Designing Self-Stabilizing Systems Using Game Theory

LI-HSING YEN, National Chiao Tung University
JEAN-YAO HUANG, National University of Kaohsiung
VOLKER TURAU, Hamburg University of Technology

Self-stabilizing systems tolerate transient faults by always returning to a legitimate system state within a finite time. This goal is challenged by several system features such as arbitrary system states after faults, various process execution models, and constrained process communication means. This work designs self-stabilizing distributed algorithms from the perspective of game theory, achieving an intended system goal through private goals of processes. We propose a generic game design for identifying a maximal independent set (MIS) or a maximal weighted independent set (MWIS) among all processes in a distributed system. From the generic game several specific games can be defined which differ in whether and how neighboring players influence each other. Turning the game designs into self-stabilizing algorithms, we obtain the first algorithms for the MWIS problem and also the first MIS self-stabilizing algorithm that considers node degree (including an analysis of its performance ratio). We also show how to handle simultaneous moves of processes in some process execution models. Simulation results indicate that, for various representative network topologies, the new algorithm outperforms existing methods in terms of MIS size and convergence rate. For the MWIS problem, the new algorithms performed only slightly worse than centralized greedy counterparts.

CCS Concepts:•**Theory of computation** → **Distributed algorithms; Algorithmic mechanism design; Solution concepts in game theory;** *Self-organization;* Distributed computing models;•**Mathematics of computing** → *Graph theory;*

General Terms: Design, Algorithms, Theory

Additional Key Words and Phrases: Distributed algorithms, game theory, independent set, self-stabilization

ACM Reference Format:
Yen, L., Huang, J., and Turau, V. 2015. Designing Self-Stabilizing Systems Using Game Theory. *ACM Trans. Autonom. Adapt. Syst.* V, N, Article A (January YYYY), 27 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

This work considers a distributed system consisting of independent processes that communicate with each other via shared variables. For this purpose each process defines variables that are owned by and exclusively updated by the process. Other processes can only read the value of this variable. A snapshot capturing all variable's values of all processes comprises the system's state. A system state may be legitimate or illegitimate with respect to a predicate that specifies a desired state of the system. When a system is in a legitimate state, an occurrence of a fault may bring the system into an illegitimate state. A self-stabilizing distributed algorithm tolerates transient

This work is supported by the Ministry of Science and Technology, Taiwan, under grant MOST 103-2221-E-390-002.
Author's addresses: L.-H. Yen; Department of Computer Science, National Chiao Tung University, Taiwan; J.-Y. Huang, Department of Computer Science and Information Engineering, National University of Kaohsiung, Taiwan; V. Turau, Institute of Telematics, Hamburg University of Technology, Germany.

faults (faults that do not persist) by always reaching legitimate system states in finite time regardless of initial system configuration [Dijkstra 1974].

This paper considers designing self-stabilizing algorithms from the perspective of game theory. Representing a game using a distributed system is not new [Halpern 2003]. Apparently, processes that have their own behaviors are analogous to autonomous agents or players in a game. Variables of processes correspond to strategies of agents, and processes update their variables as agents change their strategies. However, agents in a game are rational yet selfish, which typically seek their own interests rather than a common good. In contrast, processes in a traditional self-stabilizing system have no local, private objectives. They are usually coded to achieve a global system goal. The challenge is to design the local processes such that they achieve a global goal using only their local information, i.e., despite of lacking the knowledge of the overall system state.

Some studies [Cohen et al. 2008; Gouda and Acharya 2011] argue that individual goals are negative and may be an obstacle to reach a global system goal. In contrast, we aim at an algorithmic mechanism design [Nisan and Ronen 2001] that achieves an intended system goal through private goals of processes. To this end, we provide incentives for autonomous players whose rational yet selfish interactions with one another dynamically lead the game to a stable state where the system goal is satisfied.[1]

Many algorithmic mechanisms or other game-theoretical designs have been proposed for optimization problems in distributed systems [Nisan and Ronen 2001; Grosu and Chronopoulos 2004]. However, little work has been done towards the design of self-stabilizing distributed algorithms. Particularly, the following issues are unique to self-stabilizing systems and should be addressed by the corresponding game mechanism design.

— The system state is arbitrary after a fault, meaning that the initial state of the corresponding game is also arbitrary. In contrast, researchers usually deal with games with a known and fixed initial configuration.
— To show the correctness and analyze the convergence rate of a self-stabilizing system, we need to assume a process execution model that controls the level of execution concurrency among processes. Some existing models allow only one process to execute at a time, while others allow a non-empty set of processes to execute in parallel. In the former model, process execution sequences are non-deterministic, so the corresponding game should permit non-deterministic game play sequences. This is not a common game design constraint because in some games players can pre-compute their dominant or best-response strategies. For the latter model, we may look into *simultaneous games* which allow simultaneous moves made by *all* players. However, a game which allows simultaneous moves made by a *dynamic subset* of players is not commonly seen.
— Processes are typically assumed to communicate with each other via shared variables, disallowing accesses of non-neighbor's variables. This limitation implies that players of the game only have information of neighboring players.

Yen and Chen [2014] recently proposed an algorithmic mechanism[2] for the formation of a minimal multi-dominating set [Jia et al. 2002]. The authors converted the game design into a distributed algorithm. To the best of our knowledge, this is the first game-theoretic approach to self-stabilizing algorithms. Unfortunately, their work considered

[1]Algorithmic mechanism designs usually deal with resource sharing or load distribution problems. For that reason the designs should also consider payments to participating agents, which are not really needed in the design of self-stabilizing algorithms.
[2]disregarding the payment issue

only the model of one process executing at time. The possibility of several players in game moving simultaneously and the corresponding distributed algorithms were not considered. Furthermore, their work does not fully conform to the requirement of self-stabilization: their algorithm achieves stability probabilistically.

We are concerned with a series of problems related to independent sets. In an undirected graph $G = (V, E)$, where V is the vertex set and E is the edge set, $S \subseteq V$ is an independent set if no vertices in S are adjacent to one another. An independent set that has the maximum cardinality is called maximum independent set. If each vertex is associated with a weight (a positive real number), S is a maximum weighted independent set if it is an independent set that has the maximum total weight among all such sets. Clearly, a maximum independent set is a maximum weighted independent set with uniform weight. Finding either set is known to be NP-hard [Tarjan and Trojanowski 1977; Garey and Johnson 1979], for which polynomial-time approximation is NP-hard as well. There exist many heuristics for these two problems. One well-known greedy approach works by selecting a vertex into the set, removing it and adjacent vertices from the graph, and repeating this process on the remaining graph. The vertex selection rule may prefer vertices with minimum node degrees [Halldórsson and Radhakrishnan 1997] (for maximum independent set) or, more generally, depend on some vertex ordering that is a function of weight and node degree [Sakai et al. 2003]. The result found by the greedy approach can only be a *maximal independent set* (MIS), an independent set of which no proper superset is also an independent set. If nodes are associated with weights, the result is a maximal weighted independent set (MWIS).

We propose a game-theoretic framework for identifying an MIS/MWIS in a distributed system. The framework provides a decentralized, autonomous, and self-stabilizing approach to the MIS/MWIS problem and is general enough to incorporate various vertex ordering functions concerning weight and/or node degree. To the best knowledge of the authors, the proposed framework is the first game-theoretic approach to MIS/MWIS formation. With the proposed framework, we also present the first self-stabilizing algorithms to the MWIS problem and the first self-stabilizing MIS algorithm that considers node degree. Moreover, the latter algorithm is the first self-stabilizing MIS algorithm with a guaranteed performance ratio.

Furthermore, we discuss the time complexity of the algorithms and report about simulations to study the average-case performance. The simulations covered a variety of network topologies (including unit-disk graphs [Clark et al. 1990], random graphs [Erdös and Rényi 1959], small-world networks [Watts and Strogatz 1998], and scale-free networks [Barabási and Albert 1999]). Simulation results show that, in terms of average total weight in the final set, distributed algorithms derived from our game framework performed slightly worse than corresponding centralized greedy algorithms. But compared with existing self-stabilizing MIS algorithms, our work provides a significant improvement regarding the cardinality of the MIS and also the rate of convergence. The positive results about the cardinality of the generated MIS are not surprising because of the bounded performance ratio of the algorithm.

The remainder of this paper is organized as follows: Background information and related work are presented in Section 2. In the following section, we first model the MIS/MWIS problem as a sequential game and consider two cases where players have and do not have mutual influence, respectively. We then show how to convert the game into a self-stabilizing algorithm. Section 4 discusses how to deal with simultaneous moves and analyzes the performance ratio of the proposed algorithm. Section 5 studies the performance of the proposed approach through simulations. The simulation results are compared with those of existing solutions. The last section concludes this paper.

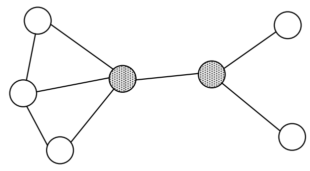


Fig. 1. Shaded nodes comprising a dominating set but not an independent set.

2. BACKGROUND AND RELATED WORK

We represent a binary relation on the set of all processes by an undirected graph, where nodes represent processes and edges represent the existence of the relation between two relevant processes.[3] Independent sets in a graph are closely related to *dominating sets*. A dominating set is a subset of nodes in a graph such that every node not in the set is adjacent to at least one node in the set. A dominating set is minimal (called a minimal dominating set) if it contains no proper subset that is also a dominating set. There are several self-stabilizing distributed algorithms proposed for identifying a minimal dominating set [Hedetniemi et al. 2003; Xu et al. 2003; Kamei and Kakugawa 2003; 2005; Kakugawa and Masuzawa 2006; Turau 2007; Goddard et al. 2008; Yen and Chen 2014]. An MIS is also a minimal dominating set. It is a dominating set because every node not in the set is adjacent to some node in the set.[4] The dominating set is *minimal* because the set will no longer be a dominating set if we remove any node from it: a node removed from the set will not be adjacent to any node remaining in the set. Therefore, an MIS approach also serves as an approach to the minimal dominating set. However, a minimal dominating set may not be an independent set. Refer to Fig. 1 for an example. Therefore, any approach to the minimal dominating set problem cannot be applied here.

Independent sets also relate to other graph problems. A *vertex cover* is a set of nodes in a given graph such that every edge of the graph is incident to at least one node in the set. Because there is no edge between any two nodes in an independent set, every edge of the graph is incident to at least one node in the complement of the independent set. This property makes the complement of an independent set a vertex cover. Accordingly, the complement of an MIS is a *minimal* vertex cover, a vertex cover that contains no proper subset that is also a vertex cover. Therefore, any MIS/MWIS approach can also be used as an approximation to the (weighted) minimum vertex cover problem.

Given a finite base set S and a collection C of subsets of S, a *set packing* is a subcollection of C such that all elements in the subcollection are mutually disjoint. The maximum set packing problem is to find a set packing that is of the maximum size. There is a polynomial-time reduction that transforms the set packing problem to the independent set problem. It works by creating a graph where one node is for each member in C and there is an edge between two nodes iff the corresponding two subsets share a common element. For this reason, any MIS/MWIS approach serves as an approximation to the (weighted) maximum set packing problem.

Existing greedy approaches to identifying an MIS/MWIS are mostly centralized. Some parallel algorithms [Karp and Wigderson 1985; Alon et al. 1986] have been proposed to speed up the computation time. In these algorithms, each node probabilistically decides whether to join the independent set. Let G denote a graph and I

[3]We use *nodes*, *vertices* and *processes* interchangeably in this paper
[4]If there were any exception, adding that node into the set would yield a larger independent set and thus the original set would not be an MIS.

the MIS/MWIS to be constructed. A generic approach to the construction of I from G works in the following way.

```
$I := \emptyset$;
**while** $G \neq \emptyset$ **do**
    select $v$ from $G$;
    $I := I \cup \{v\}$;
    $G := G \setminus \{v\} \cup N(v)$;
**end**
```

Here $N(v)$ denote the set of v's neighbors. Different approaches differ in the selection of v from G. Halldórsson and Radhakrishnan [1997] proposed an approach that selects the node with the minimum node degree among all when forming an MIS. Sakai et al. [2003] considered several selection rules when forming an MWIS. One of them termed GWMIN prioritizes node v that maximizes $W(v)/(\deg(v)+1)$, where $W(v)$ is the weight associated with v. Another rule termed GWMIN2 selects v that maximizes

$$\frac{W(v)}{W(v)+\sum_{u\in N(v)} W(u)}. \tag{1}$$

Basagni [2001] proposed a distributed algorithm for identifying an MWIS in wireless networks. In this algorithm, a node broadcasts messages to notify its neighbors of its weight and its decision to join the set. A node will join the set only if no neighbor of higher weight joins the set.

