Starting from Ch^0_v , let Ch^i_v denote the ith child of v. The i-th subsub-tree of v, denoted by T^i_v, is defined as $\{v\} \cup \{T_{\mathsf{Ch}^j_v} \mid 0 \le j \le i\}$, i.e., the sub-tree consisting of v and v's first i branches. Given a sub-problem with parameters (v, Δ) under the assumption (c, ℓ), for each $d' \in \Delta = \{0, \ldots, d\}$, a subsub-problem asks for the value of $\delta_{\rho_1,\rho_2}(T^i_v : d') - d'$, namely the smallest size of any D_1 in a minimum dom-(ρ_1, ρ_2) team (D_1, D_2) in the subsub-tree T^i_v with $|D_2| = d'$. Utilizing the similar manner in tackling sub-problems, for each subsub-problem, assign with each subsub-tree T^i_v a vector

$$ {}^{c}_{\ell}\Omega^{i}_{v} = ({}^{c}_{\ell}\Omega^{i}_{v}[0], \cdots, {}^{c}_{\ell}\Omega^{i}_{v}[d]). $$

Since the subsub-problem on the last subsub-tree is consistent with the corresponding sub-problem, the desired solution of a sub-problem on T_v is obtained by setting ${}^{c}_{\ell}\Omega^{*}_{v} = {}^{c}_{\ell}\Omega^{|\mathsf{Ch}_v - 1|}_{v}$.

Starting from the leaves, the DP algorithm processes the tree bottom-up, solving the sub-problem for each sub-tree rooted at $v \in V$ and $d' \in \Delta$. Since the root of tree T is either in D_1 or D_2, finally we have

$$ \delta_{\rho_1,\rho_2}(T : d) = \min\left\{ {}^{2}_{0}\Omega^{*}_{r}[d],\ {}^{\ 1}_{\rho_2-\rho_1}\Omega^{*}_{r}[d] \right\} + d. $$

Tracing back the matrix in each node from the root down to leaves, we can extract the elements in D_1 and D_2 that form a dom-(ρ_1, ρ_2) team. By the recursion above, for each node v, it takes at most $3d$ steps to fill in each cell in its matrix. By Lem. 2, there are $2\rho_2 - \rho_1 + 1$ legal combinations of (ℓ, c). So each node maintains a vector of length $(2\rho_2 - \rho_1 + 1) \times (d+1)$. Hence the DP algorithm runs in $O(n \times (2\rho_2 - \rho_1 + 1)(d + 1) \times 3d) = O(d^2 n)$ time and optimality of DP follows from the recursive definition above. □

Fig. 1 shows an example when DP runs on a simple tree.

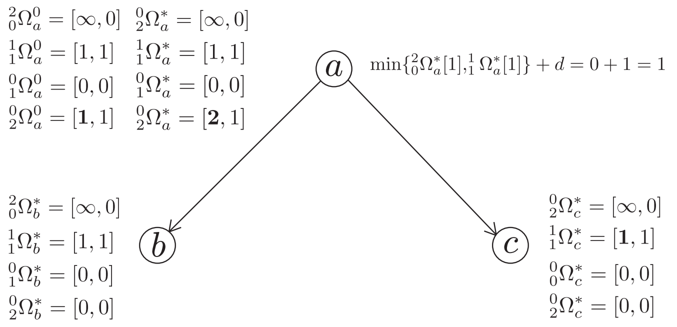
${}^{2}_{0}\Omega^{0}_{a} = [\infty, 0]$ ${}^{0}_{2}\Omega^{*}_{a} = [\infty, 0]$
${}^{1}_{1}\Omega^{0}_{a} = [1, 1]$ ${}^{1}_{1}\Omega^{*}_{a} = [1, 1]$
${}^{0}_{1}\Omega^{0}_{a} = [0, 0]$ ${}^{0}_{1}\Omega^{*}_{a} = [0, 0]$
${}^{0}_{2}\Omega^{0}_{a} = [\mathbf{1}, 1]$ ${}^{0}_{2}\Omega^{*}_{a} = [\mathbf{2}, 1]$
a
$\min\{{}^{2}_{0}\Omega^{*}_{a}[1], {}^{1}_{1}\Omega^{*}_{a}[1]\} + d = 0 + 1 = 1$
${}^{2}_{0}\Omega^{*}_{b} = [\infty, 0]$
${}^{1}_{1}\Omega^{*}_{b} = [1, 1]$
${}^{0}_{1}\Omega^{*}_{b} = [0, 0]$
${}^{0}_{2}\Omega^{*}_{b} = [0, 0]$
b
c
${}^{0}_{2}\Omega^{*}_{c} = [\infty, 0]$
${}^{1}_{1}\Omega^{*}_{c} = [\mathbf{1}, 1]$
${}^{0}_{0}\Omega^{*}_{c} = [0, 0]$
${}^{0}_{2}\Omega^{*}_{c} = [0, 0]$

