ICOSSAR 2021
The 13th International Conference on Structural Safety and Reliability (ICOSSAR 2021), June 21-25, 2021, Shanghai, P.R. China
J. Li, Pol D. Spanos, J.B. Chen & Y.B. Peng (Eds)

Tensor Network Contraction For Network Reliability Estimates

Kyle Shepherd [a] and Leonardo Dueñas-Osorio [b]

[a] *PhD Student, Dept. of Civil Engineering, Rice University, Texas, Houston, United States, E-mail: kas20@rice.edu, ORCID:0000-0002-8220-7448*

[b] *Professor, Dept. of Civil Engineering, Rice University, Houston, United States, E-mail: leonardo.duenas-osorio@rice.edu, ORCID:0000-0002-7138-7746*

ABSTRACT: Quantifying network reliability is a hard problem, proven to be #P-complete [1]. For real-world network planning and decision making, approximations for the network reliability problem are necessary. This study shows that tensor network contraction (TNC) methods can quickly estimate an upper bound of All Terminal Reliability, $Rel_{ATR}(G)$, by solving a superset of the network reliability problem: the edge cover problem, $EC(G)$. In addition, these tensor contraction methods can exactly solve source-terminal (S-T) reliability for the class of directed acyclic networks, $Rel_{S-T}(G)$.

The computational complexity of TNC methods is parameterized by treewidth, significantly benefitting from recent advancements in approximate tree decomposition algorithms [2]. This parameterization does not rely on the reliability of the graph, which means these tensor contraction methods can determine reliability faster than Monte Carlo methods on highly reliable networks, while also providing exact answers or guaranteed upper bound estimates. These tensor contraction methods are applied to grid graphs, random cubic graphs, and a selection of 58 power transmission networks [3], demonstrating computational efficiency and effective approximation using $EC(G)$.

1 Introduction

1.1 *Motivation*

Important infrastructure systems such as electrical transmission grids, potable water distribution, and roadway transportation have been modeled as networks for analysis and design [4] [5]. As these networks grow in size and become more complex, such as the addition of distributed power generation and energy storage in electrical networks [6], better algorithms are needed to analyze these networks and guarantee their safety and reliability. The tensor network contraction (TNC) algorithm we propose in this work is a step in this direction.

1.2 *Problem Definition*

A network is a graph consisting of nodes that describe a discrete component of the network (such as a power plant or household) and edges that describe a connection between two nodes (such as power lines or water pipes). We will consider a model of a graph $G = (N,E)$ where N is the set of labeled nodes $n_1,...,n_{|N|}$ and E is the set of labeled edges $e_1,...,e_{|E|}$. Each node $n_i \in N$ has a list of attributes $[S,T]$. In particular, $n_i.S$, the S attribute of variable n_i, is a Boolean variable equal to $True$ if n_i is a source node, and $n_i.T$ is a Boolean variable equal to $True$ if n_i is a terminal node. Each edge $e_i \in E$ has a list of attributes $[n_p, n_s, bi, p]$. In particular, $e_i.n_p$ is the predecessor node, $e_i.n_s$ is the successor node, $e_i.bi$ is a Boolean variable equal to *True* if the edge is bidirectional and *False* if the edge is directed, and $e_i.p$ is the edge reliability, the probability the edge exists.

To measure graph reliability, we define a function $C(G_r)$, where $C(G_r) = 1$ if there is a path from a source node to every terminal node for the particular graph realization G_r, indicating a Connected graph, and $C(G_r) = 0$ otherwise. A graph realization is defined with a vector r of length $|E|$ where $r_i = 1$ if e_i exists, and $r_i = 0$ otherwise. The probability of a given graph realization is defined below:

$$P(G_r) = \prod_{i=1}^{|E|} 1 - r_i - e_i.p + 2 * r_i * e_i.p$$

The set of all possible realization vectors is R. This set contains $2^{|E|}$ elements.

Reliability is defined as $Rel(G) = \sum_{r\in R}[C(G_r) * P(G_r)]$. $Rel(G)$ will take on a value between 0 and 1, where a higher $Rel(G)$ is desired in practice.

This model as defined allows us to consider two cases of network reliability that are important for managers of infrastructure systems. All Terminal Reliability is $Rel_{ATR}(G)$ where all edges are bidirectional and all nodes are terminal nodes. Source-Terminal reliability is $Rel_{S-T}(G)$ where all edges are directed, and in general the set of source and terminal nodes is much smaller than the set of nodes. For this work, only one source and one terminal node will be considered for $Rel_{S-T}(G)$ (also known as 2-terminal reliability).

1.3 *Justification and Objectives*

As the time to compute the reliability of a probabilistic graph scales exponentially with the size of the graph [1], a naïve brute force enumeration of all graph realizations is not feasible for the large graphs frequently encountered in infrastructure systems. Therefore, different approaches are needed to calculate values of $Rel(G)$.

One approach is to develop an algorithm with parameterized complexity to solve $Rel(G)$. While these algorithms may scale exponentially in the worst case, they may not scale as fast for the problems we are interested in. Another approach is to solve a superset problem of $Rel(G)$. In this work, the edge cover problem will be shown to be a superset problem of $Rel_{ATR}(G)$. A final approach is to give up on obtaining an exact answer and instead obtain an approximate answer for $Rel(G)$ using Monte Carlo (MC) simulation.

The objective of this work is to efficiently calculate values of $Rel(G)$ in a principled way. This work will formulate TNC algorithms for exactly solving $Rel_{S-T}(G)$ when the graph is directed and acyclic, and exactly solving an upper bound for $Rel_{ATR}(G)$ by solving the edge cover problem. These proposed TNC algorithms will be shown to have computational complexity parameterized by the treewidth of the graph. The performance of these algorithms will be tested on grid graphs, random cubic graphs, and a selection of real world transmission graphs.

