Evaluating Fault Tolerance Properties of Self-stabilizing Matching Algorithms in Wireless Sensor Networks

Can Umut Ileri, Orhan Dagdeviren
International Computer Institute
Ege University, Izmir, Turkey
Email: {can.umut.ileri, orhandagdeviren}@ege.edu.tr

***Abstract*—Self stabilization is an important paradigm for the autonomous recovery of a distributed system from transient failures such as energy depletion of nodes and disrupted connections. It has been used in wireless sensor networks (WSN) as these networks are expected to automatically recover from a transient fault without human intervention. Graph matching is fundamental graph theory problem which has a broad application range in WSNs and it has been studied extensively in self-stabilizing settings. In this work, we build a simulation model and perform tests to evaluate the fault tolerance properties of self-stabilizing matching algorithms. To the best of our knowledge, this is the first practical evaluation of these algorithms. Considering WSNs, we assume distributed fair and synchronous schedulers. Simulation results have shown that there is a tradeoff between stabilization time of algorithms and the quality of their results. The improvement algorithms which has better lower bounds give better matchings at the cost of longer durations of instability.**

***Keywords*— Self-stabilization, Graph Matching, Wireless Sensor Networks, Performance Evaluation.**

I. Introduction

Wireless sensor networks (WSNs) are composed of a large number of distributed smart nodes that can sense, measure and collect information from the environment. These networks have been used extensively due to their easiness of use and can be considered as the basis of Internet of Things (IoT). As they can be deployed in very large-scale areas with harsh terrain conditions, they can eventually become difficult to maintain and control. Sensor nodes may be damaged or failed due to energy depletion and the connections between nodes may be disrupted due to environmental conditions.

Self-stabilization [1], [2] is an important concept for the recovery from such failures and it has been widely used in WSNs [3] for various applications from topology control to time slot assignment [4]–[6]. From the self-stabilization point of view, a transient fault on the network may be considered as a black box which outputs an undesired arbitrary configuration of the system. A self-stabilizing algorithm is expected to autonomously and automatically reconfigure the system in finite amount of time (convergence property) and to remain in the desired configuration until occurrence of another transient fault (closure property).

Matching problem is among the most fundamental graph theoretical problems and it has been attracting a growing attention in network design, especially in WSNs [7], [8]. Some of these applications are radio resource allocation in cognitive radio networks [9], physical layer security [10], energy-efficient partner selection [11], partner selection for cooperative relaying [12], and optimizing storage capacity and power consumption [13]. General aim of these applications is to pair the nodes (or *users* or *agents*) according to their preference lists obtained by a specific criteria.

Given a graph $G(V, E)$, the purpose of a matching algorithm is to find a subset of edges $M \subseteq E$ such that any vertex in V is incident to at most one edge in M. The problem of maximizing the number of edges in M is called the *maximum cardinality matching* (MCM). A matching M is considered maximal if it is not possible to add another edge from E into M without having to remove another edge from M.

Matching problem has been studied extensively in self-stabilizing settings [14]–[21]. Hsu and Huang [14] proposed the first maximal matching self-stabilizing algorithm. It works on anonymous networks and assumes a central daemon. It consists of three rules which provides a basis for many self-stabilizing matching algorithms. Authors first proved that the move complexity of the algorithm is $O(n^3)$, which was later improved to $O(n^2)$ in [22] and finally to $O(m)$ in [23]. In [15], Gradinariu and Johnen worked on self-stabilizing unique naming problem and they applied their approach on Hsu&Huang's algorithm so that it can cope with distributed scheduler. Assuming an unfair daemon, they could not find a bound for the move complexity but they proved that the algorithm stabilizes in finite number of moves. Chattopadhyay et al. [16] offered an improved algorithm for the same problem working under read/write atomicity with linear round complexity. They assumed that the nodes have distinct identifiers within distance two. Goddard et al. proposed the first self-stabilizing algorithm for maximal matching problem under synchronous daemon [17]. They combined the three rules of Hsu&Huang's algorithm with the symmetry breaking technique of Chattopadhyay et al.'s algorithm and proved that their algorithm stabilizes in $O(n)$ rounds. Gradinariu and Tixeuil proposed a framework algorithm (conflict manager) for making it possible to run self-stabilizing algorithms designed for central daemon under distributed daemon [18]. They applied their technique on Hsu&Huang's Algorithm and proved that, with the help of conflict manager, Hsu&Huang's algorithm

TABLE I
SELF-STABILIZING ALGORITHMS FOR MAXIMAL MATCHING.