Fig. 1: The computational process of DP on a tree with $\rho_1 = 1, \rho_2 = 2$ and $d = 1$. The matrix maintained by each node is shown and we use ${}^{0}_{2}\Omega^{*}_{a}[0] = {}^{0}_{2}\Omega^{0}_{a}[0] +^{1}_{1} \Omega^{*}_{a}[0]$ (highlighted in bold) to show an example of the recursive process. Finally, we get the optimal dom-(ρ_1, ρ_2) team with size 1 such that $D_1 = \{a\}, D_2 = \varnothing$.

IV. Efficient Algorithms Over Directed Graphs

Utilizing the efficient algorithm over directed trees, we present three categories of efficient approximation algorithms to compute small collaborative dominating sets over directed networks.

A. Spanning Tree-based Algorithm

Given a directed network G, a naive method for approximation is to apply DP to spanning trees of G. However, a directed graph may be not strongly connected, which means there can be a *spanning forest*, a set of isolated spanning trees, rather than a single tree. We use Wilson's algorithm, RandomTree, to sample spanning forests over directed graphs, which randomly picks roots and constructs tree paths via random walks tracing up from leaves to ancestors [28]. The main advantage of RandomTree is that it does not depend on the assumption that the digraph is strongly connected, and each node has a uniform probability to be the root of a spanning tree in the forest. Then, we combine all trees in a forest to form a uniform tree using the following technique: a node v^* is superposed and linked to the root of each spanning tree. Hence we build a uniform tree rooted at v^*, which sets the scene for performing DP. We assume $\mathsf{lev}_{v^*} = \rho_2$, namely, the children of v^* (roots of all trees) can only be either in D_1 or D_2. Nevertheless, the size of a collaborative dominating team varies by the number and structures of spanning trees in the forest. To reduce sampling errors, we run RandomTree for k times to sample k spanning forests and pick the collaborative dominating team with minimum size as the final output.

Algorithm 1: STDP-k. Given a directed graph G and $\rho_1, \rho_2, d \in \mathbb{N}$, run RandomTree for k times to sample k forests $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k\}$. Let $\mathcal{T}_i = \{T_{i,1}, T_{i,2}, \ldots\}$, where $T_{i,j}$ denotes the jth spanning tree in the forest $\mathcal{T}_i$. For each forest $\mathcal{T}_i$, add a new node v_i^* and add a directed edge from v_i^* to the root of each $T_{i,j} \in \mathcal{T}_i$. Denote by $(D_{i,1}, D_{i,2})$, the minimum dom-(ρ_1, ρ_2) set for $\mathcal{T}_i$ is computed using DP under the assumption $\mathsf{lev}_{v_i^*} = \rho_2$. Finally output (D_1^*, D_2^*) such that $(D_1^*, D_2^*) = \arg\min_{1 \le i \le k} |D_{i,1} \cup D_{i,2}|$. See Procedure 2.

Algorithm 2: STDP-$k(G, d, \rho_1, \rho_2)$: Approximate the minimum dom- (ρ_1, ρ_2) team (D_1, D_2) for graph G with $|D_2| \le d$

```
Input: Graph $G = (V, E)$, $v \in V$
Output: $(D_1, D_2)$ where $D_1, D_2 \subseteq V$
for $i = 1 \to k$ do
    $\mathcal{T}_i \leftarrow$ RandomTree$(G)$;
    Add a directed edge from $v_i^*$ to the root of each $T_{i,j} \in \mathcal{T}_i$;
    $(D_{i,1}, D_{i,2}) \xleftarrow{\mathsf{lev}_{v_i^*} = \rho_2}$ DP$(\mathcal{T}_i \cup \{v_i^*\}, d)$;
return $(D_1^*, D_2^*) = \arg\min_{1 \le i \le k} |D_{i,1} \cup D_{i,2}|$
```

