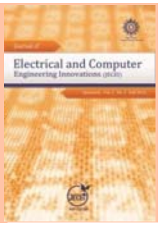
SRTTU

Journal of Electrical and Computer Engineering Innovations

JECEI, Vol. 2, No. 1, 2014

Regular Paper



DALD: Distributed Asynchronous Local Decontamination Algorithm in Arbitrary Graphs

Maryam Rahmaninia[1,*], Elnaz Bigdeli[2], and Manouchehr Zaker[3]

[1]Department of Computer Engineering, Islamic Azad University, Ghasr-E-Shirin Branch, Kermanshah, Iran.

[2]Electrical Engineering and Computer Science, University of Ottawa, Ontario, Canada.

[3]Mathematics and Computer Science Department, Institute for Advanced Studies in Basic Sciences, Zanjan, Iran.

[*]Corresponding Author's Information: ma.rahmaninia@gmail.com

ARTICLE INFO

ARTICLE HISTORY:
Received 15 February 2014
Revised 27April 2014
Accepted 8 May 2014

KEYWORDS:
Network decontamination
Mobile agents
Distributed algorithm

ABSTRACT

Network environments always can be invaded by intruder agents. In networks where nodes are performing some computations, intruder agents might contaminate some nodes. Therefore, problem of decontaminating a network infected by intruder agents is one of the major problems in these networks. In this paper, we present a distributed asynchronous local algorithm for decontaminating a network. In most of prior algorithms, there is a coordinator agent that starts from a node and decontaminates the network. Since this procedure is handled by an agent and in centralized mode decontamination algorithm is very slow. In our algorithm, the network is decomposed to some clusters and a coordinator is advocated to each cluster. Therefore, there is more than one coordinator that each of them starts from different nodes in the network and decontaminates network, independently. In this case, network is decontaminated faster. In addition, in previous works the upper bound of the number of moves and the number of cleaner agents required to decontaminate network are given only for networks with special structures such as ring or tori while our algorithm establishes these upper bounds on networks with arbitrary structure.

1. Introduction

A particularly important security concern in a network is to protect it from unwanted and possibly dangerous intrusions. At an abstract level, an intruder is an alien process that moves on the network to sites unoccupied by the systems agents and contaminating the nodes while it passes by. In such cases, a primary task is to decontaminate the infected network. There are two main types of decontamination: internal decontamination and external decontamination. In internal decontamination, local faulty behavior can be repaired by local majority mechanism with possibly different rules which are applied only to contaminated nodes [1]. In the external decontamination of network, a team of mobile agents is deployed to decontaminate a network. The network decontamination problem has been studied extensively under various terms like intruder capture, decontamination, and graph search [2]–[4]. Beside network security, the decontamination problem has many other applications. Consider the leader of a team of agents who searches a person, moving object, or contaminant in a building. Such a scenario may occur in urban search and rescue [5] and military operations [6].

The remainder of this paper is organized as follows. We first discuss related works in Section 2. Backgrounds and problem statement are presented in Section 3. We then define the overview of the proposed algorithm in Section 4. Section 5 contains a method for decomposing the network and the analysis of the proposed algorithm and the proof of its correctness are presented in Section 6. The computational complexity and the upper bound for the number of moves and the number of cleaner agents required to decontaminate a network are given in Section 7. Finally, we conclude and discuss avenues for future work in Section 8.

2. Related Works

The network decontamination problem was first proposed by Breisch in which the goal is to decontaminate network using few number of agents [7]. In decontamination problem, the goal is typically to devise a strategy for agents to collaboratively decontaminate the whole network using the smallest possible team of agents or in graph search the aim is to find a strategy that minimizes the number of searchers [8]. Finding the minimum number of agents or searches is NP-hard for an arbitrary graph [9]. Finding the optimal number of agents and searches are investigated on special structures such as mesh, tori and ring [2]-[10]. So far, there is not any algorithm which optimizes time complexity and number of agents simultaneously in any arbitrary graphs [11]. Nevertheless, decontamination is successful in various domains since many issues in this problem are related to graph concepts such as cut-width and graph minors. For example, minimum number of search in a graph is equal to cut-width of a graph with maximum degree 3 [12]. Other example is a pebbling problem in graphs [13].

