Yugoslav Journal of Operations Research
xx (20xx), Number xx, xxx–xxx
DOI: https://doi.org/10.2298/YJOR230815016K

INTEGER PROGRAMMING MODEL FOR DISTANCE-EDGE-MONITORING PROBLEM

Aleksandar KARTELJ
Faculty of Mathematics, University of Belgrade,
Studentski trg 16, 11 000 Belgrade, Serbia
aleksandar.kartelj@gmail.com

Vladimir FILIPOVIĆ
Faculty of Mathematics, University of Belgrade,
Studentski trg 16, 11 000 Belgrade, Serbia
vladofilipovic@hotmail.com

Jozef KRATICA
Mathematical Institute, Serbian Academy of Sciences and Arts,
Kneza Mihaila 36, 11 000 Belgrade, Serbia
jkratica@mi.sanu.ac.rs

Received: August 2023 / Accepted: April 2024

Abstract: The paper considers the recently introduced distance-edge-monitoring problem. For a given graph $G = (V, E)$, the set M is called distance-edge-monitoring if it is a subset of V and for every edge e of E there is a vertex x of M and a vertex y of V such that e belongs to all the shortest paths between x and y. The goal is to find the smallest distance-edge-monitoring set of the graph. We propose the first-ever integer programming model for this problem and present empirical results for five instance classes with graphs up to 4096 vertices and 1599797 edges. The proposed model was able to solve all instances of four instance classes optimally, while the remaining fifth class of graphs proves to be more difficult for the proposed model.

Keywords: Distance-edge-monitoring problem, network monitoring, *dem* number, integer programming, combinatorial optimization.

MSC: 90C10, 90C27.

1. INTRODUCTION

The concept of *distance-edge-monitoring* was introduced in [1]. The motivation arises from computer networks, where the vertices represent computers/routers and the edges represent communication links. It is well known that computer networks are complex systems that use heterogeneous devices and interconnection technologies, such as optical fibers, copper wires, wireless microwave transmission, etc. Using explicit mechanisms of detecting link failures requires expensive measurement equipment. An alternative approach would be to implicitly detect link failures based on the routing distances readily available to the computers/routers, e.g. via the *traceroute* command. More specifically, goal is to localize the broken communication link when the failure occurs. This is done by selecting a subset of vertices, called *probes*, that can measure the distance (the length of the shortest path) to any other vertex in the network. Once an edge fails, some of the distances change, which in turn allows the probes to detect this situation. Real-world applications of such probes are reported in papers on routing and network verification [2, 3, 4, 5, 6].

In this paper, we provide an exact procedure (integer linear programming model) to propose the probe set with the smallest cardinality, i.e. the one with the cheapest price. The applicability of this approach is demonstrated on random graphs that resemble some real networks, as shown in this paper. However, we emphasize that this approach does not target a specific type of graphs and is therefore applicable in a general case.

Let $G = (V(G), E(G))$ be a simple connected graph. By $d_G(x, y)$ we denote the distance in the graph, i.e., the length of the shortest path between the vertices x and y.

Definition 1. *Given a set $M \subseteq V$ of vertices and an edge $e \in E$, let $P(M, e)$ be the set of pairs (x, y) such that $x \in M$ and $y \in V$ and $d_G(x, y) \neq d_{G-e}(x, y)$, i.e., removing the edge e from G changes the distances between x and y. In other words, that the edge e must belong to all shortest paths between x and y.*

Definition 2. *Given a vertex x, $EM(x)$ is the set of edges e such that $(x, y) \in P(x, e)$ for any $y \in V$. In other words, the edges of $EM(x)$ are all monitored by x.*

Definition 3. *A set $M \subseteq V(G)$ is called distance-edge-monitoring set if every edge $e \in E$ is monitored by some vertex of M. More precisely, $\bigcup_{x \in M} EM(x) = E$. The cardinality of the minimal M is called the distance-edge-monitoring number of the graph and is denoted by $dem(G)$.*

Note that V is itself a distance-edge-monitoring set. From the previous definition, it can be seen that the selection of probes using the distance-edge-monitoring set guarantees the detection of edge defects. This is because each edge belongs to all the shortest paths between certain pair of vertices, one of which is within the probe set. Therefore, the faulty edge will increase the distance to at least one other vertex observed by this probe. In [1], the authors even proved that it is possible to determine exactly which edge has failed.