B. Greedy Algorithm

One natural drawback of STDP-k is that a number of edges are not considered in sampling spanning forest, which may cause considerable redundancy. To utilize each edge indiscriminately, we propose a greedy algorithm (Greedy), based on heuristics, which repeatedly picks nodes greedily. Given $G = (V, E)$ and $d \in \mathbb{N}$, the initial configuration is $U = V$ and $D_1, D_2 = \varnothing$. The algorithm consists of two stages:

Algorithm 2: Greedy**.**

1. Repeatedly select a node $s \in U$ according to the corresponding heuristic and add s to D_2. After each selection, compute all nodes at distance within ρ_2 from s and remove them form U. Terminate if $U = \varnothing$ or go to step 2 when $|D_2| = d$.
2. Repeatedly select a node $s' \in U$ according to the corresponding heuristic and add s' to D_1. After each selection, compute all nodes at distance within ρ_2 from s' and remove them form U. Terminate when $U = \varnothing$.

Heuristic 1: Max **(Maximum outdegree).** The intuition of this heuristic is that selecting the node with maximum outdegree can monitor a large number of agents. At each iteration, this heuristic selects a node with maximum outdegree.

Heuristic 2: Min **(Minimum indegree).** Based on minimum indegree, the second heuristic is inspired by the following intuition: a node with a small indegree is less likely to be reached from a randomly selected node. Hence this heuristic gives priority to nodes with small indegrees. At each iteration, the heuristic adds to D_2 (D_1) a node that has distance at most ρ_2 (ρ_1). More precisely, the heuristic first locates a leaf v in U, then applies a sub-procedure to find the node w to be added to D_2 (D_1). The sub-procedure determines a directed path $p = (v \leftarrow u_1 \leftarrow u_2 \leftarrow \ldots)$ in G iteratively as follows:

1) Suppose u_i is being checked. If $i = \rho_2$ (ρ_1) or u_i has no predecessor node in U, set u_i as w and terminate the process.
2) Otherwise select a u_{i+1} among predecessors of u_i with maximum outdegree.

Overall, Min is a mixed heuristic by combining minimum indegree and maximum outdegree.

Algorithm 3: Greedy(G, d, ρ_1, ρ_2): Approximate the minimum dom- (ρ_1, ρ_2) team (D_1, D_2) for graph G with $|D_2| \le d$

```
Input: Graph $G = (V, E)$, $v \in V$
Output: $(D_1, D_2)$ where $D_1, D_2 \subseteq V$
Initialize: $D_1, D_2 \leftarrow \varnothing, U \leftarrow V$;
while $|D_2| < d$ and $U \neq \varnothing$ do
    Pick a node $s \in U$ according to the corresponding heuristic;
    Set $D_2 \leftarrow D_2 \cup \{s\}$, $U \leftarrow U \setminus \{u \in U \mid \mathrm{dist}(u, s) \le \rho_2\}$;
while $U \neq \varnothing$ do
    Pick a node $s \in U$ according to corresponding heuristic;
    Set $D_1 \leftarrow D_1 \cup \{s\}$, $U \leftarrow U \setminus \{u \in U \mid \mathrm{dist}(u, s) \le \rho_1\}$;
return $(D_1, D_2)$
```

C. Replacement-based Algorithms

We next propose two algorithms based on replacement, which also use heuristics but are slightly more complicated than the greedy algorithm. A replacement algorithm functions as follows: First compute a dom-ρ_1 or dom-ρ_2 set using the greedy algorithm, which is then expanded to a collaborative dominating team via replacement.

Algorithm 3: REPL1**.** This algorithm determines D_1 by replacing nodes already in a dom-ρ_2 set. Given a directed graph G and $d \in \mathbb{N}$ with $0 < \rho_1 < \rho_2$, this algorithm involves two stages, preprocessing and replacing. Initiation: $D_1 = \varnothing$. ***Preprocessing***: Firstly invoke Greedy$(G, \infty, \rho_2, \rho_2)$ to get a dom-ρ_2 set and denote it by D_2. Then for each $u \in D_2$, generate a sub-graph G'_u as follows: Determine nodes in a sub-graph using a depth first search; During the search, backtrack when encountering a node w such that either w has no successor or $w \in D_2$; Add all searched nodes with all edges among those to G'_u. Next, run Greedy$(G'_u, d = \infty, \rho_1, \rho_1)$ on each such sub-graph to get a dom-ρ_1 set U_u. Afterwards, order all $u \in D_2$ in increasing $|U_u|$. ***Replacing***: Pick the first node u^* in the sorted list of D_2. "De-select" u^* from D_2 and put all nodes in U_{u^*} to D_1. Repeat the Replacing stage until $|D_2| = d$. Namely, each time we replace a node in D_2 with minimum number of nodes in D_1. See Procedure. 4.

