Algerian Journal of Sciences AJS / Volume X Issue Y PP. Z 2024

A Survey on Distributed Self-Stabilizing Algorithms for Finding Independent Sets

Yousra Touahra[1], Nesrine Hadil Khaloua[2], Imane Derradj[3], Nabil Guellati[4*]

[1] Department of Computer Science,
University of Ferhat Abbas (Setif 1), Algeria, tyousra29@gmail.com
[2] Department of Computer Science,
University of Ferhat Abbas (Setif 1), Algeria, kolopolo1206@gmail.com
[3] Department of Computer Science,
University of Ferhat Abbas (Setif 1), Algeria, derradjimane77@gmail.com
[4] LRSD laboratory, Department of Computer Science,
University of Ferhat Abbas (Setif 1), Algeria, nabil.guellati@gmail.com

Abstract

Self-stabilization is a technic for handling transient faults in distributed systems introduced by Dijkstra in 1973. A distributed system is considered to be self-stabilizing if it can recover a correct state after a finite period of time, regardless of its initial state. Several issues observed in networks and distributed systems can be modeled using graph parameters. In this paper, we survey distributed self-stabilizing algorithms for finding independent sets in graphs. We evaluate the performance of five algorithms by performing simulations. Independent sets finding algorithms are useful for node clustering in networks and distributed systems.

Keywords: Distributed systems, Fault-tolerance, Self-stabilization, Graph theory, Independent sets.

1 Introduction

Distributed systems play a crucial role in computing, providing a flexible architecture to meet the growing needs for connectivity and large-scale data

*Corresponding author: nabil.guellati@gmail.com

processing. Composed of a set of interconnected autonomous computers, these systems enable collaboration between different processes to achieve common objectives. However, their distributed nature raises unique challenges in terms of coordination, data consistency and fault tolerance.

Self-stabilization is a revolutionary concept introduced by the great scientist Dijkstra in 1973 [1]. It offers an innovative way to guarantee the reliability of distributed systems. It allows these systems to converge, without external interventions, towards a correct state independently of their initial state. Self-stabilization offers an effective solution to deal with transient faults. Several papers and books explaining the fundamentals of this field may be found here [2, 3, 4, 5, 6, 7, 8, 9].

An independent set [10] is a parameter of graph theory that has a lot of applications in computer science, and a lot of centralized and distributed algorithms for this parameter have been proposed. An independent set is a set of nodes in a graph that are not adjacent. An independent set is maximal (MIS) if it is not contained in any other independent set. An independent set is maximum if it has the largest cardinality of all possible independent sets for a graph G. Finding the maximum independent in a graph is known to be NP-hard, so most of the proposed algorithms focuses on finding a maximal independent set.

Independent sets in the context of networks and distributed systems are used in node clustering (figure 1). Clustering in computer networks is the creation of node groups called clusters where each group has a cluster head. Some clustering solutions use the nodes selected in an independent set as cluster heads, and together with their neighbors they form a cluster. Such solutions may be found here [11, 12].

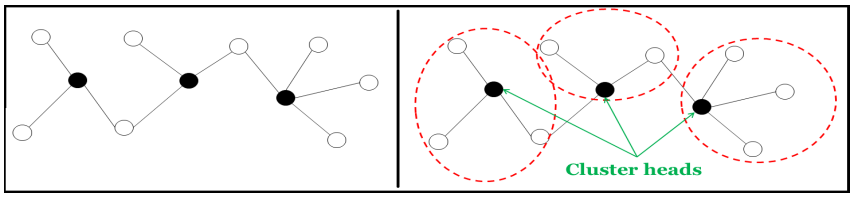

Figure 1: An independent set (on the left), and the same independent set used in clustering (on the right)

In this paper, we provide a description of distributed self-stabilizing algorithms for finding independent sets proposed in the literature and we compare five algorithms using simulations based on the application of Lukasz

Kuszner. This application has been developed for facilitating implementation and visualization of self-stabilizing algorithms.

2 Definitions

In a graph G (V, E), where V is the set of vertices and E is the set of edges, an independent set C is a subset of V such that no two members of C are adjacent in G. C is maximal if no proper super set of C is an independent set. Figure 2 shows examples of independent sets. We denote by N(i) the set of neighbors of node i in a graph G.