Decontamination problem is investigated by considering various assumptions. There are a bunch of studies in graph search in which agents are able to jump across the network. This kind of search is called non-contiguous search, but this search is not valid in many domains [14]. On the other side, there are lots of studies in which agents cannot jump in a network and cannot be removed from network. This kind of search is called contiguous search [6]. It has been proved that the contiguous searching number is always greater or equal to then on contiguous searching number [15]. In some recent works, there are some new and interesting assumptions in decontamination context. For example, the propagation patterns of faults can follow different dynamics, depending on the behavior of the affected nodes, and topology of the network. At one extreme, we have a full spread behavior: when a site is affected by a virus or any other malfunction, such a malfunction can propagate to all its neighbors; in other cases, faults propagate only to sites that are susceptible to be affected; the definition of susceptibility depends on the application but oftentimes it is based on local conditions, for example, a node could be vulnerable to contamination if a majority of its neighbors are faulty, and immune otherwise [16]-[17] or it could be immune to contamination for a certain amount of time after being repaired [18].

Some further works in the same model was done in [19], where a two dimensional lattice is considered.

In this paper, we introduce a distributed algorithm which decomposes graph to sub-graphs and decontaminates nodes in each sub-graph. In all previous papers, cleaner agents start from a home-base to decontaminate the graph but in our algorithm there is more than one home-base [20]-[21]. This algorithm decontaminates the network in an acceptable time complexity and number of agents. The proposed algorithm for decontamination in this paper can be a proper step toward finding algorithms which solve decontamination problems in optimal time with optimal number of agents.

3. Background and Problem Statement

Network is modeled as a simple undirected connected graph $G = (V, E)$. The network structure is considered arbitrary in this paper. Here, nodes are colored black or white. A node is black if it is contaminated and otherwise it is white. A contamination rule is a local majority-based rule applied to white nodes only. Color of a node is changed at discrete time steps on the basis of the majority of colors held by its neighbors. Updating is performed simultaneously at discrete time steps by all nodes subject to majority voting.

In this paper, there are two types of agents which have different capabilities. The first type of agents is cleaner agent which we call it from then a cleaner. Cleaner agents can clean the infected node and have visibility ability. It means that the cleaner agent can be aware of the state of its direct neighbors. The other type of agents is coordinator agent which we call it from then coordinator. A coordinator agent moves in the network and decides about allocating a cleaner to a node. Each agent in the network has unique identifier and is distinguished from others using this identifier. Moreover, agents can move in network from a node to another node which has direct link to it and they cannot jump to the other nodes. At any point in time each node of the network can be in one of three possible states: clean, contaminated, or guarded. A node is guarded when it contains at least one agent, clean when a cleaner agent has been on the node and the majority of its neighbors are clean or guarded, and is contaminated otherwise.

A cleaning strategy used by agents should guarantee that after a finite amount of time all nodes are clean and it should be monotone. A strategy is called monotone if it guarantees that after decontaminating a node, it will not be re-contaminated. Initially all nodes are contaminated except for the home-bases, which are obviously guarded. We consider there are enough cleaners in network and they cannot be removed from the network. As a matter of fact, contamination and decontamination processes occurring simultaneously in a network. Their interaction creates a dynamic of faults propagation and nodes mending.