Algorithm 4: REPL2**.** In contrast to REPL1, this algorithm determines D_2 by replacing nodes already in a dom-ρ_1 set. ***Preprocessing***: Initiation: Set $D_2 = \varnothing$. Firstly run Greedy$(G, d = \infty, \rho_1, \rho_1)$ to get a dom-ρ_1 set and denote it by D_1. Then for each $v \in V$, compute the descendants of v which are in D_1 and within distance $\rho_2 - \rho_1$ from v; let the set of such nodes be U_v. The distance bounder $\rho_2 - \rho_1$ guarantees that any node is covered by some node in $D_1 \cup D_2$ after we replace multiple nodes in D_1 with a single node in

D_2. ***Replacing***: Add u^* into D_2 and remove all node in U_{u^*} from D_1. Repeat Checking and Replacing for d rounds. See Procedure. 5.

Algorithm 4: REPL1(G, d, ρ_1, ρ_2): Approximate the minimum dom-(ρ_1, ρ_2) team (D_1, D_2) for graph G with $|D_2| \leq d$

```
Input: Graph $G = (V, E)$, $v \in V$
Output: $(D_1, D_2)$ where $D_1, D_2 \subseteq V$
Initialize: $D_1, D_2 \leftarrow \varnothing$;
▷ Preprocessing:;
Run Greedy$(G, \infty, \rho_2, \rho_2)$ to obtain $D_2$;
for $u \in D_2$ do
    Use DFS to compute $G'_u$;
    Run Greedy$(G, d = \infty, \rho_1, \rho_1)$ to get a dom-$\rho_1$ set $U_u$;
Store $u \in D_2$ in a min-heap with value $|U_u|$;
▷ Replacing:;
while $|D_2| > d$ do
    Set $u^* \leftarrow \arg\min_{u \in D_2} |U_u|$  ▷ pop from the heap of $D_2$;
    Set $D_1 \leftarrow D_1 \cup U_{u^*}$; $D_2 \leftarrow D_2 \setminus \{u^*\}$;
return $(D_1, D_2)$
```

Algorithm 5: REPL2(G, d, ρ_1, ρ_2): Approximate the minimum dom-(ρ_1, ρ_2) team (D_1, D_2) for graph G with $|D_2| \leq d$

```
Input: Graph $G = (V, E)$, $v \in V$
Output: $(D_1, D_2)$ where $D_1, D_2 \subseteq V$
Initialize: $D_1, D_2 \leftarrow \varnothing$;
▷ Preprocessing:;
Run Greedy$(G, d = \infty, \rho_1, \rho_1)$ to obtain set $D_1$;
for $v \in V$ do
    Set $U_v \leftarrow \{u \mid u \in D_1, \mathrm{dist}(u, v) \leq \rho_2 - \rho_1$
Put nodes $v \in V$ in a max-heap with value $|U_v|$;
▷ Replacing:;
while $|D_2| < d$ do
    Set $v^* \leftarrow \arg\max_{v \in V \setminus D_2} |U_v|$   ▷ Pop from the heap;
    Set $D_1 \leftarrow D_1 \setminus U_{v^*}$; $D_2 \leftarrow D_2 \cup \{v^*\}$;
return $(D_1, D_2)$
```