2 Background

2.1 *Exact Solvers*

Binary Decision Diagram Methods

The current state-of-the-art for exactly solving undirected K-Terminal reliability problems (which includes ATR and S-T) is using binary decision diagrams [7]. These methods consider one edge at a time, factoring the graph into subgraphs, and pruning by identifying isomorphic graphs [8].

However, the pruning is not very efficient. In the worst case for All Terminal Reliability problems, the number of subgraphs to be considered is proportional to $BELL(F_{max})$, where $BELL(k)$ is the k^{th} bell number and F_{max} is the linear-width, or pathwidth, of the graph. If the pathwidth of the graph is small, this algorithm is still useful, but $BELL(k) > (\frac{k}{e\ln(k)})^k$ [9], growing faster than 2^n, so binary decision diagram methods quickly become computationally infeasible.

#SAT solvers

The edge cover problem $Rel_{EC}(G)$ is a superset of $Rel_{ATR}(G)$. We can define

$$Rel_{EC}(G) = \sum_{r \in R} [EC(G_r) * P(G_r)]$$

where $EC(G_r) = 1$ if every node has at least one existing edge, and $EC(G_r) = 0$ otherwise. $Rel_{ATR}(G) \subset Rel_{EC}(G)$ because for every case of $Rel_{ATR}(G)$ every node must have at least one existing edge to ensure connectivity, but $Rel_{EC}(G) \not\subset Rel_{ATR}(G)$ because unconnected "islands" of nodes can satisfy $EC(G_r)$ while not satisfying global connectivity for $C(G_r)$.

Rewritten in conjunctive normal form (CNF) as a monotone #SAT problem [10] [11]:

$$EC(G_r) = \bigwedge_{n_i \in N} (\bigvee_{j \in n_i} e_j(r))$$

where each node n has a set of associated edges $j \in n_i$ if edge e_j has node n_i as a predecessor or successor. The function $e_j(r)$ is equal to True if $r_j = 1$ for graph realization r.

Our ability to write this problem in CNF form, a set of clauses which all must be true, and each clause is satisfied if at least one variable in the clause is true, allows it to be solved by powerful existing model counting solvers such as cachet [12], miniC2D [13], and d4 [14]. However, these solvers are considered to be "black-box" solvers, so there is no ability to estimate their computation time for an arbitrary graph G.

2.2 *Probabilistic Solvers*

Monte Carlo (MC) Methods

If a set of K independent random realizations of a given graph are generated, G_{rand}, then $Rel(G)$ can be estimated from this random sample of graphs. $C(G_{rand})$ can be considered to be a set of Bernoulli trials, a binomial experiment $B(K,p)$ so MC methods can be used to estimate the p of the Bernoulli process and provide a confidence interval.

Given a specific simulation of G_{rand}, the log-likelihood profile of p can be obtained.

Figure 1: Needed MC trials for a given ε at a 95% confidence interval shown on a log-log scale graph: As p_{true} decreases, the number of needed MC trials for a given relative error increases.

../figures/MCTrials.png

The most likely value of p, p_{true}, is obtained by finding the maximum of the profile. To obtain a confidence interval, we can use the profile likelihood method [15]. For a given desired $1-\alpha$ and $\varepsilon = \frac{p_{true}}{p_{-(1-\alpha)\%}}$, the number of needed samples K can be calculated.

For a range of ε values, Figure 1 shows on a log-log scale how the number of needed MC samples decreases as p_{true} increases. Specifically, if p_{true} increases by a factor of 10, the number of trials needed decreases by a factor of 10 while $p_{true} < 0.1$.

The advantage of MC Methods is the number of samples required is only proportional to $\frac{1}{p_{true}}$ and is not proportional to the problem size. The drawback is that p_{true} is not known ahead of time, so the number of samples required could be large if p_{true} is very small and cannot be known ahead of time, and stopping rules must be used, adding uncertainty.

Fully Polynomial-time Randomized Approximation Scheme (FPRAS)

From the analysis above, MC methods become infeasible for estimating graph failure rates when $Rel(G)$ is close to 1. In response to this drawback, Karger [16] developed a Fully Polynomial-time Randomized Approximation Scheme (FPRAS) for estimating $(1-Rel_{ATR}(G))$ that runs in $O(\frac{|E||N|^4}{\varepsilon^3}\ln(|N|))$ time.

However, the FPRAS algorithm only works when $(1-Rel_{ATR}(G)) < |N|^{-4}$. For most engineering applications, the failure chances we care about are small. For example, 1% for 50 year earthquake hazards [17] or $2.0*10^{-6}$ per year for nuclear power plants [18]. Therefore, highly reliable engineering networks with more than $(1-2.0*10^{-6})^{-\frac{1}{4}} \approx 27$ nodes cannot be practically solved using known FPRAS methods. In addition, this FPRAS algorithm is limited and only works for All-Terminal Reliability. While extensions exist for K-Terminal problems, such as the one developed by Paredes [19] which works well in practical settings, these extensions rely on NP-oracles and therefore have exponential worst case behavior.

Therefore, we desire an approximation algorithm that is not dependent on $Rel(G)$, and is instead parameterized by some other graph property that is small for engineering networks of interest. TNC algorithms fit this desire, being parameterized by the treewidth of a graph, which is usually small and constrained for the almost-planar engineering networks we care about.

