

A Game Theoretic Approach for a Minimal Secure Dominating Set

Xiuyang Chen, Changbing Tang, *Senior Member, IEEE*, and Zhao Zhang

Abstract—The secure dominating set (SDS), a variant of the dominating set, is an important combinatorial structure used in wireless networks. In this paper, we apply algorithmic game theory to study the minimum secure dominating set (MinSDS) problem in a multi-agent system. We design a game framework for SDS and show that every Nash equilibrium (NE) is a minimal SDS, which is also a Pareto-optimal solution. We prove that the proposed game is an exact potential game, and thus NE exists, and design a polynomial-time distributed local algorithm which converges to an NE in $O(n)$ rounds of interactions. Extensive experiments are done to test the performance of our algorithm, and some interesting phenomena are witnessed.

Index Terms—Algorithmic game theory, multi-agent systems, potential game, secure dominating set.

I. INTRODUCTION

WITH the rapid progress of wireless network technology, some distributed network systems such as multi-agent systems, sensor networks, and wireless ad hoc networks have become more and more popular for network monitoring which only requires little intervention from people. To save cost, it is desirable to use small number of intelligent devices to collect information from the whole system. Such a consideration leads to the minimum dominating set (MinDS) problem [1].

Note that in a MinDS problem, each sensor is only responsible for detecting security issues of the system, and is not endowed with the power to handle them. This limits the scope of application of dominating sets. For example, in a museum, each pavilion is viewed as a vertex of a graph, and the corridors between the pavilions form the edge set of the graph. A guard in pavilion v_i is responsible for monitoring the surrounding pavilions $N[v_i]$, where $N[v_i]$ is the closed neighbor set of v_i . If every pavilion could be monitored, then those pavilions with guards form a dominating set. If a problem

occurs in a guarded pavilion, then the guard in the corresponding pavilion can deal with the issue. If a problem occurs in an unguarded pavilion, then a nearby guard could move to deal with the issue. At the same time, it is expected that all pavilions remain to be monitored after movement. This leads to the minimum secure dominating set (MinSDS) problem. In addition to the above scenario, SDS problems are also applied in the context of the protection system, network security system, military strategy analysis, etc. [1]–[4].

The concept of SDS was first introduced by Cockayne *et al.* [5] in 2005. Burger *et al.* [6] designed two exponential-time algorithms for the MinSDS problem in general graphs. It is known that the MinSDS problem is NP-hard. In fact, Boumediene Merouane and Chellali [7] proved that the MinSDS problem is NP-hard even when restricted to bipartite graphs and split graphs. Therefore, many people attempted to solve the MinSDS problem with special classes of graphs [4], [8]–[15]. These algorithms are centralized. Note that centralized algorithms are vulnerable to external attacks. Damage to the center may lead to a breakdown of the whole system. This observation motivates us to seek distributed algorithms for the MinSDS problem.

A distributed algorithm can effectively reduce the damage of attacks, especially in multi-agent systems, in which each node is an autonomous agent and can determine its strategy for the next step based on currently collected information (which is often local since an agent might not be as powerful as a center). Such an autonomy eliminates the dependence on a center, and greatly improves anti-attack capability. Note that individual benefits and social welfare are often conflicting. Thus, a disadvantage resulting from such autonomy is that a self-organized algorithm take a longer time to run, and may fail to get satisfactory solutions.

Using game theory can effectively cope with the above disadvantages, and the hypothetical definition of a player is exactly what we expect from an agent: rational, intelligent, and selfish. Game theory has the advantage whereby providing interaction frameworks (game design) and specific local rules (distributed methods), a satisfactory collective behavior may be theoretically guaranteed through competition and cooperation among individuals. For studies of game theory in wireless and communication networks, readers may refer to the monograph [16] and references therein.

Compared with the large quantities of approximation algorithms for the MinDS problem and its variants [17], studies on the domination problems using game theory are not as common. Yen and Chen [18] designed a multi-domination game

Manuscript received June 13, 2022; accepted July 14, 2022. This work was supported in part by the National Natural Science Foundation of China (U20A2068, 11771013) and Zhejiang Provincial Natural Science Foundation of China (LD19A010001). Recommended by Associate Editor Lei Ding. (Corresponding author: Zhao Zhang.)

Citation: X. Y. Chen, C. B. Tang, and Z. Zhang, “A game theoretic approach for a minimal secure dominating set,” *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 12, pp. 2258–2268, Dec. 2023.

X. Y. Chen is with the College of Mathematics and System Science, Xinjiang University, Urumqi 830000, China (e-mail: xiuyangchen@126.com).

C. B. Tang is with the College of Physics and Electronic Information Engineering, Zhejiang Normal University, Jinhua 321004, China (e-mail: tangcb@zjnu.edu.cn).

Z. Zhang is with the School of Mathematical Sciences, Zhejiang Normal University, Jinhua 321004, China (e-mail: hxhzz@sina.com).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/JAS.2023.123315

and proved that every Nash equilibrium (NE) of the game is a minimal multi-dominating set, and is also a Pareto optimal solution. Based on the game, a distributed algorithm was designed. Note that the algorithm uses a central daemon which allows only one process at a time. Later, Yen and Sun [19] designed an independent domination game and proved that every NE is a minimal independent dominating set. A distributed algorithm was also presented under a central daemon. Chen and Zhang [20] designed a connected domination game and proved that every NE is a minimal connected dominating set and a distributed algorithm can find an NE in linear rounds of interactions. An example was given showing that the game cannot guarantee Pareto optimality. All these game theoretical algorithms are sequential, i.e., players have to make decisions in turn. As far as we know, algorithms for the secure dominating set problem are rare, and existing algorithms are all used for special graphs. Our paper is the first one using game theory to compute a secure dominating set in general graphs. Furthermore, our algorithm allows players to make decisions simultaneously, only using local information.

Another closely related area of research is game theoretical algorithms for the minimum vertex cover (MinVC) problem [21]–[25]. Different from the dominating set problem which uses vertices to monitor vertices, the vertex cover problem uses vertices to monitor edges. Nevertheless, many ideas used in these studies are very inspiring to our work. Yang and Li [21] used snowdrift games to study the MinVC problem, and devised a distributed algorithm, trying to find a vertex cover with a small size. Tang *et al.* [22] generalized the study to the weighted version, and designed an asymmetric game algorithm, trying to find a vertex cover with small weight. Sun *et al.* [23], [24] proposed a distributed algorithm for the MinWVC problem based on relaxed greed and finite memory. They used potential game theory to prove its convergence to an NE. Chen *et al.* [25] proposed a weighted vertex cover game using a 2-hop adjustment scheme, trying to get a better solution. In all of these works, minimality of NEs were proved, but Pareto optimality was not discussed, which is a preferred quality of solutions from a social point of view.

In addition to the above works which are most closely related with our work, there are also many game theoretical studies on various coverage problems from different perspectives. For example, Li *et al.* [26], [27] employed cost sharing methods and mechanisms for a generalized set cover game. The focus was on whether players chose to lie. Fang and Kong [28] studied the core stability of vertex cover games. The core of a game is an important concept in cooperative game theory, which ensures that players have a willingness to participate. The cores of a domination game were studied by Velzen in [29], and the cores of a connected domination game were studied by Kim in [30]. Another measure of the quality of a game theoretical solution is the price of anarchy (PoA), which is the ratio between the cost of a worst NE and the cost of a socially optimal solution. Ai *et al.* [31] studied PoA and computational complexity of a coverage game. Note that their algorithm is centralized.

In this paper, we study the MinSDS problem in a multi-agent system using potential game theory. The main contribu-

tions are summarized as follows.

1) *Game Design*: We design a security domination game in multi-agent systems, in which every player determines himself to be a dominator or a dominee. By carefully designing utility functions for the players, we show that every Nash equilibrium (NE) is a minimal secure dominating set (MSDS), as well as a Pareto-optimal solution. Furthermore, we prove that this game is an exact potential game, and thus NE exists.

2) *Algorithm Design*: Based on the above security domination game, we propose a best response dynamic distributed local algorithm (BRDDLA) for the secure dominating set (SDS) problem. The algorithm is distributed and local: in each round of the algorithm, every player decides by himself on his strategy based on local information in his 6-hop neighborhood.

3) *Efficiency of Algorithm*: We prove that BRDDLA can converge to an NE in $O(n)$ rounds, where n is the number of players. Hence the algorithm can find a minimal secure dominating set, which is also a Pareto-optimal solution, in linear time. Since the MinSDS problem is NP-hard, so minimal solutions and Pareto-optimal solutions that can be obtained in linear time are fairly good. Furthermore, an NE solution achieves a kind of balance in a multi-agent system, which is also a desirable property.

