IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 29, No. 2, April, 2024, pp. 581-590
Journal homepage: http://iieta.org/journals/isi

Self-Stabilizing Algorithms for Computing Maximal Distance-2 Independent Sets and Minimal Dominating Sets in Networks

Djamila Bouhata, Souheila Bouam, Hamouma Moumen*, Badreddine Benreguia, Chafik Arar

LAMIE Laboratory, Department of Computer Science, University of Batna 2, Batna 05078, Algeria

Corresponding Author Email: hamouma.moumen@univ-batna2.dz

https://doi.org/10.18280/isi.290219

Received: 29 May 2023
Revised: 4 September 2023
Accepted: 11 October 2023
Available online: 25 April 2024

Keywords:
self-stabilizing algorithm, expression model, maximal independent set, minimal dominating set, distributed system, network

ABSTRACT

This study devises self-stabilizing algorithms that leverage the expression distance-2 paradigm to compute (i) a maximal distance-2 independent set, wherein selected nodes maintain a separation exceeding two edges, ensuring non-adjacency; and (ii) a minimal dominating set, wherein each external node has at least one node as a neighbor in the dominating set. The efficacy and convergence of the algorithms are established through rigorous proofs within the framework of the expression model. Extensive simulation tests validate the algorithms' proficiency in selecting a minimal subset of nodes across expansive network topologies. These algorithms find practical applications in network operations, particularly in ad hoc and wireless sensor networks for the selection of cluster heads that facilitate critical services. Moreover, the self-stabilizing property of the algorithms guarantees the robust reconfiguration of cluster heads post-failure, thereby preserving network functionality amidst disruptions.

1. INTRODUCTION

Self-stabilization emerges as a pivotal fault tolerance paradigm within distributed systems and networks, facilitating convergence to a correct global state from an unspecified initial configuration. Such a self-stabilizing algorithm, operational without the necessity for external intervention, is capable of amending the distributed system's global state within a determinate timeframe. The literature is replete with a spectrum of self-stabilizing algorithms derived from graph theory. These algorithms address an assortment of computational problems including leader election, node coloring, and the provision of solutions to domination and independent set issues, as well as the construction of spanning trees. The practical significance of these algorithms is accentuated in their application to network architectures such as sensor and ad-hoc networks, where they are instrumental in defining cluster heads [1-3].

Let $G=(V, E)$ be a graph where V represents the node set and E is the edge set. The present work employs several variants of independent and dominating sets, which are frequently referenced throughout subsequent sections. The definitions of these terms are as follows:

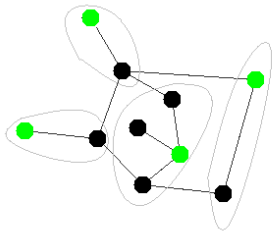
- **Independent Set:** is defined as a subset of nodes in V, denominated S, such that no two nodes within S share an edge.
- **Maximal Independent Set (MIS):** It is an independent set, denoted as S, for which no larger set exists that preserves independence. It is implicit that within MIS, the distance between any node pair in S exceeds one.
- **Maximal Distance-2 Independent Set (MD2IS):** This is a particular set of the independent set, S, wherein no node in S is reached by another node in S only through at least 3 edges. In MD2IS, there are no two independent nodes in S that could be reached through two edges.
- **Minimal Dominating Set (MDS):** is a subset of nodes, denoted as S, where every node in the graph is either included in S or is adjacent to at least one node of S. The dominating set is considered minimal if no smaller subset of S can satisfy the conditions of being a dominating set.

Figure 1 serves as an illustrative guide for conceptualizing the Maximal Independent Set, Maximal Distance-2 Independent Set, and Minimal Dominating Set. In this figure, nodes delineated in green represent members of the respective set. It is noteworthy that, as depicted in Figure 1(a), the MIS showcases a pairwise node distance of at least two edges. In contrast, Figure 1(b) reveals that within the MD2IS, the inter-node distance is strictly greater than two edges. In the case of MDS, any node not inside the MDS (represented with black) is at least adjacent to one member of the MDS, thereby satisfying the domination property.

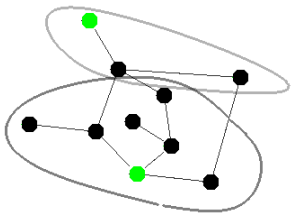
Historically, the pursuit of solutions for the MDS and MIS conundrums has been vigorous within the realm of self-stabilization research. Traditional approaches predominantly employ the distance-one model, which permits interactions solely between immediate neighboring nodes. In a novel departure, the distance-2 model for self-stabilization-introduced by Turau [4] extends this interaction to nodes within a two-hop radius. This study seeks to highlight the capacity of the distance-2 model to underpin the development of novel self-stabilizing algorithms. The abstraction inherent in the distance-2 model paves the way for the forthcoming design of algorithms targeting both MD2IS and MDS.

However, the application of such algorithms within the distance-1 paradigm necessitates the utilization of a transformer. Despite the ease of algorithm design afforded by the distance-2 model, direct application is untenable; hence, transformation to the executable distance-1 model is imperative. The transformer introduced by Turau [4] will be used in this research.

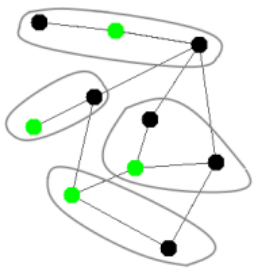
In practical scenarios, the implementation of MIS, MD2IS, and MDS finds resonance in ad hoc and wireless sensor networks. Here, nodes within these sets assume the role of servers or cluster heads, providing a suite of services to adjacent nodes, designated as cluster members. These services encompass but are not limited to routing data, key distribution for encryption, and management of member identity. Given the propensity for failures in ad hoc and, particularly, sensor networks (often attributed to battery limitations) the indispensability of self-stabilizing algorithms becomes pronounced. Such algorithms are integral to the restoration of cluster headsets, thereby ensuring network resilience and operational continuity.



(a) MIS: Maximal Independent Set



(b) MD2IS: Maximal Distance-2 Independent Set



(c) MDS: Minimal Dominating Set

Figure 1. Three variants of independent and dominating sets composed of green nodes

1.1 Paper organization

The structure of the paper is methodically divided into seven sections for clarity and coherence. Section 2 discusses the prior art and delineates the contributions of the current study. Section 3 presents the model and the terminological foundation used in self-stabilization. The algorithms proposed for identifying the Maximal Independent Set at a distance-2 and then for the computation of the Minimal Dominating Set using the distance-2 model are introduced in Sections 4 and 5. In section 6, we illustrate the simulation tests to demonstrate the efficiency of the proposed algorithms. Lastly, in Section 7, the conclusion summarizes the main points presented in this paper.

2. RELATED WORK AND PAPER CONTRIBUTION

Shukla et al. [5] have introduced a self-stabilizing algorithm to compute the maximal independent set. Their approach is based on two simple rules: (1) a node v has to enter into the set S if v has no neighbor within S, (2) and each node inside S has to exit S if it has a neighbor that belongs to S to preserve independence. Hedetniemi et al. [6] have presented a self-stabilizing algorithm to calculate the minimal dominating set. Subsequently, numerous algorithms have emerged, using additional domination parameters such as total domination [7, 8], efficient domination [9, 10], connected dominating set [11, 12], influence domination [13, 14], and distance-k domination [15].