4. Decontamination Algorithm

In this section we introduce Distributed Asynchronous Local Decontamination (DALD), an algorithm which can decontaminate an arbitrary graph. DALD has four phases that we describe below:

1. **Selecting home-base nodes:** At first, some nodes are chosen randomly and are considered as home-base nodes. Home-base nodes are those nodes that coordinator agents with unlimited number of cleaners are located there.
2. **Clustering:** The graph is divided into some clusters. The home-base nodes are the center of each cluster. Clusters are developed using a t-distance method which guarantees to comprise the whole graph.
3. **Tree construction:** In each cluster, a breadth-first traversal is used to construct a tree which uses a lexical order. The constructed trees guarantee the monotonicity of the decontamination algorithm.
4. **Cleaning:** All coordinators in all clusters decontaminate at the equivalent trees of their clusters in parallel.

The first three phases are executed only once in the beginning of the algorithm. The cleaning phase is executed until the network is decontaminated totally. All phases are discussed in more detail in the following sections.

5. Clustering

Graph clustering is the task of grouping vertices of the graph into clusters, taking into consideration the edge structure of the graph in such a way that there should be many edges within each cluster and relatively few between the clusters [22].

There are various ways to cluster graphs with different properties. We cluster graph using *t*-distance property.

Definition 1: For a pair of nodes n_i and n_j, let$T(n_i, n_j)$ be the shortest distance between them in the graph. Let$\varphi_t(n_i) = \{n_j | T(n_i, n_j) \leq t, n_i, n_j \in V\}$denotes a set of nodes that can be reached from n_i within *t* hops.

According to this definition, above nodes in distance of *t* hops from a home-base node is in the cluster of that home-base. Nodes in distance of $t + 1$hops from home-base nodes are boundary nodes. Boundary nodes are nodes which belong to other clusters and the coordinator can obtain information about their status. As a matter of fact, boundary nodes guarantee the convergence of algorithm in each cluster and consequently the convergence of the algorithm in the whole graph.

Example 1: Consider the graph as shown in Figure 1. Let's consider nodes n_1 and n_{12} are home-base nodes. Clusters for these two nodes with *t=2* are

$$\varphi_2(n_1) = \{n_1, n_2\ , n_3, n_4, n_5, n_{13}\},$$
$$\varphi_2(n_{12}) = \{n_6, n_7\ , n_8, n_9, n_{10}, n_{11}, , n_{12}\}$$

Boundary nodes for clusters n_1 and n_{12} are $\{n_6, n_7, n_8, n_9\}$, $\{n_4, n_5, n_{13}\}$ consecutively.

Determination of Parameter t: The key point in clustering graphs is to determine *t* in a way that each node belongs to one cluster at least. To this end, we specify the value of *t* in a way that the whole graph is covered by our clustering method

Definition 2: Consider a graph $G = (V, E)$ with $|V| = n$ and with home-bases $\{H_1, \ldots, H_h\}$. $D_i = min\{T(n_i, H_1), T(n_i, H_2), \ldots, T(n_i, H_h)\}$ indicates distance of node n_i of the nearest home-base. Parameter *t* is determined by finding the maximum of all minimum distances of all nodes in the graph, $t = max\{D_1, \ldots, D_n\}$.

5.1 Tree constructor

In graph theory, breadth-first search (BFS) is a graph search algorithm that starts from arbitrary node in a graph and makes it the root node and explores all neighbor nodes [23]. Then, for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it visits all the nodes. The order of visiting neighbors is considered the lexical order in which the nodes with smaller ID are visited first. For the graph in the Figure 1, equivalent trees for clusters $n_1$and n_{12} are depicted in Figure 2.

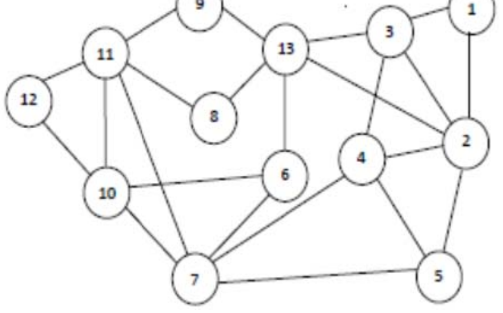


Figure 1: An example for a graph with 13 nodes.

