Efficient Network Reliability Evaluation for Client-Server Model

Kengo Nakamura*, Takeru Inoue†*, Masaaki Nishino* and Norihito Yasuda*
*NTT Communication Science Laboratories, NTT Corporation, Kyoto, Japan
†NTT Network Innovation Laboratories, NTT Corporation, Kanagawa, Japan
Email: {kengo.nakamura.dx, takeru.inoue.dr, masaaki.nishino.uh, norihito.yasuda.hn}@hco.ntt.co.jp

GLOBECOM 2021 - 2021 IEEE Global Communications Conference | 978-1-7281-8104-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/GLOBECOM46510.2021.9685283

***Abstract*—Network reliability, i.e., the probability of connecting a set of specified nodes under stochastic link failure, is a key indicator of network infrastructure, such as communication and power. Since network reliability evaluation is a computationally heavy task, several methods have been proposed to efficiently perform it. However, modern network infrastructures follow the client-server model, where many clients are served independently, so we have to evaluate the network reliability for *every* set consisting of the servers and each client. This evaluation process involves repetitive evaluations while changing the set, which imposes a heavy burden on network operators. This paper proposes a method that efficiently performs network reliability evaluation for the client-server model. Since our method is designed to evaluate reliability for multiple clients without explicit repetition, the computational complexity does not increase compared to the case where existing methods evaluate reliability for a single client. Numerical experiments using datasets of various topologies, including real communication networks, reveal great efficiency. Our method is more than 100 times faster than an existing method that requires repeated evaluation, e.g., it takes only 27 seconds to compute the reliability for 670 clients on a large network with 821 links.**

***Index Terms*—Network reliability, dynamic programming**

I. Introduction

Current society largely depends on a number of network infrastructures [1], such as telecommunications and electric power. Network reliability is traditionally defined as the probability that specified k terminals can be connected on a probabilistic network where link failure reflects a stochastic process. The problem of computing such probability is called *k-terminal network reliability* (k-NR) [2]–[4]. However, k-NR is known to be in #P-complete [5], a computationally tough class, since we must examine exponentially many link combinations in which the terminals are connected. The research community is devoting much effort to developing efficient methods [2], [3], and it is now possible to exactly compute k-NR for a real network with nearly 200 links [4], [6], [7].

Today's network infrastructures usually follow the client-server model. For example, cloud services are provided from web servers to browsers (clients) distributed in a network, and electric power is delivered from a substation (server) to consumers (clients) through a power distribution network. Since clients are served independently, their reliability should also be evaluated independently to assess their service level agreement. However, this evaluation task is very challenging for network operators, because heavy k-NR have to be solved for many clients; a terminal set of k-NR (Fig. 1a) corresponds to each set consisting of servers and variable clients (Fig. 1b). Since this new problem generalizes k-NR for variable clients with fixed servers, we call it *k-terminal network reliability for fixed servers and variable clients* (k-NR+) in this paper. To the best of our knowledge, no method has been proposed to efficiently perform k-NR+ for many clients.

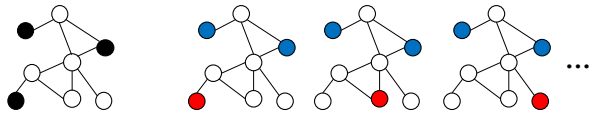
(a) k-NR. (b) k-NR+.

Fig. 1. k-NR and k-NR+ problems ($k = 3$). (a) k-NR computes the probability of connecting the set of terminals (black). (b) k-NR+ computes the probability of connecting each client (red) with the fixed servers (blue).

Before describing our contributions in this paper, we provide a brief overview of the state-of-the-art method for k-NR [2], [3] (HH method) using Fig. 2, and details of the HH method are given in Section III-A. This method determines the state (working or failing) of each link, from the top to the bottom of the search diagram in Fig. 2. To avoid generating an exponential number of states in the diagram, the method merges states "equivalent" in terms of connectivity between the k terminals, as shown by the bottom-left state in Fig. 2. Although the method is efficient for k-NR, we have to repeat it for every client in k-NR+.