Other algorithms to address specific challenges of independent sets have appeared, aiming to minimize algorithmic complexity [16] or to align with distributed daemon environments [7, 17]. A detailed study of existing algorithms in the self-stabilization context is provided in the survey [18], particularly for independent and dominating sets. Moreover, additional constraints beyond independence are imposed in certain algorithms for independent sets. For instance, Neggazi et al. [19] have proposed an algorithm aimed at identifying a maximal independent set S, wherein each node u outside of S must have a neighbor v in S, satisfying $degree(v)>degree(u)$. All the mentioned above algorithms are developed under the habitual distance-1 model.

Arapoglu and Dagdeviren [2] introduced an algorithm for Maximal Independent Sets (MIS), using distance-2 information. This algorithm, designed to compute MIS within a distributed daemon, was specifically designed for wireless sensor networks to minimize energy consumption. Under the distance-2 model, nodes allocate 50% of their transmission power for data transfer within distance-1, while utilizing 100% of their transmission power for propagating information across distance-2. Employing the message-passing model, every node could diffuse information to neighbors within a distance-2 only when its state changes.

Johnen [20, 21] has introduced a self-stabilizing algorithm aimed at identifying an independent set S where any two nodes can only reached through $k+1$ or more edges. The algorithm presented in the study by Johnen [21] marks an enhancement in memory management compared to the study [20]. In a separate work by Datta et al. [15], an algorithm is proposed to determine a dominating set S, (not necessarily independent) where every node outside of S remains reached at a distance of at most k from S. Despite the bounded complexity of $O(n+k)$ in rounds for the algorithms, the authors highlight the possibility of non-convergence under a distributed daemon, as an enabled node could be overlooked. In the literature on self-stabilization, a finite round complexity does not guarantee convergence [15]. Hence, determining the convergence time remains an unresolved issue [15, 20, 21] to compute the independence between nodes in the general case of distance $k\geq 2$. In this paper where $k=2$, we show that our self-stabilizing algorithm converges during a period of finite time.

2.1 Contribution

Using the expression distance-2 model, this paper proposes two new self-stabilizing algorithms called MD2IS and MDSD2. The first algorithm MD2IS aims to find the maximal independent set *S* where any pair of nodes in *S* are only linked through 3 or more edges. The second algorithm MDSD2 seeks to find the minimal dominating set using the expression model. The proofs of the correctness and convergence of the proposed algorithms are formally presented. The main goal is to show the power of the expression distance-2 model to facilitate the design of new self-stabilizing algorithms.

The complexity of the proposed algorithms is *O(n)* to reach the legitimate configuration, assuming using the expression model and the central daemon. The conversion of the algorithms into the distance-one model provides an equivalent algorithm that converges to the correct configuration in *O(nm)* moves under a distributed daemon, where *m* is the number of edges. Simulation tests confirm the correctness and convergence of the proposed algorithms as has been illustrated in the proofs. Tests provide small-reduced independent dominating sets, which is considered as a valuable result for many applications in the case of large-scale networks.

3. MODEL AND TERMINOLOGY

Distributed systems are typically represented as graphs, denoted as *G(V, E)*. The set *V* of *vertices,* called also *nodes,* represents the processing units of the distributed system. The connections between units are represented by the set of *links E,* called also *edges*. Nodes u which shares a link with $v \in V$ is the set of *v neighbors* defined as $N(v) = \{u \in V: vu \in E\}$. Two nodes v, u are considered *adjacent* if $u \in N(v)$.

We define the concept of the neighborhood at distance-2, denoted as $N(v)_{dist2} = N(v) \cup \{w \in V: \exists u \in N(v): w \in N(u)\}$. Consequently, all neighbors of *v* at distance 2 are either at a distance of 1 or 2 from *v*.

A subset S in V is said *independent* if there are no adjacent nodes in S, or there are no pair of nodes in S that are linked by an edge. Similarly, a subset S is called *distance-k independent* if there are no two nodes in S that are linked by k (or less) edges. Any pair of nodes in S are connected only through $k+1$ (or more) edges [20]. Thus, we call a set S *distance-2 independent*, if any two nodes in S are reached only through a number of edges that is strictly greater than 2. It is worth noting that in the Maximal Independent Set (MIS), every node *v* is either *independent* if *v* is inside the MIS, or *dominated* at least by an independent node *u* if *v* is outside the MIS [4]. In the literature, the MIS is also known as the independent dominating set.

Any node u outside of the maximal distance-2 independent set, called MD2IS, is:

-either, *u* has at least an adjacent node $v \in MD2IS$, (hence, *u* is dominated),

-or u is dominated (at distance-2) by a node $v \in MD2IS$ where *u* could reach *v* only through 2 edges.

Note that multiple nodes from the MD2IS set could dominate a node *u* outside the MD2IS set. Figure 1(b) illustrates an example of a node *u* that could be dominated by a couple of nodes v_1 and v_2. The distance between *u* and v_1 is 1 and the distance between *u* and v_2 is 2.

Definition 1: *In a graph G(V, E), a subset S in V is said a* ***distance-2 independent set*** *if and only if any two nodes in S are reached only through a number of edges that is greater or equal to 3. S is maximal if there is no other set S' that contains S such that S' is a distance-2 independent set.*

Definition 2: *In a graph G(V, E), a* ***dominating set*** *is a subset S in V where every node u outside of S has at least an adjacent node v inside S.*

In the self-stabilization context, an algorithm needs to (1) attain a globally *legitimate* configuration (2) within a finite period called *convergence*. Upon reaching the legitimate configuration, the algorithm must remain within the legitimate configuration and cannot move again outside the legitimate configuration. This condition is referred to, in the literature, as *closure*. Consequently, to demonstrate the self-stabilizing nature of an algorithm, it is necessary to show the *closure* and the *convergence.* The *closure* aims to illustrate that the reached legitimate configuration remains legitimate without any risk of moving again to an illegitimate state. The *convergence* is the certainty of achieving a legitimate configuration in a finite period.

In a self-stabilizing system, each node has a number of rules: R1, R2, … Ri. For every rule, written generally as *guard→statement,* the guard is checked using an infinite loop. The statement can be executed if the guard is true. If all nodes execute the same rules i.e. the same code, the distributed system is known as *uniform* in contrary to other systems where some nodes could execute a different code such as semi-uniform systems wherein one node has a different code. Generally, local variables are used at the level of each node to define the node *state*. The local *state* can be read and modified only by the local owner. Adjacent nodes can read the state of the local node, but cannot write in the local variable of their neighbor node.

Besides a local variable, the expression model proposes that every node can use (read/write) a local expression *exp* to have a partial view in the neighborhood at distance-1. Since any other node can read (but not write) the expression *exp* of its neighbors, it reaches in fact, information at distance 2. Thus, the expression model is executed based on information obtained at distance 2.