None of the abovementioned approaches possesses the property of self-stabilization. A distributed algorithm is self-stabilizing if it meets the *convergence* and *closure* condition [Dijkstra 1974]. The former requires that starting from arbitrary state (possibly illegitimate) the algorithm eventually reaches a legitimate state. The latter property states that any state following a legitimate state is also legitimate. For the MIS problem, a state is legitimate if all nodes that decided to be in the set indeed constitute an MIS.

Self-stabilizing algorithms typically assume that processes communicate with one another via shared variables. In this model, each process owns local variables to which it can exclusively write values. Local variables can only be read by the owner and the owner's neighboring processes. A distributed algorithm is anonymous if it does not assume the availability of unique process identifiers. The program executed by an individual process is often expressed in the form of *guarded commands* [Dijkstra 1975], a.k.a. *rules*. Each rule consists of a guard (Boolean expression) and an action (assignment statements). A rule R is *enabled* if the guard of R evaluates to true. Only at that time the action of R can be executed. The execution of an action is called a *move*. A process with at least one rule enabled is called *privileged*. The whole system enters a quiescent state (called a fixed point in [Gouda and Acharya 2011]) if no process is privileged.

The level of execution concurrency among processes is specified by three possible execution models a.k.a. *daemons*, namely, central, synchronous, and distributed daemon. The central daemon allows only one privileged process to make a move at a time. With synchronous daemon, all privileged processes make moves simultaneously. These two daemons are subsumed by the distributed daemon, which allows a non-empty subset of privileged processes to execute in parallel. In particular that means that a privileged node can wait for an arbitrary time before it makes its move. The purpose of this daemon is to model arbitrary communication delays in a network. Thus, if an algorithm

Table I. Existing Self-Stabilizing MIS Algorithms

Work	Objective	Control daemon	Topology	Anonymous?
Shukla et al. [1995]	MIS	Central	General	Yes
Ikeda et al. [2002]	MIS	Distributed	General	No
Turau [2007]	MIS	Distributed	General	No
Goddard et al. [2003]	MIS	Synchronous	General	No
Shi et al. [2004]	1-MIS	Central	Tree	Yes
This work	MWIS/MIS	Central	General	Yes
	MWIS/MIS	Distributed/Synchronous	General	No

stabilizes under the distributed daemon it will also stabilize under the central and the synchronous daemon.

A related concept is called *weak stabilization* [Gouda 2001]. Whereas for a self-stabilizing algorithm each sequence of legal moves leads from any arbitrary state to a legitimate state, weak stabilization only guarantees that for each arbitrary state there exists *at least one* sequence of moves with this property. Thus, any self-stabilizing algorithm is also weakly stabilizing but not vice versa. Gouda [2001] shows that weak stabilization of a system implies stabilization of the same system under some reasonable conditions.

Shukla et al. [1995] proposed an anonymous self-stabilizing algorithm that finds an MIS under a central daemon. This algorithm consists of two simple rules. First, a node joins the MIS if none of its neighbors is in the MIS. Second, a node leaves the MIS if one of its neighbors is also in the MIS. The same algorithm was also proposed by other researchers [Hedetniemi et al. 2003].

These two rules are general, but the proposed algorithm was designed to run under a central daemon. It does not work with the distributed or synchronous daemon. Shukla et al. [1995] proved that there is no anonymous self-stabilizing algorithm that finds an MIS under a distributed daemon. Ikeda et al. [2002] modified the second rule using unique node identifiers. More specifically, a node should leave the MIS iff any of its neighbors with a smaller identifier is also in the MIS. This algorithm works under a distributed daemon.

Turau [2007] proposed a self-stabilizing algorithm that has a lower time complexity than the one proposed by Ikeda et al. This algorithm runs under a distributed daemon. Besides an identifier, each node maintains a variable that indicates the current status of the node. While values IN and OUT indicate the node's membership in the MIS, value WAIT indicates that the node wants to join the MIS under construction.

Goddard et al. [2003] proposed a self-stabilizing algorithm that identifies an MIS under the synchronous daemon. Each node in this algorithm executes the following two rules. First, a node joins the MIS if it is not currently in the MIS and there is no neighboring node in the MIS that with a larger identifier. Second, a node leaves the MIS if there is a neighboring node in the MIS with a larger identifier than itself. The performance ratio of all these algorithms is unknown.

Unlike all approaches mentioned above, Shi et al. [2004] proposed an anonymous self-stabilizing algorithm that identifies an 1-MIS under a central daemon. An 1-MIS is an MIS with an extra property that the removal of any node from the MIS does not allow the addition of more nodes into the MIS. This algorithm works for tree-structured topologies only. Recently, Hedetniemi et al. [2013] proposed self-stabilizing algorithms that identify two disjoint maximal independent sets on a given graph. Their methods are designed to run under a central or distributed daemon.

Table I summarizes the related work. A recent literature survey on relevant self-stabilizing algorithms can be found in [Guellati and Kheddouci 2010]. Note that no prior work considers node degree in forming an MIS. Also, to the best knowledge of the authors, our work is the first self-stabilizing algorithm to the MWIS problem.

Cohen et al. [2008] first introduced the notion of selfish stabilization, where processes may seek their private goals while cooperating with one another in achieving a common goal. The private goals of different processes may be conflicting. Under this assumption, the authors devised a stabilizing distributed algorithm that forms a spanning tree. If there are two types of processes that have conflicting private goals, the authors proved that the algorithm is weakly stabilizing.

Gouda and Acharya [2011] assumed that gain functions (i.e., utility functions) of processes are defined only at quiescent states. The actions of all processes are assumed independent of the gain functions and designed to lead the system into a quiescent state. After entering a quiescent state, processes may cause perturbations as a result of attempting to increase their individual gains. The main concern of this work is whether such behavior can bring the system into illegitimate states.

Our game framework is a *noncooperative* game, meaning that players do not cooperate with each other for a system goal. It is also a graphical game [Kearns et al. 2001], because the utility of each player (to be shown in the next section) is only affected by the strategies of its neighbors, not by all other players. Our game is closely related to congestion games [Rosenthal 1973]. In a congestion game, each player has to select a collection of resources yet minimize (resp. maximize) the cost (resp. payoff) of such a selection. The cost (resp. payoff) of a selection (i.e., a strategy in game) is the summation of all individual resource costs (resp. payoffs) involved in that selection. The cost (resp. payoff) of selecting a specific resource is typically a monotonically increasing (resp. decreasing) function of the number of players that also select the same resource. Bilò et al. [2011] considered *graphical congestion games*, in which the cost function of each resource e with respect to player p_i is a linear function of the number of p_i's neighboring players that also select e. As we shall see in Sec. 3.2, the game model proposed in this paper also falls into the category of graphical congestion games. Nevertheless, several properties of the proposed game model are not yet addressed in prior work.

A related yet a little different game model is graphical coordination game with multiple players [Montanari and Saberi 2010]. In a graphical coordination game, players receive a higher payoff from adopting the same strategy as their neighbors. In contrast, in a graphical congestion game, players raise their payoffs by not selecting the same strategy with their neighbors. In this regard a congestion game is an anti-coordination game. Another related problem is distributed graph coloring [Kearns et al. 2006; Marden et al. 2015], which demands that neighboring nodes do not use the same color. For the MIS/MWIS problem, we can view the strategy of being in the set as one color (say, black) and the opposite as another color (say, white). While an independent set demands that no node colored black can have a neighbor that is also colored black, it is totally alright for white nodes to have neighbors that are also colored white. Therefore, the MIS/MWIS problem is not equivalent to the graph coloring problem.

Our work is most closely related to the work by Yen and Chen [2014]. Their approach shares several similarities with our framework:

- Both achieve a system goal by designating private objectives of individual processes.
- Both assume the model of *sequential game*, where one player moves after another and no simultaneous moves are allowed.
- Both assume *perfect information*, meaning that when making a decision, a player is aware of all relevant decisions that have been made (by other players) previously.
- Both are guaranteed to converge to a quiescent yet legitimate game state regardless of initial game configuration and game play sequence.
- Both have been converted into distributed algorithms expressed in guarded commands.

Nevertheless, our game framework differs from the approach proposed by Yen and Chen [2014] in the following ways.

— Although the game design of Yen and Chen [2014] is guaranteed to converge, the corresponding distributed algorithm converted from the game is only weakly stabilizing. The reason is that, in their game design, a player needs strategy information from non-neighbor players. Although this requirement conforms to the assumption of perfect information, when converting the game into a distributed algorithm a process has to access non-local variables (i.e., variables that are neither owned by the process nor by any of its neighbors.) Such an access can only be done indirectly due to the inherent limitation imposed by the shared variable communication model. In contrast, because a process in our algorithm does not need to access non-local variables, all algorithms derived from our framework are self-stabilizing rather than merely weak-stabilizing.
— Besides sequential moves, we also consider simultaneous moves that are allowed by distributed and synchronous daemons for self-stabilizing distributed algorithms (Sec. 4). In contrast, Yen and Chen [2014] considered only sequential moves and thus central daemon.
— Although Yen and Chen [2014] did convert their game design into distributed algorithms expressed in guarded commands, they did not explicitly show how this is done generally. In contrast, we present a generic transformation that virtually applies to any noncooperative game (Sec. 3.4).
— Some game notion was not fully explored by Yen and Chen [2014]. For example, Pareto optimality is not an essential property in their work, while it corresponds to the "maximal" property of an MIS in our problem. Similarly, social welfare of a game, which is defined as the total utility across all players, does not bear any meaning in prior work. In contrast, maximizing social welfare in our MIS game corresponds to the identification of an independent set that is of maximal cardinality.

In short, this work reveals the link between game theory and self-stabilization in a more comprehensive manner than prior work [Yen and Chen 2014].

3. THE PROPOSED APPROACH

This section is concerned with a model for sequential games, where one player moves after another and no simultaneous moves are allowed. This assumption fits the central daemon. Sec. 4 will relax this assumption and discuss how to deal with simultaneous moves when the game design is turned into a self-stabilizing algorithm for the synchronous or distributed daemon.

3.1. Generic MWIS Game

Let $P = \{p_1, p_2, \cdots, p_n\}$ be the set of all processes in a distributed system. A symmetric binary relation on P can be defined as a set of unordered pairs and represented by an undirected graph $G = (P, E)$, where $E \subseteq P \times P$ is the set of all pairs of processes that are related. To formulate the MWIS problem, let $W : P \to \mathbb{R}^+$ be a weighting function that gives each process a non-negative weight. We model the problem of identifying an MWIS in G as *MWIS* game $\Gamma = [P; E; W; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n]$, where $S_i = \{1, 0\}$ is process p_i's *strategy set* indicating whether p_i chooses to be in the MWIS (1) or not (0), and u_i is p_i's utility function. A *strategy profile* is an n-tuple $C = (c_1, \ldots, c_n)$, where $c_i \in S_i$ represents player p_i's strategy. For convenience we sometimes express C as (c_i, c_{-i}), where c_{-i} indicates a tuple of all players' strategies other than p_i's. Let $N_i = \{p_j \mid (p_i, p_j) \in E\}$ denote the set of p_i's neighbors in G. Let $L_i \subseteq N_i$ be a subset of N_i and $\alpha > 1$ be a constant. Given a strategy profile $C = (c_i, c_{-i})$, we define the utility of p_i

associated with C as

$$u_i(C) = \sum_{p_j \in L_i} w(c_i, c_j) + c_i, \tag{2}$$

where

$$w(c_i, c_j) = -\alpha c_i c_j. \tag{3}$$

The private goal of each player is to dynamically select a strategy that maximizes its own utility. Therefore, p_i chooses $c_i = 1$ only if no node in L_i is also in the set. It would rather choose $c_i = 0$ if $c_j = 1$ for any $p_j \in L_i$.