4) *Verification of Performance via Simulation*: Simulations are done to experiment on the performance of our algorithm on randomly generated graphs. Since there is no previous algorithm on MinSDS in general graphs (GA), we compare BRDDLA with a natural greedy algorithm. It turns out that BRDDLA can obtain a much better solution than GA. Furthermore, we compare the algorithm with the exact algorithm on trees. It turns out that the output of BRDDLA is close to optimal solutions. Furthermore, the number of rounds of BRDDLA is less than those of reference algorithms, especially on the random tree graphs. It is also interesting to note that when testing the effect of decision priority on the performance of the algorithm in a Barabasi-Albert (BA) graph, a better performance can be reached if we let players decide in the order that the vertices are generated by the BA model. This might suggest that the earlier a vertex is generated in a BA graph, the more important it might be.

The remaining parts of the paper are organized as follows. Section II introduces preliminaries in game theory which are used in this paper. Section III designs the security domination game. Section IV provides strict theoretical analysis for the game. Section V describes the algorithm in details and provides theoretical analysis on its convergence. Section VI evaluates the performance and complexity of the algorithm through extensive simulation. Section VII concludes the paper with some discussions on future work.

II. PRELIMINARIES

In this section, we give the formal definition of SDS, and some basic terminologies and notations of graph theory and game theory. Main notations are summarized in Table I.

Let $G = (V, E)$ be a graph with vertex set V and edge set E . We say that two vertices v_i and v_j are adjacent if $(v_i, v_j) \in E$, in this case, we also say that v_i and v_j are neighbors of each

TABLE I
LIST OF NOTATIONS

Notation	Meaning
Γ	The game
n	The number of vertices, and also the number of players
v_i	Both the vertex in the graph and the player in the game
V	The set of players or vertices $V = \{v_1, v_2, \dots, v_n\}$
S_i	$S_i = \{0, 1\}$ is the strategy set of player v_i
Σ	$\Sigma = S_1 \times \dots \times S_n$ is the strategy space
C	$C = (c_1, c_2, \dots, c_n) \in \Sigma$ is a strategy profile C also denotes the vertex set $\{v_i : c_i = 1\}$
$d(v_i, v_j)$	The length of shortest path between vertex v_i and v_j
N_i^k	$N_i^k = \{v_j \in V : d(i, j) \leq k\}$
$\tilde{N}_i^1(C)$	$\tilde{N}_i^1(C) = \{v_j \in N_i^1 : \prod_{v_k \in N_j^1 \setminus \{v_i\}} (1 - c_k) = 1\}$
$D_i(C)$	$D_i(C) = \{v_k \in N_i^1 : c_k = 1\}$
$\deg(v_i)$	The degree of vertex v_i in graph G
$u_i(C)$	v_i 's utility with respect to strategy profile C

other. The open neighbor set of v_i , denoted as $N(v_i)$, consists of those neighbors of v_i in G . The degree of vertex v_i is $\deg(v_i) = |N(v_i)|$. The closed neighbor set of v_i is $N[v_i] = N(v_i) \cup \{v_i\}$. The distance between v_i and v_j , denoted as $d(v_i, v_j)$, is the length of a shortest path between v_i and v_j in G . The k -hops neighborhood of v_i , denoted as N_i^k , consists of all those vertices at distance at most k from v_i . Clearly, v_i also belongs to N_i^k and $N_i^1 = N[v_i]$.

For a vertex subset $C \subseteq V$, a vertex $v \in V \setminus C$ is dominated by C if v has a neighbor in C . Vertices in C are called dominators and vertices in $V \setminus C$ which are dominated by C are called dominatees. A dominating set of G is a subset $C \subseteq V$ which dominates every vertex of $V \setminus C$. If C is not a dominating set, those vertices which are not dominated by C are called orphans.

Definition 1 (Dominating Set (DS)): Given a graph $G = (V, E)$, a subset $C \subseteq V$ is a dominating set of G if every vertex $v \in V \setminus C$ is adjacent to at least one vertex $u \in C$.

Definition 2 (SDS): Given a graph $G = (V, E)$, a dominating set $C \subseteq V$ is a secure dominating set of G if for each vertex $u \in V \setminus C$, there exists a vertex $v \in C$ such that $uv \in E$ and $(C \setminus \{v\}) \cup \{u\}$ is also a dominating set of G .

Definition 3 (MinSDS Problem): Given a graph $G = (V, E)$, let S_{SDS} be the set of all SDSs of G . The goal of the MinSDS problem is to find an SDS with the minimum cardinality, that is, $C^* = \arg \min_{C \in S_{SDS}} |C|$.

A game Γ can be written as $\Gamma = (V; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n)$, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of players, S_i is v_i 's strategy set, and u_i is v_i 's utility function. The strategy space of the game is $\Sigma = S_1 \times S_2 \times \dots \times S_n$. A strategy profile is an n -tuple $C = (c_1, c_2, \dots, c_n) \in \Sigma$. For player v_i , we may express C as $C = (c_i, C_{-i})$, where $C_{-i} = (c_1, \dots, c_{i-1}, c_{i+1}, \dots, c_n)$ indicates the strategies of those players except v_i . Function $u_i(C)$ is the utility of v_i under strategy profile C . Players are assumed to be selfish, intelligent and rational, which means that the goal of every player is to maximize his own utility. The best response of

player v_i to current strategy profile C is

$$BR(v_i, C) = \arg \max \{u_i(c'_i, C_{-i}) : c'_i \in S_i\}.$$

A Nash equilibrium (NE) is a strategy profile C such that no player wants to deviate from C unilaterally. The formal definition of NE is as follows.

Definition 4 (NE [32]): Given a game $\Gamma = (V; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n)$, a strategy profile $C^* = (c_1^*, c_2^*, \dots, c_n^*)$ is an NE if

$$u_i(c_i^*, C_{-i}^*) \geq u_i(c_i, C_{-i}^*) \text{ for any } v_i \in V \text{ and any } c_i \in S_i.$$

Note that an NE is not necessarily a global optimal solution. Especially for an NP-hard problem, in many cases, people are satisfied with minimal solutions or Pareto-optimal solutions.

Definition 5 (MSDS): A secure dominating set C of graph G is an MSDS if for any vertex $v \in C$, the vertex set $C \setminus \{v\}$ is no longer a secure dominating set of G .

Definition 6 (Pareto-Optimal Solution): Given a game $\Gamma = (V; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n)$, a strategy profile $C' = (c'_1, \dots, c'_n)$ strictly dominates strategy profile $C = (c_1, \dots, c_n)$ if $u_i(C') \geq u_i(C)$ holds for any index $i \in \{1, \dots, n\}$ and there exists an index $j \in \{1, \dots, n\}$ with $u_j(C') > u_j(C)$. A strategy profile $C^* = (c_1^*, \dots, c_n^*)$ is a Pareto-optimal solution if there is no strategy profile which strictly dominates C^* .

III. SECURITY DOMINATION GAME

In this section, we design a game framework for the secure dominating set problem.

For a vertex v_i , its c -value $c_i = 1$ indicates that v_i is a dominator. Denote by N_i^k the k -th closed neighborhood of vertex v_i , including v_i itself. A vertex $v_j \in \tilde{N}_i^1(C)$ means that v_j is in the closed neighborhood of v_i such that v_i is the only vertex in the closed neighborhood of v_j that is possibly a dominator. The set $D_i(C)$ contains those vertices in the closed neighborhood of v_i which are dominators.

Definition 7 (Secure Vertex): For a vertex set C (which might not necessarily be a dominating set), we call a vertex $v_j \in V \setminus C$ to be *secure* if $\exists v_k \in C$ such that $v_k v_j \in E$ and $\tilde{N}_k^1 \subseteq N_j^1$. Otherwise we say that v_j is *insecure*.

Note that a secure vertex must be a dominatee because of the existence of a neighbor v_k in C . Insecure vertices include both insecure dominatees and orphans. The idea behind such a terminology is based on the observation that if v_j is a secure vertex and v_k satisfies Definition 7, then $(C \setminus \{v_k\}) \cup \{v_j\}$ is still a dominating set.

Assume that all players are selfish, intelligent and rational, in the sense that they will not consider the benefit of the other players while seeking to maximize their own benefits. The most critical part of the game is to design good utility functions for the players such that a stable and reasonable social status can be reached through cooperation and competition among players. The following are some preferred features of a game.

1) *Self-Stability:* Starting from any initial state, the game can end up in an NE which corresponds to a secure dominating set.

2) *Small Size:* The cardinality of the SDS corresponding to an NE should be reasonably small. Since the computation of a

minimum SDS is NP-hard even using centralized algorithms, we cannot hope for a minimum SDS in reasonable time. An alternative requirement is that the computed SDS should be minimal, that is, removing any vertex will no longer be an SDS.