The *guard* consists of Boolean conditions using variables and/or expressions of the local node or the neighbor nodes. During the infinite check loop, when a *guard* is true, its corresponding rule is known as *enabled* or *privileged*. Thus, a node is *enabled* or *privileged* if there is an enabled rule among its set of rules. The execution of a statement leads to a change in the local state and consequently, to a new configuration of the global configuration. This transition through an enabled node is known in the literature as a *move*. However, the execution of a move is only allowed after getting the permission of a *scheduler* or a *daemon*. The algorithm is self-stabilizing if the sequence of executed moves is finite (whatever serial or distributed) leading certainly to a legitimate configuration.

3.1 Daemon notion

Before starting to design a new self-stabilizing algorithm, an important hypothesis must be assumed whether the algorithm is executed in a parallel or sequential way. Despite the parallel execution seems more efficient, it is difficult to formally show the correctness and the convergence under the distributed execution compared to the sequential execution.

In this direction, the *daemon concept*, known also as a *scheduler*, comes to provide researchers with an abstract tool

to make a suitable hypothesis according to the context in which is facing.

In a *central* daemon, it is supposed that privileged nodes are executed one by one in a sequential order determined by the daemon which is also referred to as *serial* daemon. Conversely, in a *distributed* daemon, the daemon may select many privileged nodes to execute a move simultaneously, forming a *round*. In the case the daemon allows all privileged nodes to execute a move simultaneously, this scheduler is said *synchronous*.

3.2 Transformers

A commonly employed method, as discussed in the work of Turau et al. [4, 22], involves the conversion of a self-stabilizing algorithm A, which operates within specific assumptions, into a new algorithm A^T, that runs under other assumptions. Despite this transformation, both algorithms are equivalent and ensure to achieve the same final configuration. Various types of transformers, such as *distance* transformers and *daemon* transformers, can be found in the literature. However, the transformation typically introduces additional time complexity by algorithm A^T to reach the final correct configuration.

In this paper, we adopt the transformer of Turau [4], to convert the initially proposed algorithm A under the expression model and a central daemon to obtain a distributed algorithm A^T. The expression distance-2 model provides an abstraction on the proposed algorithm A which facilitates proving the correctness and convergence. On the contrary, the design of new algorithms is complicated under the distance-1 context due to the details presented in this case. However, the hypotheses, on which the algorithm A is supposed to run, are either not feasible or not efficient for the execution of A. For instance, self-stabilizing algorithms cannot use directly distance-2 in reality and need to use gradually distance-1. Since distributed execution is more efficient than serial execution in terms of time of convergence, the transformation is an imperative necessity for an algorithm developed under the expression model.

3.3 Our execution model

We assume first that our algorithm is executing with the expression model and a central scheduler. The transformer is adopted then to allow our algorithm to function under other hypotheses.

4. SELF-STABILIZING ALGORITHM MD2IS

Algorithm 1 is a uniform self-stabilizing algorithm designed to identify the maximal distance-2 independent set MD2IS. Every node v maintains a local variable *state* and an expression *exp*. The *state* variable can hold either *In* or *Out* as values to indicate whether node v is inside or outside the set MD2IS. The *exp* expression helps to count the number of neighbors that are inside MD2IS. Once the legitimate configuration is reached, all nodes are disabled, and the set $S = \{v \in V : v.state = In\}$ is the maximal distance-2 independent set.

During the illegitimate configuration, enabled nodes execute serial moves until the system converges to the correct global configuration. After each move, a randomly enabled node is chosen by the central daemon to execute a new move.

Algorithm 1. Maximal Distance-2 Independent Set – MD2IS

(1) $v.exp:: |\{u \in N(v): u.state = In\}|$
(2) **R1**: $v.state = Out \wedge \forall u \in N(v): (u.state = Out \wedge u.exp = 0) \rightarrow v.state = In$
(3) **R2**: $v.state = In \wedge \exists u \in N(v): (u.state = In \vee u.exp > 1) \rightarrow v.state = Out$

Each node checks its state and its neighborhood by utilizing *state* and *exp*. The expression *exp* is specifically employed to compute the number of neighbors belonging to the set S. The expression model facilitates the detection of neighbors that are inside S within a distance of 2. When a node accesses to expressions of its neighbors, it gets information about the neighbors located at a distance of 2.

In the MD2IS algorithm, the first rule $R1$ allows the local node to move from outside to inside S, however $R2$ moves nodes from inside to outside S. Given a node v outside of S, $R1$ examines the states and the expressions of neighbor nodes. If the *state* of all neighbors is *Out* and the expression of all neighbors is equal to 0, then v moves from outside to inside S. When a node v is inside S, $R2$ checks if there exists at least a neighbor u with either $u.state = In$ or $u.exp > 1$ (u is dominated at least by two nodes: v and another node), thus v needs to move from inside to outside S due to the existence of a neighbor u within a distance of 2 inside S.

The execution of $R1$ guarantees that the independence property within distance 2 remains respected after moving a new node v from outside to inside S. However, if a given node v violated the independency condition, the execution of $R2$ removes v from S to outside.

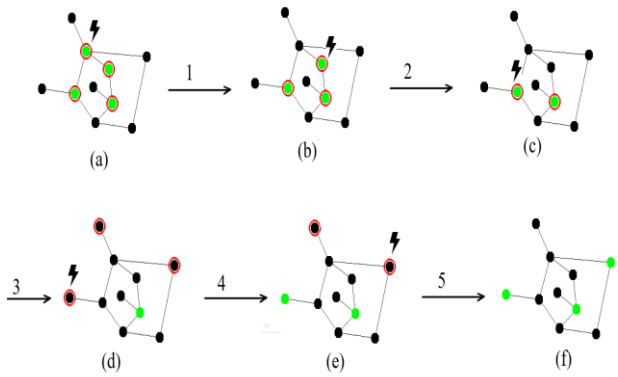


Figure 2. Convergence to the final legitimate MD2IS configuration

Figure 2 illustrates how algorithm MD2IS reaches the legitimate configuration after it starts from a random initial configuration. Black nodes are outside S while green nodes denoted elements inside S. If nodes are surrounded with red circles, the system is in an illegitimate configuration. Nodes surrounded in red color represent privileged nodes that are capable of executing a move. Using a central daemon, at each step, the daemon selects only one privileged node to carry out a move. The MD2IS set is achieved after five moves according to the sequence of rules: R2, R2, R2, R1, R1. Once no node (surrounded with red color) exists, the graph is considered in the final legitimate configuration. Note that, certain moves, such as the transition from configuration c to configuration d, may activate additional nodes in subsequent configurations.

4.1 Closure

Lemma 1 *The set S={v∈V, v.state=In} is maximal distance-2 independent when there is no activate node.*

Proof When the system reaches the final legitimate configuration, all the nodes are inactivated. Thus, no node can execute *R2* because the guard of *R2* is false. Hence, for each node *v* in *S*, all neighbors *u* of *v*: *(u.state=Out ∧ u.exp ≤ 1)*. Thus, all neighbor nodes *u* are outside of *S*, and within a distance of 2 from *v*, there is no node inside *S* because *u.exp≤1*, (When *u.exp*=1, only *v* is in *S* since *u* has only *v* as a neighbor in *S*).

We prove the maximality by contradiction assuming that *S* is not yet maximal. Suppose in the legitimate configuration where all the nodes are inactivated, we add into the set *S* a node *v* (with *v.state=Out*) and *S* remains independent. However, since *R1* is inactivated, *v* has at least one neighbor in *S* within distance 2. This is a contradiction to the distance-2 independence property.