3 Tensor Network Contraction (TNC)

3.1 *Definitions*

The goal of TNC is to write the underlying satisfiability problem as a series of tensor products. Similar techniques have been investigated in the physics community to solve specific quantum mechanics problems [20] [21]. Each clause in the satisfiability problem is represented as a tensor $T^c_{x_1,x_2,\ldots,x_k}$. If the variables $x_1,x_2,\ldots,x_k$ satisfy the underlying clause, then $T_{x_1,x_2,\ldots,x_k}=1$. For example, the T^c that encodes the Boolean clause $(x_1\vee x_2\vee x_3\vee x_4)$ is:

$$T^c_{x_1,x_2,x_3,x_4} = \begin{cases} 0, & \text{if } x_1=x_2=x_3=x_4=0 \\ 1, & \text{otherwise} \end{cases}$$

The number of solutions to the satisfiability problem can be calculated by applying the tensor product to every clause tensor T^c. The tensor product is defined as

$$T^p_{x_1,\ldots,x_k,z_1,\ldots,z_k} = T^c_{x_1,\ldots,x_k,y_1,\ldots,y_k} \otimes T^c_{y_1,\ldots,y_k,z_1,\ldots,z_k}$$

where $\otimes$ expands into

$$T^p_{x_1,\ldots,x_k,z_1,\ldots,z_k} = \sum_{y_i\in Y}\sum_{y_i=0}^{|y_i|} T^c_{x_1,\ldots,x_k,y_1,\ldots,y_k} * T^c_{y_1,\ldots,y_k,z_1,\ldots,z_k}$$

where Y is the set of all variables in common between the two tensors, and $|y_i|$ is the number of states that variable y_i can take. While y_i can take an arbitrary number of states, the remainder of this work will only consider a two-state Boolean variable.

One complication is that a tensor product is only clearly defined if each variable appears exactly once or twice, while in many satisfiability problems a variable can appear more than twice. This complication can be addressed by assigning each T^c a unique set of variables, and then creating additional variable tensors T^v to apply constraints on the variables.

Two common constraints are defined below. To constrain a Boolean variable x_1 to take the opposite value of x_2 (as needed for a Boolean formula containing x_1 and $\neg x_1$), the following tensor $T^v_{x_1,x_2}$ is set as

$$T^v_{x_1,x_2} = \begin{cases} 1, & \text{if } x_1=0 \text{ and } x_2=1 \\ 1, & \text{if } x_1=1 \text{ and } x_2=0 \\ 0, & \text{otherwise} \end{cases}$$

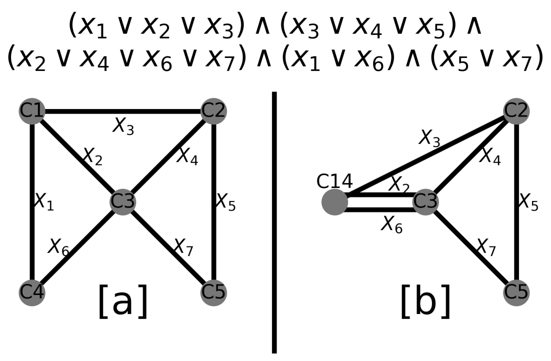
To constrain a Boolean variable x_1 to take the same value of x_2 and apply a probability p of both variables being true (as needed to define an unreliable network edge), the tensor $T^v_{x_1,x_2}$ is set as

$$T^v_{x_1,x_2} = \begin{cases} p, & \text{if } x_1=1 \text{ and } x_2=1 \\ 1-p, & \text{if } x_1=0 \text{ and } x_2=0 \\ 0, & \text{otherwise} \end{cases}$$

3.2 *Graph Representation*

Tensor multiplications can be represented as a node and edge graph, G_T, where each tensor

Figure 2: Example Tensor Graph [a] and example tensor contraction of edge X_1 [b]


$(x_1 \vee x_2 \vee x_3) \wedge (x_3 \vee x_4 \vee x_5) \wedge$
$(x_2 \vee x_4 \vee x_6 \vee x_7) \wedge (x_1 \vee x_6) \wedge (x_5 \vee x_7)$
C1
C2
C3
C4
C5
C14
X_1
X_2
X_3
X_4
X_5
X_6
X_7
[a]
[b]

is a node and each variable is an edge. An example of a tensor graph can be seen in Figure 2a. A tensor product can be represented as an edge contraction on this graph. The contraction of edge X_1 is visually shown in Figure 2b. Once all edges are contacted, only a scalar value remains, counting the number of solutions to the Boolean problem.

3.3 *Contraction Ordering*

Care must be taken when choosing the order to perform the edge contractions. Assuming Boolean variables, the product $T^c_{x_1,\ldots,x_n,y_1,\ldots,y_n} \otimes T^c_{y_1,\ldots,y_n,z_1,\ldots,z_n}$ requires $2^{|x|+|y|+|z|}$ multiplications and additions, and $2^{|x|+|z|}$ numbers need to be stored in memory for the resulting tensor. Markov, Igor L and Shi, Yaoyun show how to determine an optimal edge contraction ordering to minimize $|x|$, $|y|$, and $|z|$, also known as elimination ordering, from an optimal tree decomposition of the line graph of G_T, $LG(G_T)$ [22]. Dumitrescu et al. [23] demonstrate how algorithms from the PACE 2017 challenge [2] can be used to obtain better approximate tree decompositions for some tensor graphs representing quantum many body problems.

Harvey, Daniel J and Wood, David R provide a few different upper bounds for the treewidth of $LG(G_T)$, $tw(LG(G_T))$, bounding the size of the largest tensor [24]:

$$tw(LG(G_T)) < (tw(G_T)+1) * D_m(G_T) - 1$$

where $D_m(G_T)$ is the maximum degree of graph G_T. Dudek et al. [25] also show how high-rank tensors can be factored into a tensor tree to further minimize memory and computational requirements of the TNC.

Overall, in the worst case for infrastructure networks with bounded max degree (due to physical limitations), the largest number of variables for a single tensor is linearly proportional to the treewidth of G_T. Therefore, the computational complexity is at most $2^{C*tw(G)}$, where C is a constant between 1 and $D_m(G_T)$.

For the following formulations, if every variable tensor is contracted into an adjacent clause tensor, the resulting tensor graph is isomorphic to the underlying graph G. Therefore, for these formulations, $tw(G_T)$ is equal to $tw(G)$.

3.4 *All Terminal Reliability Formulation*

For $Rel_{ATR}(G)$ there is no known polynomial sized satisfiability equation, unless auxiliary variables are used [19]. Therefore, a tensor graph for the edge cover problem, $Rel_{EC}(G)$ will be formulated instead. The edge cover problem is satisfied if every node in the graph G has at least one existing edge. Therefore, the clause for a node n with connecting edges $e \in E$ is $(e^n_1 \vee e^n_2 \vee \ldots \vee e^n_i)$.