			Complexity[a]		
Paper	**Scheduler**	**Approx.**	**Step**	**Round**	**Assumption**
[14]	Seq. unfair	$1/2$	$O(m)$	-	Anonymous
[15]	Dist. unfair	$1/2$	-	-	Distance 2
[16]	Dist. fair	$1/2$	$\Omega(n^2)$	$O(n)$	Distance 2
[17]	Synchronous	$1/2$	$O(n^2)$	$O(n)$	Unique ID
[18]	Dist. unfair	$1/2$	$O(\Delta m)$	-	Unique ID
[19]	Dist. unfair	$1/2$	$O(m)$	$O(n)$	Distance 2
[20]	Dist. unfair	$2/3$	$O(\Omega^{\sqrt{n}})$	$O(n^2)$	Distance 2
[21]	Dist. unfair	$2/3$	$O(n^3)$	-	Distance 2

[a] Δ is the maximum degree.

stabilizes under distributed daemon and stabilizes in $O(\delta m)$ moves. Manne et al. [19] proposed a self-stabilizing maximal matching algorithm which assumes an unfair daemon and has the best move ($O(m)$) and round ($O(n)$) complexities to the best of our knowledge. Cohen et al. extended this result to anonymous networks with randomized self-stabilization [24].

All algorithms mentioned so far guarantees that the stabilized solution is a maximal matching which in turn is at least a $\frac{1}{2}$-approximation of the MCM. The following studies make use of augmenting paths to improve this ratio. Given a graph $G(V, E)$ and a matching $M \subseteq E$, an augmenting path is a path which (1) alternatively traverses matched and unmatched edges and (2) starts and ends in an unmatched edge. It is known that if G does not contain an augmenting path with length smaller or equal to 3 with respect to M, then M is a $\frac{2}{3}$-approximation to the MCM [25]. Manne et al. [20] used this finding and presented an algorithm which detects and augments all augmenting paths with length 3 in a graph with respect to a maximal matching. Its worst case complexity is $O(2^n \Delta n)$ under distributed unfair scheduler. Cohen et al. [26] presented a proof that Manne et al.'s algorithm is sub-exponential. They later [21] provided an adaptation of the algorithm which converges to at least $2/3$-approximation matching in polynomial ($O(n^3)$) number of steps.

All these works on self-stabilizing matching algorithms provide the theoretical worst case approximation ratios and time complexities (step, move or round) as well as the algorithm description and the proofs of correctness, convergence and closure. In this work, we address the practical performance of self-stabilizing maximal matching algorithms in simulation environments which model a WSN. To the best of our knowledge, this is the first study which provides practical evaluation of self-stabilizing maximal matching algorithms through simulations. We run algorithms on instances of geometric networks assuming distributed fair schedulers. The rest of the paper is organized as follows. We further describe our model in Section II, present our experiments in Section III and report computational results in Section IV.

II. THE MODEL

We assume a wireless sensor network and a message passing model over a reliable transport protocol. Each node has a unique identifier within distance 2. Nodes share the values of their variables with their neighbors via messages upon making a move (in distributed fair scheduler) or at the end of each synchronous round (in distributed fair or synchronous scheduler). Each node has local variables holding values of its neighbors. Note that the algorithm which require the highest number of variables among the algorithms we consider is [21] and it requires 4 integer (holding *id*s) and 2 binary variables.