4.2 Convergence

Lemma 2 *When node v executes R1, then:*
v becomes independent and remains independent because every neighbor node within distance-2 remains outside of S and cannot be activated.

Proof Suppose a node *v* has executed *R1*. Thus, all neighbor nodes *u* within a distance of 2 from *v* are outside of *S*. Consequently, none of *u* nodes could enable *R1* because there is at least node *v* in *S* as a neighbor of *u* within a distance of 2. Therefore, *v* remains in *S*, and all neighbors within a distance of 2 still be outside of *S*.

Lemma 3 *Any node in V will be activated at most twice by R2 followed by R1. Consequently, MD2IS terminates at most after 2n moves.*

Proof As demonstrated by Lemma 2, once every node executes *R1*, it remains stationary and cannot move again. This implies that, in the worst-case scenario, each node could be activated solely by *R2* followed by *R1*. Hence, *n* nodes need a maximum of *2n* moves to reach the legitimate configuration.

Theorem 1 *MD2IS is a self-stabilizing algorithm that computes the Maximal Distance-2 Independent Set in a linear time complexity of O(n) moves, using the expression model under the central daemon.*

Proof The proof is followed by Lemma 1 and Lemma 3.

Although the distance-2 model simplifies the creation of new self-stabilizing algorithms, the developed algorithm cannot be applied directly. Instead, it must be converted into an executable distance-1 model. This transformer suggests converting any self-stabilizing algorithm *A* operating under a central daemon, to an equivalent algorithm A^T that executes under a distributed daemon. The latter algorithm is more efficient in real contexts. The transformer converts also the theoretical distance-2 model to a real applicable distance-1 model.

Theorem 2 *Using a distributed daemon with a distance-1 model, MD2IS^D reaches the legitimate configuration of the Maximal Distance-2 Independent Set in O(nm) moves.*

Proof The proof is followed from Theorem 18 of the paper of Turau [4], where *m* represents the number of edges.

5. SELF-STABILIZING ALGORITHM MDSD2

In the literature, Hedetniemi et al. [6] have introduced an algorithm for MDS based on using *In* and *Out* states. To converge to the legitimate configuration, nodes move from *In* to *Out* or from *Out* to *In* until reaching stabilization. The authors use also a *pointer* for nodes out of *S* to determine nodes that have exactly one neighbor inside *S*. For nodes having more than a neighbor in *S*, the pointer is null. Figure 3 illustrates this additional process of pointing (node out *S* points its unique neighbor in *S*) in moves 2 and 3. This section aims to show how the expression model could simplify designing a new algorithm in the self-stabilization context. The expression model has the power to make abstraction on distance-2 in contrast to the habitual distance-1 model. It allows masking details like the above-mentioned process of pointing which is considered as an additional complication in proposing new algorithms. Using the expression distance-2 model, we propose an algorithm called MDSD2 to compute the minimal dominating set.

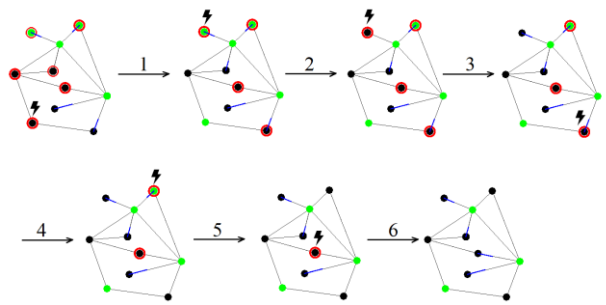


Figure 3. Convergence to the MDS using Hedetniemi algorithm [7]

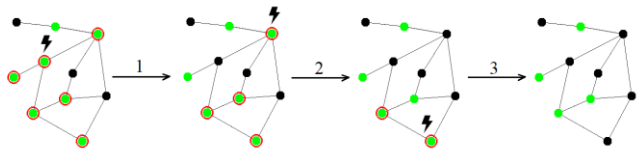


Figure 4. Convergence to the MDS using expression distance-2 model

In Algorithm 2, each node *v* includes a local variable *state* and an expression *exp*. The *state* value can be either "*In*" or "*Out*", indicating whether a node is inside the dominating set. Once the system reaches a legitimate state, the set *S={v∈V: v.state=In}* represents the dominating set. For the local expression *exp,* it calculates the number of neighbors in *S* of the current node. It helps to determine whether every node outside of *S* is dominated. Rule *R1* guarantees that every node outside *S* must be dominated, otherwise, it moves any node outside of *S* that is not dominated from *Out* to *In*. In contrast, *R2* aims to minimize the set *S* as small as possible. If a given

node v inside S exits S so that S remains a dominating set, then *R2* forces v to leave S. Note that before a node v decides to exit S, *exp* is used to ensure that all neighbors $u \in N_{V-S}(v)$ maintain the domination condition after v's departure. If it is determined that after v switches from *In* to *Out*, at least one neighbor $w \in N_{V-S}(v)$ would become undominated, and v cannot exit S. Figure 4 illustrates an example of the algorithm's execution converging to the minimal dominating set within 3 moves.

Algorithm 2. Minimal Dominating Set - MDSD2

(1) $v.exp : |\{u \in N(v): u.state = In\}|$
(2) **R1**: $v.state = Out \wedge v.exp < 1 \rightarrow v.state = In$
(3) **R2**: $v.state = In \wedge v.exp \geq 1 \wedge \forall u \in N_{V-S}(v): u.exp > 1 \rightarrow v.state = Out$

5.1 Closure

Lemma 4 *The set S is a minimal dominating set when all nodes are inactivated.*

Proof We demonstrate that (a) S is a dominating set and (b) S is minimal.

(a) Given that $R1$ is inactivated, it follows that for any node v outside of S, the local expression $v.exp \geq 1$. Consequently, every node v outside of S is dominated. Thus, S is a dominating set.

(b) We show that S is minimal by contradiction. Let's assume that all system nodes are inactive, and there exists a dominating set S_0 which is a subset of S such that $S_0 = S - \{v\}$. Consequently, $|N_{S0}(v)| \geq 1$ (since S_0 is a dominating set), implying that $|N_S(v)| \geq 1$ due to $N_{S0}(v) = N_S(v)$. Since none of the S nodes is activated ($R2$ is not enabled for v) and $v.exp \geq 1$, there exists a node u neighbor of v outside of S such that $u.exp \leq 1$, leading to $|N_S(u)| \leq 1$. Also, since v is a neighbor of u and $S_0 = S - \{v\}$ thus $|N_{S0}(u)| = |N_S(u)| - 1$. Hence $|N_{S0}(u)| < 1$ due to $|N_S(u)| \leq 1$. This is a contradiction because every node outside of S_0 must be dominated.

5.2 Convergence and Complexity Analysis

Lemma 5 *For every node w outside of S which is dominated (w.exp≥ 1), the value w.exp remains greater than or equal to* 1 *and cannot decrease below* 1.