The initial strategy of each player is arbitrary, resembling an arbitrary system state after a fault. The goal of the game design is to lead the system from an arbitrary state to a stabilized state where $\{p_i \mid c_i = 1\}$ constitutes a maximal independent set. This is done through a non-deterministic sequence of strategy changes. Players can change their strategies if necessary, following the so-called *best-response rule*. This rule states that player p_i selects strategy c_i^* only if

$$c_i^* = \mathbf{BR}_i(C) \triangleq \operatorname*{arg\,max}_{c_i \in S_i} u_i(c_i, c_{-i}). \tag{4}$$

The best-response rule demands that when selecting a strategy, a player is aware of what strategies have already been selected by other players.

The generic MWIS game is for both the MIS and the MWIS problems. For the MIS problem, the quantitative goal is to maximize the social welfare function defined as

$$\sum_{i=1}^{n} u_i(C) = |\{p_i \in P \mid c_i = 1\}| \tag{5}$$

for a game result C. For the MWIS problem, a valid game result C is evaluated by the total weight in the independent set:

$$\sum_{i=1}^{n} u_i(C) W(p_i) = \sum_{c_i = 1} W(p_i). \tag{6}$$

Specific games can be derived from the generic game by varying the definition of L_i. Doing so we can practice many node selection rules for MIS/MWIS formation. All existing self-stabilizing algorithms for the MIS problem do not prefer any node in forming an MIS. This corresponds to the definition $L_i \triangleq N_i$ for all $p_i \in P$. On the other hand, some centralized greedy approach to the MIS problem [Halldórsson and Radhakrishnan 1997] and many others for the MWIS problem [Sakai et al. 2003] select the *best* candidate in each round of the selection process based on some node ordering relation. Denote that relation by $\preceq$. Such a selection preference can be locally embedded in each player by defining $L_i \triangleq \{p_j \in N_i \mid p_j \preceq p_i\}$. We shall explore more issues about the definition of L_i in the following two subsections.

3.2. Γ_{sym}: Symmetric MWIS Game

We consider first a special instance of the MWIS game with the definition of $L_i \triangleq N_i$ for every player p_i. This game is characterized by the mutual influences on neighboring nodes: $p_j \in L_i \Leftrightarrow p_i \in L_j$. For this property the payoff given out on every edge is symmetric: $w(c_i, c_j)$ contributes $u_i(\cdot)$ whenever $w(c_j, c_i)$ contributes $u_j(\cdot)$. We refer to this game as the *symmetric* MWIS game and denote it by Γ_{sym}.

One might wonder if symmetric influences lead to instability of the game. Consider neighboring nodes p_i and p_j with $(c_i, c_j) = (1, 1)$ at some time during the game. One of these nodes may act first to reset its choice to 0. This action does not prevent the other from also resetting its choice to 0 later, as long as doing so is the other's best response. As a reaction, the one that acts first may change its strategy back to 1 for a utility gain. It is our concern whether such a potential chain reaction leads to instability of the game, i.e., a situation where some players repeatedly change their strategy.

The stability of a game is characterized by the notion of Nash equilibrium. When a game enters a Nash equilibrium, no player can further increase its utility by unilaterally deviating from its current strategy.

Definition 3.1 (*Nash equilibrium*). A strategy profile $C^* = (c_i^*, c_{-i})$ of a game $\Gamma \triangleq [P; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n]$ is a Nash equilibrium for Γ if $u_i(c_i^*, c_{-i}) \geq u_i(c_i, c_{-i})$ for all $i \in \{1, \dots, n\}$ and for all $c_i \in S_i$.

We shall show that *every* strategy profile of Γ_{sym} eventually leads to a Nash equilibrium through a series of *improvements*, where an improvement refers to a transition of strategy profile caused by the best-response strategy of a single player. This is done by showing that Γ_{sym} is an *exact potential game* [Monderer and Shapley 1996], i.e., it admits an exact potential function defined below.

Definition 3.2 (*Exact potential function*). A function $\Phi : S_1 \times \dots \times S_n \to \mathbb{R}$ is an exact potential function for a game $\Gamma \triangleq [P; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n]$ if $u_i(c_i^*, c_{-i}) - u_i(c_i, c_{-i}) = \Phi(c_i^*, c_{-i}) - \Phi(c_i, c_{-i})$ holds for all $p_i \in P$ and all $c_i, c_i^* \in S_i$ with $c_i \neq c_i^*$.

Monderer and Shapley [1996] proved that every finite potential game possesses a Nash equilibrium, and that any Nash equilibrium can be reached by a series of potential improvements caused by player's best-response strategies.

We can map the symmetric MWIS game to a graphical congestion game [Bilò et al. 2011] in the following way. Each player has to choose one resource from the resource set $\{0, 1\}$. For any player p_i, the payoff of selecting 0 ($c_i = 0$) is always 0 while the payoff of selecting 1 ($c_i = 1$) is

$$1 - \alpha |\{p_j \in N_i \mid c_j = 1\}|. \tag{7}$$

Both payoffs are defined as a (monotonically decreasing) linear function of the number of p_i's neighboring players that select the same resource. Bilò et al. [2011] showed that if the graph is undirected (i.e., neighboring nodes have mutual influences on each other), a graphical congestion game is an exact potential game.

In fact, we can prove that the following function is an exact potential function for the symmetric MWIS game. Detailed proof is given in the Appendix.

$$\Phi(C) = \frac{1}{2} \sum_{j=1}^{n} \sum_{p_k \in N_j} w(c_j, c_k) + \sum_{j=1}^{n} c_j. \tag{8}$$

Correctness is another concern of the designed game. The following theorem shows the correctness of the game.

THEOREM 3.3. *When the symmetric MWIS game $\Gamma_{sym} \triangleq [P; E; W; \{S_i\}_{i=1}^n; \{u_i\}_{i=1}^n]$ has reached a Nash equilibrium, the set $S = \{p_i \mid c_i = 1\}$ is an independent set of $G = (P, E)$.*

PROOF. By way of contradiction, assume that S at a Nash equilibrium is not an independent set of G. This implies that there exists $(p_i, p_j) \in E$ such that $\{p_i, p_j\} \subseteq S$. In that case, p_i or p_j has a negative utility value (less than or equal to $1 - \alpha$). Either player can have a utility gain if it changes its strategy to 0, this contradicts the assumption that the game has reached a Nash equilibrium. Hence S is an independent set. $\square$

We can further prove that at a Nash equilibrium S is *maximal* by showing that the Nash equilibrium is Pareto optimal.

Definition 3.4 (*Pareto optimal*). A strategy profile C is Pareto optimal if and only if there exists no other strategy profile C' such that $u_i(C') \geq u_i(C)$ for all $i \in \{1, \ldots, n\}$ and $u_j(C') > u_j(C)$ for at least one $j \in \{1, \ldots, n\}$.

In other words, a strategy profile is Pareto optimal if no player can increase its utility without decreasing the utility of any others. To see why Pareto optimality corresponds to the "maximal" property, first note that the utility of any player at a Nash equilibrium is either 0 or 1. Because any player with negative utility value can raise its utility to 0 by switching its strategy from 1 to 0, any player with strategy 1 at a Nash equilibrium can only have utility value 1. Furthermore, no player having utility value 1 can further increase its utility, so only players with utility value 0 (i.e., players choosing 0) can possibly increase their utilities. If a Nash equilibrium is Pareto optimal, which implies that no players with utility value 0 can increase their utilities without decreasing the utility of any others, we cannot add any node to the independent set S without removing any others from S. Therefore, there exists no superset of S that is also an independent set. On the other hand, if S at a Nash equilibrium is maximal, then it is impossible to increase the utility of any player $p_j \notin S$ by adding p_j into S because some neighbor of p_j must be in S. Since all players in S already have the maximal utility value, the Nash equilibrium is Pareto optimal.

THEOREM 3.5. *Every Nash equilibrium of the symmetric MWIS game is Pareto optimal.*

PROOF. We prove the theorem by showing that no players with utility value 0 at a Nash equilibrium can increase their utilities without decreasing the utility of any others. By (2), any player p_i having utility value 0 at a Nash equilibrium implies that $c_i = 0$ and there exists $p_j \in N_i$ with $c_j = 1$. Note that p_j's utility must be 1. To increase p_i's utility, c_i must be changed to 1 while c_j must be changed to 0. This means that p_j's utility must be degraded. Therefore, it is impossible to increase p_i's utility without decreasing p_j's, which means that every game result is Pareto optimal. $\square$

Although the result in any Nash equilibrium is maximal, it is not necessarily a maximum independent set. With our definition of utility, finding an independent set that is of the maximal cardinality is to maximize the social welfare defined in (5) with additional constraint $W(p_i) = 1$ for all $p_i \in P$. Clearly, the result is also not necessarily a maximum weighted independent set.

To analyze the game convergence time, let us consider a sequence of strategy profiles $C^{(0)}, C^{(1)}, \ldots, C^{(l)}$ with $l \in \mathbb{N}^{\infty}$, where $C^{(t)} = (c_1^{(t)}, c_2^{(t)}, \ldots, c_n^{(t)})$ for all t with $0 \leq t < l$.

Definition 3.6 (*Best-reply path*). [Milchtaich 1996] A sequence of strategy profiles $C^{(0)}, C^{(1)}, \ldots, C^{(l)}$ with $l \in \mathbb{N}^{\infty}$ is called a *best-reply path* if for all t with $0 \leq t < l$ the value of $C^{(t+1)}$ differs from $C^{(t)}$ in exactly one strategy c_i such that $c_i^{(t+1)} = \mathrm{BR}_i(C^{(t)})$.

Intuitively, a best-reply path is a transition sequence of strategies profiles caused by players following the best-response rules. Therefore, the maximal length of any best-reply path determines the worst-case game convergence time.

THEOREM 3.7. *Given a graph $G = (P, E)$ that contains no isolated node, the length of any best-reply path in Γ_{sym} is at most $O(|E|)$.*

PROOF. The minimum value of $\Phi(\cdot)$ is $-\alpha|E| + |P|$ (which occurs when $C = (1, 1, \ldots, 1)$). For a graph that contains no isolated node, the maximal value of $\Phi(\cdot)$ is

$|P|-1$. Consider any best-reply path $C^{(0)}, C^{(1)}, \ldots$. Without loss of generality, assume that $C^{(t+1)}$ differs from $C^{(t)}$ in some c_i for all $t \geq 0$. Because Γ_{sym} is an exact potential game, we have $\Phi(C^{(t+1)}) - \Phi(C^{(t)}) = u_i(C^{(t+1)}) - u_i(C^{(t)})$ for all $t \geq 0$. If $(c_i^{(t)}, c_i^{(t+1)}) = (0,1)$, the value of $u_i(C^{(t+1)}) - u_i(C^{(t)})$ is 1 as indicated by (19). If $(c_i^{(t)}, c_i^{(t+1)}) = (1,0)$, (22) indicates that the utility gain is at least $\alpha - 1$. Therefore, the maximal length of any best-reply path in Γ_{sym} is at most $(|P| - 1 + \alpha|E| - |P|)/\min(1, \alpha - 1) = O(|E|)$. □

3.3. Γ_{asym}: Asymmetric MWIS Game

We now consider a more general class of the MWIS game that is characterized by a binary relation $\preceq$ on P. For any two nodes p_i and p_j, $p_j \preceq p_i$ means that the preference value of p_j is at least as high as that of p_i when forming an MIS/MWIS. The definition of $\preceq$ depends on a function $f_p : P \to \mathbb{R}$ that assigns each process a metric value. By defining different $f_p(\cdot)$'s, we can mimic various heuristics proposed for MIS/MWIS and hopefully achieve a better game result (larger independent sets or higher total weights). Assume that we prefer nodes with a higher $f_p(\cdot)$ value when forming an MIS/MWIS, thus

$$p_j \preceq p_i \text{ iff } f_p(p_j) \geq f_p(p_i) \tag{9}$$

for any two nodes p_i and p_j. For example, the definition $f_p(p_i) \triangleq W(p_i)$ corresponds to the heuristic that favors nodes with higher weights when forming an MWIS [Basagni 2001]. The node ranking rule used by GWMIN [Sakai et al. 2003] corresponds to the following definition of f.

$$f_p(p_i) \triangleq \frac{W(p_i)}{\deg(p_i)+1}. \tag{10}$$

For GWMIN2 [Sakai et al. 2003] the definition would be

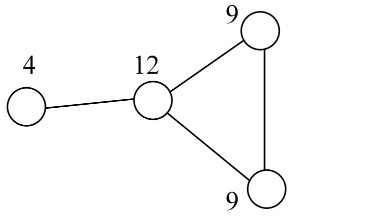
$$f_p(p_i) \triangleq \frac{W(p_i)}{W(p_i) + \sum_{p_j \in N_i} W(p_j)}. \tag{11}$$

For schemes that aim to minimize some cost function $c_p(\cdot)$, we can define $f(p_i) \triangleq 1/c_p(\cdot)$ or $f(p_i) \triangleq -c_p(\cdot)$. For example, to prioritize nodes with minimum node degrees when forming an MIS, $f_p(\cdot)$ can be defined as $f_p(p_i) \triangleq 1/(\deg(p_i)+1)$ [5]. We refer to this class of games as *asymmetric games* and denote it by Γ_{asym}.