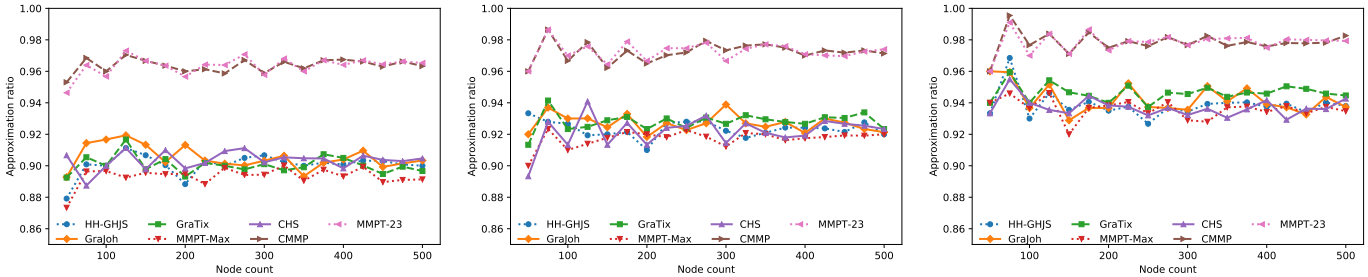
Nodes are able to detect failures of their neighbors. For a collection of node failure detection methods in WSNs, we refer to readers to [27]. In self-stabilization, values of the nodes are assumed to be finite. Actual values of local variables make up the *state* of the node. A *configuration* is an instance of the state of all nodes. Each node executes the same algorithm consisting of rules which has two components: guard and action. If the guard of the rule holds true for a node, we say the rule is *enabled* and the node is *eligible* to make a *move*. Move is the transition of a node from one state to another.

In characterization of self-stabilizing algorithms, an important notion is how the distributed processors run with respect to each other. These assumptions are modeled under the notion of *scheduler* or so called *daemon*. The two most important characteristic traits of the schedulers are distribution and fairness. (An extensive study on taxonomy of schedulers has been done in [28].) Distribution trait determines the concurrency of the processors. Particularly, exactly one processor may be scheduled at a given time (*sequential* or *central* scheduler) or more than one processor can make concurrent moves (*distributed* scheduler). Fairness trait of a scheduler considers how fairly the processors make moves, i.e., whether the scheduler guarantees that a processor which is eligible to make a move eventually makes a move (*fair scheduler*) or there is not such a guarantee for at least one eligible processor (*unfair* or *adversarial* scheduler).

III. EXPERIMENTAL SETUP

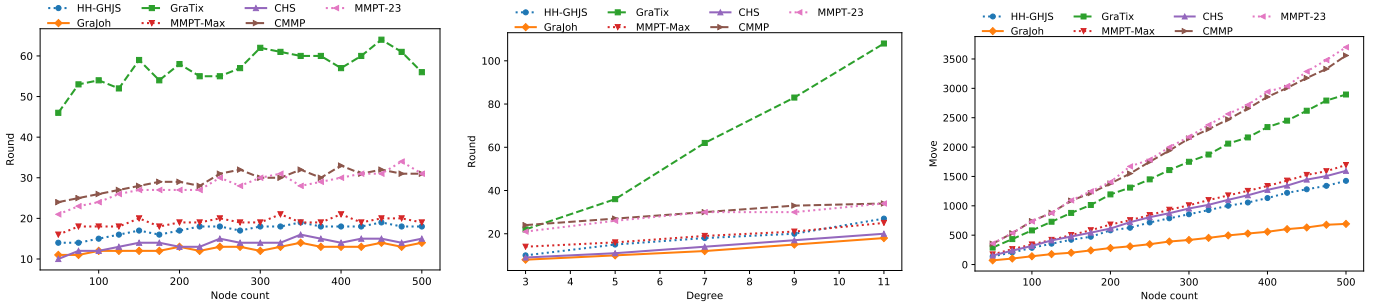
The assumption of a distributed fair scheduler fits well to the case of WSNs, since those networks generally lack of a central control mechanism for scheduling (distributed attribute) and there is nothing that prevents an enabled node from making a move (fairness attribute). Thus we run the algorithms assuming (1) distributed scheduler and (2) synchronous scheduler which is a special case of distributed fair schedulers where an enabled node is definitely privileged while in distributed fair scheduler, only global progression is guaranteed, that is, at least one node is scheduled in a round. In our simulations, we used a parameter *privProb* which determines the probability of an enabled processor to be scheduled. We set it to 0.7 for distributed scheduler. In the synchronous case, $privProb = 1$.