Another result from this work, important for our model, is the following lemma:
Lemma 1. *([1]) Let x be a vertex of a connected graph G. Then, an edge $\{u, v\}$ belongs to $EM(x)$ if and only if $u \in L_i(x)$ and v is the only neighbor of u in $L_{i-1}(x)$, for some integer i, where $L_i(x)$ denotes the set of vertices at distance i of x [1].*

2. RELATED WORK

Distance-edge-monitoring is a relatively new concept in the literature. The 2022 paper [1] formally introduces the problem. Since then, a number of papers have appeared that examine the problem from both theoretical and practical perspectives.

In [7] the authors propose lower and upper bounds for Cartesian graph products, and they also characterize the graphs that reach these lower and upper bounds. Moreover, they prove the exact *dem*-values for join, corona, cluster and some other specific graphs. Paper [8] gives some lower and upper bounds for $P(M, e)$, $EM(x)$, and $dem(G)$, and characterizes extreme graphs that reach these bounds. It also characterizes graphs with $dem(G) = 3$. In [9] it has been shown that $dem(G \setminus e) - dem(G) \leq 2$. There are some other results in this paper, for example, the authors propose the algorithm that evaluates whether the distance-edge-monitoring set still remains in the resulting graph when an edge of the graph is deleted. Paper [10] is concerned with lower bounds of distance edge monitoring number for hierarchical and corona graphs. In [11], the author proposes an algorithm for solving the distance-edge-monitoring problem. The algorithm is parameterized by the vertex cover number and feedback edge set number. This is not the first algorithm for the *dem* problem, as the authors of the original paper [1] have already proposed a polynomial algorithm based on the set cover that approximates within a factor of $\ln(|E(G)| + 1)$.

It should be noted that there is also an ongoing research [12, 13, 14] on a similar problem called edge-geodetic-monitoring (or monitoring of edge-geodetic sets). Here the goal is to find, for the given graph $G(V, E)$, a set $S \subseteq V$ such that for every $e \in E$ there is a pair $(x, y) \in S \times S$ that it monitors. The difference with distance-edge-monitoring is that both endpoints are in the selected subset S. The corresponding size of the smallest such S is called the monitoring edge-geodesic set number or $meg(G)$ for short.

3. THE PROPOSED INTEGER PROGRAMMING MODEL

The proposed integer programming model has the following form:

$$\min \sum_{v \in V} x_v \tag{1}$$

s.t.

$$\sum_{v \in V} C_{v,e} \cdot x_v \geq 1, \quad for\ e \in E \tag{2}$$

$$x_v \in \{0, 1\}, \quad for\ v \in V \tag{3}$$

The variable x_v for $v \in V$ takes the value 1 if the vertex v is within the distance-edge-monitoring set, otherwise it takes the value 0. Thus, if $x_v = 1$ for some $v \in V$, then $v \in M$. Clearly, the objective function (1) corresponds to the formulation of a distance-edge-monitoring set problem.

The constant $C_{v,e}$ for $(v, e) \in V \times E$ takes the value 1 if for a pair (v, e) there exists a vertex $u \in V$ such that e belongs to all the shortest paths between vertices v and u. Effectively, $C_{v,e} = 1$ when $e \in EM(v)$ and conversely $C_{v,e} = 0$ when $e \notin EM(v)$. Constraints (2) therefore enforce the distance-edge-monitoring condition, i.e., for every $e \in E$ there must be at least one $v \in M$ that monitors it.

Due to the small number ($|V|$) of binary variables and only $|E|$ constraints, the model can be solved quite well with ILP solvers, such as CPLEX. Note that for large graphs, the direct computation of the $C_{v,e}$-matrix can be very time consuming. Fortunately, we have used the Lemma 1 to improve the efficiency of this preparation phase. The pseudocode of the C-matrix calculation procedure is shown in Algorithm 1.