Each edge e_i has a probability p of existing, and each variable e^n_i takes the same correlated state for every superscript n. Contracting the tensor graph G_T of these tensors will yield the probability of a satisfying edge cover for the graph G.

3.5 *S-T Reliability Formulation*

For $Rel_{S-T}(G)$, the problem is satisfied if any inbound edge connected to the terminal node n_t is connected to a "marked" node. A node is marked if any of its inbound edges is connected to a "marked" node or a "source" node. For an acyclic directed network, a node n_b is "marked" if and only if there is a path from the source node to node n_b (This statement does not hold true for graphs with cycles). Even in this restricted case, $Rel_{S-T}(G)$ is still a #P-complete problem [26].

The clause for the terminal node n_t with inbound edges $e \in E$ is $(e^{n_t}_1 \vee e^{n_t}_2 \vee \ldots \vee e^{n_t}_i)$.

For a node between the source and terminal nodes n_b with inbound edges $e \in E$, it must satisfy the following clause:

$$(m_{n_b} \wedge (e_1^{n_b} \vee \ldots \vee e_i^{n_b})) \vee$$
$$(\neg m_{n_b} \wedge \neg(e_1^{n_b} \vee \ldots \vee e_i^{n_b}))$$

where m_{n_b} is a variable indicating if node n_b is marked.

For each variable e_i, it must be constrained to only be True with probability $e_i.p$ when the tail is connected to a "marked" or source node, $m_{n_b} = 1$, and always False when the tail is not connected to a "marked" node. Therefore, the corresponding variable tensor for edge i outbound from node b and inbound to node y is:

$$T^{v_e}_{m_{n_{b,y}}, e_i^{n_y}} =$$
$$\begin{cases} e_i.p, & \text{if } m_{n_{b,y}} = 1 \text{ and } e_i^{n_y} = 1 \\ 1 - e_i.p, & \text{if } m_{n_{b,y}} = 1 \text{ and } e_i^{n_y} = 0 \\ 1, & \text{if } m_{n_{b,y}} = 0 \text{ and } e_i^{n_y} = 0 \\ 0, & \text{if } m_{n_{b,y}} = 0 \text{ and } e_i^{n_y} = 1 \end{cases}$$

In addition, the directed "marked" variable $m_{n_{b,y}}$ must be constrained to the same value as m_{n_b}. Contracting the tensor graph G_T of these tensors will yield the exact probability of a satisfying path from the source node to the terminal node for the graph G.

3.6 *Tensor Network Contraction (TNC) Advantages*

TNC algorithms have many advantages over the previously described exact solvers and probabilistic solvers. The upper bound computational complexity of $2^{C*tw(G)}$ is significantly better than the $BELL(Pathwidth(G))$ of the binary decision diagram methods and the unknown upper bounds of the #SAT methods. This bound is not dependent on $Rel(G)$, so TNCs can solve some highly reliable networks faster than probabilistic solvers. The computational effort of a TNC can be known ahead of time (after the approximate tree decomposition), so reliability engineers can confidently choose the most efficient reliability solver algorithm. In addition, for infrastructure networks of interest, they are usually near-planar, which bounds treewidth to $2 * \sqrt{6*(k+1)*|N|}$ [27] where k is the number of allowed crossings for each edge, and treewidth is frequently lower than this bound [28].

TNCs only require vectorized multiplication and addition operations which are very efficient for CPUs and GPUs to compute, while binary decision diagram methods and #SAT methods require many conditional if-then statements which are more difficult to optimize. While probabilistic solvers are perfectly parallel (each sample can be done on a separate computer), the individual tensor contractions can also be broken up and dispatched to multiple parallel computing units.

4 Results and Discussion

4.1 *Benchmark Graphs*

To evaluate the empirical performance of the proposed TNC algorithm, a few classes of graphs will be considered. The first considered class of graphs are grid graphs. As most infrastructure networks are usually near-planar, grid graphs can be considered as the ideal case of planar graphs.

Second, random connected cubic graphs will be considered. Using a set of reliability preserving transformations [29], and by splitting high degree nodes into a chain of degree 3 nodes connected by unfailing edges, all graphs can be converted to a cubic graph with equivalent $Rel(G)$. A 1-Flipper Markov Chain Monte Carlo (MCMC) algorithm will be used to uniformly generate these random cubic graphs [30].

Third, a collection of 58 US power transmission networks [3] will be considered. These graphs will be reduced using reliability preserving transformations before reliability calculations are performed.

All benchmarks are performed on a Intel Core i7-4810MQ CPU @ 2.90GHz, with 16 GB of RAM. All code is single threaded. The code used to generate these graphs can be seen at this link [LINK GITHUB HERE].

4.2 *Grid Graphs*

4.2.1 *Computational Time*

As seen in Figure 3a, as the grid dimension increases, both the number of subgraphs for the binary decision diagram method and the number of floating point operations for the TNC increases exponentially. Figure 3b shows the wall clock time taken for each method. In both cases, the slope of the TNC is significantly smaller than the binary decision diagram method, showing significant computational advantages for calculating $EC(G)$ and $Rel_{S-T}(G)$.

Further computational advantages can be seen if one dimension of the grid graph is fixed in size (8 nodes is large enough for a non-trivial treewidth size and small enough to be computed quickly by both methods). As seen in Figure 3c, as n increases for the $8xn$ grids, both the number of subgraphs for the binary decision diagram method and the number of floating point operations for the TNC increases linearly. However, as seen in Figure 3d, the wall clock time taken for the binary decision diagram method increases quadratically (each subgraph needs an $O(|E|)$ connectivity check) while the TNC time only increases linearly. For graphs of bounded treewidth, TNCs show significant computational improvement.