3) *Time Efficiency*: The time for the game to reach an NE should be polynomial in the size of the instance.

We define utility functions as follows. For a strategy profile $C = (c_1, c_2, \dots, c_n)$, the utility function of v_i is defined as

$$u_i(C) = g_i(C) + q_i(C) \quad (1)$$

where $g_i(C)$ and $q_i(C)$ are defined to be

$$g_i(C) = -\lambda_1 c_i \quad (2)$$

$$q_i(C) = -\lambda_2 \sum_{v_j \in N_i^3} (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C) \quad (3)$$

where $m_{k,j}(C)$ is an indicator function defined as

$$m_{k,j}(C) = \begin{cases} 1, & |\tilde{N}_k^1(C) \setminus N_j^1| \neq \emptyset \\ 0, & |\tilde{N}_k^1(C) \setminus N_j^1| = \emptyset \end{cases} \quad (4)$$

and λ_1, λ_2 are constants satisfying $0 < \lambda_1 < \lambda_2$. When $D_j(C) = \emptyset$, we regard $\prod_{v_k \in D_j(C)} m_{k,j}(C)$ to be 1. So, an orphan $v_j \in N_i^3$ always contributes $-\lambda_2$ to $q_i(C)$.

The idea behind the definition of $m_{k,j}(C)$ comes from the observation that if v_j is a secure vertex, then there exists a vertex $v_k \in D_j(C)$ satisfying Definition 7, which has $\tilde{N}_k^1(C) \setminus N_j^1 = \emptyset$ and thus $m_{k,j}(C) = 0$. Therefore, secure vertex v_j contributes 0 to $q_i(C)$. On the other hand, if v_j is an insecure dominatee, then for any vertex $v_k \in D_j(C)$, we have $\tilde{N}_k^1 \setminus N_j^1 \neq \emptyset$, and thus $\prod_{v_k \in D_j(C)} m_{k,j}(C) = 1$. Hence an insecure dominatee in N_i^3 will contribute $-\lambda_2$ to $q_i(C)$. Notice that a dominator also contributes 0 to $q_i(C)$ since its c -value equals 1. Combining these observations with the comment in the last sentence of the last paragraph, we have proved the following property.

Property 1: For any vertex v_i , a vertex in N_i^3 contributes $-\lambda_2$ to $q_i(C)$ if and only if it is an insecure vertex, otherwise, the contribution is 0. Hence, $q_i(C) = -\lambda_2 |IS_i(C)|$, where $IS_i(C)$ is the set of insecure vertices in N_i^3 .

The reason why we use N_i^3 in the definition of $q_i(C)$ is because of the following property:

Property 2: The status of vertex v_i does not affect the value of $m_{k,j}(C)$ for any $v_j \notin N_i^3$ and $v_k \in D_j(C)$.

Property 2 follows from the definition of $m_{k,j}$, and an intuition can be gained by considering the example in Fig. 1. Notice that $m_{k,j}(C) = 1$ since $\tilde{N}_k^1(C) = \{v_l, v_k, v_j\} \not\subseteq N_j^1$, while $m_{k,j}(C') = 0$ since $\tilde{N}_k^1(C') = \{v_k, v_j\} \subseteq N_j^1$. This is because of the influence of vertex v_i . But if v_j is at least 4 hops away from v_i , such an influence will not occur.

To gain some intuition about u_i , consider the example in Fig 2, in which v_1, v_2, v_4 are dominators, v_3 is a secure dominatee, v_5, v_6 are insecure dominatees (for example, $D_5(C) = \{v_4\}$ and $\tilde{N}_4^1 = \{v_5, v_6\}$, so $\tilde{N}_4^1 \setminus N_5^1 = \{v_6\} \neq \emptyset$, and thus v_5 is an insecure dominatee), and v_7 is an orphan.

1) A dominator v_i has $c_i = 1$, and thus gets a punishment of λ_1 from $g_i(C)$. For this instance, $g_1(C) = g_2(C) = g_4(C) = -\lambda_1$.

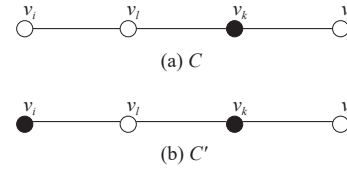


Fig. 1. An example illustrating Property 2.

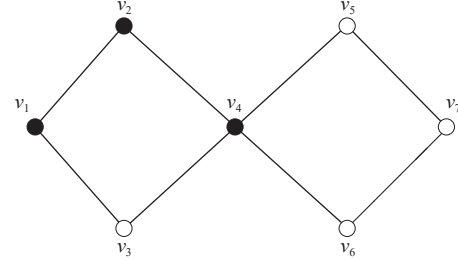


Fig. 2. An example illustrating the ideas behind the definition of utility function u_i . Blackened vertices are selected vertices in strategy profile C .

2) If v_i is an insecure dominatee, then it will encourage himself to be a dominator, and at the same time help some insecure neighbors become secure. For example, v_5 is an insecure dominatee with $u_5(C) = -3\lambda_2$. If we change the status of v_5 to be a dominator, then $u_5(C'_5 = 1, C_{-5}) = -\lambda_1$. Because $0 < \lambda_1 < \lambda_2$, we have $u_5(C'_5 = 1, C_{-5}) > u_5(C_5 = 0, C_{-5})$. Thus, u_5 has an incentive to change his strategy from 0 to 1, while at the same time, both v_6 and v_7 become secure dominatees.

3) If v_i is a secure dominatee which cannot help any insecure dominatee to become secure, then he has no incentive to change his current strategy. For example, v_3 is a secure dominatee with $u_3(C) = -3\lambda_2$. If v_3 changes from 0 to 1, then vertices v_5, v_6, v_7 are still insecure, and $u_3(C'_3 = 1, C_{-3}) = -\lambda_1 - 3\lambda_2 < u_3(C)$. So, v_3 prefers to stay 0.

4) The above 2) and 3) show that the designed utility can stimulate an evolution towards a feasible solution. We can show that the designed utility can also provide an incentive for redundant players to leave C . For example, v_2 is redundant in the sense that if v_2 changes his strategy from $c_2 = 1$ to $c'_2 = 0$, then he becomes a secure dominatee and does not cause other secure dominatees to become insecure. In view of utility, $u_2(C) = -\lambda_1 - 3\lambda_2 < u_2(C'_2 = 0, C_{-2}) = -3\lambda_2$. Thus, v_2 is willing to retreat from the dominating set.

IV. THEORETICAL ANALYSIS

In this section, we analyze the theoretical properties of the security domination game designed in the above section. In the following, we shall use C to denote both a strategy profile $(c_1, \dots, c_n) \in \Sigma$ and the vertex set $\{v_i : c_i = 1\}$ it corresponds to. We assume that a player is willing to change his strategy only when he can be *strictly* better off.

A. Nash Equilibrium Solution

In this subsection, we show that an NE of the game always corresponds to a minimal secure dominating set. The proof makes use of the following property and Lemma 1.

Property 3: For any player $v_i \in V$ and two profiles $C \subseteq C'$, we have $m_{k,j}(C) \geq m_{k,j}(C')$ for any $v_j \in V$ and $v_k \in D_j(C)$.

Proof: Notice that a vertex $v_l \in \tilde{N}_k^1(C')$ if and only if $(N_l^1 \setminus \{v_k\}) \cap C' = \emptyset$. Since $C \subseteq C'$, $(N_l^1 \setminus \{v_k\}) \cap C' = \emptyset$ implies $(N_l^1 \setminus \{v_k\}) \cap C = \emptyset$, which is equivalent to say that $v_l \in \tilde{N}_k^1(C)$. Thus, $\tilde{N}_k^1(C') \subseteq \tilde{N}_k^1(C)$. Then the observation follows from the definition of $m_{k,j}$. ■

Lemma 1: Vertex set C is an SDS if and only if all vertices in $V \setminus C$ are secure.

Proof: To prove the “only if” part, consider an SDS C . Then all vertices in $V \setminus C$ are dominatees. If there exists an insecure dominatee $v_i \in V \setminus C$, then for any dominator v_j of v_i , $\tilde{N}_j^1 \setminus N_i^1 \neq \emptyset$. In such a case, any vertex $v_k \in \tilde{N}_j^1 \setminus N_i^1$ must be an orphan with respect to $C \setminus \{v_j\} \cup \{v_i\}$, and thus $C \setminus \{v_j\} \cup \{v_i\}$ is not a DS. By the arbitrariness of v_j , C is not an SDS, which is a contradiction.