V. Experiments

A. Random Networks

We run our algorithms on synthetic networks that are generated using standard network models. **(1) Directed Barabási-Albert** (BA) **Networks.** Barabási-Albert model generates scale-free networks with a power-law degree distribution. The process initiates a cycle of n nodes and from each node (successor), it adds at most $m < n$ edges to others (predecessors) using a preferential attachment scheme. Namely, the in-degree follows a power-law distribution. **(2) Directed Erdös-Rényi** (ER) **Networks.** The model starts with a number n. For each pair of nodes, add an edge from the one the other uniformly randomly with probability $p \in [0, 1]$ [29]. **(3) Navigable Small World** (NSW) **Networks.** A navigable small-world graph is a directed grid with additional long-range connections that are chosen randomly. The model starts with an $n \times n$ grid where a node is identified by the coordinate $(i, j), i, j \in \{1, 2, \dots, n\}$. The distance between two nodes (i, j) and (k, l) is defined as the number of lattice steps, i.e., $d((i, j), (k, l)) = |k-i|+|l-j|$. For a universal constant $p \geq 1$, the node u has a directed edge to every other node within lattice distance p, which are considered as local contacts. For universal constants $q \geq 0$ and $r \geq 0$, directed edges are constructed from u to q other nodes (the long-term contacts) using independent random trials; the u has endpoint v the probability proportional to $1/d(u, v)^r$ [30].

We conduct two tests: (1) **Test 1:** To investigate how our algorithms performs under different sizes of D_2, we fix (ρ_1, ρ_2) to $(1, 2)$ and vary d in range $[0, 50]$. We compare the size of D_1 output by our algorithms (e.g., the first row in Fig. 2). (2) **Test 2:** To reveal how algorithms perform under different influence capacities, we fix d to 10 and vary (ρ_1, ρ_2) (e.g., the second row in Fig. 2). For each model and each test, results are averaged over 10 independent runs.

Results. The results for each type (BA, ER and NSW) of random networks are summarized as follows.

BA **networks.** We generate 10 BA networks each containing 1000 agents with $m = 2$. See Fig. 2 for the results of two tests. The first row and second row show the results for test 1 and 2, respectively. A few facts stand out from the plots: (a) as expected, STDP-k shows an incremental performance as the number of trials k increases, but still has considerable errors compared to Greedy algorithms and replacement-based algorithms. (b) In Greedy algorithms, heuristic Max produces a better outcome than Min. (c) Among all four categories of algorithms, replacement-based algorithms achieve the best performance. Particularly, REPL2's outcomes have the lowest size of D_1 for every pair of ρ_1 and ρ_2.

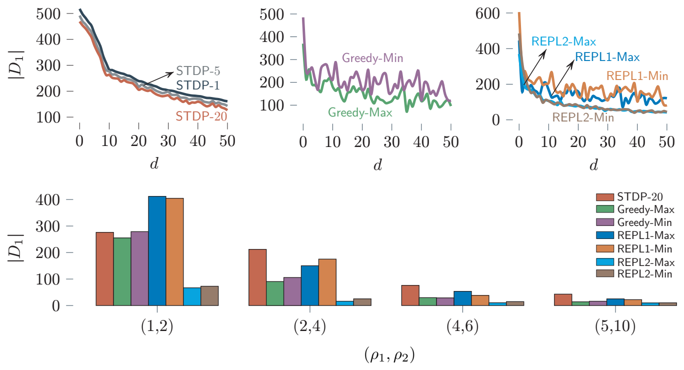

Fig. 2: Experimental results for BA networks.

ER **networks.** We generate 10 ER networks each consisting of 1000 nodes with probability $p = 0.07$. See Fig. 3 for results. STDP-k and Greedy show the similar tendency as in BA networks. Surprisingly, REPL1 produces the optimal outcome (D_1 = 0) when d is large, while REPL2 fails. This is due to that when the number of nodes in D_2 is sufficient to cover the whole network, REPL1 skips the replacing procedure and terminates directly after preprocessing procedure. Hence, in this case, the size of D_1 output by REPL1 is 0. Due to the same reason, in cases where ρ_1 and ρ_2 are large, all algorithms except STDP-k output the result that $|D_1| = 0$.

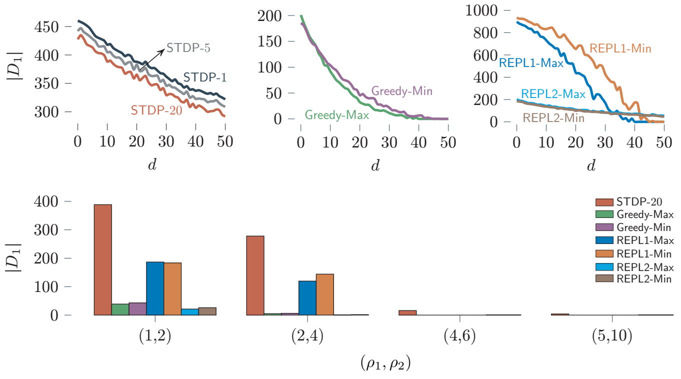

Fig. 3: Experimental results for ER networks.