4.2.2 *Monte Carlo (MC) Comparison*

As the TNC only bounds $Rel_{ATR}(G)$ by calculating $EC(G)$, we can evaluate the quality of this estimate by determining the number of MC trials needed to obtain bounds of $Rel_{ATR}(G)$ better than $EC(G)$. Using a 95% confidence interval, we can calculate the number of MC trials needed to create a confidence interval that excludes $EC(G)$.

For an edge failure rate of 0.01, both $Rel_{ATR}(G)$ and $EC(G)$ are approximately constant at 0.9996. Approximately 24 million MC trials are needed to rule out $EC(G)$, and this count is insensitive to the size of the grid. Therefore, for reliable grid graphs, $EC(G)$ is a good bound.

Figure 3: Computational complexity and wall clock time for nxn and $8xn$ grid graphs. Subgraphs and $REL_{ATR}(G)$ are BDD calculations, and floating point operations and $EC(G)$ are TNC calculations.

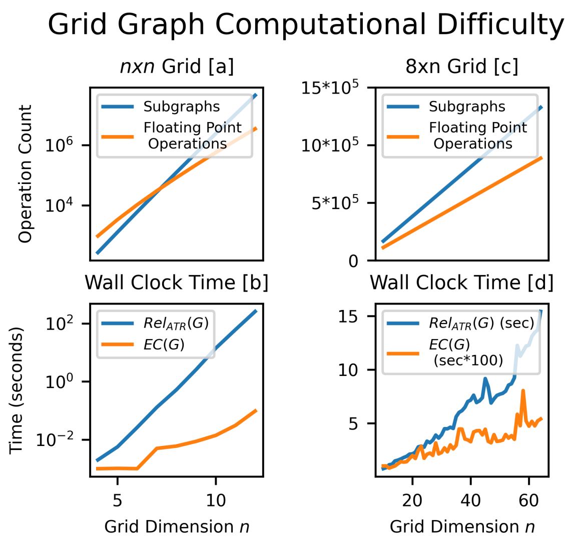
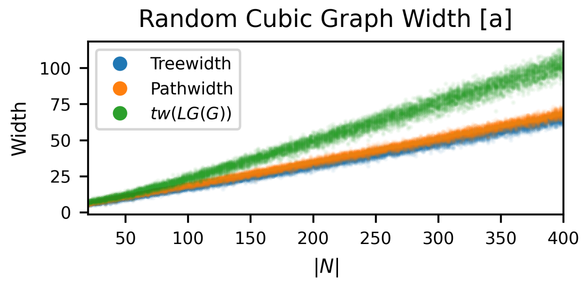


Figure 4: Approximated treewidth, pathwidth, and treewidth of the line graph of G_{rc} for 10,000 randomly generated connected cubic graphs G_{rc}, and the ratios between these widths.

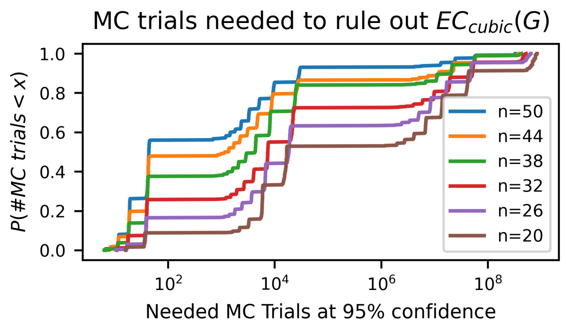


4.3 *Random Connected Cubic Graphs*

4.3.1 *Computational Time*

A random selection of 10,000 random connected cubic graphs G_{rc} from node count $|N| = 20$ to $|N| = 400$ were generated. The treewidth of G_{rc} and the treewidth of $LG(G_{rc})$ were computed using an approximate treewidth solver [2] for 6 seconds. The pathwidth of G_{rc} was estimated from the tree decomposition of G_{rc}. As seen in Figure 4, there is a linear increase in approximated treewidth as graph size increases.

Figure 5: The empirical cumulative distribution function of randomly generated connected cubic graphs of size |N| that need less than X MC trials of $Rel_{ATR}(G)$ to rule out the bound from $EC(G)$ at the 95% confidence level.



4.3.2 *Monte Carlo (MC) Comparison*

A random selection of 10,000 random connected cubic graphs G_{rc} from node count $|N| = 20$ to $|N| = 50$ were generated and solved for $EC(G)$ and $Rel_{ATR}(G)$. Each edge had a failure rate of 0.01 to represent a network with high reliability. $EC(G)$ is very tightly constrained, with a standard deviation of $1.1 * 10^{-12}$ for all |N|, and can be estimated as $EC(G) \approx (1 - |N| * 0.01^3)$. $Rel_{ATR}(G)$ had greater variance, with a range of [0.92265,0.99998].

For each graph, the number of MC trials needed to rule out the $EC(G)$ bound at the 95% confidence interval was calculated. Figure 5 shows the proportion of cubic graphs that need less than X MC trials to rule out the $EC(G)$ bound. The observed segmented stair-step pattern is unusual, and is likely caused by the graphs in each segment sharing some topological feature such as a bridge.

As graph size increases, the empirical cumulative distribution function pushes up and to the left, indicating more graphs need fewer MC trials to rule out the $EC(G)$ bound. This means as highly reliable cubic graphs become larger, $EC(G)$ becomes a worse bounding value of $Rel_{ATR}(G)$.

Overall, despite the computational advantages of TNCs, the treewidth of these graphs scales linearly with size, resulting in computational complexity growing exponentially with the size of the graph. In addition, $EC(G)$ as measured by TNCs is a poor bounding value for many cubic graphs, only a few MC trials are needed to achieve a better bounding value. This bounding value becomes worse as the graph size increases.