Algorithm 1 Calculation of C-matrix

```
C_{|V(G)|×|E(G)|} ← false_{|V(G)|×|E(G)|}
d_G ← FloydWarshall(G)
for e = {u, v} ∈ E(G) do
    for x ∈ V(G) do
        if d_G(x, v) = d_G(x, u) then
            continue
        end if
        if d_G(x, v) = d_G(x, u) + 1 then
            x_monitors ← true
            for w ∈ V(G) \ {u} do
                if d_G(x, v) = d_G(x, w) + 1 then
                    x_monitors ← false
                    break
                end if
            end for
            if not x_monitors then
                continue
            end if
        end if
        if d_G(x, u) = d_G(x, v) + 1 then
            x_monitors ← true
            for w ∈ V(G) \ {v} do
                if d_G(x, u) = d_G(x, w) + 1 then
                    x_monitors ← false
                    break
                end if
            end for
            if not x_monitors then
                continue
            end if
        end if
        C_{x,e} ← true
    end for
end for
```

Namely, for each edge $e = \{v, u\}$ and each vertex x in graph G, one needs to compare the distances $d_G(x, v)$ and $d_G(x, u)$. If $d_G(x, v) = d_G(x, u)$, the vertex x cannot monitor the edge e because if the optimal path passes through this edge, the distances to its endpoints cannot be equal (one distance should be greater than the other by one). Therefore, the remaining interesting cases are when $d_G(x, v) = d_G(x, u) + 1$ or $d_G(x, v) + 1 = d_G(x, u)$. For these two cases, one should check whether v is the only neighbor of u at distance $d_G(x, u) - 1$ from x and, conversely, whether u is the only neighbor of v at distance $d(x, v) - 1$ from x. For the calculation of the distances between the vertices we used the Floyd-Warshall algorithm, with a computational complexity of $O(|V(G)|^3)$. As can be seen from the presented pseudocode, the total computational complexity is $O(|V(G)|^2 \cdot |E(G)|)$.

4. EXPERIMENTAL RESULTS

In this section, we present the results obtained by solving the proposed IP model using IBM CPLEX 22.1.0 solver in 64-bit mode with 16 threads. All tests were performed on an Intel i9-9900KF CPU @3.6 GHz with 64 GB of memory running the Windows 10 operating system. The maximum runtime was set to one hour.

Instance classes are shown in Table 1, where each row corresponds to an instance class, while the columns contain the full name, abbreviation, number of instances, minimum and maximum number of vertices, and minimum and maximum number of edges. All instances, except the random ones, are already known from the literature. They are all available in the public GitHub repository dedicated to the paper `https://github.com/kartelj/distance_edge_monitoring_public`. This repository also contains binaries and scripts needed to run and reproduce the results reported in this paper.

Table 1: Instance classes characteristics

Instance class	Abbreviation	#Instances	$\|V\|_{min}$	$\|V\|_{max}$	$\|E\|_{min}$	$\|E\|_{max}$
Crew scheduling	`mcsp`	10	50	500	173	16695
Graph coloring	`mgcol`	30	100	300	2420	22601
Pseudo boolean	`frb`	40	450	1534	17794	127011
Hypercubes	`hypercube`	12	2	4096	1	24576
Erdős-Rényi	`random`	24	200	2000	539	1599797

The tables 2-6 contain results for tested instance classes. Each of these tables is organized in the same way, with rows corresponding to different instances within their instance class, while columns correspond to the instance name (label), the number of vertices, the number of edges, the *dem* value obtained from CPLEX, the CPLEX runtime, and the CPLEX exit status (*opt* means that optimality was verified, while empty "" means that the time limit was reached).

Table 2 contains the results for `mcsp` instances. It can be observed that CPLEX found all optimal solutions in 5 seconds or less.