SW **networks.** We generate 10 SW networks each initiated with a 32 by 32 grid (1024 nodes) with $p = q = 1$ and $r = 2$. See Fig. 4 for results. Observe that (a) STDP-k again performs worst among all algorithms. (b) REPL2 surpasses REPL1 remarkably in both tests. (c) For REPL2, the heuristic Min shows a slightly better performance than Max.

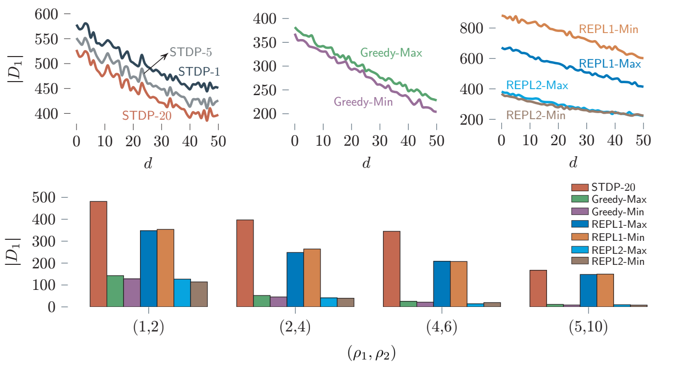

Fig. 4: Experimental results for NSW networks.

Overall, we observe that among all categories of algorithms, replacement based algorithms (especially REPL2) show the averagely best performance on all synthetic networks. Moreover, the heuristic Max and Min should be chosen accordingly as the structural properties may vary over networks.

B. Real-world Networks

We then test our algorithms on real-world networks. We conduct experiments on three datasets: (1) **Wiki-Vote (Wiki)** contains all the Wikipedia voting data from the inception of Wikipedia till January 2008 [31]. Nodes in the network represent Wikipedia users and a directed edge from i to j represents that i voted on j. (2) **Bitcoin OTC trust network (Bitcoin)** record anonymous Bitcoin trading on Bitcoin OTC with temporal information [32], where a directed edge ij denotes a trade between user i and user j. (3) **Cit-HepPh network (Cit)** is a high-energy physics citation network [33], which collects all papers from 1993 to 2003 on arXiv; A directed edge ij denotes that paper i cites j. The statistics of three real-world networks are summarized in Tab. I.

TABLE I: Statistics of real-world networks

Property	**Wiki**	**Bitcoin**	**Cit**
# nodes	7115	5881	27770
# edges	103689	35592	352807
# connected components	24	4	143
diameter (largest component)	7	9	13

We adopt the following settings for our experiments on the real-world networks: for Test 1, we fix (ρ_1, ρ_2) to $(1, 2)$ and vary d from 0 to 400, at an interval of 50; for Test 2, we fix d to 50 and vary (ρ_1, ρ_2) in $\{(1, 2), (2, 4), (4, 6), (5, 10)\}$, respectively. Considering that the error of STDP is considerably large compared to all other algorithms, we only use Greedy and replacement-based algorithms in this experiment. See Tab. II for results. From the results, one observes: **(a)** For Test 1, the outcome by REPL1 show significant errors on Wiki and Cit networks, on which Greedy and REPL2 produce similar better results. While REPL2-Max stand outs for its best output on the Bitcoin network. This may be due to there is a large number of "leaves" that locate in the edge of Bitcoin network, which implies that covering them with priority can reduce the size of a broker team. **(b)** For Test 2, it is observed that for Wiki and Cit, the size of D_1 decreases at a much slower speed as the values of ρ_1 and ρ_2 grow. This is because there are a number of isolated components with lots of zero-indegree nodes in Wiki and Cit. There, We have to assign each of such zero-indegree nodes in either D_1 and D_2 even though ρ_1 and ρ_2 are large.