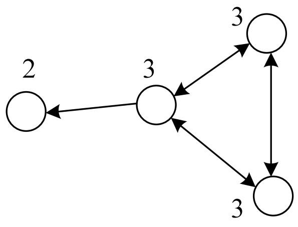
Note that a game design here is not equivalent to the corresponding greedy approach due to the decentralization nature of the game, arbitrary initial game configuration, and non-deterministic game dynamics. Moreover, every time a greedy approach selects a node, the node and all its neighbors are removed from the graph, so any $f_p(\cdot)$ function related to node degree should be recomputed for all the remaining nodes. In contrast, $\deg(p_i)$ is a unchanged for any node p_i during a game play.

Clearly, $\preceq$ is a node ordering that is well defined for *every* pair of nodes. This is essential for a centralized algorithm that repeatedly looks for a node with maximum $f(\cdot)$ value. For a distributed approach where only local information is available, we restrict our attention to the same relation defined on neighboring nodes only. That is, $p_j \preceq p_i$ is of interest only if $p_j \in N_i$. More specifically, $L_i \triangleq \{p_j \in N_i \mid p_j \preceq p_i\}$ in an MWIS^a game. Because the relation $\preceq$ defined above is neither symmetric nor asymmetric, we have $p_j \in L_i \not\Rightarrow p_i \in L_j$ but $p_j \preceq p_i \land p_i \preceq p_j$ is still possible and so is $p_j \in L_i \land p_i \in L_j$. This possibility makes the symmetric MWIS game a special case of asymmetric MWIS games.

[5]This definition avoids possible zero denominators that may occur to isolated nodes.



(a) A given weighted graph with weights shown beside nodes.



(b) The influence graph corresponding to (a) where the value of $f_p(\cdot)$ defined as (10) is shown beside each node.

Fig. 2. An example of influence graph

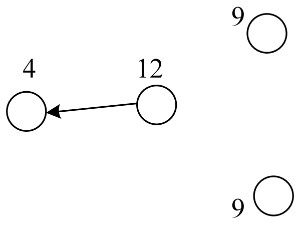


Fig. 3. The new influence graph that corresponds to (12).

Because now player's influences may not be mutual, we should remodel the interdependence relationship between players. For each undirected graph $G = (P, E)$ given for the MWIS/MIS problem, we define an *influence graph* to be a directed graph $G_i = (P, E_i)$ constructed from G such that G and G_i share the same vertex set P and for every $(p_i, p_j) \in E$, $(p_i, p_j) \in E_i$ iff $p_i \preceq p_j$. In other words, there is a directed arc from p_i to p_j in an influence graph if and only if $f_p(p_i) \geq f_p(p_j)$. Note that G_i may contain directed cycles. Each cycle is a cyclic sequence of adjacent nodes in G that are of the same $f_p(\cdot)$ value. Figure 2 shows an example of influence graph that contains two directed cycles (one clockwise and the other counterclockwise).

It appears difficult to show that Γ_{asym} is an exact potential game. In fact, it has been pointed out that an equilibrium may not exist in a graphical congestion game when the associated social knowledge graph is directed (i.e., influences are non-mutual) [Bilò et al. 2011; Tekin et al. 2012]. If the social graph is directed (which is essentially an influence graph) but acyclic, Bilò et al. [2011] showed that the game always converges. However, this result does not help since the influence graph considered here may contain cycles.

One might consider excluding the possibility of equal preference from the definition of $\preceq$ to avoid possible cycles in an influence graph. More specifically, define

$$p_j \preceq p_i \text{ iff } f_p(p_j) > f_p(p_i). \tag{12}$$

For the example shown in Figure 2, the resulting influence graph will be like that shown in Figure 3. Although the influence graph no longer contains cycles, such a redefinition of $\preceq$ can lead to incorrect results. In Figure 3, the three nodes that are originally involved in the cycles can now all join the set regardless of which one moves first. The result will no longer be an independent set.

We therefore look into the dynamics of Γ_{asym} in the time domain. We assume *reasonably fair opportunities* among players in any best-reply path $C^{(0)}, C^{(1)}, \ldots$. such that for all $p_i \in P$, if $c_i^{(t)} \neq \mathrm{BR}_i(C^{(t)})$ for some $t \geq 0$ then there exits $t' > t$ such that

$c_i^{(t')} = \mathrm{BR}_i(C^{(t')})$. This assumption implies that a best-reply path either ends at a Nash equilibrium or is infinite. In the latter case, at least one player changes its strategy for infinitely many times. The following definition decomposes the stability of a game into a function of the stabilities of individual players.

Definition 3.8 (*Player's stability*). Player p_i is *stable* in a best-reply path $\xi = C^{(0)}, C^{(1)}, \dots$ if there exists some $t \geq 0$ such that $g_i(t)$ is true, where

$$g_i(t) = \begin{cases} \text{true} & \text{if } c_i^{(t)} = c_i^{(t')} \text{ for all } t' \geq t \\ \text{false} & \text{otherwise} \end{cases} \tag{13}$$

If there exists $p_i \in P$ such that $g_i(t) =$ false for all $t \geq 0$, then ξ must be infinite. On the other hand, the existence of a finite number t such that $g_i(t) =$ true for all $p_i \in P$ in ξ implies that ξ ends at a Nash equilibrium. We shall prove that every possible best-reply path in an asymmetric MWIS game ends at a Nash equilibrium.

If only one player is allowed to change her/his strategy at one time, the best-response rule implies the following two properties of Γ_{asym}.

PROPERTY 1. *If* $\mathit{BR}_i(C^{(t)}) = 1$ *at some time* $t \geq 0$ *then* $c_j^{(t)} = 0$ *for all* $p_j \in L_i$.

PROPERTY 2. *If* $\mathit{BR}_i(C^{(t)}) = 0$ *at some time* $t \geq 0$ *then there exits* $p_j \in L_i$ *such that* $c_j^{(t)} = 1$.

These two properties are needed to prove Lemma 3.10. We also define a *preference relation* $\prec_f$ on nodes as follows.

Definition 3.9. (Preference relation $\prec_f$) For any two nodes p_i and p_j, $p_j \prec_f p_i$ if and only if $p_j \preceq p_i$ and $p_i \not\preceq p_j$.

Equivalently, $p_j \prec_f p_i$ if and only if $f_p(p_j) > f_p(p_i)$. We define $U_i \triangleq \{p_j \in L_i \mid p_j \prec_f p_i\}$ for each p_i. Lemma 3.10 indicates that, though any player in L_i has an influence on p_i's utility, only those in U_i can have an influence on p_i's stability.

LEMMA 3.10. *Let* ξ *be a best-reply path. For each player* p_i*, if* $U_i = \emptyset$ *or there exists* $t \geq 0$ *such that* $g_j(t) =$ *true holds in* ξ *for all* $p_j \in U_i$*, then* $g_i(t') =$ *true for some* $t' \geq t$ *(*$t = 0$ *if* $U_i = \emptyset$*).*

PROOF. By way of contradiction, assume that either $U_i = \emptyset$ or there exists $t \geq 0$ such that $g_j(t) =$ true holds for some player p_i and all $p_j \in U_i$ but $g_i(t') =$ false for all $t' \geq t$. This implies that ξ is infinite and p_i constantly changes its strategy. If $L_i = \emptyset$, then $\mathrm{BR}_i(C^{(t)}) = 1$ for all $t \geq 0$ and we have $g_i(t) =$ true for some $t \geq 0$ by the reasonably fair opportunities assumption. Assume $L_i \neq \emptyset$. Consider first the case $U_i = L_i$ (thus $U_i \neq \emptyset$). It may happen that $c_i^{(t)} = \mathrm{BR}_i(C^{(t)})$. If $c_i^{(t)} \neq \mathrm{BR}_i(C^{(t)})$, the fair opportunity assumption ensures that $c_i^{(t')} = \mathrm{BR}_i(C^{(t')})$ for some $t' > t$. Either case implies that $g_i(t') =$ true for some $t' \geq t$ because all players in L_i do not revise their strategies at and after t, which contradicts with the assumption. Consider next the case $U_i \subset L_i$ (which includes the special case that $U_i = \emptyset$). The condition that p_i constantly changes its strategy implies that p_i changes c_i from 1 to 0 for infinitely many times. Suppose that $c_i^{(t_1)} = \mathrm{BR}_i(C^{(t_1)}) = 1$ for some time $t_1 > t$ and $c_i^{(t_2)} = \mathrm{BR}_i(C^{(t_2)}) = 0$ for some later time $t_2 > t_1$. Properties 1 and 2 imply that $c_j^{(t_1)} = 0$ for all $p_j \in L_i$ and there exists $p_j \in L_i$ with $c_j^{(t_2)} = 1$. It turns out that there must exist some $p_j \in L_i$ that changes c_j from 0 to 1 at some time $t_3 \in (t_1, t_2)$. Refer to Fig. 4 for an illustration. Because either $U_i = \emptyset$ or $g_k(t') =$ true for all $p_k \in U_i$ at all time $t' > t$, p_j must be in

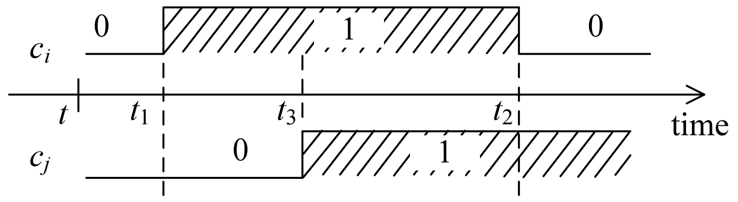


Fig. 4. Interaction between neighboring players p_i and p_j

Table II. Mapping from Game to Self-Stabilizing Algorithm

Game	Self-Stabilizing Algorithm
Player p_i	Process p_i
Strategy c_i	Variable c_i
Strategy profile C	Global state C
Utility function and best-response rule	Guards and actions of guarded commands
Best-reply path	Process execution sequence
Nash equilibrium	Quiescent state

$L_i \setminus U_i$. Therefore, $p_j \preceq p_i$ and $p_i \preceq p_j$ and thus $p_i \in L_j$. However, because $c_i^{(t_3)} = 1$, $\text{BR}_j(C^{(t_3)}) = 0 \neq c_j^{(t_3)} = 1$. This contradicts with the best response rule and thus is impossible. □

Given an influence graph $G_i = (P, E_i)$, the preference relation $\prec_f$ on P corresponds to a *preference graph* $G_f = (P, E_f)$ that is obtained by removing all (u, v)'s from E_i for which both (u, v) and (v, u) are in E_i. Without loss of generality, let $\xi_f = p_1, p_2, \ldots, p_m$, where $2 \leq m \leq n$, be a sequence of nodes in G_f such that $(p_j, p_{j+1}) \in E_i$ for all j, $1 \leq j \leq m-1$, and $(p_m, p_1) \in E_i$. If ξ_f is a directed cycle in G_f, then either $f_p(p_1) > f_p(p_2) > \ldots > f_p(p_m) > f_p(p_1)$ or $f_p(p_1) < f_p(p_2) < \ldots < f_p(p_m) < f_p(p_1)$. Both cases are impossible. Therefore, G_f must not contain any directed cycles and thus there exists at least one source (a node with zero indegree) in G_f. This property guarantees the stability of the asymmetric MWIS game, as the following theorem shows.