To prove the “if” part, suppose all vertices in $V \setminus C$ are secure. Since secure vertices are dominatees, vertex set C is a DS. If C is not an SDS, then there exists a vertex v_i such that for any vertex $v_j \in N_i^1 \cap C$, $C \setminus \{v_j\} \cup \{v_i\}$ is not a DS. This implies that $\tilde{N}_j^1 \setminus N_i^1 \neq \emptyset$, and thus v_i is not a secure vertex, which is a contradiction. ■

Theorem 1: Every Nash equilibrium of the security domination game is a minimal secure dominating set.

Proof: We first prove that every Nash equilibrium C is a secure dominating set. By Property 1, this is equivalent to showing that any player $v_i \in V \setminus C$ is secure. Suppose this is not true and consider an insecure vertex v_i . By Property 1, v_i contributes $-\lambda_2$ to $q_i(C)$, and thus

$$u_i(C) = -\lambda_2 - \lambda_2 \sum_{v_j \in N_i^3 \setminus \{v_i\}} (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C).$$

Let $C' = (c'_i = 1, C_{-i})$, where we have

$$u_i(C') = -\lambda_1 - \lambda_2 \sum_{v_j \in N_i^3 \setminus \{v_i\}} (1 - c_j) \prod_{v_k \in D_j(C')} m_{k,j}(C').$$

By Property 3 and because $D_j(C) \subseteq D_j(C')$, we have

$$\prod_{v_k \in D_j(C')} m_{k,j}(C') \leq \prod_{v_k \in D_j(C)} m_{k,j}(C') \leq \prod_{v_k \in D_j(C)} m_{k,j}(C).$$

Combining this with the assumption $0 < \lambda_1 < \lambda_2$, we have $u_i(C') > u_i(C)$, which contradicts the fact that C is a Nash equilibrium. Thus, C is a secure dominating set.

Next we show that C is a minimal SDS. Otherwise, there is a vertex $v_i \in C$ such that $C' = C \setminus \{v_i\}$ is still an SDS. Then any vertex in $V \setminus C'$ is secure, and the same holds for any vertex in $V \setminus C$. By Property 1, $u_i(C) = -\lambda_1$ and $u_i(C') = 0$. Thus, v_i is willing to change from $c_i = 1$ to $c'_i = 0$, which contradicts that C is an NE. ■

B. Potential Game and Existence of NE

Note that a game in strategic form does not necessarily have a pure Nash equilibrium [32]. However, a potential game always has a pure Nash equilibrium [33]. In this subsection, we prove the existence of NE for our game by showing that it is a potential game, and show that an NE can be reached in linear rounds of interactions among the players.

Definition 8 (Exact Potential Game): We call a game

$\Gamma = (V; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n)$ an exact potential game if there exists a potential function $\pi: \Sigma \mapsto \mathbb{R}$ such that for any player $v_i \in V$ and any strategies $c_i, c'_i \in S_i$, $c_{-i} \in S_{-i}$, the following equality holds:

$$\pi(c_i, C_{-i}) - \pi(c'_i, C_{-i}) = u_i(c_i, C_{-i}) - u_i(c'_i, C_{-i}).$$

Lemma 2: The proposed security domination game is an exact potential game.

Proof: We prove that the following function is a potential function:

$$\pi(C) = -\lambda_1 \sum_{j=1}^n c_j - \lambda_2 \sum_{j=1}^n (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C).$$

Denote the two terms of $\pi(C)$ as $\pi^{(1)}(C)$ and $\pi^{(2)}(C)$, respectively.

Let $C = (c_i, C_{-i})$ and $C' = (c'_i, C_{-i})$ be two strategy profiles before and after some v_i changes its strategy from c_i to c'_i . We may assume, without loss of generality, that $c_i = 0$ and $c'_i = 1$. It can be calculated that

$$\pi^{(1)}(C) - \pi^{(1)}(C') = \lambda_1 = g_i(C) - g_i(C') \quad (5)$$

and

$$\begin{aligned} \pi^{(2)}(C) - \pi^{(2)}(C') &= -\lambda_2 \sum_{v_j \in N_i^3} (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C) \\ &\quad - \lambda_2 \sum_{v_j \notin N_i^3} (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C) \\ &\quad + \lambda_2 \sum_{v_j \in N_i^3} (1 - c'_j) \prod_{v_k \in D_j(C')} m_{k,j}(C') \\ &\quad + \lambda_2 \sum_{v_j \notin N_i^3} (1 - c_j) \prod_{v_k \in D_j(C')} m_{k,j}(C') \\ &= -\lambda_2 \sum_{v_j \in N_i^3} (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C) \\ &\quad + \lambda_2 \sum_{v_j \in N_i^3} (1 - c'_j) \prod_{v_k \in D_j(C')} m_{k,j}(C') \\ &= q_i(C) - q_i(C') \end{aligned} \quad (6)$$

where the second equality holds because of Property 2 and thus the second and the fourth terms are cancelled. Combining (5) and (6), we have

$$\pi(C) - \pi(C') = u_i(C) - u_i(C').$$

■

Theorem 2: The secure domination game always has an NE. Furthermore, starting from any initial state, the number of iterations needed for the security domination game to reach an NE is at most $\frac{(\lambda_1 + \lambda_2)n}{\min\{\lambda_2 - \lambda_1, \lambda_1\}} = O(n)$.

Proof: We consider the minimum benefit that a player can achieve in each round. Suppose v_i changes his strategy from c_i to c'_i to achieve a positive profit. Let $C = (c_i, C_{-i})$ and $C' = (c'_i, C_{-i})$.

If $c_i = 0$ and $c'_i = 1$, by the definition of utility functions and

Property 1, the incentive for player v_i to change from 0 to 1 is that he can help some insecure vertex become a secure vertex, hence $q_i(C') - q_i(C) = \lambda_2(|IS_i(C)| - |IS_i(C')|) \geq \lambda_2$. Combining this with $g_i(C') = -\lambda_1$ and $g_i(C) = 0$, we have $u_i(C') - u_i(C) \geq \lambda_2 - \lambda_1 > 0$.

In the case $c_i = 1$ and $c'_i = 0$, changing a vertex from 1 to 0 cannot reduce the number of insecure vertices, and thus $q_i(C') - q_i(C) \leq 0$. If $q_i(C') - q_i(C) < 0$, then $q_i(C') - q_i(C) \leq -\lambda_2$, the decrease is too large for v_i to be willing to change. Thus, v_i has an incentive for such a change only when $q_i(C') = q_i(C)$ and $u_i(C') - u_i(C) = g_i(C') - g_i(C) = 0 - (-\lambda_1) = \lambda_1$.

By Lemma 2, $\pi(C') - \pi(C) = u_i(C') - u_i(C) \geq \min\{\lambda_2 - \lambda_1, \lambda_1\}$. Notice that $-(\lambda_1 + \lambda_2)n \leq \pi(C) \leq 0$. Hence the number of iterations is at most $\frac{0 - (-(\lambda_1 + \lambda_2)n)}{\min\{\lambda_2 - \lambda_1, \lambda_1\}}$, which is $O(n)$ since λ_1 and λ_2 are constants satisfying $0 < \lambda_1 < \lambda_2$. For example, if taking $\lambda_2 = 2\lambda_1$, then the number of iterations is no more than $3n$. ■

C. Pareto Optimality

Note that an NE is a stable state from the view of individual players, in which no single player can be strictly benefited by unilateral deviation. While a Pareto optimal solution is a stable state in which a strict benefit for some players will definitely harm the interest of some others, which is from the view of social welfare. This subsection shows that any NE of the game is also Pareto optimal.

Theorem 3: Every NE of the security domination game is a Pareto-optimal solution.

Proof: First, notice that for any strategy profile $C \in 2^V$, $u_i(C) \leq -\lambda_1 < 0$ for any player $v_i \in C$, and $u_i(C) \leq 0$ for any player $v_i \in V \setminus C$.

Suppose $C = (c_1, \dots, c_n)$ is an NE but not a Pareto-optimal solution. Then there is a strategy profile $C' = (c'_1, \dots, c'_n)$ such that $u_i(C') \geq u_i(C)$ holds for any $i \in \{1, 2, \dots, n\}$, and $\exists j \in \{1, 2, \dots, n\}$ with $u_j(C') > u_j(C)$.

By Theorem 1, C is an SDS. Then by Property 1, $q_i(C) = 0$ for any player v_i . Thus, $u_i(C) = -\lambda_1$ for any $v_i \in C$ and $u_i(C) = 0$ for any $v_i \in V \setminus C$. As a consequence, for $v_i \in V \setminus C$, $0 \geq u_i(C') \geq u_i(C) = 0$ leads to $u_i(C') = 0$, which implies $v_i \in V \setminus C'$. Hence, $V \setminus C \subseteq V \setminus C'$, or equivalently, $C' \subseteq C$. Because C' and C are different, C' is a proper subset of C .