This paper proposes a dynamic programming (DP) method to solve k-NR+. Surprisingly, the computational complexity of our method is the same as that of the HH method, even though our method computes k-NR+ while the HH method performs k-NR. Our contributions are summarized as follows:

- We design an efficient k-NR+ method that first constructs a search diagram downward by only considering servers and then computes reliability for every client while scanning the diagram upward. This process involves no repetition for clients, and it imposes no overhead in terms of computational complexity. In addition, the search process allows our method to solve k-NR more efficiently than does the HH method when k is a small constant.
- Our method is evaluated numerically with synthetic graphs and real communication networks. As predicted by the computational complexity, our method runs much faster than the HH method. Our method outperforms the

978-1-7281-8104-2/21/$31.00 ©2021 IEEE

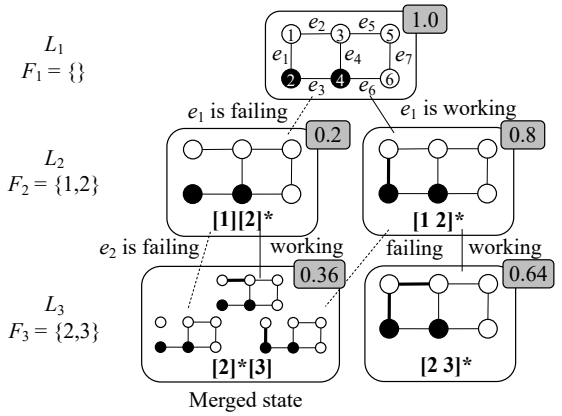

Fig. 2. Part of previous works' search diagram [2], [3].

HH method by more than 100 times; e.g., for a topology with 821 links and 670 clients, our method takes less than 27 seconds to solve k-NR+, while the repeated use of the HH method finishes only 2% of the clients in an hour.

II. Problem Statement

Our network model is a connected undirected graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. Each edge $e_i \in E$ has a probability $p_i \in [0, 1]$ of working, i.e. e_i fails with probability $1-p_i$. We assume that the state of e_i (working or failing) is statistically independent of the other edges' states. For subset E' of E, we use the term "the state of E'" to represent a collection of the states of edges in E'. Given state X of all edges E, we can consider the corresponding graph $G^X = (V, E^X)$, where $E^X \subseteq E$ is the set of working edges. The probability of this state is computed as

$$p(X) = \prod_{i:e_i \in E^X} p_i \cdot \prod_{i:e_i \in E \setminus E^X} (1 - p_i). \tag{1}$$

Given $K \subseteq V$, the reliability $R_{G,p}(K)$ is the probability that all vertices in K are connected, which can be written as

$$R_{G,p}(K) = \sum_{X \in \Gamma_K} p(X), \tag{2}$$

where Γ_K is the set of states X of E such that all vertices in K are connected in G^X.

To be agnostic on infrastructure types, we introduce the neutral terms "sources" and "destinations" instead of servers and clients, respectively. Now the k-NR+ problem can be formally stated as follows: Given undirected graph $G = (V, E)$, working probability p_i for each edge $e_i \in E$, and the set of $k-1$ sources $T \subseteq V$, compute $R_{G,p}(T \cup \{v\})$ for each destination $v \in V \setminus T$. Note that this paper assumes all vertices except sources are considered as destinations, since this situation maximizes the computational load; our method works even when only a subset is destinations.

For simplicity, we impose some assumptions on G: G has no self-loops, and all vertices in G have a degree not less than 2. Note that these assumptions can be easily removed. First, the existence of self-loop edges does not affect the reliability measure. Second, the degree 1 vertices can be handled as follows: Let w be a degree 1 vertex in G and $e_i = \{u, w\}$ be the only edge incident to w. We can compute the reliability on G by that on $G - w$, the graph obtained by deleting w from G, assuming that $R_{G',p}(\{v\}) = 1$ for any vertex v in G'.

$$R_{G,p}(K) = \begin{cases} p_i \cdot R_{G-w,p}(K \cup \{u\} \setminus \{w\}) & (w \in K), \\ R_{G-w,p}(K) & (w \notin K). \end{cases} \tag{3}$$

By recursively removing degree 1 vertices, we obtain a graph with only degree ≥ 2 vertices, or a singleton graph.