THEOREM 3.11. *Any best-reply path in Γ_{asym} is finite.*

PROOF. We prove the theorem by showing that all players are stable in any game play. Let $\deg^-(p_i)$ denote the indegree of p_i in G_f. Define $N(k) \triangleq \{p_i \in P \mid \deg^-(p_i) = k\}$ for all k, $0 \leq k \leq n-1$. For any source p_i in G_f (i.e., $p_i \in N(0)$), $U_i = \emptyset$ and thus p_i is stable by Lemma 3.10. For all $0 < k \leq n-1$, assume that all players in $N(0) \cup N(1) \cup \ldots \cup N(k-1)$ are all stable. Then all players in $N(k)$, if any, are stable as well by Lemma 3.10. Thus, the theorem can be proved by induction on k. □

Theorem 3.11 proves that any asymmetric MWIS game eventually reaches a Nash equilibrium regardless of initial game configurations and best-reply path. We can show that when an asymmetric MWIS game enters a Nash equilibrium, the set $S = \{p_i \mid c_i = 1\}$ is indeed an MWIS. The proof is analogous to that of the symmetric MWIS game and is omitted.

3.4. Transformation to Self-Stabilizing Algorithms

We now show how to transform a game design into a self-stabilizing distributed algorithm expressed in guarded commands. Table II shows the mapping for such a transformation. In particular, each process p_i acting as a player uses a local variable c_i to

Table III. Self-Stabilizing MIS/MWIS Algorithms Derived from MWIS Game

Game Type	Objective	Definition of $f_p(\cdot)$	Name	New?
Symmetric	MIS	$f_p(p_i) = f_p(p_j)$ for all $p_i \neq p_j$		No
Asymmetric	MIS	$1/(\deg(p_i) + 1)$	$\mathcal{A}_{\text{deg}}$	Yes
Asymmetric	MWIS	$W(p_i)$	$\mathcal{A}_{\text{weight}}$	Yes
Asymmetric	MWIS	Eq. (10)	$\mathcal{A}_{\text{GWMIN}}$	Yes
Asymmetric	MWIS	Eq. (11)	$\mathcal{A}_{\text{GWMIN2}}$	Yes

keep p_i's current strategy. We designate one rule for each possible strategy. The action of each rule sets c_i to some possible choice c, while the guard of each rule is a precondition ensuring that the corresponding action is p_i's best response. More specifically,

For each $c \in S_i$, formulate the following rule:
guard: $c_i \neq c \wedge c_{-i} \in \{c_{-i} \mid u_i(c, c_{-i}) > u_i(c_i, c_{-i})\}$
action: $c_i := c$

This transformation is generic in the sense that it virtually applies to all noncooperative game models.

For both symmetric and asymmetric MWIS games, the correct value of c_i is either IN or OUT, indicating whether p_i chooses to be in the independent set or not. Therefore, each p_i has two guarded commands, one for c_i = IN and the other for c_i = OUT:

R1 $c_i \neq \text{OUT} \wedge \exists p_j \in L_i : c_j = \text{IN}$
$\rightarrow c_i := \text{OUT}$
R2 $c_i \neq \text{IN} \wedge (N_i = \emptyset \vee \forall p_j \in L_i : c_j \neq \text{IN})$
$\rightarrow c_i := \text{IN}$

Here we use $c_i \neq$ OUT (resp. IN) instead of c_i = IN (resp. OUT) because the value of c_i could be arbitrary (neither IN nor OUT) after a fault. The assumption of reasonably fair opportunities among players corresponds to fairness assumption of the daemon: if an command is enabled infinitely often, then it is eventually executed.

Each process p_i needs the information of L_i, which includes the set of neighbors of p_i, and the weight, degree, and strategy of each neighbor. Accessing a neighbor's local information (weight, degree, and strategy) is accordant to the shared-variable assumption. If the underlying graph can change dynamically, we may let each process p_i locally maintain N_i and shared variables d_i that respectively represent p_i's degree. In this case there should be another guarded command that corrects the values of d_i and N_i after a topology change or a transient fault. To ease our discussion, we simply assume a static topology and that the graph information (i.e., N_i and d_i) is preserved when transient faults occur. Similarly, we assume that the weight of each process is not affected by transient faults.

Table III lists some self-stabilizing MIS/MWIS algorithms that are converted from the game designs. For the symmetric MWIS game, the transformation result resembles conventional self-stabilizing MIS algorithms [Shukla et al. 1995; Hedetniemi et al. 2003]. For asymmetric MWIS games, the results are several brand new self-stabilizing MWIS algorithms (with different definitions of $f_p(\cdot)$). We refer to the set of these algorithms as $\mathcal{A}_{\text{MWIS}}$. In particular, $\mathcal{A}_{\text{deg}}$ is an $\mathcal{A}_{\text{MWIS}}$ algorithm that prefers nodes with minimum node degrees when forming an MIS. This is the first self-stabilizing algorithm for MIS that considers node degree.

In the following we analyze the *approximation ratio* ρ of $\mathcal{A}_{\text{deg}}$. Formally,

$$\rho(G) = \max_{G} \frac{\alpha(G)}{\mathcal{A}_{\text{deg}}(G)},$$

where G ranges over all finite graphs, $\alpha(G)$ is the size of a maximum independent set of G, and $\mathcal{A}_{\text{deg}}(G)$ is the size of the solution obtained by algorithm $\mathcal{A}_{\text{deg}}$ for G. There exist only a few self-stabilizing algorithms with known approximation factor, e.g., [Turau 2010; Turau and Hauck 2011]. In this section we compute ρ for algorithm $\mathcal{A}_{\text{deg}}$. For this purpose we prove that algorithm $\mathcal{A}_{\text{deg}}$ produces maximal independent sets that are also computed by the sequential greedy algorithm of Halldórsson and Radhakrishnan [1997]. Thus, $\mathcal{A}_{\text{deg}}$ has the same approximation ratio as their algorithm.

Consider a configuration where no node is privileged with respect to $\mathcal{A}_{\text{deg}}$. Let I be the set of nodes that are IN. For each subset $U \subseteq P$ denote by $minDeg(U)$ the subset of nodes of U that have minimal degree in the subgraph induced by U. Note that $minDeg(U) \neq \emptyset$ if $U \neq \emptyset$.

LEMMA 3.12. *There exists an ordering $p_1, \ldots, p_{|I|}$ of the nodes in I such that for $i = 1, \ldots, |I|$ node p_i has minimum degree in the graph induced by $P_{i-1} = P \setminus \bigcup_{j=1,\ldots i-1} \hat{N}_j$, where $\hat{N}_j = N_j \cup \{p_j\}$.*

PROOF. If $minDeg(P) \cap I = \emptyset$ then p_i is OUT for all $p_i \in minDeg(P)$. This is impossible since rule R2 would be enabled for all $v \in minDeg(P)$, thus $minDeg(P) \cap I \neq \emptyset$.

The construction of the ordering is by induction. Choose any node p_1 from $minDeg(P) \cap I$. Note that w is OUT for all $w \in N(p_1)$ by rule R1. Suppose there already exist $p_1, \ldots, p_i$ in I that fulfill the stated property. Observe that w is OUT for all $w \in N(p_j)$ for $j = 1, \ldots, i$ by rule R1. If $P_i = \emptyset$ then $i = |I|$ and the proof is complete. Hence, $P_i \neq \emptyset$. Assume $minDeg(P_i) \cap I = \emptyset$. Let $p \in minDeg(P_i)$ and w a neighbor of p with $\deg(w) \leq \deg(p)$ such that w is IN. Then $w \notin \{p_1, \ldots, p_i\}$ since otherwise $p \notin P_i$. Hence, $w \notin \bigcup_{j=1,\ldots i-1} \hat{N}_j$ since w is IN. This implies $w \in P_i$ and thus $w \in minDeg(P_i)$. Then w is OUT by assumption. This contradicts with the choice of w. Therefore, $minDeg(P_i) \cap I \neq \emptyset$. So choose any node p_{i+1} from $minDeg(P_i) \cap I$. □

THEOREM 3.13. *The performance ratio of algorithm $\mathcal{A}_{deg}$ is bounded by $(\Delta + 2)/3$, where Δ is the maximal node degree in the given graph.*

PROOF. By Lemma 3.12 algorithm $\mathcal{A}_{\text{deg}}$ computes a maximal independent set that is also computed by the Greedy algorithm shown above. By Theorem 5 of [Halldórsson and Radhakrishnan 1997] the performance ratio of algorithm $\mathcal{A}_{\text{deg}}$ is $(\Delta + 2)/3$. □

4. DEALING WITH SIMULTANEOUS MOVES

The proposed games till now are all sequential games, which disallow simultaneous moves. This model well fits a central daemon which allows only one privileged process to execute at a time. For distributed or synchronous daemon, the execution time of a distributed algorithm can be divided into a series of *rounds*. In each round, some or all privileged processes make moves simultaneously. The action of each process in a round depends on the status of its neighboring processes in the previous round.

Allowing two neighboring processes p_i and p_j to move simultaneously may cause ping-pong effects and thus instability. To illustrate, suppose that the rounds are sequentially numbered as $T_1, T_2, \ldots$. If p_i and p_j in $\mathcal{A}_{\text{deg}}$ both have R2 enabled in round T_k, they will join the set in round T_{k+1}. Then, because $p_i \in L_j$ and $p_j \in L_i$, p_i and p_j will have R1 enabled and consequently leave the set in round T_{k+2} (see Fig. 5) under a synchronous daemon. This possibility can lead to a state transition cycle that lasts forever. In case of distributed daemon, the algorithm is weakly stabilizing because the algorithm stabilizes whenever only one privileged process is scheduled to execute.

A game that allows simultaneous moves by players is a simultaneous game. The example shown in Fig. 5 indicates that the proposed MWIS game may become a cyclic game [Selten and Wooders 2001] and thus can be instable if all players move simulta-

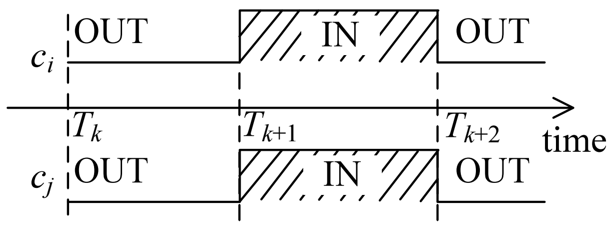


Fig. 5. Simultaneous moves of two neighboring processes p_i and p_j

Table IV. A possible state transition sequence

k	$c_1^{(k)}$	$c_2^{(k)}$	$c_3^{(k)}$	$c_4^{(k)}$	$c_5^{(k)}$	$c_6^{(k)}$	$c_7^{(k)}$
0	OUT	IN	IN	OUT	IN	OUT	OUT
1	IN	IN	OUT	OUT	IN	IN	IN
2	IN	OUT	OUT	OUT	OUT	IN	OUT
3	IN	OUT	OUT	IN	OUT	IN	OUT

neously. If a non-empty subset of players (rather than all players) are allowed to move simultaneously, the proposed game turns into a weakly acyclic game [Young 1993] which is stable only asymptotically. To ease our discussion, we skip game design challenges concerning these issues and address directly how to deal with simultaneous moves made by a dynamic non-empty subset of processes under a distributed or synchronous daemon.