On the other hand, $u_i(C') \geq u_i(C) = -\lambda_1 > -\lambda_2$ implies that any player $v_i \in C$ has $u_i(C') = 0$ or $-\lambda_1$. This is possible only when $q_i(C') = 0$, in other words, $IS_i(C') = \emptyset$. Then, C' is also an SDS, contradicting the fact that C is a minimal SDS. ■

D. Relation of the Solutions

Let S_{SDS} , S_{MSDS} , S_{NE} and S_{POS} be the set of secure dominating sets, minimal secure dominating sets, Nash equilibria of the secure domination game, and Pareto-optimal solutions of the SDS problem, respectively. According to Theorems 1 and 3, we have $S_{NE} \subseteq S_{MSDS} \cap S_{POS}$. The relations between these sets are illustrated by Fig. 3

V. ALGORITHM DESIGN AND ANALYSIS

To realize the above SDS game, one may let players make their decisions in order until no player can improve his utility.

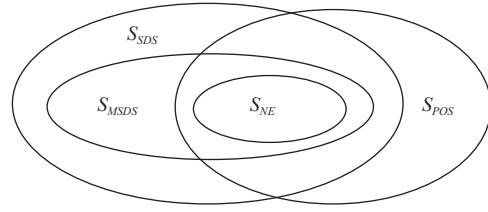


Fig. 3. Relations between solution sets S_{SDS} , S_{MSDS} , S_{NE} and S_{POS} .

Theorem 2 guarantees that in this way, the algorithm converges to an NE in $O(n)$ rounds. However, this algorithm is not distributed and is quite time consuming (since every player has to wait for the decisions of the other players).

To design a distributed algorithm, an idea is to let all players make decisions simultaneously. However, such a method may cause a failure of convergence, and may even be trapped in an infinite loop. For example, given a connected graph $G = (V, E)$, if the current strategy profile is $C = (0, \dots, 0)$, then for every player v_i , since $u_i(c'_i = 1, C_{-i}) - u_i(C) \geq \lambda_2 - \lambda_1 > 0$, he is inclined to change his strategy from 0 to 1. Thus, the next strategy profile becomes $C' = (1, \dots, 1)$. Then for every player v_i , $u_i(c''_i = 0, C'_{-i}) - u_i(C') = \lambda_1 > 0$, and thus he is inclined to change his strategy back to 0, leading to the next strategy profile back to $C = (0, \dots, 0)$. Thus, simultaneous decisions may cause the algorithm to be stuck in an infinite loop.

The reason why a mess might be created by a simultaneous decision is as follow: A best response of a player is based on the assumption that all the other players keep their strategies; thus, if two correlated players change their strategies simultaneously, then their best responses for the previous strategy profile are no longer best responses for the strategy profile after the simultaneous change.

An idea to avoid such a mess is to let only a set of independent players make decisions simultaneously. By the definition of utility function $u_i(C)$, it is easy to see that player v_i can make his decision on local information which is at most six hops away from v_i . The following argument shows that a little more storage space may further reduce the dependence on local information to at most three hops.

Notice that $\tilde{N}_k^1(C)$ only uses c -values of players in N_k^2 . Thus, for any player v_j and $v_k \in D_j(C)$, the value of $m_{k,j}(C)$ only depends on information in N_k^2 . As a consequence, every player v_j can compute $\prod_{v_k \in D_j} m_{k,j}(C)$ locally using information in N_j^3 . Suppose every v_j stores information

$$SS_j = (c_j, \prod_{v_k \in D_j} m_{k,j})$$

with respect to current solution C . Then SS_j can be computed using local information N_j^3 , and player v_i can compute $g_i(C)$ and $q_i(C)$ by accessing local information in N_i^3 .

The distributed algorithm is described in Algorithm 1. To emphasize the range of information needed for a computation, we use a subscript. For example, $BR(v_i, C_{N_i^3})$ indicates that only information in N_i^3 is needed for v_i to compute a best response. The details for the execution of the algorithm are

explained as follows.

Algorithm 1 is a distributed realization of the SDS game presented in Section II, where parameter $T = \frac{(\lambda_1 + \lambda_2)n}{\min\{\lambda_2 - \lambda_1, \lambda_1\}}$, the reason of which will be clear after proving Lemma 4. For current strategy profile C , we denote the marginal utility of player v_i by $mu_i(C) = u_i(BR(v_i, C_{N_i^3}), C_{-i}) - u_i(c_i, C_{-i})$. All players compute their marginal utility simultaneously, but not all of them change their strategy at the same time. A player v_i decides to change his strategy only when

$$i = \arg \min\{j : v_j \in N_i^6 \text{ and } mu_j(C) > 0\}. \quad (7)$$

Algorithm 1 Best Response Dynamic Distributed Local Algorithm (BRDDLA)

Input: An initial strategy profile $C^{(0)} = (c_1^{(0)}, \dots, c_n^{(0)})$

Output: A minimal SDS C'

```

1:  $C \leftarrow C^{(0)}$ 
2: for  $t = 1, 2, \dots, T$  do
3:   for every player  $v_i$  (this is done simultaneously) do
4:      $c'_i \leftarrow BR(v_i, C_{N_i^3})$  by accessing  $SS_j$  for  $v_j \in N_i^3$ 
5:      $mu_i \leftarrow u_i(c'_i, C_{-i}) - u_i(c_i, C_{-i})$ 
6:     if  $v_i$  satisfies (7) then
7:        $c_i \leftarrow c'_i$ 
8:     end if
9:   end for
10:  if  $C = C^{(t-1)}$  then
11:    Break and go to line 16
12:  else
13:     $C^{(t)} \leftarrow C$ 
14:  end if
15: end for
16: Output  $C' \leftarrow C$ 

```

That is, the player v_i has the priority to change his strategy only when he has the smallest ID among players in N_i^6 with a strictly positive marginal utility.

The reason why we use (7) to determine who can change strategy is based on the following property.

Property 4: Suppose \mathcal{H} is the set of players who have changed their strategies simultaneously in one execution of the inner for loop of Algorithm 1; then \mathcal{H} is a 3-hop independent set in the SDS game, that is,

$$N_i^3 \cap N_j^3 = \emptyset, \quad \forall v_i, v_j \in \mathcal{H}.$$

As a consequence, if a strategy profile C changes into C' , then for any player v_i who has changed his strategy by rule (7), his best response to C is also a best response to C' .

Intuitively, since decisions are made locally, only allowing players who are somewhat distant to change strategies can effectively decouple mutual influences, and thus leads to a guaranteed convergence rate. We give a strict analysis for this intuition in the following.

Lemma 3: Suppose C is the current strategy profile. After a round of the inner for loop of Algorithm 1, the new strategy profile is $C' = (C'_{\mathcal{H}}, C_{-\mathcal{H}})$, where \mathcal{H} is the set of players who have their strategies changed in this round. Then,

$$\pi(C') - \pi(C) = \sum_{v_i \in \mathcal{H}} (u_i(c'_i, C_{-i}) - u_i(c_i, C_{-i})).$$

Proof: Suppose $\mathcal{H} = \{v_{i_1}, v_{i_2}, \dots, v_{i_t}\}$. According to Property 4, \mathcal{H} is a 3-hop independent set, thus the player set V can be decomposed into disjoint union of sets $V = N_{i_1}^3 \cup N_{i_2}^3 \cup \dots \cup N_{i_t}^3 \cup V_t$, where $V_t = V \setminus \bigcup_{l=1}^t N_{i_l}^3$. Hence, the potential function $\pi(C)$ can be rewritten as

$$\begin{aligned} \pi(C) = & -\lambda_1 \sum_{v_i \in \mathcal{H}} c_i - \lambda_1 \sum_{v_i \notin \mathcal{H}} c_i \\ & - \lambda_2 \sum_{v_i \in \mathcal{H}} \sum_{v_j \in N_i^3} (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C) \\ & - \lambda_2 \sum_{v_j \in V_t} (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C). \end{aligned}$$

Similarly, $\pi(C')$ can be rewritten as

$$\begin{aligned} \pi(C') = & -\lambda_1 \sum_{v_i \in \mathcal{H}} c'_i - \lambda_1 \sum_{v_i \notin \mathcal{H}} c_i \\ & - \lambda_2 \sum_{v_i \in \mathcal{H}} \sum_{v_j \in N_i^3} (1 - c'_j) \prod_{v_k \in D_j(C')} m_{k,j}(C'_{\mathcal{H}}, C_{-\mathcal{H}}) \\ & - \lambda_2 \sum_{v_j \in V_t} (1 - c'_j) \prod_{v_k \in D_j(C')} m_{k,j}(C'_{\mathcal{H}}, C_{-\mathcal{H}}). \end{aligned}$$