4.4 *Power Transmission Grids*

The $Rel_{ATR}(G)$ and $EC(G)$ of 58 transmission power grids [3] were calculated. Table 1 shows the node and edge count of these graphs after reliability preserving reductions, and the result of the $Rel_{ATR}(G)$ and $EC(G)$ calculations at edge failure rate 0.5, 0.1 and 0.01. Some graphs were omitted due to trivial structure or inability to compute $Rel_{ATR}(G)$.

Figure 6 shows the treewidth and $tw(LG(G))$ of the power grids in relation to the previously analyzed graphs. In general, the treewidth of the power grids is smaller than equal sized cubic graphs, making them very computationally efficient to solve. However, $tw(LG(G))$ of the power grids are significantly greater than their treewidth, comparable to $tw(LG(G_{rc}))$ of equal sized cubic graphs, due to the presence of high degree nodes in the power grids. The tensor factoring techniques in [25] may reduce these large values of $tw(LG(G))$. Despite this, TNCs still quickly solve $EC(G)$ and $Rel_{S-T}(G)$ of these graphs in comparison to the binary decision diagram techniques.

When each edge only has a 1% chance of failure, $EC(G)$ is a good approximation for 37 of the power grids as seen in the tall green bars in Figure 7. It would take more than 1,000,000 MC trials to rule out the $EC(G)$ approximation for these graphs. For the largest power grids, only 1,000 MC trials are needed to rule out $EC(G)$.

5 Conclusion

5.1 *Results Summary*

Overall, TNCs for solving $Rel_{S-T}(G)$ and estimating $Rel_{ATR}(G)$ demonstrate many computational advantages on many practical networks. These methods are parameterized by

Figure 6: Treewidth and $tw(LG(G))$ comparison between grid graphs, random cubic graphs, and power transmission grids.

../figures/WidthCompare.png

the treewidth of the network, so graphs with low treewidth such as grids and the 58 power transmission networks can be quickly solved. In the general case as represented as random cubic graphs, TNCs are not as computationally efficient due to the linear relationship between treewidth and random cubic graph size. In addition, the presence of high degree nodes in the power transmission networks introduces a large constant factor between treewidth and the computationally relevant treewidth of the line graph. Despite these limitations, the tensor methods are still 10 to 100 times faster than the state-of-the-art binary decision diagram methods as measured by wall clock time.

When estimating $Rel_{ATR}(G)$ by calculating $EC(G)$, TNCs show excellent results on grid graphs. As edge failure rate decreases, as is the case for highly reliable networks, $EC(G)$ becomes a better estimator. This increase in estimation accuracy is likely due to the fact that as edge failures become less likely, multiple edge failures needed to disconnect the graph become exponentially less likely, which heavily discounts occurrences of disconnected “islands” of nodes that satisfy $EC(G)$ and do not satisfy $Rel_{ATR}(G)$.

However, $EC(G)$ is only a good estimate

Figure 7: Number of MC Trials needed to rule out $EC(G)$ at the 95% confidence level for each power grid at different edge failure rates p.

../figures/PowerGridMC.png

for $Rel_{ATR}(G)$ for a small subset of cubic graphs and 37 out of 55 of the power grids. This work did not investigate why $EC(G)$ was a good estimate for these graphs, although it is likely due to topological bottlenecking effects.

5.2 *Future Work*

In relation to algorithm design, the primary bottleneck to TNCs is the memory requirement. Techniques such as sparse arrays, online matrix compression, or tensor factoring can be used to reduce the memory footprint of large tensors. Additionally, many quantum computer algorithms can be described as tensor contractions [31] [32], so these TNCs may be able to exploit quantum computers to achieve a quantum speedup over traditional algorithms.

As $EC(G)$ is not always a good approximation for $Rel_{ATR}(G)$, it would be beneficial to classify the graphs where $EC(G)$ is a good approximation of $Rel_{ATR}(G)$. If an algorithm can quickly identify these classes of graphs, then tensor methods can quickly and confidently estimate $Rel_{ATR}(G)$ using $EC(G)$. In addition, determining how to incorporate TNCs with other #SAT solvers into a virtual best solver will greatly expand the classes of graphs where $EC(G)$ can be quickly solved.

While $Rel_{S-T}(G)$ can be exactly solved by tensor methods for directed acyclic graphs, the introduction of cycles causes drastic multiplicative overcounting of solutions if the given $Rel_{S-T}(G)$ formulation is used. Determining a better tensor graph for solving $Rel_{S-T}(G)$ or determining how to compensate for the multiplicative overcounting can expand the number of graphs that can be exactly solved by TNCs.