A commonly-adopted solution to this problem is to use process identifiers to break the symmetry that occurs to neighboring processes. In case of the symmetric MWIS game, the result will be similar to that proposed by Ikeda et al. [2002] or Goddard et al. [2003]. For asymmetric MWIS games, special treatment is needed to deal with neighboring nodes of the same preference value.

Suppose that every node u has a unique identifier denoted by $id(u)$. We define *precedence relation* between nodes as follows.

Definition 4.1. (Precedence relation $\prec_d$) For any two nodes p_i and p_j, $p_i \prec_d p_j$ if and only if (1) $f_p(p_i) > f_p(p_j)$ or (2) $f_p(p_i) = f_p(p_j)$ and $id(p_i) < id(p_j)$.

Define $M_i \triangleq \{p_j \in N_i \mid p_j \prec_d p_i\}$. Let $c_i^{(k)}$ denote the value of c_i in round T_k. The $\mathcal{A}_{\text{MWIS}}$ algorithm modified for the synchronous or distributed daemon consists of two guarded commands as shown below.

```
R1  $c_i^{(k)} \neq \text{OUT} \wedge \exists p_j \in M_i : c_j^{(k)} = \text{IN}$
      $\to c_i^{(k+1)} := \text{OUT}$
R2  $c_i^{(k)} \neq \text{IN} \wedge (M_i = \emptyset \vee \forall p_j \in M_i : c_j^{(k)} \neq \text{IN})$
      $\to c_i^{(k+1)} := \text{IN}$
```

We use $\mathcal{A}'_{\text{MWIS}}$ to denote the set of modified algorithms. For a specific algorithm $\mathcal{A}_x$ that runs under a central daemon, we denote the corresponding modified version by $\mathcal{A}'_x$. In particular, $\mathcal{A}'_{\text{deg}}$ is the modified $\mathcal{A}_{\text{deg}}$. Consider the graph shown in Fig. 6. Assume that $id(p_i) = i$ for all i, $1 \leq i \leq 7$. Table IV shows a possible state transition sequence when running $\mathcal{A}'_{\text{deg}}$ on this graph to find an MIS.

Before we prove the stabilization of $\mathcal{A}'_{\text{MWIS}}$, we first define stabilities of processes under a synchronous or distributed daemon.

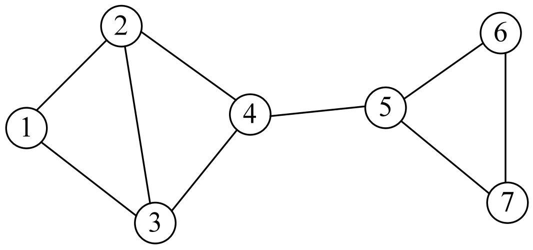


Fig. 6. A sample topology

Definition 4.2. (Process's stability) When running a distributed self-stabilizing algorithm under a distributed daemon, process p_i is *stable* in round T_k if none of p_i's guarded commands are enabled *after* T_k. We say that p_i *becomes* stable in round T_k if it is stable in T_k but not in round T_{k-1}.

The stability of nodes running $\mathcal{A}'_{\text{MWIS}}$ relies on the precedence relation between nodes. A *precedence graph* $G_d = (P, E_d)$ is a directed graph constructed from a given undirected graph $G = (P, E)$ such that G and G_d share the same vertex set P and for every $(p_i, p_j) \in E$, $(p_i, p_j) \in E_d$ iff $p_i \prec_d p_j$. Similar to G_f, G_d does not contain directed cycles. So there exists at least one source in G_d. Note also for every $(p_i, p_j) \in E$, either $p_i \prec_d p_j$ or $p_j \prec_d p_i$ but not both. Therefore, G_d is an acyclic orientation of G.

THEOREM 4.3. *Let P be a finite set of nodes. Let $R_i \subseteq P$ be the set of nodes that are stable after the i-th round ($i \geq 1$) when running $\mathcal{A}'_{MWIS}$ under a synchronous daemon. If $R_i \neq P$ then $R_i \subset R_{i+1}$.*

PROOF. Assume that $R_i = R_{i+1}$, which means that no node in $P \setminus R_i$ becomes stable in the $(i+1)$-th round. Consider any node $p_j \in P \setminus R_i$. If $M_j \subseteq R_i$, either R1 or R2 of p_j is enabled and p_j can become stable in the $(i+1)$-th round. Therefore, there must be some $p_k \in M_j$ such that $p_k \notin R_i$. By the same argument, we can show that there must be some node $p_l \in M_k$ such that $p_l \notin R_i$, and so forth. It turns out that every node in $P \setminus R_i$ must have a predecessor in G_d that is also in $P \setminus R_i$. Because $P \setminus R_i$ is finite, there must be at least one cycle in G_d, which contradicts the property that G_d is acyclic. This completes the proof. □

Next we analyze the time complexity and performance ratio of $\mathcal{A}'_{\text{MWIS}}$. Theorem 4.3 indicates that n nodes will become stable within n rounds when running $\mathcal{A}'_{\text{MWIS}}$ under a synchronous daemon, so the time complexity of $\mathcal{A}'_{\text{MWIS}}$ is $O(n)$. Consider running $\mathcal{A}'_{\text{MWIS}}$ on a complete graph G consisting of n nodes. Because G is a complete graph, the corresponding precedence graph will contain only one source. The source node will join the MWIS in the first round, while all other nodes will be out of the set in the second round. This is valid regardless of n, which represents a best-case execution time.

As an example of the worst-case execution, consider running $\mathcal{A}'_{\text{MWIS}}$ on a ring graph. Suppose that all n nodes having the same $f_p(\cdot)$ value are consecutively assigned identifiers $0, 1, \ldots, n-1$ along the ring. Refer to the corresponding digraph shown in Fig. 7. Assume that all nodes are initially OUT. In the first round, all nodes will become IN. Only node 0 will become stable in this round. In the second round, nodes $1, 2, \ldots, n-1$ will change to OUT, and nodes 1 and $n-1$ will become stable. In the third round, nodes $2, 3, \ldots, n-2$ will change back to IN, and only node 2 will become stable. In general, node k, $0 \leq k < n-1$ will become stable in Round $k+1$. The whole execution takes $n-1$ rounds.

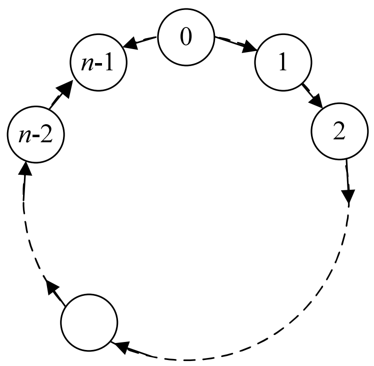


Fig. 7. The precedence graph corresponding to a ring with monotonically increasing identifiers

5. PERFORMANCE EVALUATION

We conducted simulations to investigate the performance of the proposed approach. In case of the MWIS problem, we are concerned with total weight in an independent set. For the MIS problem, we are concerned with the size of MIS and also the rate of convergence for all approaches running under a synchronous daemon.

5.1. Experimental Setup

The performance metrics depends on network topology. For a fair comparison, we tested four representative types of network topologies: Unit Disk Graph (UDG) [Clark et al. 1990], ER model [Erdös and Rényi 1959], WS model [Watts and Strogatz 1998], and BA model [Barabási and Albert 1999].

In a UDG, nodes are characterized by their locations and communication ranges. A link exists between two nodes if and only if these two nodes are within the communication rage of each other. We randomly deployed n nodes in a 1000×1000 m^2 region. Each node has a communication range of 200 m.

In the ER model, whether an edge exists between any two nodes is determined by an edge probability p_e. The resulting topology is a random graph.

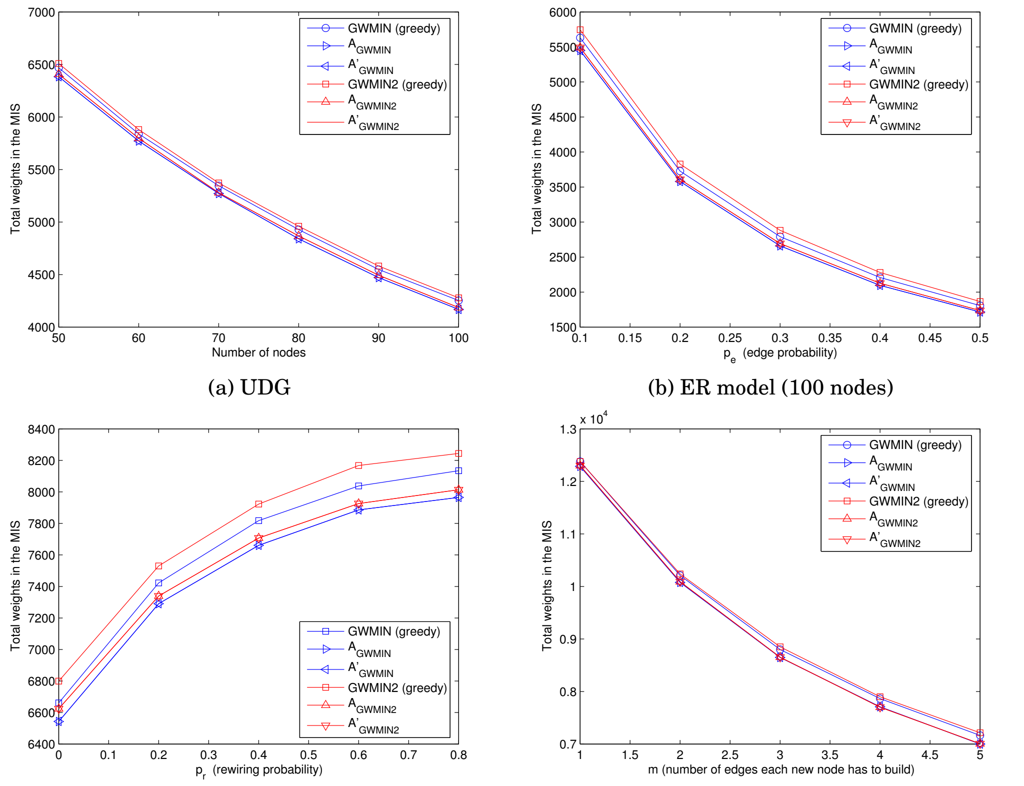
In the WS model, a regular graph is first formed, where each node has $2k$ edges connecting to its $2k$ nearest neighbors. Then, for each node, we rewire every edge of this node to a randomly selected node with probability p_r. The result is a small-world network. We assumed a graph of 100 nodes and $k = 2$.

In the BA model, the graph has n_0 nodes initially. Other nodes are incrementally added to the graph. When adding a node into the graph, we build m edges that connect this node to m nodes already in the graph. The other end of each edge is randomly determined. The probability that a new edge connects to node u is proportional to the degree of u. The result is a scale-free network. We tested a 100-node graph with $n_0 = 5$.

To simulate arbitrary system states after a transient fault, all process variables were initialized with randomly-determined values. Process identifiers were randomly yet uniquely set. For a system consisting of n processes, the weight of each process was set to an integer randomly selected from $[0, n-1]$. Each result was averaged over 1000 trials.