By the partition of V and Property 2, for any $\ell \in \{1, \dots, t\}$ and any $v_j \in N_{i_\ell}^3$, the status of vertex $v_{i_{\ell'}}$ with $\ell' \neq \ell$ will not affect the value of $m_{k,j}(C)$ for any $v_k \in D_j(C)$, and thus $m_{k,j}(C'_{\mathcal{H}}, C_{-\mathcal{H}}) = m_{k,j}(c'_i, C_{-i})$ and $D_j(C') = D_j(c'_i, C_{-i})$. Similar argument shows that for any $v_j \in V_t$, we have $D_j(C') = D_j(C)$ and $m_{k,j}(C') = m_{k,j}(C)$ for any $v_k \in D_j(C')$. It follows that:

$$\begin{aligned} \pi(C') - \pi(C) &= -\lambda_1 \sum_{v_i \in \mathcal{H}} c'_i + \lambda_1 \sum_{v_i \in \mathcal{H}} c_i \\ & - \lambda_2 \sum_{v_i \in \mathcal{H}} \sum_{v_j \in N_i^3} (1 - c'_j) \prod_{v_k \in D_j(c'_i, C_{-i})} m_{k,j}(c'_i, C_{-i}) \\ & + \lambda_2 \sum_{v_i \in \mathcal{H}} \sum_{v_j \in N_i^3} (1 - c_j) \prod_{v_k \in D_j(C)} m_{k,j}(C) \\ &= \sum_{v_i \in \mathcal{H}} (u_i(c'_i, C_{-i}) - u_i(c_i, C_{-i})). \end{aligned}$$

Lemma 4: Starting from any initial strategy profile $C^{(0)}$, Algorithm 1 converges in at most $\frac{(\lambda_1 + \lambda_2)n}{\min\{\lambda_2 - \lambda_1, \lambda_1\}}$ rounds, which is $O(n)$.

Proof: Based on Lemma 3 and the assumption that a player is willing to change his strategy only when he can be strictly better off, which is similar to the proof of Theorem 2, in every round of Algorithm 1, the potential π is reduced by at least $\min\{\lambda_2 - \lambda_1, \lambda_1\}$. Then, the lemma follows.

The following theorem shows the quality of the solution.

Theorem 4: The output C' of algorithm BRDDLA is a minimal secure dominating set and a Pareto-optimal solution.

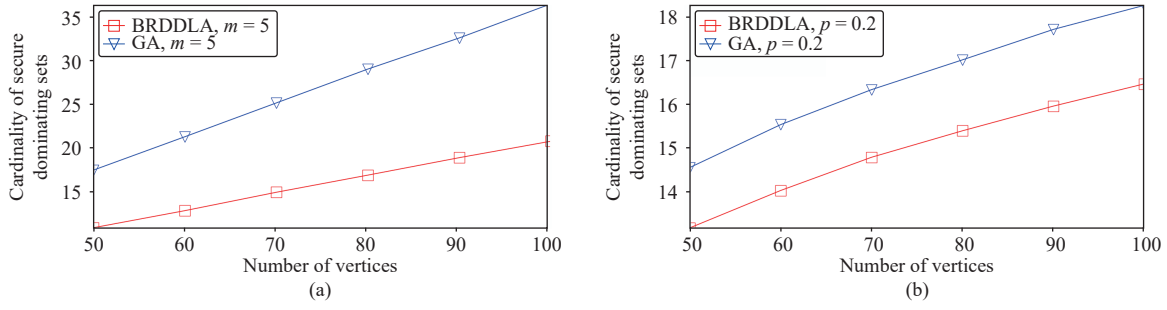
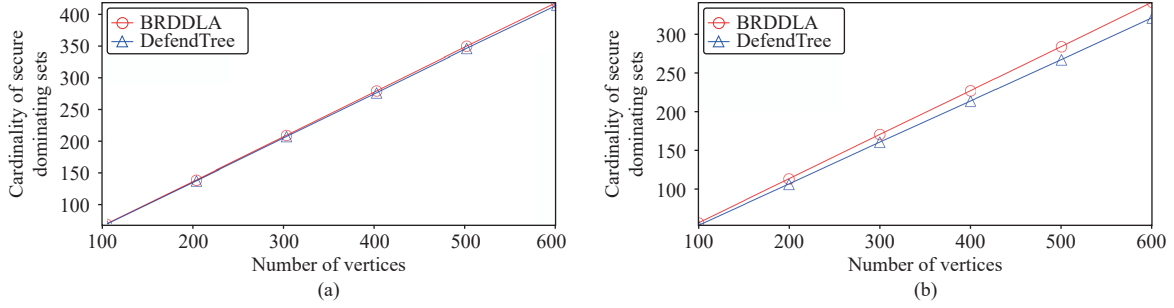
Fig. 4. Comparison between BRDDLA and GA. (a) BA, $m_0 = 5$; (b) ER, $p = 0.2$.

Fig. 5. Comparison of BRDDLA and DefendTree on trees. (a) BAT; (b) RT.

Proof: We claim that when $C = C^{(t-1)}$, C must be an NE. In fact, C stops being updated only when no player can be strictly better off. Thus, the reason why $C = C^{(t-1)}$ is because $C^{(t-1)}$ consists of best responses for every player, and thus is an NE. Then, the theorem follows from Theorems 1 and 3. ■

VI. SIMULATION RESULTS

In this section, we experiment on the performance of our algorithm BRDDLA. All experiments are coded in Python and run with an identical configuration: AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx and 16 GB of RAM.

A. Comparison With a Greedy Algorithm

Although there are some polynomial-time algorithms for MinSDS on special classes of graphs [4], [8]–[15], we have not found any algorithm on MinSDS designed for a general graph. Hence, we compare our algorithm with a natural greedy algorithm (GA) described in Algorithm 2, where $IS(C)$ denotes the set of insecure vertices in V with respect to current solution C . The algorithm iteratively selects a vertex the addition of which reduces the number of insecure vertices most. By Lemma 1, the whole vertex set V is an SDS, and thus GA terminates in at most n rounds.

Algorithm 2 Greedy Algorithm (GA)

Input: graph G
Output: an SDS C
1: **while** $IS(C) \neq \emptyset$ **do**
2: $v_i \leftarrow \arg \max_{v_i \in V \setminus C} \{|IS(C)| - |IS(C \cup \{v_i\})|\}$
3: $C \leftarrow C \cup \{v_i\}$
4: **end while**

Graphs for the experiments are generated randomly using the following two models.

1) *The Barabási-Albert Graph (BA)* [34]: Starting from a

graph with a small number m_0 of vertices, new vertices are iteratively added. When a new vertex is added, it is connected to m existing vertices, where $m \leq m_0$, and the probability that an existing vertex is linked with the new vertex depends on its current degree.

2) *The Erdős-Rényi Graph (ER)* [35]: In this graph, every edge is presented with probability p independent of the other edges.

In Fig. 4, the horizontal axis is the number of vertices n . For each n , 1000 sampled graphs are generated. The vertical axis is the average size of computed solutions. It turns out that BRDDLA is superior to GA, especially in BA, where the average sizes of the solutions computed by BRDDLA are roughly 59% of those computed by GA. For clarity of figures, we only show the case when $m = m_0 = 5$ for the BA, and $p = 0.2$ for the ER. In fact, when we tested on other m , m_0 and p , all experiments support the same results.

B. Comparison With an Exact Solution on Trees

To see the accuracy of our algorithm, we compare BRDDLA with the exact algorithm DefendTree proposed by Burger *et al.* [4] for MinSDS on trees. Trees for the experiments are constructed in the following two manners.

1) *Barabási-Albert Tree (BAT)*: A tree is constructed by the Barabási-Albert model with $m_0 = 2$ and $m = 1$.

2) *Random Tree (RT)*: Let V be a vertex set on n vertices and F be the edge set consisting all possible edges between vertices of V . Starting from an empty graph on vertex set V , iteratively add an edge e from F randomly and uniformly as long as no cycle is created, until we obtain a spanning tree on V .

Again, for each n , 1000 trees of size n are sampled. Fig. 5 shows the average sizes of solutions computed by BRDDLA (red line) and the average sizes of optimal solutions computed by DefendTree (blue line). It can be seen that these two lines

are very close. We use $\gamma = \frac{|C'|-|C^*|}{|C^*|}$ to measure relative error of BRDDLA, where C' is the output of BRDDLA and C^* is the optimal solution computed by DefendTree. It can be seen from Table II that γ is around 1% for BAT and 6% for RT.

