A Connected-Component Improvement Procedure for Relative Isolation Probability Calculation

Renzo Roel P. Tan and Kazushi Ikeda

***Abstract*—Due to the prevalence of natural disasters, the resilience of networks that provide essential services to the population has been studied. A recent contribution is the probability of relative isolation, which measures the likelihood of a demand node to be unreachable by supply nodes. The existing procedure for calculation is an enumerative decision-diagram-based approach that may be resource-intensive especially on occasions when time and memory constraints are imposed. In response, the study reverse-engineers the current method towards increased efficiency. The proposed solution hinges on the breadth-first search, producing connected components based on the sources. The algorithm is then benchmarked on foundational networks from literature, presenting a 99.77% decrease in computation time on average.**

***Index Terms*—breadth-first search, graph measure, Monte Carlo method, probability of relative isolation.**

I. Introduction

RESILIENCE refers to the ability of a network to maintain an acceptable level of operation in the event of damage. Since natural phenomena that pose risk to service-providing infrastructure are inevitable, network resilience has been the subject of research [1]. Accordingly, measures to estimate the resilience of lifelines such as road networks, water pipelines, power lines, and communication systems, among others, have been crafted by studying the graphs in which they may be represented [2], [3].

Recently, an original metric for resilience called the probability of relative isolation was proposed [4]. In brief, the relative isolation probability measures how likely it is that a vertex would be unreachable by any source given the failure of edges. It seeks to translate the edge-centric failure probabilities into vertex-centric isolation probabilities to shed light on the resilience of a graph structure. With the new probability measure, the effect of weak supply links to the over-all network operation may be gauged. Demand nodes that may require emergency response after destructive episodes may also be identified. Towards mitigation, the metric may be used as basis for network improvements such as component reinforcement and source addition.

While the utility of the relative isolation probability is promising, its calculation may prove costly at times as the current approach is enumerative and relies on the construction of decision diagrams [4]. In cases where computational resources are scarce, an alternative strategy is required to generate the probabilities without the need for the enumeration of surviving supply routes. In the paper, the existing decision-diagram-based solution is reverse-engineered to compose an efficient method based on the breadth-first search. The proposed improvement procedure builds source-rooted connected components, determining the relative isolation of nonsource vertices depending on whether or not they are part of the said components. Overall, the novel algorithm decreases computation time by 99.74% on average.

The study is outlined as follows. A recapitulation of the relative isolation probability as a novel resilience measure for graphs is in the second section. The next section covers the methodology, under which both the algorithm implementation and machine specifications fall. The fifth section lays out the results. A conclusion of the work is contained in the final section.

Notation used is as follows. This is based on [4].

V	The set of vertices $\{v_1, v_2, \ldots, v_{\lvert V\rvert}\}$.
E	The set of edges $\{e_1, e_2, \ldots, e_{\lvert E\rvert}\}$.
G	The graph (V, E) defined by V and E.
S	The set of source vertices $\{s_1, s_2, \ldots, s_{\lvert S\rvert}\}$.
$f : A \to B$	A function mapping the set A to the set B.

II. Probability of Relative Isolation

The section introduces the probability of relative isolation and some accompanying concepts [4].

Definition 1 (Path)**.** A path is a sequence $e_1, e_2, \ldots, e_p$, where $e_i = \{v_{i-1}, v_i\}$ is an edge of the graph for $i = 1, 2, \ldots, p$ with vertex $v_i \neq v_j$ if $i \neq j$.

A path is therefore a sequence of edges that connects two vertices v_0 and v_p. The definition of the relative isolation probability of a vertex in a given graph is below. The following is slightly modified for the purposes of the paper.

Definition 2 (Probability of Relative Isolation)**.** Consider the graph $G = (V, E)$ and the set of source vertices $S \subset V$. Let the probability of failure of an edge be $\pi(e_i)$ for $i = 1, 2, \ldots, |E|$, with function $\pi : E \to [0, 1]$.

The number of iterations, which is essentially how many times the Monte Carlo experiment is executed, is set to be N. A random number $\rho_k(e_i)$ with function $\rho_k : E \to [0, 1]$ is assigned to each edge e_i on the kth iteration for $k = 1, 2, \ldots, N$. For all $e_i \in E$, an indicator function μ_k is defined to be

$$\mu_k(e_i) = \begin{cases} 1 & \text{if } \rho_k(e_i) > \pi(e_i) \\ 0 & \text{otherwise} \end{cases},$$

Manuscript submitted 15 June 2023; revised 13 February 2024. The Japan Society for the Promotion of Science funds the study through the Grants-in-Aid for Scientific Research Program (KAKENHI 18K19821 and 22K19833). Support is also given by Kyoto University and Toyota Motor Corporation through the Advanced Mathematical Science for Mobility Society joint project.