5.2. Results

As mentioned, several brand new self-stabilizing MWIS algorithms can be derived from asymmetric MWIS games, each with a different definition of $f_p(\cdot)$. We considered $\mathcal{A}_{\text{GWMIN}}$, $\mathcal{A}_{\text{GWMIN2}}$, $\mathcal{A}'_{\text{GWMIN}}$, and $\mathcal{A}'_{\text{GWMIN2}}$. We measured the performance of these four



(a) UDG (b) ER model (100 nodes)

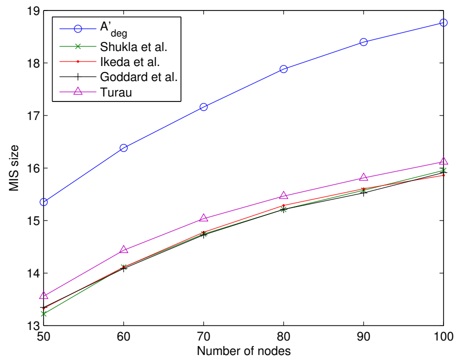
(c) WS model ($k = 2$; 100 nodes) (d) BA model ($n_0 = 5$ forming a clique; 100 nodes)

Fig. 8. Average total weights in an MWIS

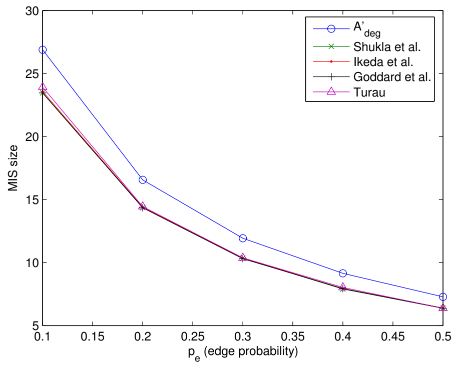
algorithms in terms of average total weight in identified independent sets. The results were compared with that of GWMIN and GWMIN2 [Sakai et al. 2003]. Note that the counterparts are not self-stabilizing algorithms. We conducted such comparisons simply because up to now there exists no self-stabilizing algorithm for the MWIS problem.

Figure 8 shows the average total weight in various types of network topologies. The performance gap between GWMIN2 and GWMIN is hardly noticeable in UDG and BA models. However, GWMIN2 significantly outperformed GWMIN in WS model. These two greedy approaches are both superior to all the four self-stabilizing algorithms in all settings. This result is justifiable since the counterparts are centralized algorithms with a global view. Another difference is that these centralized algorithms start with a fixed initial configuration (all processes were out of the independent set initially) while the initial configuration for the self-stabilizing algorithms was purely random. There is no significant performance difference between $\mathcal{A}_{\text{GWMIN}}$ and $\mathcal{A}'_{\text{GWMIN}}$, and between $\mathcal{A}_{\text{GWMIN2}}$ and $\mathcal{A}'_{\text{GWMIN2}}$. This suggests that process execution models do not have significant impact on the quality of the results.

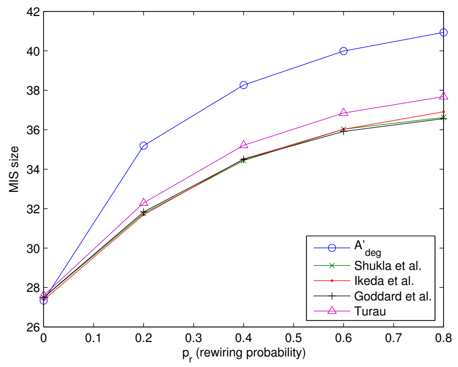
For the MIS problem, we considered existing approaches proposed by Shukla et al. [1995], Ikeda et al. [2002], Goddard et al. [2003] and Turau [2007]. We ran the first under a central daemon and the others under a synchronous daemon. We did not consider the symmetric MWIS game because its performance would be similar to [Shukla et al. 1995] under a central daemon and [Ikeda et al. 2002; Goddard et al. 2003] under a synchronous daemon. Instead, we tested $\mathcal{A}'_{\text{deg}}$, the degree-aware self-stabilizing



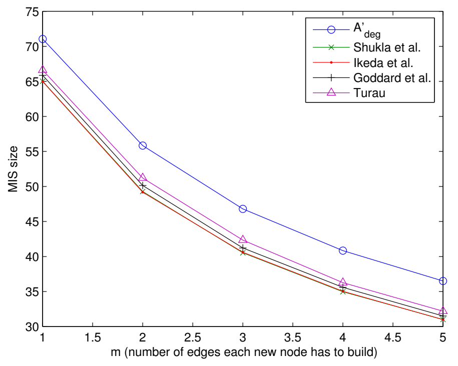
(a) UDG



(b) ER model (100 nodes)



(c) WS model ($k = 2$; 100 nodes)



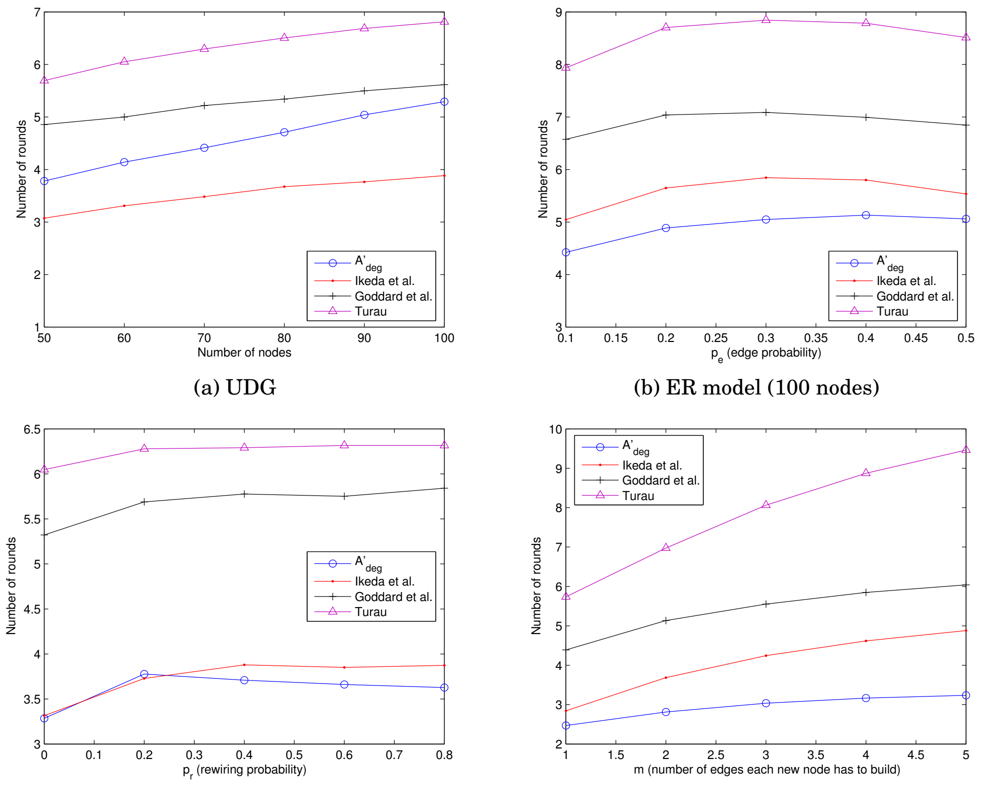
(d) BA model ($n_0 = 5$ forming a clique; 100 nodes)

Fig. 9. Average cardinality of MIS

algorithms running under distributed and synchronous daemons. Our major concern is the size of MIS. A larger MIS indicates a better result. We also measured the rate of convergence for all approaches running under a synchronous daemon.

Figure 9 shows the average MIS size in these four types of network topologies. The average MIS size generally increases with increasing number of nodes in UDGs. It also increases with p_r in small-world networks. In ER model, the average MIS size decreases as p_e changes from 0.1 to 0.5. The reason is that the network diameter (i.e., the length of the longest shortest path between any pair of nodes) decreases with an increasing p_e. In scale-free networks, the performance of all methods degrades as m increases. This is because a larger m generally implies a higher probability of a small set of nodes dominating all other nodes and thus a smaller MIS.

In all settings, $\mathcal{A}'_{\text{deg}}$ generally outperforms all the counterparts, thanks to the consideration of node degree in its design. Node degree in a random graph or UDG has a binomial distribution [Bollobás 1998; Hekmat and Van Meighem 2003]. The BA model creates a network topology for which the distribution of node degrees follows a power law. Because $\mathcal{A}'_{\text{deg}}$ prefers small-degree nodes in forming an MIS, diverging node degrees in these settings explain the superiority of $\mathcal{A}'_{\text{deg}}$ in MIS size over the counterparts. However, each node has uniformly $2k$ neighbors in regular graphs ($p_r = 0$ in WS model). In this case, the preference for small-degree nodes in the formation of an MIS



(a) UDG (b) ER model (100 nodes)

(c) WS model ($k = 2$; 100 nodes) (d) BA model ($n_0 = 5$ forming a clique; 100 nodes)

Fig. 10. Rate of convergence (average number of rounds before stabilization)

does not really affect the result. Consequently, $\mathcal{A}'_{\text{deg}}$ does not outperform the counterparts. $\mathcal{A}'_{\text{deg}}$ retakes its first place as p_r increases.

We also measured the rate of convergence (defined as the average number of rounds before stabilization) for each approach. Only the synchronous daemon was used here because it allows exact process execution sequences. Fig. 10 shows the results in each type of network topology. The approach proposed by Shukla et al. was excluded because it cannot be run under a synchronous daemon. The results indicate that $\mathcal{A}'_{\text{deg}}$ has the fastest convergence rate, followed in turn by those proposed by Ikeda et al., Goddard et al., and Turau. The only exception is in UDG, where $\mathcal{A}'_{\text{deg}}$ is next to that proposed by Ikeda et al.

6. CONCLUSIONS

Game theory provides a mathematical framework for the study of competitions and potential cooperations among participating agents. In this paper, we have proposed a generic game for identifying MIS/MWIS in a general graph. From the generic game many specific games can be defined depending on whether neighboring players have mutual influence and how such influence is defined. All games under consideration eventually enter a Nash equilibrium regardless of initial game configurations and best-reply path, and the game result is correct yet Pareto optimal. We have transformed specific game designs to corresponding self-stabilizing distributed algorithms under a central daemon. In particular, a self-stabilizing algorithm that considers node degree when forming an MIS has been proposed with its performance ratio analyzed. We also have shown how to handle simultaneous moves which is essential to the execution of self-stabilizing algorithms running under a distributed or synchronous daemon. The performance of several algorithms transformed from games has been studied through simulations with four types of representative network topologies. The simulation results indicate that the self-stabilizing algorithms for MWIS are only slightly inferior to sequential greedy approaches in terms of total average weight. Nevertheless, the self-stabilizing algorithm that is degree-aware generally outperforms other existing approaches in terms MIS size and convergence rate.

The proposed MIS algorithms also serve as approaches to the minimal dominating set. By taking the complement of the results yielded by the MIS/MWIS algorithms, we have approximations to the (weighted) minimum vertex cover problem. With a polynomial-time transformation, the proposed MIS/MWIS approaches can also serve as approximations to the (weighted) maximum set packing problem. Finally, the proposed approaches might be used to solve similar problems (e.g., [Shi et al. 2004; Hedetniemi et al. 2013]) with some modifications.

REFERENCES

Noga Alon, László Babai, and Alon Itai. 1986. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *Journal of Algorithms* 7, 4 (Dec. 1986), 567–583.

Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286 (Oct. 1999), 509–512.

Stefano Basagni. 2001. Finding a Maximal Weighted Independent Set in Wireless Networks. *Telecommunication Systems* 18 (2001), 155–168.

Vittorio Bilò, Angelo Fanelli, Michele Flammini, and Luca Moscardelli. 2011. Graphical Congestion Games. *Algorithmica* 61 (2011), 274–297.

B. Bollobás. 1998. *Modern Graph Theory*. Springer-Verlag New York.

Brent N. Clark, Charles J. Colbourn, and David S. Johnson. 1990. Unit disk graphs. *Discrete Mathematics* 86, 1-3 (Dec. 1990), 165–177.

Johanne Cohen, Anurag Dasgupta, Sukumar Ghosh, and Sébastien Tixeuil. 2008. An Exercise in Selfish Stabilization. *ACM Trans. on Autonomous and Adaptive Systems* 3, 4 (Nov. 2008).

E. W. Dijkstra. 1974. Self-stabilizing systems in spite of distributed control. *Comm. ACM* 17, 11 (Nov. 1974), 643–644.

E. W. Dijkstra. 1975. Guarded Commands, Nondeterminacy, and Formal Derivation of Programs. *Comm. ACM* 18, 8 (Aug. 1975), 453–457.

P. Erdös and A. Rényi. 1959. On Random Graphs I. *Publications Mathematicae, Debrecen* 6 (1959), 290–297.

M. R. Garey and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York.

Wayne Goddard, Stephen T. Hededtniemi, David P. Jacobs, Pradip K. Srimani, and Zhenyu Xu. 2008. Self-Stabilizing Graph Protocols. *Parallel Process. Lett.* 18, 1 (2008), 189–199.