Figure 2: An independent set (on the left), and a maximal independent set (on the right)

We model a distributed system by a graph where the set of vertices represents the computers and the set of edges represents the links connecting them.

Self-stabilizing algorithms are written as a set of guarded rules, where a rule is of the form "**if** condition **then** action". A process for which a condition is true is called an enabled process.

Self-stabilizing algorithms assume the existence of a scheduler (a daemon) that chooses at every step of the system execution the set of processes to execute an action. The central daemon selects at every execution step one process to execute an action. The synchronous daemon selects at every execution step all enabled processes to execute an action. The distributed daemon selects at every execution step a sub-set of enabled processes to execute an action. The authors may assume only one of these schedulers for their algorithm.

Another assumption widely used by self-stabilizing algorithms is the shared memory model of communication. Here, they assume that every process can read directly the variables of its neighbors.

3 Self-stabilizing independent set algorithms

In [13], Hedetniemi et al. propose an algorithm that uses a central daemon. This algorithm finds a maximal independent set (MIS) in a graph. Here, each node maintains a boolean variable s(i) to indicate its membership in the set being constructed. The algorithm works by applying two rules: rule 1 adds a node to the set if this node does not belong to the set and if all its neighbors do not belong to the set. On the other hand, rule 2: allows removing a node from the set if it already belongs to the set and if at least one of its neighbors belongs to the set.

In [14], Turau proposes an algorithm that uses a distributed daemon. This algorithm assumes that each node in the graph has a unique identifier and starts in one of the predefined initial states: WAIT (indicates that the node is waiting for a decision), OUT (means that the node does not belong to the maximal independent set) or IN (indicates that the node is part of the maximal independent set). The algorithm works as follows : if a node is "out" and has no neighbor "in" then it passes to the state "wait". Secondly, if a node is "wait" and has a neighbor "in" then it passes to state "out". Thirdly, if a node is "wait" and has no neighbor "in" and no neighbor "wait" with lower identifier then it passes to state "in". Finally, a node in the state "in" and has a neighbor "in" passes to state "out".

In [15], Srimani et al. propose an algorithm for computing a maximal independent set (MIS) in a distributed system. This algorithm relies on assumptions such as the use of a synchronous daemon and distinct identifiers for the nodes. Each node maintains a boolean variable to indicate its membership in the maximal independent set being constructed. The algorithm works by applying two rules. Firstly, a node out of the set joins the set if it has no neighbor in the set with higher identifier. Also, a node in the set leaves the set if it has a neighbor with higher identifier in the set.

In [16], Neggazi et al. propose an algorithm for finding a set of node in an arbitrary graph that is both independent and strongly dominating. The authors assume a distributed daemon and unique identifiers for the nodes. A set of nodes in a graph is a strongly dominating set if each node in the graph is either in the set or has a neighbor in the set with higher degree. This is the first self-stabilizing algorithm for this graph parameter. For this algorithm, the authors define a lexicographic order between the nodes of a graph G which takes into account their degrees and their identifiers. Thus, the nodes are first classified according to their degrees and if two nodes have the same degree, they are classified according to the highest identifier. Thus, node i is stronger than node j means that the degree of i is higher than the

degree of j or they have the same degree and the identifier of i is higher than the identifier of j. The algorithm uses two rules. The first one corrects the degree of a node if it is not correct, and the second one changes the state of a node to "out" if it is strongly dominated and to "in" otherwise.

In [17], Michiyo et al. propose an algorithm for finding an MIS assuming a distributed daemon and unique identifiers for the nodes. The rules of the algorithm allow a node to join the set if it has no neighbor in the set, and to leave the set if it has a neighbor with a lower identifier in the set.

In [18], Arapoglu et al. propose an algorithm for the capacitated maximal independent set problem. Here, in a capacitated independent set each node in the set has a capacity of domination that it can't exceed. The algorithm assumes a distributed scheduler, unique identifiers for the nodes and a capacity of domination to each node. The algorithm has 7 rules where R1, R2, R3 and R4 are executed by IN nodes and the other rules are executed by OUT nodes. R1 and R7 construct an MIS and the other rules are given to provide the capacity constraint. Moreover, the authors implemented their algorithm in a Wireless Sensor Network.

In [19], Yen et al. model the problem of finding an MIS as a non-cooperative graphical game and propose an algorithmic mechanism design for the problem. Then, the authors turn the design into a self-stabilizing algorithm assuming a synchronous daemon.