To simulate WSNs, we created network instances with the geometric network model. Given a graph $G(V, E)$ where vertices in V are distributed over a 2D area and they all have a unit range r, any two vertices are considered connected if their euclidean distance is smaller than r. In real WSNs, ranges of nodes may vary, however unit ranges do not affect the generality. We generated 1000 topologies having sizes from



(a) Average Degree = 5 (b) Average Degree = 7 (c) Average Degree = 9

Fig. 1. Approximation performance of algorithms with respect to size of the network



(a) Time vs. Node count. Average Degree: 7 (b) Time vs. Density. Node Count: 300 (c) Move count vs. Node count. Average degree: 7

Fig. 2. Time required to stabilize from random configuration

25 to 500 and average degrees from 3 to 11. Each size-degree pair have 10 different instances and we use their averages.

We had two sets of experiments. In the first set, starting from a randomly initialized configuration, we ran algorithms on each test instances and measured the *move count*, *round count* and *approximation ratio*. In the second set, we fixed the number of nodes and average degree and, starting from a stabilized (actually optimum) configuration, we ran algorithms along with several fault scenarios. Each scenario has 4 parameters: n_{init}: number of nodes in the initial configuration. n_{round}: number of rounds that the algorithms are allowed to run. p_{insert}: probability of a new node to join network. p_{remove}: probability of a node to fail (leave the network). We had 5 scenarios with the parameters shown in Table II.

TABLE II
FAULT SCENARIOS FOR EXPERIMENTS

Scenario	n_{init}	n_{round}	p_{insert}	p_{remove}
1	100	3000	0.1	0
2	500	3000	0	0.1
3	300	3000	0.05	0.05
4	300	3000	0.10	0.10
5	300	3000	0.15	0.15

As an example, in scenario 3, the network initially has 300 nodes with a configuration in which the values of the variables of nodes correspond to the optimum matching. After the execution starts, in each round, an active node fails with a probability of 0.1 and an new node joins the network with a probability of 0.1. The execution lasts 3000 rounds.

Throughout the evaluations, we consider [14] and [17] as the same algorithms since they have the same rules and the latter is a transformed version of the former to the synchronous scheduler. We implemented and compared 7 algorithms. We name those algorithms as follows: HH-GHJS [14] [17], GraJoh [15], CHS [16], GraTix [18], MMPT-Max [19], MMPT-23 [20] and CMMP [21]. Note that MMPT-23 and CMMP are improvement algorithms and require to be composed with a self-stabilizing maximal matching algorithm. For these two algorithms, we used MMPT-Max as the initial matching algorithm as suggested in papers. We used two metrics for the second simulation set: (1) the number of rounds in which the system is stabilized during execution and (2) approximation to the optimum matching at the end of each round.

IV. COMPUTATIONAL RESULTS

A. Stabilization from Random Configuration

Approximation performance. All $1/2$-approximation algorithms gave similar approximation ratios ranging between 0.88 and 0.92 for average degree of 5 (Figure 1a). As the average degree increases to 7 and 9, their approximations rise to 0.9-0.94 and 0.92-0.95 (Figures 1b and 1c). Note that in complete graphs any maximal matching is optimum. This explains the increase in the approximation performance. $2/3$-approximation algorithms performed very well as they resulted with matchings which approximate the optimum matching by

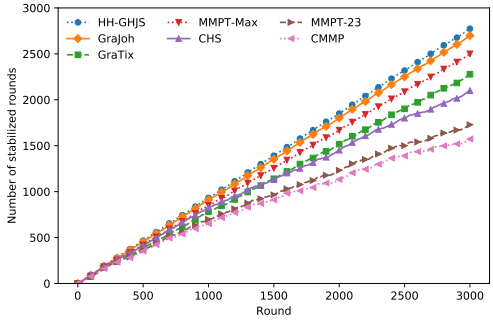


Fig. 3. Number of rounds at the end of which the system is stabilized (Scenario 1: Growing network)