Table 2: DEM results for `mcsp` instances

	Dimension		CPLEX		
Instance	$\|V\|$	$\|E\|$	*dem*	T (sec)	Status
mcsp50	50	173	23	0.047	opt
mcsp100	100	715	58	0.178	opt
mcsp150	150	1355	83	0.391	opt
mcsp200	200	2543	116	0.653	opt
mcsp250	250	4152	145	1.108	opt
mcsp300	300	6108	170	1.638	opt
mcsp350	350	7882	216	2.641	opt
mcsp400	400	10760	251	3.360	opt
mcsp450	450	13510	273	5.001	opt
mcsp500	500	16695	318	4.014	opt

For `mgcol` instances listed in Table 3 optimality is verified in all 30 instances. The instances with 300 vertices took several hundred seconds, while the smaller instances with 100 vertices finished within 1 second. There also appears to be certain regularity in the $dem(G)$ values, as they are 90 or 91 for instances with 100 vertices, while they are 288 for all instances with 300 vertices.

Table 3: DEM results for `mgcol` instances

	Dimension		CPLEX		
Instance	$\|V\|$	$\|E\|$	*dem*	T (sec)	Status
mgcol1	100	2487	91	0.952	opt
mgcol2	100	2487	91	0.969	opt
mgcol3	100	2482	91	0.848	opt
mgcol4	100	2503	91	0.903	opt
mgcol5	100	2450	91	0.978	opt
mgcol6	100	2537	91	0.878	opt
mgcol7	100	2505	91	0.876	opt
mgcol8	100	2479	90	0.827	opt
mgcol9	100	2486	90	0.811	opt
mgcol10	100	2506	91	0.883	opt
mgcol11	100	2467	91	0.886	opt
mgcol12	100	2531	91	0.917	opt
mgcol13	100	2467	91	0.849	opt
mgcol14	100	2524	91	0.903	opt
mgcol15	100	2528	91	0.914	opt
mgcol16	100	2493	91	0.840	opt
mgcol17	100	2503	91	0.881	opt
mgcol18	100	2472	91	0.861	opt
mgcol19	100	2527	91	0.767	opt
mgcol20	100	2420	91	0.903	opt
mgcol21	300	22482	288	215.6	opt
mgcol22	300	22569	288	636.2	opt
mgcol23	300	22393	288	257.9	opt
mgcol24	300	22446	288	806.2	opt
mgcol25	300	22360	288	293.0	opt
mgcol26	300	22601	288	798.0	opt
mgcol27	300	22327	288	666.9	opt
mgcol28	300	22472	288	695.7	opt
mgcol29	300	22520	288	187.8	opt
mgcol30	300	22543	288	655.5	opt

The `frb` instances proved to be the most difficult for CPLEX in solving the IP model. As can be seen from Table 4, the optimal solutions were achieved in only

8 out of 40 cases – in all of these cases the number of vertices was either 450 or 595.

Table 4: DEM results for `frb` instances

	Dimension		CPLEX		
Instance	$\|V\|$	$\|E\|$	*dem*	T (sec)	Status
frb30-15-1	450	17827	420	425.4	opt
frb30-15-2	450	17874	420	231.7	opt
frb30-15-3	450	17809	420	442.9	opt
frb30-15-4	450	17831	420	225.2	opt
frb30-15-5	450	17794	420	50.50	opt
frb35-17-1	595	27856	562	3616	
frb35-17-2	595	27847	560	1633	opt
frb35-17-3	595	27931	560	1193	opt
frb35-17-4	595	27842	560	2663	opt
frb35-17-5	595	28143	562	3616	
frb40-19-1	760	41314	722	3622	
frb40-19-2	760	41263	723	3642	
frb40-19-3	760	41095	722	3642	
frb40-19-4	760	41605	723	3620	
frb40-19-5	760	41619	721	3609	
frb45-21-1	945	59186	903	3623	
frb45-21-2	945	58624	903	3620	
frb45-21-3	945	58245	903	3624	
frb45-21-4	945	58549	903	3626	
frb45-21-5	945	58579	905	3627	
frb50-23-1	1150	80072	1103	3630	
frb50-23-2	1150	80851	1105	3634	
frb50-23-3	1150	81068	1104	3637	
frb50-23-4	1150	80258	1103	3627	
frb50-23-5	1150	80035	1103	3633	
frb53-24-1	1272	94227	1225	3642	
frb53-24-2	1272	94289	1223	3636	
frb53-24-3	1272	94127	1224	3631	
frb53-24-4	1272	94308	1225	3642	
frb53-24-5	1272	94226	1224	3634	
frb56-25-1	1400	109676	1350	3641	
frb56-25-2	1400	109401	1349	3638	
frb56-25-3	1400	109379	1349	3639	
frb56-25-4	1400	110038	1348	3648	
frb56-25-5	1400	109601	1350	3650	
frb59-26-1	1534	126555	1482	3647	
frb59-26-2	1534	126163	1481	3647	
frb59-26-3	1534	126082	1483	3655	
frb59-26-4	1534	127011	1480	3650	
frb59-26-5	1534	125982	1480	3653	