In [20], Benreguia et al. propose an algorithm for finding a maximal distance-2 independent set. This parameter is maximal independent set where nodes are far from each other at least with distance 3. The algorithm uses a central daemon. It allows a node that is "out" and has no direct neighbor "in" and no distance-2 neighbor "in" to join the set. Also, a node that is "in" and has a neighbor "in" or has a distance-2 neighbor "in" leaves the set. It is to be noted that the authors assume that each node may directly read the variables of its direct neighbors and distance-2 neighbors.

In [21], Kakugawa el al. propose an algorithm for the 1-MIS problem assuming the distance-3 model. The 1-MIS problem is a problem of computing a maximal independent set such that flipping the membership of any three nodes does not improve the size of the independent set. This means that, in a 1-MIS we may not increase the cardinality of the MIS by removing a node and adding two or more nodes. Also, the authors assume the distance-3 model of communication, which means that a node can read directly the variables of its direct neighbors, distance-2 neighbors, and distance-3 neighbors. The proposed algorithm assumes a central daemon. It uses 6 rules for finding a 1-MIS and it implements a procedure called "state-flipping transaction" allowing a node to leave the set if it may allow more nodes to join

the set.

4 Discussion

It is to be noted that some of the algorithms presented in this paper use a central daemon. This assumption is not realistic since execution of processes in a distributed system is performed in a parallel way. Thus, the central daemon eliminates all parallelism in the execution of the distributed system. However, it is possible to transform a self-stabilizing algorithm using a central daemon to an algorithm using a distributed daemon by applying the rules of transformers like the one described here [22] by Goddard and Srimani. An algorithm that uses a distributed daemon is more suitable for implementation and the distributed daemon is far more realistic in a distributed context than the central daemon.

On another hand, all of the algorithms presented in this paper use the shared memory model of communication which is not realistic in a distributed context. However here too, transformation is possible and we may transform these algorithms and implement them using the message passing model of communication. For example, Arapoglu et al. described in their paper how to transform and implement their algorithm in a wireless sensor network. Also, a way to transform self-stabilizing algorithm to the message passing model of communication has been described in this book [23].

5 Performance evaluation

We implemented the five algorithms : Hedetniemi, Turau, Michiyo, Srimani, and Neggazi using the application of Lukasz Kuszner in order to perform simulations and to compare the convergence speed of the five algorithms. The figure 3 shows the number of moves (actions executed by the algorithms) needed by every algorithm to reach a stable state. According to the results, the algorithm of Michiyo is the best algorithm in terms of convergence speed, and convergences quicker than any other algorithm. From another side, the algorithm of Neggazi is the slowest algorithm, and it converges very slowly in comparison to the other algorithms. However, this algorithm finds a strong dominating set and not only an independent set, and putting this algorithm aside the other four algorithms have quite close convergence speed.

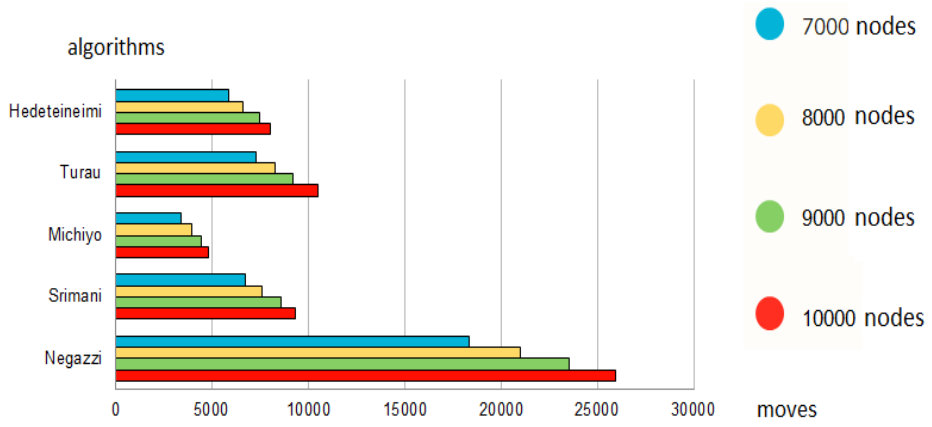

Figure 3: Number of moves to reach a stable state

6 Conclusion

In this paper we survey distributed self-stabilizing algorithms for the independent set parameter in graphs. A lot of algorithms have been proposed for this parameter and it is important to study these propositions and to compare them. Here, we described the proposed algorithms and the assumptions made for each algorithm, and we performed simulations to compare five algorithms in terms of convergence speed. As perspective we may consider to survey the self-stabilizing algorithms for finding dominating sets, which is a quite close parameter.