R. R. P. Tan is an assistant professor in the Division of Information Science, Nara Institute of Science and Technology, Ikoma City, Nara Prefecture, Japan, the Graduate School of Informatics, Kyoto University, Kyoto City, Kyoto Prefecture, Japan, and the John Gokongwei School of Management, Ateneo de Manila University, Quezon City, National Capital Region, Philippines (corresponding author, e-mail: rr.tan@is.naist.jp).

K. Ikeda is a professor in the Division of Information Science, Nara Institute of Science and Technology, Ikoma City, Nara Prefecture, Japan (e-mail: kazushi@is.naist.jp).

assigning either a 1 or a 0 to each edge indicating its survival or failure, respectively, for iteration k.

A subgraph $G_k = (V_k, E_k)$ is produced, with $e_i \in E_k$ if and only if $\mu_k(e_i) = 1$ and $v_j \in V_k$ for $j = 1, 2, \ldots, |V|$ if and only if there exists $e \in E_k$ such that $v_j \in e$ or if $v_j \in S$. More precisely, $E_k = \{e \in E \mid \mu_k(e) = 1\}$ and $V_k = \{v \in V \mid v \in e,\ e \in E_k\} \cup S$. For all $v_j \in V \setminus S$, a second indicator function is then defined as

$$\lambda_k(v_j) = \begin{cases} 0 & \text{if } v_j \text{ is connected to some } s \in S \text{ in } G_k \\ 1 & \text{otherwise} \end{cases},$$

specifying whether or not each vertex is connected to a source for iteration k. If $v_j \in S$ then $\lambda_k(v_j) = 0$.

The relative isolation probability of a vertex v_j within N instances, denoted by $\Pi_\iota^N(v_j)$, is

$$\Pi_\iota^N(v_j) = \frac{\sum_{l=1}^{N} \lambda_l(v_j)}{N}.$$

III. Breadth-First Search

The breadth-first search is a fundamental algorithm for graph traversal. As the name suggests, the algorithm explores the vertices of the graph in a systematic manner starting from a given vertex and subsequently visiting its neighbors in order of increasing distance. A single connected component containing the starting vertex is produced [5]. The algorithm may thus be be used to calculate the probability of relative isolation since $\lambda_k(v)$ equivalently indicates whether vertex v belongs to a connected component containing a source for the kth iteration.

The search begins with a queue containing the starting vertex v. Briefly, a queue is a data structure that follows the first-in-first-out principle. Initially, vertex v is marked as visited. Iteratively, the algorithm dequeues the first element v' and enqueues all unvisited neighbors of v'. Termination happens when all vertices reachable from vertex v have been visited or when the queue is empty.

The worst-case complexity of the algorithm is $\mathcal{O}(|V| + |E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges in the graph. This is significantly less than the complexity of a decision-diagram-based approach with a worst-case complexity of $\mathcal{O}(n(\mathcal{D}))$, where $n(\mathcal{D})$ is the number of nodes in the constructed diagram [4]. The breadth-first search, however, does not store information on all paths between two vertices as in a method using decision diagrams.

IV. Methods

Graphs from literature are used as benchmark and translated into edge lists. Every row represents an edge defined by two vertices. The source vertices and failure probabilities of the edges are kept in a separate file in order to conveniently create multiple scenarios.

As outline, the relative isolation probability of a vertex is computed by performing a number of randomized simulations, checking for the connection of the vertex to a source per iteration, and providing the percentage of simulations in which the vertex is relatively isolated.

First, a Monte Carlo simulation is done as a preliminary processing step. The survival or failure of each edge is determined and G_k is produced for $k = 1, 2, \ldots, N$. The pseudocode is shown in Algorithm 1. Line 5 generates random numbers uniformly over $(0, 1)$. The random numbers are then compared to the failure probabilities assigned to the edges. The N instances of the original graph are the results of the preparatory step.

Algorithm 1 MCPrePro

```
1: Assume G = (V, E)
2: Let EDGESTATE be an array whose indices are an edge and an integer
3: for e ∈ E do
4:     for k = 1, 2, ..., N do
5:         if UNIFORM(0, 1) ≥ e.p then
6:             EDGESTATE[e][k] ← 1
7:         else
8:             EDGESTATE[e][k] ← 0
```

Note that $e.p$ is the failure probability of edge e.