TABLE II
RELATIVE ERROR OF BRDDLA MEASURED BY γ

Graph	$ V = 100$	200	300	400	500	600
BAT	0.95%	1.03%	0.98%	1.04%	0.99%	1.11%
RT	6.05%	6.27%	6.14%	6.22%	6.31%	6.25%

C. Effect of Decision Priority

Theoretical analysis guarantees that BRDDLA could always find a minimal SDS, but different minimal solutions may have different sizes. According to decision rule 7, a player with a smaller ID has a higher priority to make a change. So, a question is: will different ID of players lead to solutions with much difference?

To test the effect of decision priority, for a randomly generated graph, we randomly and uniformly generate 1000 permutations of the players which are used as players' ID used by BRDDLA. Denote by \bar{C} and \underline{C} the solution with the maximum size and the solution with the minimum size among these 1000 solutions, respectively. Still use C' to denote the solution computed using ID $(1, 2, \dots, n)$. Then parameter $\omega = \frac{|\bar{C}|-|\underline{C}|}{|\underline{C}|}$ can be used to measure the relative gap between the worst solution and the best solution influenced by different ID, and parameter $\eta = \frac{|\underline{C}|-|C'|}{|C'|}$ can be used to measure the improvement on the natural ID $(1, 2, \dots, n)$.

Tables III and IV show the values of ω and η in various types of experimental graphs.

From Table III, it can be seen that for a general graph, there is a big difference between the best solution and worst solution, and the difference is relatively smaller for ER than for BA. This indicates that sizes of different minimal solutions might vary greatly, and the variance is smaller in an evenly distributed graph than in a power-law graph. While for trees, the difference is much smaller, and smaller for BA than for ER. This might be because our algorithm is already fairly accurate for trees, especially for power law graphs. These experiments certify that different IDs of players do make a difference in the quality of the solution obtained by BRDDLA, which also suggests that in order to obtain a better solution, one may repeat the algorithm several times based on different ID arrangements.

What is most interesting is to observe from Table IV that the improvement brought about by decision priority is significantly smaller for BA than for ER. The reason might be explained as follows. For BA, $(1, 2, \dots, n)$ is the order of vertices added in the construction of the graph. A vertex which is added earlier has a larger chance to be an important individual, and thus letting such the vertex to make a decision earlier may have a better chance to create a good result. While in ER, importance of vertices is distributed evenly, and thus trying different players' IDs might result in more diversified solutions. In summary, changing players' ID might improve the

performance of the algorithm for evenly distributed random graphs, while for BA, letting players' IDs be equivalent to construction order might be good enough.

D. Effect of Initial Solution

In the above experiments, the initial strategy profile is always taken to be $C^{(0)} = (0, \dots, 0)$. Although Lemma 2 guarantees that BRDDLA could converge to an NE starting from any initial strategy profile, we would like to know the influence of initial strategy profile on the final output. For this purpose, we use a simple method: restart.

The results are shown in Fig. 6. For each type of graphs, 10 sample-graphs are generated. For each graph, 100 initial strategy profiles are taken uniformly and randomly, and the smallest size of these 100 solutions is recorded as the "result of restart". In Fig. 6, the green lines mark the results of restart, and the red lines mark the sizes of those solutions starting from strategy profile $(0, \dots, 0)$. It turns out that the effect of restart is obvious on ER and RT, but not so obvious on BA and BAT. The reason might be similar as before: our algorithm is already good enough for the BA model, which leaves only a little space for improvement.

E. Convergence Rate

In this subsection, we test on the number of rounds needed by DefendTree, GA and BRDDLA on BA, ER, BAT and RT. The number of vertices is 1000. The results are shown in Table V. As Lemma 4 shows, BRDDLA converges in $O(n)$ rounds, where n is the number of vertices. This is a theoretical result. It turns out that in a practical setting, BRDDLA converges in much less rounds than n . Furthermore, BRDDLA converges faster than the reference algorithms, especially on BAT and RT. The reason might be that both BAT and RT are sparse, and thus more players can change their strategies simultaneously.

VII. CONCLUSION

In this paper, we use the game theoretic method to solve the MinSDS problem in a multi-agent system. An SDS game was proposed and proven to be an exact potential game, converging to an NE in linear rounds of interactions. Moreover, we proved that every NE of the game is a minimal secure dominating set and a Pareto optimal solution. Then, a distributed algorithm was designed to simulate the game process. In each round of the algorithm, every player only uses local information at most three hops away to make his decision. The performance of our algorithm was tested through experiments on randomly generated graphs using both ER and BA. We compare its performance with a natural greedy heuristic, and it turns out that our algorithm out-performs the greedy strategy by a large amount. Compared with an existing exact algorithm for MinSDS on trees, our algorithm turns out to be fairly accurate. Furthermore, our algorithm is not only better in accuracy, but also faster in convergence.

There might be a lot of interesting topics to be further explored, such as price of anarchy (PoA) [36], allocation games [37], evolutionary games [38] and networked games [39], [40].

TABLE III
INFLUENCE OF DECISION PRIORITY MEASURED BY ω

Graph	$ V = 50$	60	70	80	90	100
BA	63.64%	100.00%	86.67%	100.00%	75.00%	71.43%
ER	42.86%	38.46%	41.18%	50.00%	50.00%	38.89%

Graph	$ V = 100$	200	300	400	500	600
BAT	1.39%	1.46%	1.42%	0.72%	1.72%	1.24%
RT	10.17%	7.02%	7.60%	7.14%	5.21%	5.78%

TABLE IV
IMPROVEMENT BY DECISION PRIORITY MEASURED BY η

Graph	$ V = 50$	60	70	80	90	100
BA	-3.06%	-3.67%	-3.24%	-3.33%	-3.59%	-3.41%
ER	-28.57%	-15.38%	-23.53%	-14.29%	-25.00%	-22.22%

Graph	$ V = 100$	200	300	400	500	600
BAT	-1.21%	-0.73%	-0.47%	-0.36%	-1.44%	-0.99%
RT	-6.78%	-2.63%	-3.51%	-4.02%	-2.78%	-3.18%

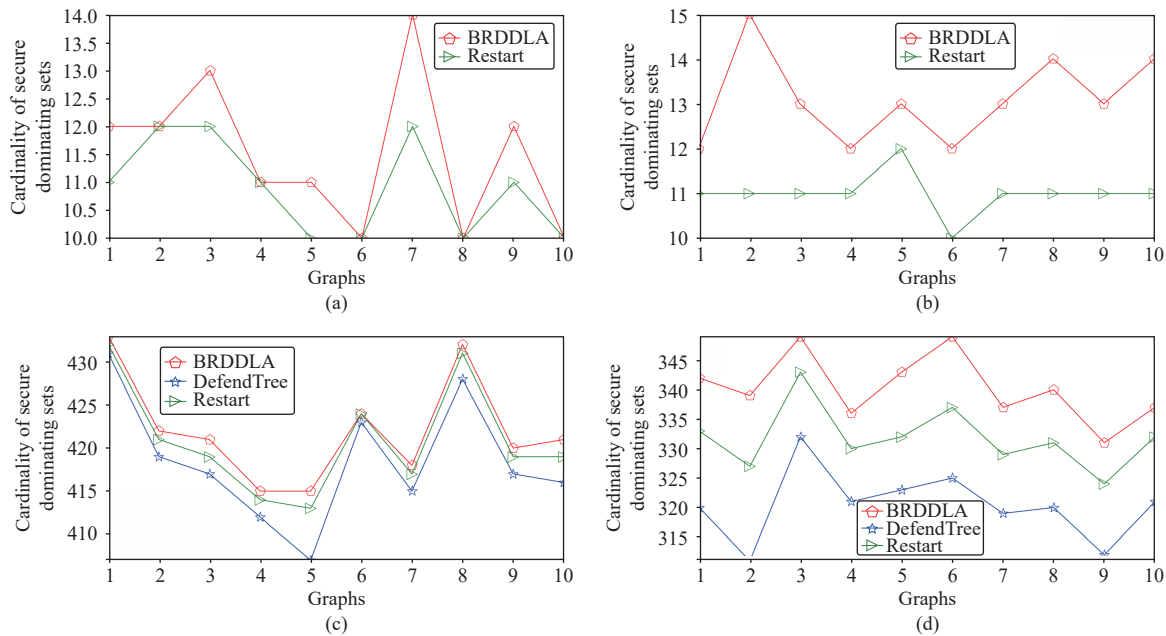


Fig. 6. Effect of Initial Solution by Restart. (a) BA, $m_0 = m = 5$, $|V| = 50$; (b) ER, $p = 0.2$, $|V| = 50$; (c) BAT, $|V| = 600$; (d) RT, $|V| = 600$.