Table 1: Power Transmission Network Attributes

ID	|V|	|E|	TW	PW	LGW	Rel_{time}	$Rel_{0.5}$	$Rel_{0.1}$	$Rel_{0.01}$	EC_{time}	$EC_{0.5}$	$EC_{0.1}$	$EC_{0.01}$
1	16	29	4	5	8	0.017332	0.12167154	0.98957460	0.99999087	0.174876	0.25502051	0.99065582	0.99999096
2	19	33	4	4	8	0.000186	0.05365770	0.98211723	0.99998379	0.000333	0.17822816	0.98764111	0.99998795
3	9	18	5	5	9	0.000597	0.42333221	0.99636464	0.99999694	0.000723	0.52156830	0.99651838	0.99999695
4	12	20	4	4	6	0.000000	0.12807083	0.97834075	0.99988980	0.000172	0.31490231	0.99091973	0.99999098
5	4	6	4	3	5	0.000000	0.59375000	0.99581400	0.99999597	0.000000	0.64062500	0.99605700	0.99999600
6	24	48	5	6	11	0.004095	0.06873492	0.98659392	0.99998786	0.000753	0.15051040	0.98749687	0.99998794
7	13	24	5	5	9	0.004037	0.19829893	0.99110197	0.99999191	0.000333	0.31878930	0.99177120	0.99999197
8	16	32	5	5	9	0.004749	0.16953775	0.99180050	0.99999289	0.000913	0.31583457	0.99342016	0.99999394
9	19	34	4	4	7	0.000333	0.05261123	0.98166897	0.99998374	0.001684	0.17841807	0.98687403	0.99998697
10	9	15	4	4	6	0.000000	0.27203369	0.99197148	0.99999289	0.000256	0.42657471	0.99376760	0.99999397
11	7	11	4	3	6	0.000000	0.34082031	0.99218972	0.99999290	0.000513	0.48193359	0.99398015	0.99999399
12	17	36	6	6	12	0.023612	0.22980855	0.99329227	0.99999393	0.002440	0.32829193	0.99430582	0.99999493
13	6	9	4	3	5	0.000333	0.38671875	0.99237749	0.99999291	0.000334	0.51367188	0.99409283	0.99999400
14	18	32	4	3	8	0.000342	0.05216674	0.97114616	0.99988185	0.000333	0.19721385	0.98784985	0.99998797
15	23	39	6	6	9	0.005238	0.02955951	0.97957456	0.99998174	0.000927	0.10569971	0.98202115	0.99998197
16	23	40	5	6	10	0.003775	0.02555826	0.96167545	0.99978280	0.000506	0.12043086	0.98454274	0.99998493
17	6	11	4	3	7	0.000000	0.55371094	0.99753011	0.99999796	0.000336	0.62939453	0.99760764	0.99999796
18	4	6	4	3	5	0.000000	0.59375000	0.99581400	0.99999597	0.000000	0.64062500	0.99605700	0.99999600
19	9	17	4	4	8	0.000269	0.36923218	0.99458268	0.99999496	0.000000	0.47051239	0.99482186	0.99999498
20	16	30	5	5	9	0.002691	0.14746502	0.99081730	0.99999189	0.001519	0.27150461	0.99154743	0.99999195
21	18	37	5	5	11	0.002053	0.16092585	0.99022499	0.99999092	0.000337	0.25048232	0.99066939	0.99999096
22	7	14	5	4	8	0.001577	0.55395508	0.99764972	0.99999797	0.000501	0.62353516	0.99768485	0.99999797
23	9	16	4	3	8	0.000250	0.31793213	0.99331619	0.99999393	0.003037	0.43721008	0.99385534	0.99999398
24	31	51	5	5	8	0.004045	0.00333280	0.96024725	0.99996437	0.003970	0.04349889	0.97436290	0.99997398
25	9	15	4	3	6	0.000757	0.25952148	0.99102678	0.99999189	0.000000	0.41384888	0.99297602	0.99999299
26	27	49	5	5	8	0.001722	0.01736987	0.96797843	0.99987974	0.000333	0.09222602	0.98204488	0.99998198
28	27	47	5	6	9	0.006189	0.01239560	0.97220986	0.99997466	0.000000	0.07123664	0.97908383	0.99997897
29	34	71	6	7	15	0.046117	0.02852902	0.98122226	0.99998281	0.019774	0.07020389	0.98231394	0.99998291
30	12	23	4	5	10	0.002861	0.24995208	0.99155426	0.99999195	0.000594	0.34471893	0.99195646	0.99999199
31	21	37	5	5	8	0.000335	0.03919450	0.97994898	0.99998177	0.000334	0.14345905	0.98499664	0.99998498
32	19	33	5	5	10	0.003519	0.04548891	0.96999886	0.99988083	0.000334	0.15963011	0.98581925	0.99998596
33	29	60	6	8	13	0.044070	0.04883569	0.98456505	0.99998585	0.005141	0.10738096	0.98535319	0.99998592
34	36	70	6	7	12	0.049390	0.01668428	0.97998377	0.99998179	0.002529	0.05712185	0.98122264	0.99998190
35	14	25	5	5	9	0.002144	0.15695018	0.98911478	0.99998990	0.000334	0.28797701	0.99087288	0.99999098
36	29	56	5	6	10	0.007635	0.02254811	0.97219655	0.99988380	0.001594	0.08928589	0.98340645	0.99998392
37	20	38	5	5	9	0.004598	0.07215021	0.98532712	0.99998682	0.003260	0.18448659	0.98851388	0.99998894
38	31	64	5	6	9	0.002900	0.03717040	0.98529150	0.99998684	0.001087	0.11263403	0.98646343	0.99998694
39	61	124	9	10	18	1.551559	0.00116882	0.95569496	0.99986768	0.047763	0.00951425	0.96999788	0.99997085
40	23	43	4	4	9	0.003473	0.03814892	0.98081026	0.99998276	0.000774	0.12076441	0.98385333	0.99998396
41	18	32	4	4	8	0.003706	0.05750333	0.96609719	0.99978686	0.000333	0.19745638	0.98784986	0.99998797
42	36	68	5	5	14	0.006670	0.00381684	0.81463673	0.98037497	0.007386	0.02868238	0.97142746	0.99997098
43	31	58	5	7	11	0.027855	0.01692225	0.97759178	0.99997973	0.003529	0.06477194	0.98050149	0.99998092
44	78	150	7	10	17	0.459141	0.00003289	0.93844113	0.99994142	0.076132	0.00077407	0.94594519	0.99994492
45	55	102	6	9	11	0.101259	0.00035646	0.94503474	0.99985754	0.002354	0.00611823	0.95975279	0.99995896
46	60	116	6	7	16	0.081021	0.00023098	0.92515124	0.99965566	0.026080	0.00335799	0.95504980	0.99995397
47	48	86	7	8	12	0.075847	0.00080723	0.95996061	0.99996351	0.001003	0.01131853	0.96602777	0.99996592
48	52	97	8	10	18	0.528176	0.00066992	0.93887114	0.99975969	0.100093	0.00760182	0.96237592	0.99996194
49	60	111	6	8	13	0.130469	0.00014183	0.93194376	0.99975358	0.008036	0.00387670	0.95751108	0.99995693
50	36	74	6	7	14	0.072151	0.02141654	0.97943954	0.99998083	0.006922	0.05905184	0.98061142	0.99998094
51	58	123	7	8	18	0.167014	0.00129092	0.96131942	0.99996365	0.095085	0.00715189	0.96402050	0.99996392
52	77	146	8	10	20	2.546974	0.00002134	0.76559700	0.97996221	0.214653	0.00090534	0.94845804	0.99994788
53	73	122	6	7	13	0.121916	0.00000214	0.89256304	0.99962324	0.009748	0.00057446	0.93907596	0.99993696
55	208	390	12	14	31	649.388035	0.00000000	0.66512161	0.97965552	313.083091	0.00000001	0.86434283	0.99985575
56	184	364	10	14	28	93.468759	0.00000000	0.85395647	0.99967281	97.081757	0.00000011	0.88667166	0.99988172
58	406	785	10	15	24	293.777338	0.00000000	0.56241914	0.98813658	12.744727	0.00000000	0.74302255	0.99970647