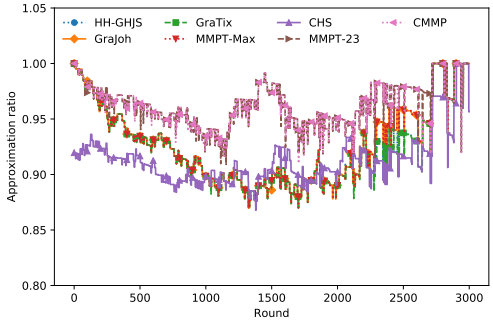


Fig. 4. Approximation ratio at each round during execution of 3000 rounds (Scenario 2: Diminishing network)

0.96 to 0.98. Under synchronous scheduler, experiments give similar results (We omit plots for space restrictions.).

Stabilization time. The time required for stabilization of the algorithms with respect to the graph size (Figure 2a) and average degree (Figure 2b) are shown in Figure 2. The quickest algorithms to stabilize are GraJoh and CHS. GraTix takes the most time to stabilize. Recall that this algorithm allows only one node to make a move at a time in 1-hop distance. As shown in Figure 2b, this algorithm's running time increases as average degree in the network increases. Other algorithms require only slightly more time as the network becomes denser.

The total number of moves of nodes to stabilize increases linearly with respect to the size of the network (Figure 2c). GraJoh requires the least number of moves (1 move per node in average) to stabilize. MMPT-Max, CHS and HH-GHJS require around 3 moves per node. GraTix again gives the worst performance among $1/2$-approximation algorithms. Improvement algorithms CMMP and MMPT-23 require around 7 moves per node. Recall that these algorithms use MMPT-Max as the initial matching algorithm. That means the improvement procedure itself requires more move than finding a maximal matching.

We have another important result obtained from the comparison of Figures 2a and 2c. Note that both results are obtained under synchronous scheduler. As we consider the number of moves per round, we see that both improvement algorithms and CHS have relatively better round-time performance. In other words, the nodes running these algorithms make more concurrent moves than the other algorithms.

B. Fault Tolerance

Growing and Diminishing Networks. Scenario 1 is an example of growing network where new nodes continuously join the network. Note that from the self-stabilization point of view, joining of a node is a transient fault, too. Similarly, Scenario 2 is an example of diminishing network. Figure 3 illustrates the number of stabilized rounds through an execution of 3000 rounds in Scenario 1. Results show that HH-GHJS is the most agile algorithm in responding failures. During the execution of HH-GHJS, the system remains stabilized at the end of nearly 90% of the rounds. GraJoh gives a similar performance to HH-GHJS. Other $1/2$-approximation algorithms are ordered as MMPT-Max, GraTix and CHS according to their performance. In the case of improvement algorithms of CMMP and MMPT-23, the system is not stable half of the time. As the respective results of Scenario 2 is quite similar, we omit it.

In Figure 4 we show the actual approximation ratio of algorithms with respect to the optimum matching at each round. Recall that the system is initialized with an optimum matching. An interesting result is the case of CHS as it does not keep the optimum solution in the first round, although the system is in a desired configuration. The reason is that, CHS always tries to match the node having the lowest id with the node having the second lowest id, and so on. It does so at the expense of breaking an already formed matching.