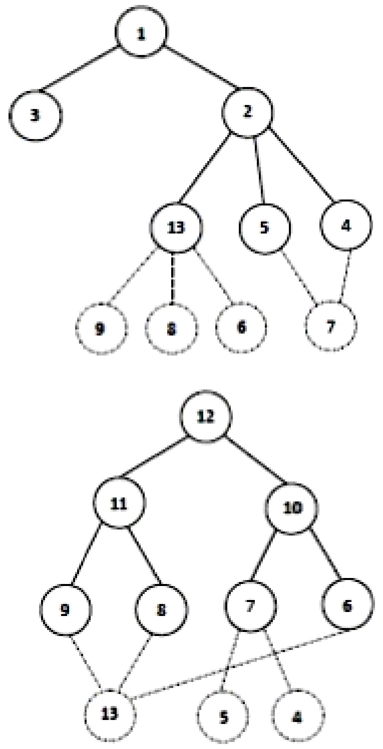


Figure 2: Equivalent tree for clusters with home-bases n_1 and n_{12}.

5.2 Cleaning

In each sub-graph, the home-base node is the root of BFS tree and the start node for our algorithm. In each sub-graph, there is a coordinator which tries to decontaminate the sub-graph. We assume that coordinator has enough cleaners to clean the sub-graph.

A coordinator starts from the root and traverses the graph in BFS order. It means that coordinator cleans all the nodes in level l and then it goes to the level $l + 1$ and it guarantees that by moving to the next level the cleaned nodes will not be contaminated again.

Below, the DALD algorithm is described with more details.

DALD Algorithm:

– Cleaning from level 0 to level 1:

- •Coordinator sends cleaners to all k offsprings of the root node.

– Cleaning offsprings of nodes in levels $l > 0$:

- • Let's assume that the constructed tree is cleaned up to level l. To clean nodes in level $l + 1$ coordinator returns back to the first node in lexical order in level l and determines the number of the node offsprings. Consider coordinator is in node n_i in level l. If the node n_i has k offsprings and the majority of its boundary neighbors are contaminated, coordinator moves to the root and advocates k cleaners to clean all children. Otherwise, if the majority of boundary neighbors are clean, the coordinator advocates $k - 1$ cleaners to clean offsprings using the cleaner in node n_i.
- • All cleaners in node n_i are sent to k offsprings of node n_i to clean these nodes. According to the previous description, if the majority boundary neighbors of n_i are contaminated, a cleaner is retained in node n_i.

  That is why coordinator requests k nodes when the majority of nodes are contaminated. On the other side, if the majority of boundary neighbors are not contaminated coordinator request $k-1$ nodes because it can use the cleaner in node n_i.
- • Whenever the coordinator settles in a leaf node, if all boundary neighbors are cleaned, cleaner in this node is sent back to the home-base.

This algorithm is executed in parallel in all clusters.

Example 2: Let's consider all nodes in the graph in Figure 1 are contaminated. We choose n_1 and n_{12} as home-base nodes, randomly.

Based on description in Section 5, parameter t is set to 2.

The equivalent trees for clusters n1 and n12 are constructed.

Trees for these two clusters are depicted in Figure 2. Coordinators start from n1 and n12 with unlimited number of cleaners in these two nodes. Hence, nodes n1 and n12 are cleaned by cleaners at first as it is clear in Figure 3a.

Then, coordinators send cleaners to the children of n1 and n12 simultaneously as it is shown in Figure 3b.

Then, each coordinator finds the first node in lexical order in the level 1 to move on. For cluster n1, this node is n2 and for cluster n12 this node is n10. Node n2 has three children and node n10 has two children.

Coordinators in each cluster move back to home-bases and bring 2 and 1 cleaners to clean offspring nodes of nodes n2 and n10 and as a result all children of n2 and n10 are clean as shown in Figure 3c.

Since nodes n2 and n10 are not treated by any nodes in the graph, coordinators use cleaners in nodes n2 and n10 and send cleaners to offspring nodes.