VI. Conclusion and Future Work

This paper introduces the notion of distance-(ρ_1, ρ_2) dominating set and the corresponding collaborative brokerage problem. The problem has a polynomial time algorithm on directed trees despite being NP-complete in general. We present three classes of efficient algorithms to solve the problem on directed graphs. Experiments on generated and real-world networks reveal that replacement-based algorithms make a good trade-off between efficiency and accuracy.

A natural future work is to go beyond two brokers to assume an arbitrary number of brokers with potentially different influencing power. For this, one can generalize our definition to the notion of dom-$(\rho_1, \dots, \rho_n)$ team with integers $\rho_1 \le \rho_2 \le \dots \le \rho_n$. A similar dynamic programming strategy can be used to solve this problem over directed trees. We also expect that similar algorithms proposed in this paper can be adapted to compute optimal solutions for this more general case. Another future work is to theoretically evaluate the performance of the algorithms proposed above, and derive bounds on the approximation ratio.

TABLE II: Experimental results for real-world networks. The minimum size of D_1 output by algorithms is highlighted in bold for each test.

		Wiki						**Bitcoin**						**Cit**					
(ρ_1, ρ_2)	d	Greedy		REPL1		REPL2		Greedy		REPL1		REPL2		Greedy		REPL1		REPL2	
		Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
(1,2)	0	4821	**4812**	7035	7047	4821	**4812**	**2715**	3205	3650	5655	2725	3205	8892	**8327**	23005	22784	8892	**8327**
(1,2)	50	**4702**	**4702**	6975	6996	4747	4744	1672	2651	**1020**	3723	1202	1523	8055	7959	22324	22447	8542	**7898**
(1,2)	100	**4645**	4651	6924	6944	4694	4694	910	2571	**634**	3110	857	1136	7812	**7631**	21938	22289	8362	7692
(1,2)	150	**4592**	4601	6872	6894	4643	4644	715	2199	**472**	2580	695	952	7640	**7468**	21624	22057	8205	7555
(1,2)	200	**4542**	4551	6815	6843	4612	4594	538	1908	**378**	2196	567	796	7497	**7303**	21278	21691	8081	7442
(1,2)	250	**4491**	4501	6763	6793	4576	4544	406	1797	**307**	1830	475	666	7348	**7151**	20946	21410	7967	7330
(1,2)	300	**4441**	4451	6710	6743	4526	4494	328	1695	**257**	1536	408	573	7202	**7042**	20621	21152	7873	7224
(1,2)	350	**4390**	4401	6658	6693	4476	4444	256	1509	**207**	1257	345	505	7083	**6907**	20328	20851	7782	7152
(1,2)	400	**4340**	4351	6608	6642	44226	4394	189	1331	**157**	1018	303	455	6991	**6815**	20031	20531	7682	7076
(1,2)	50	**4702**	5709	6975	8346	4747	5686	1672	1844	**1020**	1185	1202	1282	**8055**	9522	22324	26406	8542	10050
(2,4)	50	4690	4694	6071	6405	4687	**4685**	**109**	218	1271	1503	137	222	5766	**5243**	22547	22793	6107	5426
(4,6)	50	4689	4690	5515	5818	**4686**	4688	**22**	35	337	261	76	95	5396	**4624**	20826	21201	5363	4645
(5,10)	50	4689	4693	5323	5537	**4684**	4685	**17**	25	275	165	30	34	5381	**4574**	19405	19872	5349	4620

References

[1] A. Moskvina and J. Liu, "How to build your network? a structural analysis," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, 2016, pp. 2597–2603.

[2] B. Yan, Y. Liu, J. Liu, Y. Cai, H. Su, and H. Zheng, "From the periphery to the center: information brokerage in an evolving network," in *Proc. IJCAI 2018*, 2018, pp. 3912–3918.

[3] B. Yan, Y. Chen, and J. Liu, "Dynamic relationship building: exploitation versus exploration on a social network," in *International Conference on Web Information Systems Engineering*. Springer, 2017, pp. 75–90.

[4] E. Ravasz and A. L. Barabsi, "Hierarchical organization in complex networks," *Physical Review E Statistical Nonlinear & Soft Matter Physics*, vol. 67, no. 2, p. 026112, 2003.

[5] J. Liu and A. Moskvina, "Hierarchies, ties and power in organizational networks: model and analysis," *Social Network Analysis and Mining*, vol. 6, no. 1, p. 106, 2016.