References

[1] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communication of the Acm*, 17(11):643–644, November 1974.

[2] S. Dolev. *Self-Stabilization*. MIT Press, 2000.

[3] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, second edition, September 2000.

[4] L. C. Wuu and S. T. Huang. Distributed self-stabilizing systems. *Journal of Information Science and Engineering*, 11(2):307–319, June 1995.

[5] M. G. Gouda. The triumph and tribulation of system stabilization. In *Proceedings of the 9th International Workshop on Distributed Algorithms*, pages 1–18, 1995.

[6] M. Schneider. Self-stabilization. *ACM Computing Surveys*, 25(1):45–67, March 1993.

[7] J. Brzezinski and M. Szychowiak. Self-stabilization in distributed systems - a short survey. *Foundations of Computing and Decision Sciences*, 25(1), 2000.

[8] J. E. Burns, M. G. Gouda, and R. E. Miller. Stabilization and pseudo-stabilization. *Distributed Computing*, 7:35–42, 1993.

[9] S. Tixeuil. *Algorithms and Theory of Computation Handbook*, chapter Self-stabilizing Algorithms. CRC Press, Taylor & Francis Group, November 2009.

[10] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of domination in graphs*. CRC press, 2013.

[11] K. Erciyes, O. Dagdeviren, D. Cokuslu, and D. Ozsoyeller. Graph theoretic clustering algorithms in mobile ad hoc networks and wireless sensor networks. *Applied and Computational Mathematics*, 6(2):162–180, 2007.

[12] O. Senouci, S. Harous, and Z. Aliouat. Survey on vehicular ad hoc networks clustering algorithms: Overview, taxonomy, challenges, and open research issues. *International Journal of Communication Systems*, 33(11), 2020.

[13] S. M. Hedetniemi, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani. Self-stabilizing algorithms for minimal dominating sets and maximal independent sets. *Computers & Mathematics with Applications*, 46(5-6):805–811, 2003.

[14] V. Turau. Linear self-stabilizing algorithms for the independent and dominating set problems using an unfair distributed scheduler. *Information Processing Letters*, 103(3):88–93, 2007.

[15] W. Goddard, S. T. Hedetniemi, D. P. Jacobs, P. K. Srimani, and Z. Xu. Self-stabilizing graph protocols. *Parallel Processing Letters*, 18(01):189–199, 2008.

[16] B. Neggazi, N. Guellati, M. Haddad, and H. Kheddouci. Efficient self-stabilizing algorithm for independent strong dominating sets in arbitrary graphs. *International Journal of Foundations of Computer Science*, 26(06):751–768, 2015.

[17] M. Ikeda, S. Kamei, and H. Kakugawa. 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)*, pages 70–74, 2002.

[18] O. Arapoglu and O. Dagdeviren. Distributed self-stabilizing capacitated maximal independent set construction in wireless sensor networks. *Wireless Personal Communications*, 114(4):3271–3293, 2020.

[19] L. H. Yen and J. Y. Huang. Selfish self-stabilizing approach to maximal independent sets. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 3, pages 9–16, 2015.

[20] B. Benreguia, H. Moumen, S. Bouam, and C. Arar. Self-stabilizing algorithm for maximal distance-2 independent set. *arXiv preprint arXiv:2101.11126*, 2021.

[21] H. Kakugawa, S. Kamei, M. Shibata, and F. Ooshita. A self-stabilizing distributed algorithm for the 1-mis problem under the distance-3 model. In *2023 Eleventh International Symposium on Computing and Networking Workshops (CANDARW)*, pages 100–106, 2023.

[22] W. Goddard and P. K. Srimani. Daemon conversions in distributed self-stabilizing algorithms. In *International Workshop on Algorithms and Computation*, pages 146–157. Springer, 2013.

[23] V. K. Garg. *Concurrent and distributed computing in Java*. John Wiley & Sons, 2005.