Algorithm 2 PICCC

```
1: Execute MCPrePro
2: Let ISOLATIONS and FOUND be arrays whose index is a vertex
3: ISOLATIONS[v] ← 0  for v ∈ V
4: for k = 1, 2, ..., N do
5:     FOUND[v] ← false  for v ∈ V \ S
6:     for s ∈ S do
7:         Execute BFS(s, k)
8:     for v ∈ V \ S do
9:         if not FOUND[v] then
10:            ISOLATIONS[v] ← ISOLATIONS[v] + 1
11: for v ∈ V do
12:     PRINT(v, ISOLATIONS[v]/N)
```

Note that BFS(s, k) is a search from vertex s in the kth simulation.

The actual probability calculation is the second step, seen in Algorithm 2. In each iteration, a breadth-first search is executed from each source vertex s as in Line 7. The search traverses edges that survive given the current scenario derived from EDGESTATE$[\cdot][k]$ from Algorithm 1. This results in the list of vertices that are reachable by any source vertex. For a vertex v, the algorithm adds 1 to the isolation count of the vertex and moves on to the next vertex if it is not reachable by any source; otherwise, it simply moves on to the next. In the end, the isolation count of each vertex is divided by the number of simulations to get the probability of relative isolation. It is worth noting that a source vertex is assigned a probability of 0 since there always exists a path to itself.

The implementation is entirely done in the C++ programming language and compiled with version 9.3.0 of the g++. Refer to the PICCC (https://github.com/renzopereztan/PICCC) repository for the complete code. The PIZDD (https://github.com/renzopereztan/PIZDD) program serves as basis. Regarding the machine, the operating system used is the Ubuntu 18.04.4 Long Term Support (Bionic Beaver). The processor is the 1.80GHz Intel® Core™ i7-8565U. The experiments are done with a memory allocation of 16GB.

V. Results

The algorithm is tested on six benchmark and three real-world networks. Shown in Figures 1 to 6 are the graphs of different structures with $6 \leq |V| \leq 40$ and $15 \leq |E| \leq 58$ carefully selected from foundational research [6], [7]. These include a complete graph, a Petersen graph, two distinctive graphs, a square grid, and a rectangular mesh. Figures 7 to 9 show the water distribution networks from Bursa in Turkey, Hanoi in Vietnam, and Kobe in Japan [4] with $12 \leq |V| \leq 32$ and $15 \leq |E| \leq 34$.

The number of simulations is set to 1000, 10000, and 100000 to gauge the scaling of computation costs with N. In further examining performance, the edge failure probabilities are initialized at 0.50, 0.05, and 0.95 for the benchmarks to represent an acceptable breadth of cases. On one hand, simulating with 0.05 edge failure probabilities yields relatively dense G_k; on the other, simulating with 0.95 edge failure probabilities yields relatively sparse G_k. The 0.50 edge failure probability experiments signify the middle ground. The probabilities of failure for the real-world networks are set based on previous studies.

Results are summarized in Tables I, II, III, IV, V, and VI. In the tables, $|E|$ is the number of edges and $|V|$ is the number of vertices. The time elapsed during calculation is presented both per stage and in aggregate. As illustration, $t_p^{\mathbf{z}}$ and $t_p^{\mathbf{c}}$ are the total computation time by the algorithm in the decision-diagram-based approach and the connected-components-based approach, respectively, when the edge failure probability is p. The average across all three probabilities is indicated as $t^{\mathbf{z}}$ and $t^{\mathbf{c}}$. The column δ then indicates the percent decrease of the proposed approach based on connected components versus the existing approach based on decision diagrams.

Seen in Tables VII and VIII is the capability of the new method. It handles cases in which $N = 1000000$ with ease, whereas the decision-diagram-based algorithm is not able to finish the calculation within reasonable time. As a remark on convergence, at the core of the computation is dividing the number of simulations yielding relative isolation by the total number of simulations. On the one-hundredth iteration, for example, changes larger than a percent that succeeding iterations may make is unlikely. Moreover, the entire set of experiments is executed multiple times to ensure accuracy. It is found that the variability between runs is not significant.