[6] D. Bouhnik and Y. Giat, "Information gatekeepers–aren't we all?" *Informing Science: The International Journal of an Emerging Transdiscipline*, vol. 18, pp. 127–144, 2015.

[7] L. Lü, D. Chen, X.-L. Ren, Q.-M. Zhang, Y.-C. Zhang, and T. Zhou, "Vital nodes identification in complex networks," *Physics Reports*, vol. 650, pp. 1–63, 2016.

[8] A. Moskvina and J. Liu, "Integrating networks of equipotent nodes," in *International Conference on Computational Social Networks*. Springer, 2016, pp. 39–50.

[9] ——, "Togetherness: an algorithmic approach to network integration," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 2016, pp. 223–230.

[10] Y. Cai, H. Zheng, J. Liu, B. Yan, H. Su, and Y. Liu, "Balancing the pain and gain of hobnobbing: Utility-based network building over atributed social networks," in *Proceedings of 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 193–201.

[11] Y. Tang, J. Liu, W. Chen, and Z. Zhang, "Establishing connections in a social network," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2018, pp. 1044–1057.

[12] S. P. Borgatti, "The key player problem," in *Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers*. National Academies Press, 2003, p. 241.

[13] T. Milenković, V. Memišević, A. Bonato, and N. Pržulj, "Dominating biological networks," *PloS one*, vol. 6, no. 8, p. e23016, 2011.

[14] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[15] M. A. Henning, "Distance domination in graphs," *Pure and Applied Mathematics*, vol. 209, pp. 321–350, 1998.

[16] E. Sampathkumar and H. Walikar, "The connected domination number of a graph," *J. Math. Phys*, 1979.

[17] J. Wu and H. Li, "On calculating connected dominating set for efficient routing in ad hoc wireless networks," in *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*. ACM, 1999, pp. 7–14.

[18] E. J. Cockayne, R. Dawes, and S. T. Hedetniemi, "Total domination in graphs," *Networks*, vol. 10, no. 3, pp. 211–219, 1980.

[19] C.-S. Liao and G. J. Chang, "k-tuple domination in graphs," *Information Processing Letters*, vol. 87, no. 1, pp. 45–50, 2003.

[20] J. K. Lan and G. J. Chang, "On the mixed domination problem in graphs," *Theoretical Computer Science*, vol. 476, pp. 84–93, 2013.

[21] P. Jain, M. Jayakrishnan, F. Panolan, and A. Sahu, "Mixed dominating set: A parameterized perspective," in *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 2017, pp. 330–343.

[22] D. Moshkovitz, "The projection games conjecture and the np-hardness of $\ln n$-approximating set-cover." in *APPROX-RANDOM*. Springer, 2012, pp. 276–287.

[23] B. Khoussainov, J. Liu, and I. Khaliq, "A dynamic algorithm for reachability games played on trees," in *International Symposium on Mathematical Foundations of Computer Science*. Springer, 2009, pp. 477–488.

[24] T. M. Neeson, M. C. Ferris, M. W. Diebel, P. J. Doran, J. R. OHanley, and P. B. McIntyre, "Enhancing ecosystem restoration efficiency through spatial and temporal coordination," *Proceedings of the National Academy of Sciences*, vol. 112, no. 19, pp. 6236–6241, 2015.

[25] X. Wu, A. Kumar, D. Sheldon, and S. Zilberstein, "Robust optimization for tree-structured stochastic network design," *arXiv preprint arXiv:1612.00104*, 2016.

[26] R. Kumar, K. Punera, and A. Tomkins, "Hierarchical topic segmentation of websites," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 257–266.

[27] E. Cockayne, S. Goodman, and S. Hedetniemi, "A linear algorithm for the domination number of a tree," *Information Processing Letters*, vol. 4, no. 2, pp. 41–44, 1975.

[28] J. G. Propp and D. B. Wilson, "How to get a perfectly random sample from a generic markov chain and generate a random spanning tree of a directed graph," *Journal of Algorithms*, vol. 27, no. 2, pp. 170–217, 1998.

[29] P. Erdös and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, pp. 17–61, 1960.

[30] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," Cornell University, Tech. Rep., 1999.

[31] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2010, pp. 1361–1370.

[32] S. Kumar, F. Spezzano, V. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 221–230.

[33] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *Proceedings of the eleventh ACM SIGKDD*. ACM, 2005, pp. 177–187.