Proof Consider a node w outside of S with $u.exp \geq 1$. The value of *u.exp* may decrease if and only if any neighbor node v moves from inside to outside S. According to rule *R2*, when node v switches from inside to outside S, all neighbors u of v that are outside S have $u.exp > 1$. Let's denote S_2 as the new set of nodes with *state=In* after v is removed from S, therefore, $|N_{S2}(u)| = |N_S(u) - 1|$. Consequently, $|N_S(u) - 1| \geq 1$ implies $|N_{S2}(u)| \geq 1$, ensuring that *u.exp* remains ≥1 through recurrence.

Lemma 6 *Once a node exits set S, it cannot return to S thereafter.*

Proof As any node exits S *has exp* ≥1, *R1* cannot be executed. Lemma 5 indicates that Rule *R1* cannot be triggered again.

Lemma 7 *Using the expression distance-2 model with the central daemon, if each node executes R1 followed by R2, algorithm 2 terminates in the worst case by executing* 2*n moves.*

Proof The proof is evident according to Lemma 5 and Lemma 6.

Theorem 3 *Using a central daemon and under the expression model, the self-stabilizing algorithm MDSD2 reaches the minimal dominating set in O*(*n*) *moves.*

Proof According to Lemma 4 and Lemma 7, the proof is adopted.

In the following theorem, we employ the transformer of Turau [4] that provides an alternative self-stabilizing algorithm MDSD2^D which executes under a distributed daemon with the habitual distance-one model.

Theorem 4 *Using a distributed daemon and under the distance-1 model, the self-stabilizing algorithm MDSD2^D computes the minimal dominating set and converges to the legitimate configuration within O(nm) moves.*

Proof The proof is adopted according to Theorem 18 of Turau [4], where m is the number of edges in the graph.

6. SIMULATION RESULTS

In this section, the simulation tests are presented to evaluate the proposed algorithms MD2IS and MDSD2. Evaluations are performed according to two metrics: (1) *cardinality* (the number of elements) of the independent and/or dominating set, (2) and *convergence* (the number of moves) to reach the legitimate configuration. A self-stabilizing algorithm is more preferment if it has a smaller cardinality and converges sooner to the legitimate configuration. The convergence metric reflects the communication complexity between nodes in the network.

The algorithms MD2IS and MDSD2 have been written in Java using an expression model. For MIS [7] and MDS [6], the implementation of Kuszner [23] has been reutilized. The graphs have been generated randomly with different densities and sizes using the *Erdos-Renyi* model [24], called also *random networks*. To control the number of edges, a parameter of density is changed from 0 to 1. The more the density parameter grows, the more the number of edges increases. When the density parameter equals 1, the graph is fully connected. The generated graphs varied from 500 nodes to 20000 nodes. For each size of graphs, we have carried out from 5 to 10 executions. The average value has been considered after the algorithm converges to the legitimate configuration.

Figure 5 illustrates the size variation of MD2IS and MIS according to graph density. MD2IS produces smaller independent sets compared to those generated by MIS. The density of the graphs significantly influences the size of the obtained sets. As shown in Figure 5, when the graph density increases, the size of the independent sets tends to decrease. For instance, the number of nodes of the MD2IS set approaches 1 as the density exceeds 0.5. It is worth noting that the MD2IS set includes only one node for the fully connected graphs (density=1).

Figure 6 depicts the size variation of MD2IS and MIS sets according to the number of nodes of the overall graph. The

curves demonstrate that the maximal independent set (MIS) grows proportionally with graph size. Conversely, the cardinality of MD2IS decreases as graph size increases. For instance, in graphs comprising 10000 nodes, the MIS cardinality exceeds 100, while the MD2IS cardinality remains below 10 nodes.