VI. Conclusion

A novel approach for the calculation of the probability of relative isolation is proposed. The Monte Carlo experiment is maintained as the first step, simulating for the success or failure of edges and producing the surviving subgraphs. In the second step, a breadth-first search is executed on the resulting subgraphs to create connected components stemming from each source vertex and to determine the relative isolation of the nonsource vertices. The proposed algorithm consumes significantly less time, ideal for situations that necessitate low-resource and swift computations.

References

[1] H. Perez-Roses, "Sixty years of network reliability," *Mathematics in Computer Science Vol. 12*, pp. 275–293, 2018.

[2] H. Yang and S. Shoping, "Seismic reliability of urban pipeline network systems," *In Advancing Mitigation Technologies and Disaster Response for Lifeline Systems: Proceedings of the 6th United States Conference and Workshop on Lifeline Earthquake Engineering*, pp. 445–454, 2003.

[3] M. Javanbarg and S. Takada, "Seismic reliability assessment of water supply systems," *In Safety, Reliability, and Risk of Structures, Infrastructures, and Engineering Systems: Proceedings of the 10th International Conference on Structural Safety and Reliability*, pp. 3455–3462, 2010.

[4] R. Tan, K. See, J. Kawahara, K. Ikeda, R. de Jesus, L. Garciano, and A. Garciano, "The relative isolation probability of a vertex in a multiple-source edge-weighted graph," *Engineering Letters Vol. 30 No. 1*, pp. 117–130, 2022.

[5] J. Holdsworth, "The nature of breadth-first search," *James Cook University School of Computer Science, Mathematics, and Physics Technical Report 99-1*, 1999.

[6] F. Yeh and S. Kuo, "Obdd-based network reliability calculation," *Electronics Letters Vol. 33 No. 9*, pp. 759–760, 1997.

[7] S. Chaterjee, V. Ramana, G. Vishwakarma, and A. Verma, "An improved algorithm for k-terminal probabilistic network reliability analysis," *Journal of Reliability and Statistical Studies Vol. 10 Iss. 1*, pp. 15–26, 2017.

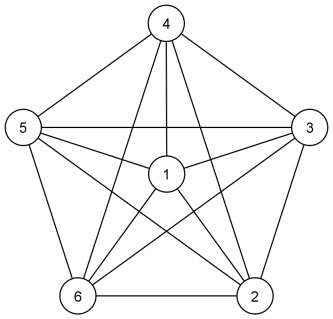

Fig. 1: Graph 1.

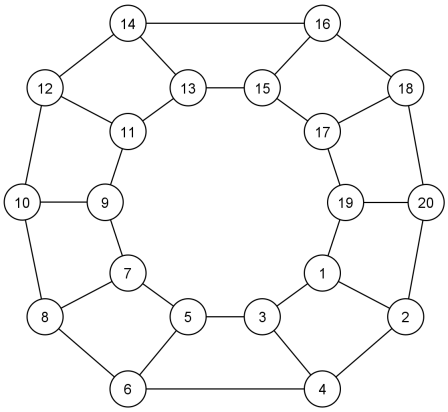

Fig. 2: Graph 2.

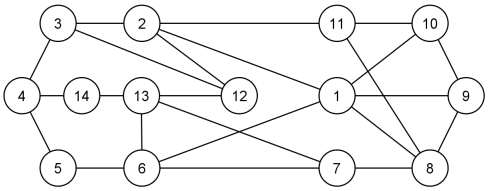

Fig. 3: Graph 3.

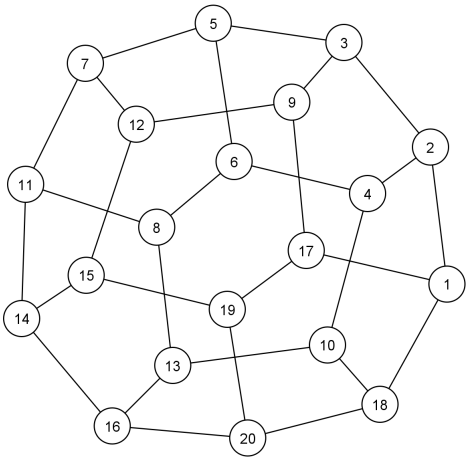

Fig. 4: Graph 4.

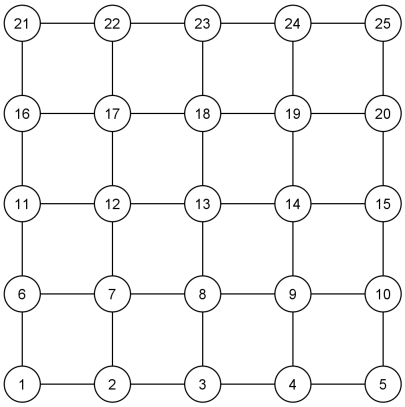

Fig. 5: Graph 5.