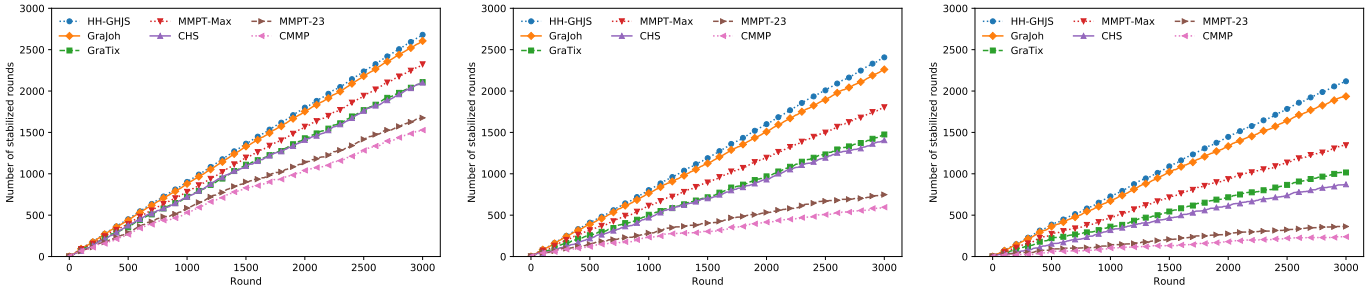
Improvement algorithms perform better in keeping the approximation ration high. They even give nearly-optimum results around round 1500, while the approximations of other algorithms drop to 90%. As more nodes are removed from the system, all algorithms give similar results and the approximation gets close to the optimum. Recall that the system starts with 400 nodes and the expected node count at round 3000 is 100 which results in a less number of neighbor for each node.

Dynamic networks. Figure 5 illustrates the fault tolerance results for Scenarios 3 to 5 in which the joining and removing may happen at the same time with various fault percentages. The order of performances of algorithms are same with the case in Scenario 1 (Figure 3). We observe in Figure 5 that the number of stable rounds decreases as the frequency of faults increases. In Scenario 5 where a fault is expected nearly in every 3 rounds, a system running improvement algorithms can be stable only in roughly 10% of the time.

V. Conclusion

In this paper, we presented evaluation of self-stabilizing matching algorithms for WSNs by conducting extensive experiments. To the best of our knowledges, this is the first practical performance evaluation study in this field.

In general algorithms perform significantly better than their theoretical approximation guarantees. In sparse networks (300 nodes and 750 edges), improvement algorithms (MMPT-23 and CMMP) are able to reach 0.96-approximations while



(a) Scenario 3: $p_{insert} = p_{remove} = 0.05$ (b) Scenario 4: $p_{insert} = p_{remove} = 0.1$ (c) Scenario 5: $p_{insert} = p_{remove} = 0.15$

Fig. 5. Number of rounds at the end of which the system is stabilized

maximal matching algorithms reach 0.9-approximation. As networks become denser, the performance of algorithms gets closer to optimum. In environments with continuous transient faults, HH-GHJS and GraJoh keeps the system more stable with respect to the other algorithms. Although improvement algorithms are able to a give better matchings, they require more time to stabilize. Our study shows that the theoretical performance measures of algorithm may not reflect their performances well in real world situations.

ACKNOWLEDGEMENT

Authors would like to thank TUBITAK for the project grant (215E115) and the PhD research grant (BIDEB-1001).

REFERENCES

[1] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of the ACM*, vol. 17, no. 11, pp. 643–644, 1974. [Online]. Available: http://portal.acm.org/citation.cfm?doid=361179.361202

[2] S. Dolev, *Self-stabilization*. MIT press, 2000.

[3] G. Siegemund, "Self-stabilizing algorithms in wireless sensor networks," Ph.D. dissertation, Technische Universität Hamburg-Harburg, 2017.

[4] N. Mitton, E. Fleury, I. G. Lassous, and S. Tixeuil, "Self-stabilization in self-organized multihop wireless networks," in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*. IEEE, 2005, pp. 909–915.

[5] H. Zhang and A. Arora, "Gs3: scalable self-configuration and self-healing in wireless sensor networks," *Computer Networks*, vol. 43, no. 4, pp. 459–480, 2003.

[6] J. Ben-Othman, K. Bessaoud, A. Bui, and L. Pilard, "Self-stabilizing algorithm for efficient topology control in wireless sensor networks," *Journal of Computational Science*, vol. 4, no. 4, pp. 199–208, 2013.

[7] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, "Matching theory for future wireless networks: fundamentals and applications," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 52–59, May 2015.

[8] S. Bayat, Y. Li, L. Song, and Z. Han, "Matching theory: Applications in wireless communications," *IEEE Signal Processing Magazine*, vol. 33, no. 6, pp. 103–122, Nov 2016.