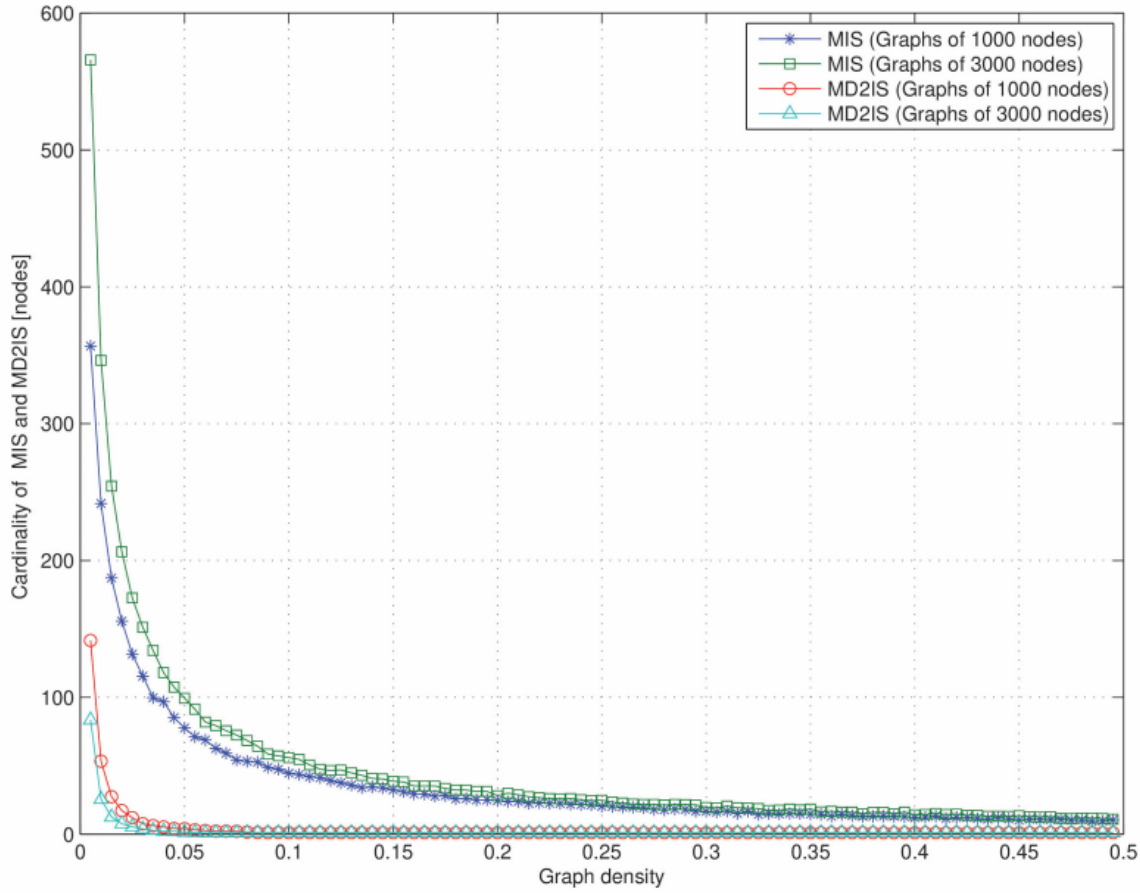


Figure 5. Number of nodes inside MD2IS and MIS according to graph density

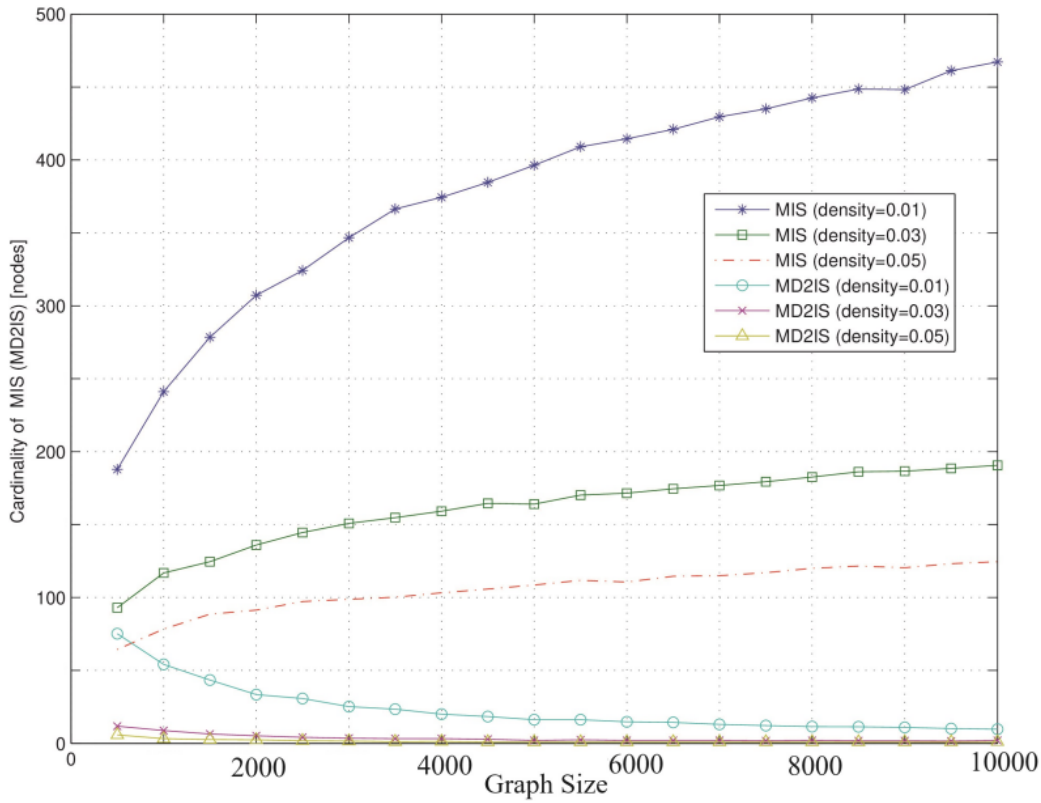


Figure 6. Number of nodes inside independent sets according to the overall size of the graphs