Hypercubes can be easily solved with CPLEX applied to the proposed IP model. As Table 5 shows, all instances were solved within ≈ 1 minute.

Table 5: DEM results for `hypercube` instances

	Dimension		CPLEX		
Instance	$\|V\|$	$\|E\|$	dem	T (sec)	Status
hypercube-1	2	1	1	0.005	opt
hypercube-2	4	4	2	0.010	opt
hypercube-3	8	12	4	0.009	opt
hypercube-4	16	32	8	0.030	opt
hypercube-5	32	80	16	0.026	opt
hypercube-6	64	192	32	0.055	opt
hypercube-7	128	448	64	0.128	opt
hypercube-8	256	1024	128	0.334	opt
hypercube-9	512	2304	256	0.749	opt
hypercube-10	1024	5120	512	2.514	opt
hypercube-11	2048	11264	1024	12.72	opt
hypercube-12	4096	24576	2048	65.97	opt

The instances denoted as `random` were generated using the parametric model $G(n, p)$ of Erdős-Rényi, in which a graph with n vertices is constructed by randomly drawing edges with probability p. Table 6 shows that all instances considered were solved optimally. It is also interesting to observe that $dem(G) \approx |V|/2$ holds for all instances, regardless of graph density.

Table 6: DEM results for `random` instances

	Dimension		CPLEX		
Instance	$\|V\|$	$\|E\|$	dem	T (sec)	Status
random-v200-p0.025	200	539	92	0.093	opt
random-v200-p0.05	200	1068	97	0.209	opt
random-v200-p0.1	200	2009	98	0.343	opt
random-v200-p0.2	200	3948	101	0.788	opt
random-v200-p0.5	200	9873	100	1.931	opt
random-v200-p0.8	200	15931	100	3.337	opt
random-v500-p0.025	500	3130	248	2.669	opt
random-v500-p0.05	500	6262	249	2.573	opt
random-v500-p0.1	500	12562	250	3.867	opt
random-v500-p0.2	500	25076	249	7.522	opt
random-v500-p0.5	500	62475	250	18.19	opt
random-v500-p0.8	500	99716	250	31.25	opt
random-v1000-p0.025	1000	12485	495	7.873	opt
random-v1000-p0.05	1000	24928	496	15.67	opt
random-v1000-p0.1	1000	50020	497	21.78	opt
random-v1000-p0.2	1000	100101	499	43.45	opt
random-v1000-p0.5	1000	250082	500	124.3	opt
random-v1000-p0.8	1000	399551	500	148.3	opt
random-v2000-p0.025	2000	50100	997	32.50	opt
random-v2000-p0.05	2000	100030	995	54.55	opt
random-v2000-p0.1	2000	200160	1000	100.5	opt
random-v2000-p0.2	2000	400478	1000	183.0	opt
random-v2000-p0.5	2000	999844	1000	865.5	opt
random-v2000-p0.8	2000	1599797	1000	1867	opt

As can be seen from the results presented, CPLEX works well with the proposed IP model. This is a consequence of the relatively small number of model variables. It can also be observed that it works slightly better on sparse graphs than on dense graphs. Although the `frb` instances are not the largest instances tested in terms of graph dimensions, they proved to be the most difficult. There-

fore, instance dimensions and instance density are not the only factors influencing instance hardness. For example, there are `frb40` instances with 760 nodes and a density of ≈ 0.15 that are not solved optimally, while `random` instances with 2000 nodes and a density of 0.2 or more are solved optimally. The other factors that may influence the instance hardness are based on the internal structure of `frb` graphs, which may be of interest for future research.