Algorithm runs in the same manner and coordinators move to n3 and n11 and send cleaner to their children and now the network is clean as depicted in Figure 3d.

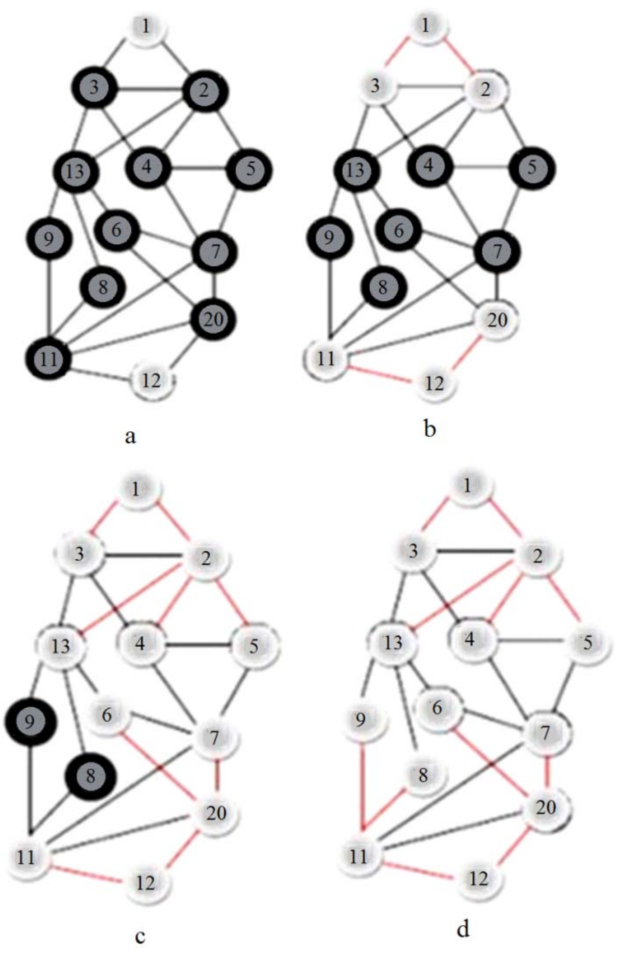


Figure 3: An example of algorithm on a graph with 13 nodes and 2 home-basesn_1 and n_{12}.

6. Algorithm Analysis

We now prove the correctness of DALD algorithm. It is shown that all nodes in the network will be cleaned and once a node has been cleaned it will never be re-contaminated.

Let's n_i be a node in level l; Let $\Gamma(i)$ denotes the neighbors of n_i in the graph $G = (V, E)$:

$$\Gamma(i) = \{n_j | (n_i, n_j) \in \mathrm{E}\} \tag{1}$$

Let $\Gamma'(i)$ denotes children of n_i at level $l + 1$ in the broadcasted tree. It is clear that$\forall\ i\ \Gamma'(i) \subseteq \Gamma(i)$

The following lemmas and theorem prove the correctness of our algorithm.

Lemma 1: All nodes in a cluster which stand on level $l< t$ in constructed tree, do not have boundary nodes.

Lemma 2: If n_i is a node in level l and n_j is a node in level $l + 1$ and $n_j \in \Gamma(i)\backslash\Gamma'(i)$then there is a node $\mathrm{n_m}$ in level l such that$n_j \in \Gamma'(m)$.

Lemma 3: In DALD algorithm, when an agent leaves unguarded node n_i at level l, all the neighbors of n_i are either clean or guarded.

Proof: This is clearly true for the node at level 0. Assume that it is true for all nodes at level $0 \le l < L$ and we prove it's true for nodes in level L. It is obvious that a node is contaminated by its neighbors. All neighbors of node n_i in level L can be one of the following nodes: nodes in level $L - 1$, nodes in level L, nodes in level $L + 1$, nodes in the boundary of group.