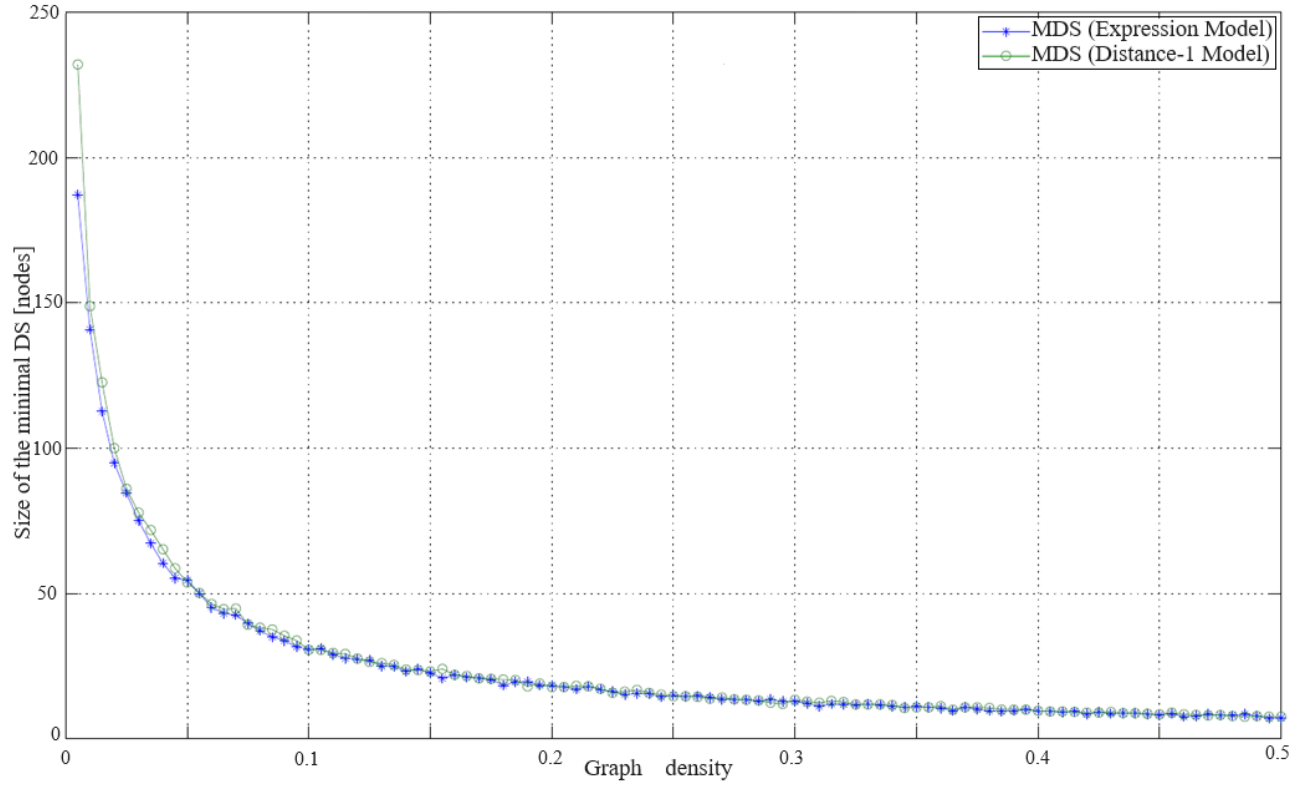


Figure 7. Number of nodes inside MDSD2 according to graph density

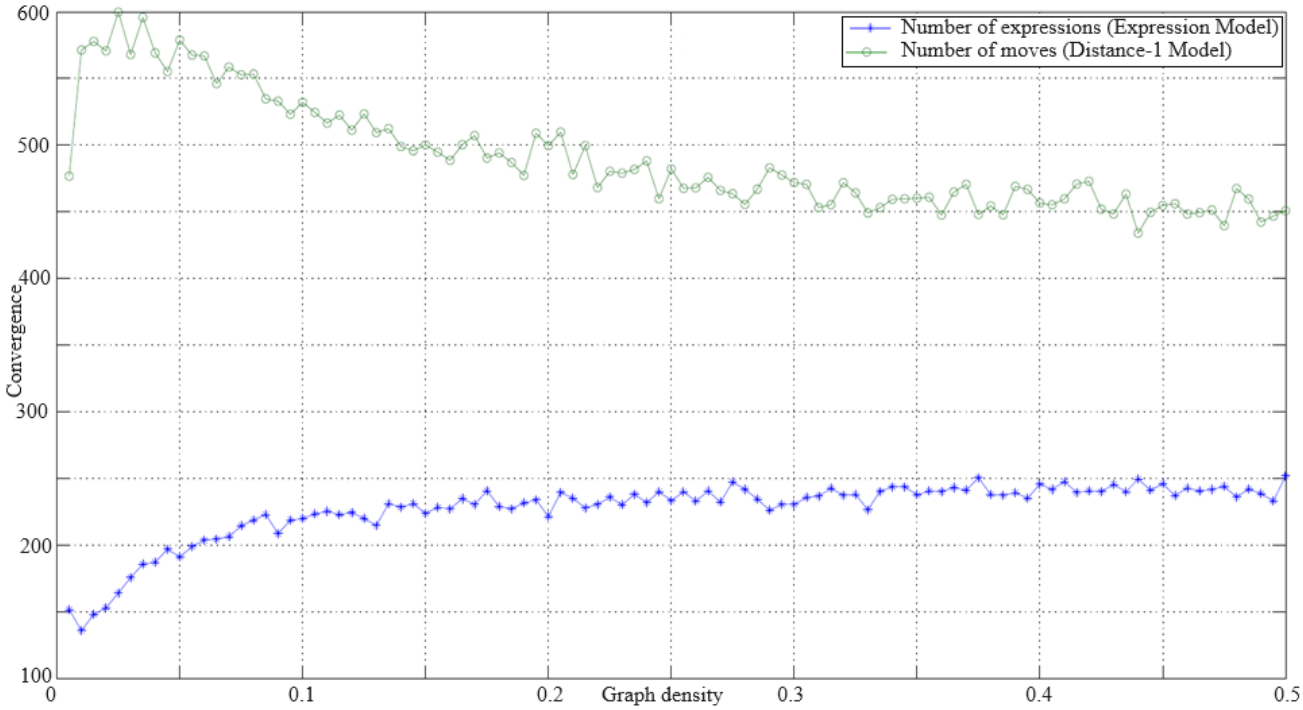


Figure 8. The convergence in computing MDS according to graph density (for graphs with 500 nodes)

Additional simulation results are presented in Table 1. The experiments utilize arbitrary graphs with a low density to be closer to modeling the real existing networks. The generated graphs on which simulations are executed can contain from 1000 nodes up to 20000 nodes. To measure the size of MD2IS/MIS and the number of moves to converge to MD2IS/MIS, average values are considered after executing 5 tests. The number of moves to reach the legitimate configuration is proportional to the size of the graphs. According to Lemma 3, MDSIS is achieved after 2n moves in the worst case, This upper bound is confirmed by the conducted tests even though our results show that n/2 is sufficient to compute MD2IS.

Figure 7 shows the size of the dominating set for the two algorithms of expression model MDSD2 and the distance-1 model MDS of Hedetniemi [6]. Although the two algorithms provide approximately the same size, recall that we aim to show that our proposed algorithm using the expression

distance-2 model has been designed in a straightforward way regarding the distance-1 algorithm which is written using complicated rules based on pointers [6].

Figure 8 illustrates that both MDS algorithms converge to the legitimate configuration in a limited duration of time. Note that the time of convergence is calculated with the number of moves for the distance-1 model, while for the expression distance-2 model, the number of expressions is considered.

Table1. Size of MD2IS and convergence (in moves) for graphs with a density of 0.001

Graph Size	MD2IS Cardinality	MIS Cardinality	MD2IS Convergence	MIS Convergence
1000	601.8(60.18%)	687.8(68.78%)	400.2	415.6
1500	703.6(46.91%)	926.4(61.76%)	652.8	636.6
2000	758.2(37.91%)	1099.4(54.97%)	910.8	845.4
2500	794.4(31.78%)	1248.2(49.93%)	1229.2	1078.4
3000	786.6(26.22%)	1385.0(46.17%)	1570.8	1327.4
3500	776.4(22.18%)	1522.4(43.50%)	1877.2	1624.6
4000	762.2(19.06%)	1614.4(40.36%)	2202.8	1887.4
4500	748.8(16.64%)	1720.4(38.23%)	2483.6	2206.0
5000	724.2(14.48%)	1801.0(36.02%)	2785.8	2454.8
5500	711.0(12.93%)	1887.0(34.31%)	3048.4	2749.2
6000	692.2(11.54%)	1964.4(32.74%)	3335.6	3061.6
6500	662.8(10.20%)	2020.8(31.09%)	3551.6	3324.6
7000	642.6(9.18%)	2111.8(30.17%)	3825.2	3691,4
7500	633.8(8.45%)	2149.8(28.66%)	4098.8	3927.2
8000	604.6(7.56%)	2216.6(27.71%)	4333.4	4229.2
8500	601.8(7.08%)	2269.2(26.70%)	4586.4	4542.8
9000	577.6(6.42%)	2320.4(25.78%)	4836.0	4830.8
9500	552.8(5.82%)	2364.6(24.89%)	5097.2	5152.4
10000	548.4(5.48%)	2415.2(24.15%)	5350.8	5379.2
10500	527.4(5.02%)	2476.2(23.58%)	5645.0	5728.4
11000	501.6(4.56%)	2507.4(22.79%)	5817.6	6063.2
11500	498.2(4.33%)	2560.4(22.26%)	6091.0	6303.6
12000	481.2(4.01%)	2602.2(21.69%)	6300.6	6622.0
12500	477.8(3.82%)	2635.2(21.08%)	6555.8	6923.8
13000	454.6(3.50%)	2665.6(20.50%)	6790.0	7195.0
13500	448.0(3.32%)	2705.0(20.04%)	7057.6	7496.6
14000	435.2(3.11%)	2747.8(19.63%)	7251.6	7758.6
14500	427.4(2.95%)	2785.0(19.21%)	7512.0	8045.0
15000	416.2(2.77%)	2793.0(18.62%)	7798.4	8234.0
15500	408.2(2.63%)	2839.4(18.32%)	7988.8	8579.8
16000	398.8(2.49%)	2867.4(17.92%)	8293.6	8857.8
16500	392.0(2.38%)	2886.6(17.49%)	8496.2	9177.6
17000	383.4(2.26%)	2934.2(17.26%)	8798.4	9433.6
17500	380.2(2.17%)	2962.6(16.93%)	9043.4	9704.0
18000	367.6(2.04%)	2976.6(16.54%)	9249.4	9972.0
18500	361.0(1.95%)	3017.8(16.31%)	9481.0	10284.0
19000	355.2(1.87%)	3036.6(15.98%)	9748.6	10535.8
19500	353.0(1.81%)	3054.8(15.67%)	9998.8	10785.0
20000	340.4(1.70%)	3080.2(15.40%)	10207.8	11032.4

7. CONCLUSION

Two new self-stabilizing algorithms have been presented to compute the maximal distance-2 independent set and the minimal dominating set, called MD2IS and MDSD2, respectively. The algorithms were proposed very smoothly through the expression distance-2 model. Thus, in this paper, we illustrated the ability of the expression model of Turau to simplify the design of new algorithms in the self-stabilizing paradigm. However, the introduced algorithms under the expression model cannot be applied directly and need to be converted into the real executable distance-1 model.

The expression model of Turau allows to design of new algorithms with a certain level of abstraction that hides unuseful algorithmic details. Under this abstraction, the correctness and convergence proofs become evident to be formulated. The proposed algorithms reach the legitimate configuration in *O(n)* moves with the expression model and under a serial execution. In the habitual distance-one model, the algorithms achieve the legitimate configuration in *O(nm)* moves under distributed execution.