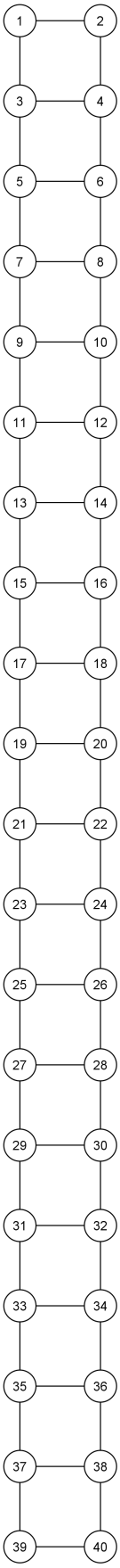

Fig. 6: Graph 6.

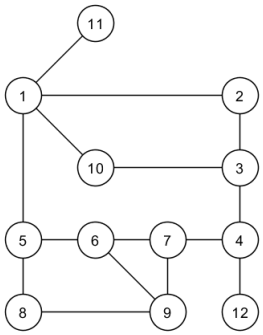

Fig. 7: Bursa network.

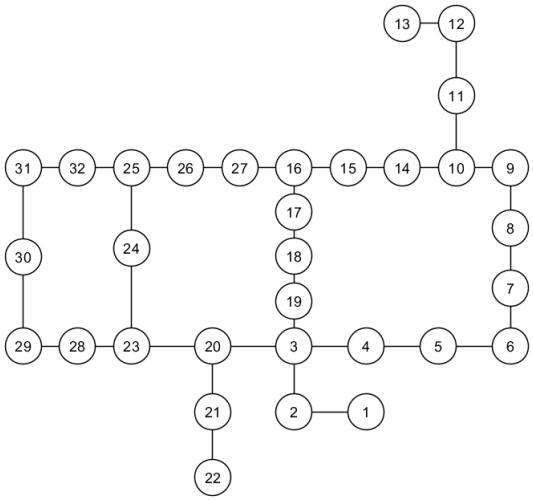

Fig. 8: Hanoi network.

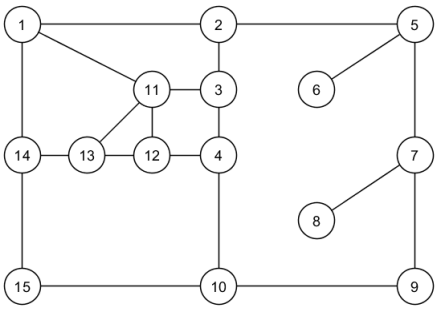

Fig. 9: Kobe network.

TABLE I: Experiment statistics on benchmark networks for $N = 1000$.

G	$\|E\|$	$\|V\|$	$t^{\mathbf{z}}_{0.50}$	$t^{\mathbf{z}}_{0.05}$	$t^{\mathbf{z}}_{0.95}$	$t^{\mathbf{z}}$	$t^{\mathbf{c}}_{0.50}$	$t^{\mathbf{c}}_{0.05}$	$t^{\mathbf{c}}_{0.95}$	$t^{\mathbf{c}}$	δ
1	15	6	1.8001	1.2530	2.1570	1.7367	0.0250	0.0278	0.0383	0.0304	98.25%
2	30	20	9.9405	5.8383	17.5536	11.1108	0.0366	0.0385	0.0248	0.0333	99.70%
3	23	14	11.0817	7.9262	12.3811	10.4630	0.0455	0.0331	0.0229	0.0338	99.68%
4	30	20	16.5617	12.7112	22.5423	17.2718	0.0380	0.0348	0.1007	0.0578	99.67%
5	40	25	32.8349	24.4346	45.9892	34.4196	0.0538	0.0465	0.0376	0.0459	99.87%
6	58	40	18.2273	12.0706	68.3635	32.8871	0.0452	0.0421	0.0380	0.0418	99.87%

TABLE II: Experiment statistics on benchmark networks for $N = 10000$.