—

6 REFERENCES

References

[1] Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[2] Holger Dell, Christian Komusiewicz, Nimrod Talmon, and Mathias Weller. The pace 2017 parameterized algorithms and computational experiments challenge: The second iteration. In *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[3] Jian Li, Leonardo Dueñas-Osorio, Changkun Chen, Benjamin Berryhill, and Alireza Yazdani. Characterizing the topological and controllability features of us power transmission networks. *Physica A: Statistical Mechanics and its Applications*, 453:84–98, 2016.

[4] Louis L Levy and Albert H Moore. A monte carlo technique for obtaining system reliability confidence limits from component test data. *IEEE Transactions on Reliability*, 16(2):69–72, 1967.

[5] Eduardo Cotilla-Sanchez, Paul DH Hines, Clayton Barrows, and Seth Blumsack. Comparing the topological and electrical structure of the north american electric power infrastructure. *IEEE Systems Journal*, 6(4):616–626, 2012.

[6] Alberto Escalera, Barry Hayes, and Milan Prodanović. A survey of reliability assessment techniques for modern distribution networks. *Renewable and Sustainable Energy Reviews*, 91:344–357, 2018.

[7] Jacques Carlier and Corinne Lucet. A decomposition algorithm for network reliability evaluation. *Discrete Applied Mathematics*, 65(1-3):141–156, 1996.

[8] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. K-terminal network reliability measures with binary decision diagrams. *IEEE Transactions on Reliability*, 56(3):506–515, 2007.

[9] Daniel Berend and Tamir Tassa. Improved bounds on bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010.

[10] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

[11] Radislav Vaisman, Ofer Strichman, and Ilya Gertsbakh. Model counting of monotone conjunctive normal form formulas with spectra. *INFORMS Journal on Computing*, 27(2):406–415, 2015.

[12] Tian Sang, Paul Beame, and Henry Kautz. Heuristics for fast exact model counting. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 226–240. Springer, 2005.

[13] Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[14] Jean-Marie Lagniez and Pierre Marquis. An improved decision-dnnf compiler. In *IJCAI*, volume 17, pages 667–673, 2017.

[15] DJ Venzon and SH Moolgavkar. A method for computing profile-likelihood-based confidence intervals. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 37(1):87–94, 1988.

[16] David R Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM review*, 43(3):499–522, 2001.

[17] International Code Council. *2018 International Building Code*. International Code Council, 2017.

[18] United States Nuclear Regulatory Commission. Safety/risk assessment results for generic issue 199, implications of updated probabilistic seismic hazard estimates in central and eastern united states on existing plants., 2010.

[19] Roger Paredes, Leonardo Dueñas-Osorio, Kuldeep S Meel, and Moshe Y Vardi. Principled network reliability approximation: A

counting-based approach. *Reliability Engineering & System Safety*, 191:106472, 2019.

[20] Feng Pan, Pengfei Zhou, Sujie Li, and Pan Zhang. Contracting arbitrary tensor networks: general approximate algorithm and applications in graphical models and quantum circuit simulations. *Physical Review Letters*, 125(6):060503, 2020.

[21] Jacob Biamonte. Lectures on quantum tensor networks. *arXiv preprint arXiv:1912.10049*, 2019.

[22] Igor L Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008.

[23] Eugene F Dumitrescu, Allison L Fisher, Timothy D Goodrich, Travis S Humble, Blair D Sullivan, and Andrew L Wright. Benchmarking treewidth as a practical component of tensor network simulations. *Plos one*, 13(12):e0207827, 2018.

[24] Daniel J Harvey and David R Wood. The treewidth of line graphs. *Journal of Combinatorial Theory, Series B*, 132:157–179, 2018.

[25] Jeffrey M Dudek, Leonardo Duenas-Osorio, and Moshe Y Vardi. Efficient contraction of large tensor networks for weighted model counting through graph decompositions. *arXiv preprint arXiv:1908.04381*, 2019.

[26] J Scott Provan. The complexity of reliability computations in planar and acyclic graphs. *SIAM Journal on Computing*, 15(3):694–702, 1986.

[27] Vida Dujmovic, David Eppstein, and David R Wood. Structure of graphs with locally restricted crossings. *SIAM Journal on Discrete Mathematics*, 31(2):805–824, 2017.

[28] Silviu Maniu, Pierre Senellart, and Suraj Jog. An experimental study of the treewidth of real-world graph data. In *22nd International Conference on Database Theory (ICDT 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[29] Andrew M Shooman and Aaron Kershenbaum. Exact graph-reduction algorithms for network reliability analysis. In *IEEE Global Telecommunications Conference GLOBECOM'91: Countdown to the New Millennium. Conference Record*, pages 1412–1420. IEEE, 1991.

[30] Tomás Feder, Adam Guetz, Milena Mihail, and Amin Saberi. A local switch markov chain on given degree graphs with application in connectivity of peer-to-peer networks. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 69–76. IEEE, 2006.

[31] Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, 2017.

[32] Leonardo Duenas-Osorio, Moshe Vardi, and Javier Rojo. Quantum-inspired boolean states for bounding engineering network reliability assessment. *Structural Safety*, 75:110–118, 2018.