According to the algorithm, the constructed tree is cleaned level by level. Therefore, when coordinator is in level L, it cleans all nodes in the previous levels. As a result, a node in level L won't be contaminated by nodes in the previous level. According to the breadth-first traverse of graph, all the neighbors of node n_i in level L are traversed before or after this node. In the algorithm, all nodes in tree which are traversed before n_i are cleaned by coordinator before n_i. Thus, there is no treat by these nodes for n_i. As it is mentioned, cleaning algorithm cleans tree level by level and cleaners in level L leave nodes in this level if all nodes of this level has been cleaned. Therefore, node n_i will not be contaminated by its neighbors in level L.

Neighbor n_j of node n_i which is in level $L + 1$ can be in two states. The n_j can be a node in level $L + 1$ that$n_j \in \Gamma(i)\backslash\Gamma'(i)orn_j \in \Gamma'(i)$. If n_j is a node such that $n_j \in \Gamma(i)\backslash\Gamma'(i)$ according to the lemma 2 there is a node n_m which is in level L traversed before node n_i and n_j is the child of n_m. Node n_j is decontaminated by node $\mathrm{n_m}$. On the other side, if n_j is a node such that$n_j \in \Gamma'(i)$ according to phase 2 the coordinator sends sufficient cleaners to this nodes.

The cleaner node can get the information from the nodes which are in the boundary of graph. A node in a cluster may be contaminated by boundary nodes. Therefore, coordinator checks all the boundary neighbors of the agent. If the majority of boundary neighbors of the current node are contaminated the coordinator decides to preserve cleaner in the node, but if the majority boundary neighbors are clean or guarded it removes cleaner from this node. Finally, it can be concluded that using this algorithm a cleaned node is not treated by its neighbors because they are clean or guarded.

Theorem 1: The proposed algorithm decontaminates all nodes.

Proof: According to the lemma 3, a node is not contaminated after cleaning. In each cluster, the algorithm decontaminates the sub-graph level by level. As a result, if the coordinator reaches to the leaf nodes in tree, it cleans all nodes in that sub graph. Since, all nodes belong to at least one group, it should be concluded that the whole graph will be cleaned in finite time.

7. Complexity Analysis

In all decontamination algorithms, the number of moves and the number of agents to decontaminate the network are important parameters. In this section, we give the upper bound for these parameters.

Theorem 2: The number of cleaner agents to clean a graph *G = (V, E)* with maximum degree, distance *t* to construct clusters and *h* home-bases is $O(h\Delta^t)$.

Proof: According to lemma 1, the middle nodes do not have boundary neighbors and based on the previous lemmas and theorems, these nodes are not treated by other neighbors because they are decontaminated or guarded. As a result, all cleaners move to leaf nodes and the number of cleaners is equal to the number of leaf nodes. In a tree with maximum degree *Δ*, this value is $O(\Delta^t)$. In the worst case, all cleaners stay on leaf nodes to prevent from re-contamination by boundary nodes. Thus, in each cluster at most $O(\Delta^t)$cleaners are used. As a result, for all *h* home-bases the total number of cleaners is $O(h\Delta^t)$.

Theorem 3:In a graph *G = (V, E)* with arbitrary structure let *|V | = n* and distance *t* is used to construct clusters and *h* is the number of home-bases. The number of coordinators moves to construct the graph is $O(ht^2\Delta^t)$

Proof: To count the number of moves in the graph, we consider the number of moves by coordinators and cleaners altogether. To simplify the problem, we first consider the number of moves in each cluster. At first, we describe the number of moves by cleaners then we count the number of moves by coordinators.

Number of moves by cleaners: As mentioned before, there are *h* clusters constructed by distance *t*. The maximum distance from the root to the farthest node in tree is *t*. So, the farthest distance that each cleaner should traverse is *t*. As explained in the last phase of algorithm, after cleaning a leaf node, the cleaner returns back to the root. Consequently, the cleaner can have *2t* moves at most.