TABLE V
NUMBER OF ROUNDS OF THREE ALGORITHMS

Algorithm	BA, $m_0 = 5$	ER, $p = 0.2$	BAT	RT
DefendTree	None	None	1000	1000
GA	377	43	711	601
BRDDLA	360	41	568	210

REFERENCES

- [1] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Fundamentals of Domination in Graphs*. Boca Raton, USA: CRC Press, Jan. 1998.
- [2] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Domination in Graphs*. New York, USA: Routledge, Oct. 1998, vol. 2.
- [3] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning, *Topics in Domination in Graphs*. Geneva, Switzerland: Springer, Cham, Jan. 2020, vol. 64.
- [4] A. Burger, A. de Villiers, and J. van Vuuren, "A linear algorithm for secure domination in trees," *Discrete Applied Math.*, vol. 171, pp. 15–27, 2014.
- [5] E. Cockayne, P. Grobler, W. Gründlingh, J. Munganga, and J. Vuuren, "Protection of a graph," *Utilitas Mathematica*, vol. 67, pp. 19–32, 2005.
- [6] A. Burger, A. de Villiers, and J. Vuuren, "Two algorithms for secure graph domination," *J. Combinatorial Math. and Combinatorial Computing*, vol. 85, pp. 321–339, 2013.
- [7] H. Boumediene Merouane and M. Chellali, "On secure domination in graphs," *Infor. Proc. Lett.*, vol. 115, no. 10, pp. 786–790, 2015.
- [8] T. Araki and H. Miyazaki, "Secure domination in proper interval graphs," *Discrete Applied Math.*, vol. 247, pp. 70–76, 2018.
- [9] T. Araki and R. Yamanaka, "Secure domination in cographs," *Discrete Applied Math.*, vol. 262, pp. 179–184, 2019.
- [10] T. Araki and I. Yumoto, "On the secure domination numbers of maximal outerplanar graphs," *Discrete Applied Math.*, vol. 236, pp. 23–29, 2018.
- [11] A. P. Burger, M. A. Henning, and J. H. van Vuuren, "Vertex covers and secure domination in graphs," *Quaestiones Math.*, vol. 31, no. 2, pp. 163–171, 2008.
- [12] A. Burger, A. de Villiers, and J. van Vuuren, "On minimum secure dominating sets of graphs," *Quaestiones Math.*, vol. 39, no. 2, pp. 189–202, 2016.
- [13] E. Castellano, R. Ugbinada, and S. Canoy Jr, "Secure domination in the joins of graphs," *Applied Math. Sciences*, vol. 8, pp. 5203–5211, 2014.
- [14] D. Corneil, H. Lerchs, and L. Burlingham, "Complement reducible graphs," *Discrete Applied Math.*, vol. 3, no. 3, pp. 163–174, 1981.
- [15] D. Pradhan and A. Jha, "On computing a minimum secure dominating set in block graphs," *J. Combinatorial Optimization*, vol. 35, pp. 613–631, 2018.
- [16] Z. Han, D. Niyato, W. Saad, T. Başar, and A. Hjørungnes, *Game Theory in WIREless and Communication Networks: Theory, Models, and*

Applications. Cambridge, UK: Cambridge University Press, Jan. 2011.

- [17] D.-Z. Du and P.-J. Wan, *Connected Dominating Set: Theory and Applications*. New York, USA: Springer 2013, vol. 77.
- [18] L.-H. Yen and Z.-L. Chen, "Game-theoretic approach to self-stabilizing distributed formation of minimal multi-dominating sets," *IEEE Trans. Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3201–3210, 2014.
- [19] L.-H. Yen and G.-H. Sun, "Game-theoretic approach to self-stabilizing minimal independent dominating sets," in *Proc. Internet and Distributed Computing Syst.*, 2018, pp. 173–184.
- [20] X. Chen and Z. Zhang, "A game theoretic approach for minimal connected dominating set," *Theoretical Computer Science*, vol. 836, pp. 29–36, 2020.
- [21] Y. Yang and X. Li, "Towards a snowdrift game optimization to vertex cover of networks," *IEEE Trans. Cyber.*, vol. 43, no. 3, pp. 948–956, 2013.
- [22] C. Tang, A. Li, and X. Li, "Asymmetric game: A silver bullet to weighted vertex cover of networks," *IEEE Trans. Cyber.*, vol. 48, no. 10, pp. 2994–3005, 2018.
- [23] C. Sun, W. Sun, X. Wang, and Q. Zhou, "Potential game theoretic learning for the minimal weighted vertex cover in distributed networking systems," *IEEE Trans. Cyber.*, vol. 49, no. 5, pp. 1968–1978, 2019.
- [24] C. Sun, X. Wang, H. Qiu, and Q. Chen, "A game theoretic solver for the minimum weighted vertex cover," in *Proc. IEEE Int. Conf. Syst., Man and Cyber.*, 2019, pp. 1920–1925.
- [25] J. Chen, K. Luo, C. Tang, Z. Zhang, and X. Li, "Optimizing polynomial-time solutions to a network weighted vertex cover game," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 2, pp. 512–523, 2023.
- [26] X.-Y. Li, Z. Sun, W. Wang, X. Chu, S. Tang, and P. Xu, "Mechanism design for set cover games with selfish element agents," *Theoretical Computer Science*, vol. 411, no. 1, pp. 174–187, 2010.
- [27] X.-Y. Li, Z. Sun, W. Wang, and W. Lou, "Cost sharing and strategyproof mechanisms for set cover games," *J. Comb. Optim.*, vol. 20, pp. 259–284, 2010.
- [28] Q. Fang and L. Kong, "Core stability of vertex cover games," in *Internet and Network Economics*. Berlin Heidelberg, Germany: Springer, 2007, pp. 482–490.
- [29] B. van Velzen, "Dominating set games," *Operations Research Letters*, vol. 32, no. 6, pp. 565–573, 2004.
- [30] H. K. Kim, "On connected dominating set games," *J. Korean Data and Information Science Society*, vol. 22, pp. 1275–1281, 2011.
- [31] X. Ai, V. Srinivasan, and C. Tham, "Optimality and complexity of pure nash equilibria in the coverage game," *IEEE J. Selected Areas in Commun.*, vol. 26, no. 7, pp. 1170–1182, 2008.
- [32] J. F. Nash, "Equilibrium points in n -person games," *Proc. National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950.
- [33] D. Monderer and L. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, pp. 124–143, 1996.
- [34] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.
- [35] P. Erdős and A. Rényi, "On random graphs I," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.
- [36] T. Roughgarden, "The price of anarchy is independent of the network topology," *J. Computer and System Sciences*, vol. 67, no. 2, pp. 341–364, 2003.
- [37] Y. Li and A. S. Morse, "The power allocation game on a network: A paradox," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 4, pp. 771–776, 2018.
- [38] G. Zhao, Y. Wang, and H. Li, "A matrix approach to the modeling and analysis of networked evolutionary games with time delays," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 4, pp. 818–826, 2018.
- [39] M. Ye, D. Li, Q.-L. Han, and L. Ding, "Distributed Nash equilibrium seeking for general networked games with bounded disturbances," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 2, pp. 376–387, 2023.
- [40] D. Cheng, T. Xu, F. He, and H. Qi, "On dynamics and Nash equilibriums of networked games," *IEEE/CAA J. Autom. Sinica*, vol. 1, no. 1, pp. 10–18, 2014.



Xiuyang Chen received the B.S. degree in mathematics and applied mathematics from Zhejiang International Studies University in 2018, the M.S. degree in mathematics and applied mathematics from Zhejiang Normal University in 2021. He is currently a Ph.D. candidate in mathematics and applied mathematics at Xinjiang University. His current research interests include game theory, mechanism design, intelligence algorithm, and approximation algorithm.



Changbing Tang (Senior Member, IEEE) received the B.S. and M.S. degrees in mathematics and applied mathematics from Zhejiang Normal University, in 2004 and 2007, respectively, the Ph.D. degree in circuits and systems from the Department of Electronic Engineering, Fudan University in 2014. He is currently an Associate Professor with the College of Physics and Electronic Information Engineering, Zhejiang Normal University. His research interests include game theory, blockchain and its applications, networks and distributed optimization.

He is the recipient of the Academic New Artist Doctoral Post Graduate from the Ministry of Education of China in 2012 and the recipient of the Academician Pairing Training Program for Young Talents of Zhejiang Province in 2019.



Zhao Zhang received the B.S. degree in computational mathematics, the M.S. degree in basic mathematics and the Ph.D. degree in applied mathematics from Xinjiang University, in 1996, 1999 and 2003, respectively. From 2003 to 2014, She was engaged in scientific research in Xinjiang University. She is currently a Distinguished Professor of Zhejiang Normal University. Her current research interests include approximation algorithm, combinatorial optimization and machine learning.