Wayne Goddard, Stephen T. Hedetniemi, David P. Jacobs, and Pradip K. Srimani. 2003. A Self-Stabilizing Distributed Algorithm for Minimal Total Domination in an Arbitrary System Graph. In *Proc. 17th Int'l Parallel and Distributed Processing Symp.*

Mohamed G. Gouda. 2001. The Theory of Weak Stabilization. In *Lecture Notes in Computer Science 2194*, A.K. Datta and T. Herman (Eds.). Springer-Verlag, 114–123.

M. G. Gouda and H. B. Acharya. 2011. Nash equilibria in stabilizing systems. *Theoretical Computer Science* 412 (2011), 4325–4335.

Daniel Grosu and Anthony T. Chronopoulos. 2004. Algorithmic Mechanism Design for Load Balancing in Distributed Systems. *IEEE Trans. on Systems, Man, and Cybernetics–Part B: Cybernetics* 34, 1 (Feb. 2004), 77–84.

Nabil Guellati and Hamamache Kheddouci. 2010. A survey on self-stabilizing algorithms for independence, domination, coloring, and matching in graphs. *J. Parallel Distrib. Comput.* 70 (2010), 406–415.

M. M. Halldórsson and J. Radhakrishnan. 1997. Greed is Good: Approximating Independent Sets in Sparse and Bounded-Degree Graphs. *Algorithmica* 18, 1 (1997), 145–163.

Joseph Y. Halpern. 2003. A computer scientist looks at game theory. *Games and Economic Behavior* 45 (2003), 114–131.

S. M. Hedetniemi, S.T. Hedetniemi, D. P. Jacobs, and P. K. Srimani. 2003. Self-Stabilizing Algorithms for Minimal Dominating Sets and Maximal Independent Sets. *Computers & Mathematics with Applications* 46, 5-6 (Sept. 2003), 805–811.

Stephen T. Hedetniemi, David P. Jacobs, and K.E. Kennedy. 2013. Linear-Time Self-Stabilizing Algorithms for Disjoint Independent Sets. *Comput. J.* 56, 11 (2013), 1381–1387.

R. Hekmat and P. Van Meighem. 2003. Degree Distribution and Hopcount in Wireless Ad-hoc Networks. In *Proc. 11th IEEE Int'l Conf. on Networks*. 603–609.

M. Ikeda, S. Kamei, and H. Kakugawa. 2002. A space-optimal self-stabilizing algorithm for the maximal independent set problem. In *Proc. 3rd Int'l Conf. on Parallel and Distributed Computing, Applications and Technologies*.

Lujun Jia, Rajmohan Rajaraman, and Torsten Suel. 2002. An Efficient Distributed Algorithm for Constructing Small Dominating Sets. *Distributed Computing* 15, 4 (2002), 193–205.

Hirotsugu Kakugawa and Toshimitsu Masuzawa. 2006. A Self-Stabilizing Minimal Dominating Set Algorithm with Safe Convergence. In *Int'l Parallel and Distributed Processing Symposium*.

S. Kamei and H. Kakugawa. 2003. A self-stabilizing algorithm for the distributed minimal k-redundant dominating set problem in tree network. In *Proc. 4th International Conference on Parallel and Distributed Computing, Applications and Technologies*.

S. Kamei and H. Kakugawa. 2005. A self-stabilizing approximation algorithm for the distributed minimum k-domination. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences* 5 (2005), 1109–1116.

Richard M. Karp and Avi Wigderson. 1985. A fast parallel algorithm for the maximal independent set problem. *J. ACM* 32, 4 (Oct. 1985), 762–773.

Michael Kearns, Siddharth Suri, and Nick Montfort. 2006. An Experimental Study of the Coloring Problem on Human Subject Networks. *Science* 313 (Aug. 2006), 824–827.

Micheal J. Kearns, Michael L. Littman, and Satinder P. Singh. 2001. Graphical models for game theory. In *Proc. 17th Conf. in Uncertainty in Artificial Intelligence*. 253–260.

Jason R. Marden, Behrouz Touri, Ragavendran Gopalakrishnan, and J. Patrick O'Brien. 2015. Impact of Information in a Simple Multiagent Collaborative Task. In *IEEE Conference on Decision and Control*. 4543–4548.

I. Milchtaich. 1996. Congestion games with player-specfic payoff functions. *Games and Economic Behavior* 13, 1 (1996), 111–124.

Dov Monderer and Lloyd S. Shapley. 1996. Potential Games. *Games and Economic Behavior* 14 (1996), 124–143.

Andrea Montanari and Amin Saberi. 2010. The spread of innovations in social networks. *Proc. Natl Acad. Sci. USA* 107 (2010), 20196–20201.

Noam Nisan and Amir Ronen. 2001. Algorithmic Mechanism Design. *Games & Economic Behavior* 35 (2001), 166–196.

R. W. Rosenthal. 1973. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory* 2, 1 (1973), 65–67.

Shuichi Sakai, Mitsunori Togasaki, and Koichi Yamazaki. 2003. A note on greedy algorithms for the maximum weighted independent set problem. *Discrete Applied Mathematics* 126 (2003), 313–322.

Reinhard Selten and Myrna H. Wooders. 2001. Cyclic Games: An Introduction and Some Examples. *Games & Economic Behavior* 34 (2001), 138–152.

Z. Shi, W. Goddard, and S. T. Hedetniemi. 2004. An anonymous self-stabilizing algorithm for 1-maximal independent set in trees. *Inform. Process. Lett.* 91, 2 (2004), 77–83.

Sandeep K. Shukla, Daniel J. Rosenkrantz, and S. S. Ravi. 1995. Observations on self-stabilizing graph algorithms for anonymous networks. In *Proc. 2nd Workshop on Self-Stabilizing Systems*.

R. E. Tarjan and A. E. Trojanowski. 1977. Finding a Maximum Independent Set. *SIAM J. Comput.* 6, 3 (1977), 537–546.

Cem Tekin, Mingyan Liu, Richard Southwell, Jianwei Huang, and Sahand H. A. Ahmad. 2012. Atomic Congestion Games on Graphs and its Applications in Networking. *IEEE/ACM Trans. on Networking* 20, 5 (Oct. 2012), 1541–1552.

Volker Turau. 2007. Linear self-stabilizing algorithms for the independent and dominating set problems using an unfair distributed scheduler. *Inform. Process. Lett.* 103, 3 (2007), 88–93.

Volker Turau. 2010. Self-Stabilizing Vertex Cover in Anonymous Networks with Optimal Approximation Ratio. *Parallel Processing Letters* 20, 2 (2010), 173–186.

Volker Turau and Bernd Hauck. 2011. A new Analysis of a Self-Stabilizing Maximum Weight Matching Algorithm with Approximation Ratio 2. *Theoretical Computer Science* 412, 40 (Sept. 2011), 5527–5540.

Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393 (June 1998), 440–442.

Z. Xu, S. T. Hedetniemi, W. Goddard, and P. K. Srimani. 2003. A synchronous self-stabilizing minimal domination protocol in an arbitrary network graph. In *Lecture Notes in Computer Science 2918*. Springer-Verlag, 26–32.

Li-Hsing Yen and Zong-Long Chen. 2014. Game-theoretic approach to self-stabilizing distributed formation of minimal multi-dominating sets. *IEEE Trans. Parallel Distrib. Syst.* 25, 12 (Dec. 2014), 3201–3210.

H. P. Young. 1993. The Evolution of Conventions. *Econometrica* 61, 1 (Jan. 1993), 57–84.

Appendix: Proof of Γ_{sym} being an exact potential game

To prove, we need the following lemma.

LEMMA 6.1. *For any player p_i and any strategy profile C, $u_i(C) = 0$ if $c_i = 0$.*

PROOF. If $c_i = 0$, then $w(c_i, c_j) = 0$ for all $p_j \in N_i$. Therefore, $u_i(C) = \sum_{p_j \in L_i} w(c_i, c_j) + c_i = 0$. □

THEOREM 6.2.

$$\Phi(C) = \frac{1}{2}\sum_{j=1}^{n}\sum_{p_k \in N_j} w(c_j, c_k) + \sum_{j=1}^{n} c_j$$

is an exact potential function for Γ_{sym}.

PROOF. Let $C = (c_i, c_{-i})$ and $C^* = (c_i^*, c_{-i})$ be two strategy profiles before and after process p_i changes its strategy from c_i to c_i^*, respectively. $\Phi(C)$ can be rephrased as

$$\Phi(C) = \frac{1}{2}(u_i(C) - c_i + \sum_{j \neq i}(u_j(C) - c_j)) + \sum_{j=1}^{n} c_j. \tag{14}$$

Similarly,

$$\Phi(C^*) = \frac{1}{2}(u_i(C^*) - c_i^* + \sum_{j \neq i}(u_j(C^*) - c_j)) + \sum_{j \neq i} c_j + c_i^*. \tag{15}$$

For any player $p_j \notin N_i \cup \{p_i\}$, $u_j(\cdot)$ is not affected by p_i's strategy change. Let $U_i = \{p_j | p_j \in N_i \wedge c_j = 1\}$. Lemma 6.1 yields $u_k(C) = u_k(C^*) = 0$ for all $p_k \in N_i \setminus U_i$. Therefore,

$$\begin{aligned}\Phi(C^*) - \Phi(C) &= \frac{1}{2}\left(u_i(C^*) - u_i(C) + c_i - c_i^*\right) \\ &+ \frac{1}{2}\sum_{p_j \in U_i}\left(u_j(C^*) - u_j(C)\right) + (c_i^* - c_i).\end{aligned} \tag{16}$$

Player p_i can make two possible transitions: $(c_i, c_i^*) = (0, 1)$ and $(c_i, c_i^*) = (1, 0)$.

Case 1. $(c_i, c_i^*) = (0, 1)$. This transition happens only if $c_j = 0$ for all $p_j \in N_i$ (otherwise $u_i(C^*) \leq -\alpha + 1 < 0$ and p_i would not have an incentive to set $c_i^* = 1$). So $U_i = \emptyset$. Furthermore, $w(c_i, c_j) = w(c_i^*, c_j) = 0$ for all $p_j \in N_i$ and thus

$$u_i(C) = \sum_{p_j \in N_i} w(c_i, c_j) + c_i = 0 \tag{17}$$

and

$$u_i(C^*) = \sum_{p_j \in N_i} w(c_i^*, c_j) + c_i^* = 1. \tag{18}$$

Therefore, (16) becomes

$$\begin{aligned}\Phi(C^*) - \Phi(C) &= c_i^* - c_i = 1 \\ &= u_i(C^*) - u_i(C).\end{aligned} \tag{19}$$

Case 2. $(c_i, c_i^*) = (1, 0)$. Because $u_i(C^*) = 0$ by Lemma 6.1, $u_i(C)$ must be negative. This implies that $U_i \neq \emptyset$. For all $p_j \in U_i$,

$$\begin{aligned}u_j(C^*) &= u_j(C) - w(c_j, c_i) + w(c_j, c_i^*) \\ &= u_j(C) + \alpha.\end{aligned} \tag{20}$$

Since

$$\begin{aligned}u_i(C) &= \sum_{p_j \in U_i} w(c_i, c_j) + \sum_{p_j \in N_i \setminus U_i} w(c_i, c_j) + c_i \\ &= \sum_{p_j \in U_i} (-\alpha) + c_i = -\alpha|U_i| + 1,\end{aligned} \tag{21}$$

we have $u_i(C^*) - u_i(C) = \alpha|U_i| - 1$. Now (16) becomes

$$\begin{aligned}\Phi(C^*) - \Phi(C) &= \frac{1}{2}(\alpha|U_i| - 1 + 1) + \frac{1}{2}\sum_{p_j \in U_i} \alpha + (-1) \\ &= \alpha|U_i| - 1 = u_i(C^*) - u_i(C).\end{aligned} \tag{22}$$

□

Since the strategy space of Γ_{sym} is finite, Γ_{sym} is a finite (exact) potential game.