Based on lemma 1, all the nodes in level *l*, $l < t$ are not connected to the boundary nodes. In this case, cleaners in nodes in level $l < t$ can leave the nodes to upper level in tree. By this description, the number of cleaners used to decontaminate a cluster is equal to the number of cleaners in level *t*. We have at most Δ^tcleaners in level *t*. Therefore, the number of moves by all cleaners in a cluster is at most $O(2t\Delta^t)$.

Number of moves by coordinators: Based on the algorithm, a coordinator visits all the nodes in the cluster to clean the cluster. To visit the nodes in level *l + 1* a coordinator should visit all nodes in level *I* that *i< l + 1*. It means that to visit nodes in upper levels the nodes in lower levels should be visited again. In this way all the number of moves for a coordinator is:

$$2\sum_{i=1}^{t} i \times \Delta^i \leq t^2\ \Delta^t \qquad (2)$$

Now, we have the number of moves by cleaners and coordinators. The upper bound for the number of moves in a graph with *h* cluster is $O(ht^2\ \Delta^t)$

8. Conclusions

In this paper, we decontaminated a network with arbitrary structure using a distributed algorithm. To decontaminate a network distributive, it is decomposed to clusters. A clustering method clusters nodes in network by t-distance method. Since, there is more than one home base in the network, decontamination is very fast. This algorithm converges in finite time and decontaminates all nodes finally. In the previous works, the upper bound of the number of moves and the number of cleaner agents required to decontaminate network are given only for networks with special structures such as ring or tori while our algorithm establishes these upper bounds on networks with arbitrary structure. The upper bound for the number of moves and the number of cleaners to clean the network in arbitrary structure are $O(ht^2\ \Delta^t)$ and $O(h\ \Delta^t)$, consecutively. In future works we will use a clustering method by which the overlaps among groups and the number of moves and cleaners are decreased as well.

References

[1] P. Flocchini, "Contamination and decontamination in majority-based systems," Journal of Cellular Automata, Vol. 4, No. 3, pp. 183-200, 2009.

[2] P. Flocchini, M.J. Huang, and F.L. Luccio, "Decontaminating chordal rings and toriusing mobile agents," International Journal of Foundations of Computer Science,Vol. 18, No. 3, pp. 547-564, 2007.

[3] F.L. Luccio, "Intruder capture in Sierpinski graphs," Proc.of the 4th International Conference on Fun.with Algorithms (FUN), Lecture Notes in Computer Science, 2007, 4475, pp. 249-261.

[4] T. Parson, "The search number of a connected graph," Proc.of the 9th Southeastern Conference on Combinatory, Graph Theory and Computing, 1978, pp. 549-554.

[5] V. Kumar, D. Rus, and S. Singh, "Robot and sensor networks for first responders," IEEE Pervasive Computing, Vol. 3, No. 4, pp. 24-33, 2004.

[6] L. Barrire, P. Flocchini, ,P. Fraigniaud, and N. Santoro, "Capture of an intruder by mobile agents," Proc. of the 14th ACM Symposium on Parallel Algorithms and Architectures, 2002, pp. 200-209.

[7] R. Breisch, "An intuitive approach to speleotopology," Southwestern Cavers VI, 1967, (5), pp. 72-78.

[8] N. Megiddo, S. Hakimi, M. Garey, D. Johnson, and C. Papadimitriou, "The complexity of searching a graph," Journal of the ACM (JACM),Vol. 35, No. 1, pp. 18-44, 1988.

[9] A. LaPaugh, "Recontamination does not help to search a graph," Journal of the ACM(JACM), Vol. 40, No. 2, pp. 224–245, 1993.

[10] P. Flocchini, F.L. Luccio, and L. Song, "Size optimal strategies for capturing an intruder in mesh networks," Proc. of the International Conference on Communications in Computing, 2005, pp. 200-206.