Simulation results illustrate also the efficiency of the proposed algorithms. Our algorithms provide results similar to the existing algorithms for some cases such as the cardinality of minimal dominating set. In other cases, comparison cannot be considered due to the differentiation of criteria. For example, MIS and MD2IS provide different sets by definition which makes the comparison of cardinality insignificant in this case.

The obtained results reveal a reduced number of nodes for MD2IS set which is considered an important result in our study. The reduced number of MD2IS can be employed efficiently for a lot of real cases such as sensor wireless networks to select cluster heads. Thus, the use of MD2IS seems promising due to its self-stabilizing nature that allows the network to continue to operate even in the presence of faults with an efficient consumption of energy for nodes and cluster heads. In future works, we hope to study self-

stabilization with a more general distance-k and to test the efficiency of our algorithms for real applications.

REFERENCES

[1] Arapoglu, O., Akram, V.K., Dagdeviren, O. (2019). An energy-efficient, self-stabilizing and distributed algorithm for maximal independent set construction in wireless sensor networks. Computer Standards & Interfaces, 62: 32-42. https://doi.org/10.1016/j.csi.2018.07.004

[2] Arapoglu, O., Dagdeviren, O. (2019). An asynchronous self-stabilizing maximal independent set algorithm in wireless sensor networks using two-hop information. In 2019 International Symposium on Networks, Computers and Communications (ISNCC), Istanbul, Turkey, pp. 1-5. https://doi.org/10.1109/ISNCC.2019.8909189

[3] Bein, D., Datta, A.K., Jagganagari, C.R., Villain, V. (2005). A self-stabilizing link-cluster algorithm in mobile ad hoc networks. In 8th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'05), Las Vegas, NV, USA, pp. 6. https://doi.org/10.1109/ISPAN.2005.12

[4] Turau, V. (2012). Efficient transformation of distance-2 self-stabilizing algorithms. Journal of Parallel and Distributed Computing, 72(4): 603-612. https://doi.org/10.1016/j.jpdc.2011.12.008

[5] Shukla, S.K., Rosenkrantz, D.J., Ravi, S.S. (1995). Observations on self-stabilizing graph algorithms for anonymous networks. In Proceedings of the second workshop on self-stabilizing systems, 7: 1-15. https://doi.org/10.1109/ALLERTON.2016.7852307

[6] Hedetniemi, S.M., Hedetniemi, S.T., Jacobs, D.P., Srimani, P.K. (2003). Self-stabilizing algorithms for minimal dominating sets and maximal independent sets. Computers & Mathematics with Applications, 46(5-6): 805-811. https://doi.org/10.1016/S0898-1221(03)90143-X

[7] Goddard, W., Hedetniemi, S.T., Jacobs, D.P., Srimani, P.K. (2003). Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks. In Proceedings International Parallel and Distributed Processing Symposium, Nice, France, pp. 14. https://doi.org/10.1109/IPDPS.2003.1213302

[8] Belhoul, Y., Yahiaoui, S., Kheddouci, H. (2014). Efficient self-stabilizing algorithms for minimal total k-dominating sets in graphs. Information Processing Letters, 114(7): 339-343. https://doi.org/10.1016/j.ipl.2014.02.002

[9] Turau, V. (2013). Self-stabilizing algorithms for efficient sets of graphs and trees. Information Processing Letters, 113(19-21): 771-776. https://doi.org/10.1016/j.ipl.2013.07.008

[10] Hedetniemi, S.M., Hedetniemi, S.T., Jiang, H., Kennedy, K.E., McRae, A.A. (2012). A self-stabilizing algorithm for optimally efficient sets in graphs. Information Processing Letters, 112(16): 621-623. https://doi.org/10.1016/j.ipl.2012.02.014

[11] Bessaoud, K., Bui, A., Pilard, L. (2013). Self-stabilizing algorithm for low weight connected dominating set. In 2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications, Delft, Netherlands, pp. 231-238. https://doi.org/10.1109/DS-RT.2013.33

[12] Ding, Y., Wang, J.Z., Srimani, P.K. (2016). A linear time self-stabilizing algorithm for minimal weakly connected dominating sets. International Journal of Parallel Programming, 44(1): 151-162. https://doi.org/10.1007/s10766-014-0335-4

[13] Wang, G., Wang, H., Tao, X., Zhang, J. (2013). A self-stabilizing algorithm for finding a minimal positive influence dominating set in social networks. In Conferences in Research and Practice in Information Technology (CRPIT), 137: 93-100.

[14] Ding, Y., Wang, J.Z., Srimani, P.K. (2014). Self-stabilizing selection of influential users in social networks. In 2014 IEEE 17th International Conference on Computational Science and Engineering, Chengdu, China, pp. 1558-1565. https://doi.org/10.1109/CSE.2014.288

[15] Datta, A.K., Devismes, S., Larmore, L.L. (2019). A silent self-stabilizing algorithm for the generalized minimal k-dominating set problem. Theoretical Computer Science, 753: 35-63. https://doi.org/10.1016/j.tcs.2018.06.040

[16] Turau, V. (2007). Linear self-stabilizing algorithms for the independent and dominating set problems using an unfair distributed scheduler. Information Processing Letters, 103(3): 88-93. https://doi.org/10.1016/j.ipl.2007.02.013

[17] Ikeda, M., Kamei, S., Kakugawa, H. (2002). A space-optimal self-stabilizing algorithm for the maximal independent set problem. In the Third International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp. 70-74.

[18] Guellati, N., Kheddouci, H. (2010). A survey on self-stabilizing algorithms for independence, domination, coloring, and matching in graphs. Journal of Parallel and Distributed Computing, 70(4): 406-415. https://doi.org/10.1016/j.jpdc.2009.11.006

[19] Neggazi, B., Guellati, N., Haddad, M., Kheddouci, H. (2015). Efficient self-stabilizing algorithm for independent strong dominating sets in arbitrary graphs. International Journal of Foundations of Computer Science, 26(6): 751-768. https://doi.org/10.1142/S0129054115500422

[20] Johnen, C. (2014). Fast, silent self-stabilizing distance-k independent dominating set construction. Information Processing Letters, 114(10): 551-555. https://doi.org/10.1016/j.ipl.2014.04.013

[21] Johnen, C. (2015). Memory efficient self-stabilizing distance-k independent dominating set construction. In Networked Systems: Third International Conference, NETYS 2015, Agadir, Morocco, pp. 354-366. https://doi.org/10.1007/978-3-319-26850-7_24

[22] Goddard, W., Srimani, P.K. (2013). Daemon conversions in distributed self-stabilizing algorithms. In International Workshop on Algorithms and Computation, pp. 146-157. https://doi.org/10.1007/978-3-642-36065-7_15

[23] Kuszner, L. (2005). Tools to develop and test self-stabilizing algorithms. http://kaims.eti.pg.gda.pl/kuszner/selfstab/main.html.

[24] Barabási, A.L., Albert, R. (1999). Emergence of scaling in random networks. Science, 286(5439): 509-512. https://doi.org/10.1126/science.286.5439.509