G	$\|E\|$	$\|V\|$	$t^{\mathbf{z}}_{0.50}$	$t^{\mathbf{z}}_{0.05}$	$t^{\mathbf{z}}_{0.95}$	$t^{\mathbf{z}}$	$t^{\mathbf{c}}_{0.50}$	$t^{\mathbf{c}}_{0.05}$	$t^{\mathbf{c}}_{0.95}$	$t^{\mathbf{c}}$	δ
1	15	6	17.3706	12.4311	20.4240	16.7419	0.0768	0.1277	0.1289	0.1111	99.34%
2	30	20	107.7092	55.7579	193.2862	118.9178	0.1475	0.1158	0.1173	0.1269	99.89%
3	23	14	122.3553	84.5313	151.3217	119.4028	0.1068	0.1862	0.1066	0.1332	99.89%
4	30	20	184.9917	140.6444	237.3762	187.6708	0.1320	0.1228	0.2081	0.1543	99.92%
5	40	25	361.2559	252.3969	435.7676	349.8068	0.3737	0.1676	0.1652	0.2355	99.93%
6	58	40	199.5222	116.2606	684.8887	333.5572	0.2131	0.1937	0.3307	0.2458	99.93%

TABLE III: Experiment statistics on benchmark networks for $N = 100000$.

G	$\|E\|$	$\|V\|$	$t^{\mathbf{z}}_{0.50}$	$t^{\mathbf{z}}_{0.05}$	$t^{\mathbf{z}}_{0.95}$	$t^{\mathbf{z}}$	$t^{\mathbf{c}}_{0.50}$	$t^{\mathbf{c}}_{0.05}$	$t^{\mathbf{c}}_{0.95}$	$t^{\mathbf{c}}$	δ
1	15	6	172.4573	110.1030	208.0723	163.5442	0.5109	0.5072	0.4732	0.4971	99.70%
2	30	20	1023.6077	522.5294	1837.2693	1127.8021	0.9434	1.0219	0.9126	0.9593	99.91%
3	23	14	1125.0595	776.0387	1262.2774	1054.4586	0.7713	0.7057	0.7312	0.7361	99.93%
4	30	20	1675.6369	1265.0377	2215.9349	1718.8698	1.0006	1.0391	1.1066	1.0488	99.94%
5	40	25	3284.2540	2464.2823	4070.9923	3273.1762	1.3436	1.1935	1.2140	1.2503	99.96%
6	58	40	2238.2904	1121.5588	6781.3090	3380.3860	1.7538	1.7005	1.8659	1.7734	99.95%

TABLE IV: Experiment statistics on real-world networks for $N = 1000$.

G	$\|E\|$	$\|V\|$	$t^{\mathbf{z}}$	$t^{\mathbf{c}}$	δ
Bursa	15	12	3.6773	0.0064	99.83%
Hanoi	34	32	12.7224	0.0370	99.71%
Kobe	20	15	5.6987	0.0230	99.60%

TABLE V: Experiment statistics on real-world networks for $N = 10000$.

G	$\|E\|$	$\|V\|$	$t^{\mathbf{z}}$	$t^{\mathbf{c}}$	δ
Bursa	15	12	36.8083	0.0518	99.86%
Hanoi	34	32	114.6825	0.1827	99.84%
Kobe	20	15	57.2328	0.1149	99.80%

TABLE VI: Experiment statistics on real-world networks for $N = 100000$.

G	$\|E\|$	$\|V\|$	$t^{\mathbf{z}}$	$t^{\mathbf{c}}$	δ
Bursa	15	12	406.5972	0.5000	99.88%
Hanoi	34	32	1080.5268	1.2970	99.88%
Kobe	20	15	544.2032	0.7791	99.86%

TABLE VII: Experiment statistics on benchmark networks for $N = 1000000$.

G	$\|E\|$	$\|V\|$	$t^{\mathbf{c}}_{0.50}$	$t^{\mathbf{c}}_{0.05}$	$t^{\mathbf{c}}_{0.95}$	$t^{\mathbf{c}}$
1	15	6	5.0462	4.7480	4.7828	4.8590
2	30	20	9.4655	9.1059	8.9982	9.1899
3	23	14	7.4822	7.1319	6.9573	7.1905
4	30	20	9.5298	9.2382	9.4393	9.4025
5	40	25	12.0434	11.6596	11.6597	11.7876
6	58	40	17.4662	17.3065	17.3124	17.3617

TABLE VIII: Experiment statistics on real-world networks for $N = 1000000$.

G	$\|E\|$	$\|V\|$	$t^{\mathbf{c}}$
Bursa	15	12	5.1248
Hanoi	34	32	12.1588
Kobe	20	15	7.3236