[11] P. Flocchini, M.J. Huang, and F.L. Luccio, "Decontamination of hypercubes by mobile agents," Networks, Vol. 52, No. 3, pp. 167-178, 2008.

[12] F. Makedon, I. H. Sudborough, "On minimizing width in linear layouts," Journal of Discrete Applied Mathematics, Vol. 23, No. 3, pp. 243- 265, 1989.

[13] L. Kirousis, C. Papadimitriou, "Searching and pebbling," Theoretical Computer Science, Vol. 47, pp. 205-218, 1986.

[14] T. Parson, "Pursuit-evasion in a graph,"Theory and Applications of Graphs, Lecture Notes in Mathematics, Springer-Verlag, 1976, pp. 426-44.

[15] L. Barri‘ere, P. Fraigniaud, N. Santoro, and D.M. Thilikos, "Searching is not jumping," 29th Workshop on Graph Theoretic Concepts in Computer Science (WG), LNCS2880, 2003, pp. 34-45.

[16] F. Luccio, L. Pagli, and N. Santoro, "Network decontamination with local immunization," International Journal of Foundation of Computer Science, Vol. 18, No. 3, pp. 457– 474, 2007.

[17] S. Kutten, D. Peleg, "Tight fault locality," SIAM Journal on Computing, Vol. 30, No. 1, pp. 247–268, 2000.

[18] P. Flocchini, M.J. Huang, and F. L. Luccio, "Decontamination of chordal rings and tori," Proc. of the 8th Workshop on Advances in Parallel and Distributed Computational Models (APDCM), Rodi, Greece, 2006.

[19] Y. Daadaa, P. Flocchini, and N. Zaguia, "Decontamination with temporal immunity by mobile cellular automata," Proc. of the International Conference on Scientific Computing (CSC), 2011, pp. 172–178.

[20] P. Flocchini, B. Mans, and N. Santoro, "Tree decontamination with temporary immunity," Proc. of the 19th International Symposium on Algorithms and Computation (ISAAC), 2008, pp. 330-341.

[21] P. Flocchini, N. Santoro, "Network decontamination from a black virus,"Proc. of the 27th IEEE International Parallel and Distributed Processing Symposium Workshops & Ph.D. Forum (IPDPSW), 2013, pp. 696 – 705.

[22] S.E. Schaeffer, "Graph clustering," Computer Science Review, 2007, pp. 27- 64.

[23] D.G. Corneil, "Lexicographic breadth first search – a survey," Proc. of the 30th International Workshop on Graph-Theoretic Methods in Computer Science, 2004, pp. 21-23.

BIOGRAPHIES

Maryam Rahmaninia was born in Ghasr-e-shirin, Kermanshah, Iran, in Aug. 1985. She received her B.Sc. degree in Computer Science from Shahid Beheshti University, Tehran, Iran, in 2008 and M.Sc. degree in Computer Science from Institute for Advanced Studies in Basic Sciences, Zanjan, Iran in 2010. Her interests are in neural network, theoretical computer science and multi agent systems.

Elnaz Bigdeli is Ph.D. student at Department of Computer Science & Engineering at the University of Ottawa. She received M.Sc. degree in Computer Science in Institute for Advanced Studies in Basic Science (IASBS) in 2011. She completed her B.Sc. degree in Information Technology Engineering in (IASBS) in 2008. Her research interests lie in the area of machine learning and data mining. In recent years, she has focused on stream data clustering. She has collaborated actively with researchers in other disciplines of computer science, particularly social networks and distributed systems.

Manouchehr Zaker is currently associate professor of Mathematics in Institute for Advanced Studies in Basic Sciences, Zanjan, Iran. He got B.Sc. degree from Tabriz University in 1994. Then, he got M.Sc. and Ph.D. degrees in Mathematical Sciences from Sharif University of Technology, Tehran, Iran, in 1997 and 2001, respectively. His research areas are Graph Theory and Combinatory, Algorithmic Graph Theory and Social and Complex Networks studies using Graph Theory.