5. CONCLUSION

We have proposed the first integer programming model for the recently introduced distance-edge-monitoring problem. The model appears to be very efficient for a variety of graph classes. It can be used as a tool for understanding $dem(G)$ regularities in some other graph classes as well.

In further research, it would be interesting to characterize difficult-to-solve graphs, starting from the pseudo-Boolean graphs that have been empirically shown to be difficult, and to understand why $dem(G) \approx |V|/2$ in case of Erdős-Rényi graphs. In addition, it would be interesting to explore the possibilities of various exact and non-exact solving methods, such as A*, Beam search, Variable neighborhood search, Genetic algorithm, etc. Non-exact methods would be suitable for large graphs, common in practical applications. Other ways to solve large graphs could be to combine exact and non-exact methods to take advantage of both worlds. For example, there are metaheuristics based on ILP models, such as Construct, Merge, Solve & Adapt, VNS+ILP hybrids and others.

Funding. A. Kartelj and V. Filipović are supported by grant 451-03-47/2023-01/200104 from the Ministry of Science Technological Development and Innovations of the Republic of Serbia.

REFERENCES

[1] F. Foucaud, S.-S. Kao, R. Klasing, M. Miller, and J. Ryan, "Monitoring the edges of a graph using distances," *Discrete Appl. Math.*, vol. 319, pp. 424–438, 2022.

[2] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *Proc IEEE INFOCOM 2000*, vol. 3. IEEE, 2000, pp. 1371–1380.

[3] L. Dall'Asta, I. Alvarez-Hamelin, A. Barrat, A. Vázquez, and A. Vespignani, "Exploring networks with traceroute-like probes: Theory and simulations," *Theor. Comput. Sci.*, vol. 355, no. 1, pp. 6–24, 2006.

[4] Z. Beerliova, F. Eberhard, T. Erlebach, A. Hall, M. Hoffmann, M. Mihal'ak, and L. S. Ram, "Network discovery and verification," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2168–2181, 2006.

[5] D. Bilò, T. Erlebach, M. Mihalák, and P. Widmayer, "Discovery of network properties with all-shortest-paths queries," *Theor. Comput. Sci.*, vol. 411, no. 14-15, pp. 1626–1637, 2010.

[6] E. Bampas, D. Bilò, G. Drovandi, L. Gualà, R. Klasing, and G. Proietti, "Network verification via routing table queries," *J. Comput. Syst. Sci.*, vol. 81, no. 1, pp. 234–248, 2015.

[7] W. Li, R. Klasing, Y. Mao, and B. Ning, "Monitoring the edges of product networks using distances," *arXiv preprint arXiv:2211.10743*, 2022.

[8] C. Yang, R. Klasing, Y. Mao, and X. Deng, "On the distance-edge-monitoring numbers of graphs," *Discrete Appl. Math.*, vol. 342, pp. 153–167, 2024. doi: https://doi.org/10.1016/j.dam.2023.09.012

[9] C. Yang, R. Klasing, C. He, and Y. Mao, "Perturbation results for distance-edge-monitoring numbers," *arXiv preprint arXiv:2301.02507*, 2023.
[10] G. Yang and C. He, "Distance-edge-monitoring sets in hierarchical and corona graphs," *J. Interconnect. Netw.*, vol. 23, no. 02, p. 2250003, 2023.
[11] L. Václav, "Distance edge monitoring set problem with respect to structural parameters," B.S. thesis, Faculty of Information Technology CTU in Prague, Department of Theoretical Computer Science, 2022.
[12] F. Foucaud, K. Narayanan, and L. Ramasubramony Sulochana, "Monitoring edge-geodetic sets in graphs," in *Conf. Algorithms Discrete Appl. Math.* Springer, 2023, pp. 245–256.
[13] J. Haslegrave, "Monitoring edge-geodetic sets: hardness and graph products," *Discrete Appl. Math.*, vol. 340, pp. 79–84, 2023.
[14] A. Tan, W. Li, X. Wang, and X. Li, "Monitoring edge-geodetic numbers of convex polytopes and four networks," *Int. J. Parallel Emergent Distrib. Syst.*, pp. 1–12, 2023.