[9] S. Bayat, R. H. Louie, B. Vucetic, and Y. Li, "Dynamic decentralised algorithms for cognitive radio relay networks with multiple primary and secondary users utilising matching theory," *Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 5, pp. 486–502, 2013.

[10] S. Bayat, R. H. Y. Louie, Z. Han, B. Vucetic, and Y. Li, "Physical-layer security in distributed wireless networks using matching theory," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 5, pp. 717–732, May 2013.

[11] M. W. Baidas and M. M. Afghah, "A matching-theoretic approach to energy-efficient partner selection in wireless networks," in *2015 International Wireless Communications and Mobile Computing Conference (IWCMC)*, Aug 2015, pp. 1260–1265.

[12] C. Hasan, E. Altman, and J. M. Gorce, "Partner selection for decode-and-forward cooperative relaying: A matching theoretic approach," in *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Sept 2013, pp. 2275–2280.

[13] A. Roumy and D. Gesbert, "Optimal matching in wireless sensor networks," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 1, no. 4, pp. 725–735, 2007.

[14] S.-C. Hsu and S.-T. Huang, "A self-stabilizing algorithm for maximal matching," *Information processing letters*, vol. 43, no. 2, pp. 77–81, 1992.

[15] M. Gradinariu and C. Johnen, "Self-stabilizing neighborhood unique naming under unfair scheduler," in *European Conference on Parallel Processing*. Springer, 2001, pp. 458–465.

[16] S. Chattopadhyay, L. Higham, and K. Seyffarth, "Dynamic and self-stabilizing distributed matching," in *Proceedings of the twenty-first annual symposium on Principles of distributed computing*. ACM, 2002, pp. 290–297.

[17] W. Goddard, S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani, "Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*. IEEE, 2003.

[18] M. Gradinariu and S. Tixeuil, "Conflict managers for self-stabilization without fairness assumption," in *Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on*. IEEE, 2007, pp. 46–46.

[19] F. Manne, M. Mjelde, L. Pilard, and S. Tixeuil, "A new self-stabilizing maximal matching algorithm," *Theoretical Computer Science*, vol. 410, no. 14, pp. 1336–1345, 2009.

[20] ——, "A self-stabilizing 23-approximation algorithm for the maximum matching problem," *Theoretical Computer Science*, vol. 412, no. 40, pp. 5515–5526, 2011.

[21] J. Cohen, K. Maamra, G. Manoussakis, and L. Pilard, "Polynomial self-stabilizing maximum matching algorithm with approximation ratio 2/3," in *LIPIcs-Leibniz International Proceedings in Informatics*, vol. 70. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[22] G. Tel, "Maximal matching stabilizes in quadratic time," *Information Processing Letters*, vol. 49, no. 6, pp. 271–272, 1994.

[23] S. T. Hedetniemi, D. P. Jacobs, and P. K. Srimani, "Maximal matching stabilizes in time o (m)," *Information Processing Letters*, vol. 80, no. 5, pp. 221–223, 2001.

[24] J. Cohen, J. Lefevre, K. Maâmra, L. Pilard, and D. Sohier, "A self-stabilizing algorithm for maximal matching in anonymous networks," *Parallel Processing Letters*, vol. 26, no. 04, p. 1650016, 2016.

[25] J. E. Hopcroft and R. M. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal on computing*, vol. 2, no. 4, pp. 225–231, 1973.

[26] J. Cohen, K. Maamra, G. Manoussakis, and L. Pilard, "The manne et al. self-stabilizing 2/3-approximation matching algorithm is sub-exponential," *arXiv preprint arXiv:1604.08066*, 2016.

[27] Z. Rehena, R. Mukherjee, S. Roy, and N. Mukherjee, "Detection of node failure in wireless sensor networks," in *Applications and Innovations in Mobile Computing (AIMoC), 2014*. IEEE, 2014, pp. 133–138.

[28] S. Dubois and S. Tixeuil, "A taxonomy of daemons in self-stabilization," *arXiv preprint arXiv:1110.0334*, 2011.