III. Method

A. Review of Existing Method for k-NR

Since our method is based on the HH method [2], [3], we first explain it. A naive way to compute (2) is to enumerate all states of E in Γ_K. This can be performed by deciding whether each edge is working one by one in a predefined order $e_1, \ldots, e_m$. That is, from a state of $E_{<i} = \{e_1, \ldots, e_{i-1}\}$, two states of $E_{<i+1}$ are generated by specifying whether e_i is working. However, since there are 2^m states of E in total, such an exhaustive search is not suited even for graphs with dozens of edges. The HH method also searches the states by deciding the state of each edge one by one. However, to avoid redundant computations, it identifies some equivalent states of $E_{<i}$ with respect to the *partition* among the *boundary vertices* during the search. In addition, the HH method employs several pruning techniques to reduce the computation.

Given a predefined order of edges, the i-th boundary vertices F_i $(i = 1, \ldots, m+1)$ are the ones appearing in both $E_{<i}$ and $E_{\geq i} = E \setminus E_{<i}$. The partition is defined for a state of $E_{<i}$.

Definition 1 The *i-th level partition* of a state $X_{<i}$ of $E_{<i}$ is defined as follows: If there is a vertex in K that appears in $E_{<i}$ and is not connected to any vertex in F_i with the working edges in $E_{<i}$, the partition is $\perp$ (called *0-pruned partition*). Otherwise, the partition is defined as the pair of *blocks* that partition F_i and *marks* on blocks. Here two vertices $x, y \in F_i$ are in the same block iff they are connected with the working edges in $E_{<i}$, and a block is marked iff at least one of the vertices in K is connected to vertices in this block.

We can identify the states $X^1_{<i}, X^2_{<i}$ of $E_{<i}$ if their partitions are identical because, for every state $X_{\geq i}$ of remaining edges $E_{\geq i}$, $G^{X^1_{<i} \cup X_{\geq i}}$ and $G^{X^2_{<i} \cup X_{\geq i}}$ are equivalent in that the vertices in K are either all connected or not.

Based on this fact, the method can be seen as a search of partitions rather than states of edges. Fig. 2 is a diagram of such a search procedure performed on the graph shown inside the top rounded rectangle, given that $K = \{2, 4\}$ (black vertices). The partition is represented with bold text; here each block is represented by brackets ([ ]) with vertices inside them and each mark is indicated by an asterisk (*). First, we begin with an empty partition of $E_{<1} = \emptyset$. Next, given an i-th level partition P of $E_{<i}$, we examine two partitions P^{LO} and P^{HI} of $E_{<i+1}$ ($i+1$-th level partition) by excluding and including e_i, respectively, which means e_i is failing and working, respectively. In Fig. 2, we see that three states of $E_{<3}$

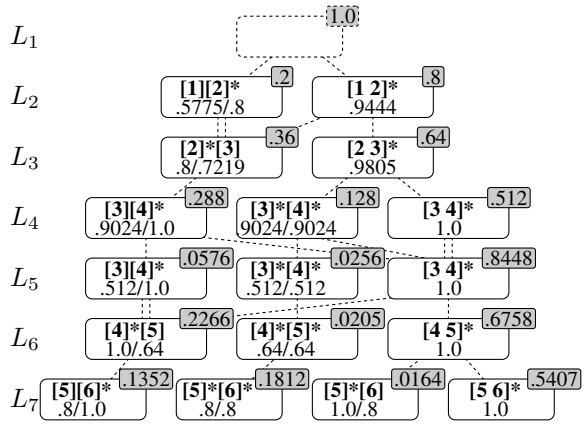

Fig. 3. Search diagram of proposed method with the graph presented in Fig. 2 given that $T = \{2, 4\}$ and the probability calculation along with this diagram when $p_i = 0.8$ for all edges. Solid and dashed lines indicate that the edge is working and failing, respectively. Here 0-pruned partitions are omitted.

are merged into the 3rd level partition [2]*[3], and the search is continued without distinguishing them; such "merging" leads to a significant reduction of computation. The search is terminated when all vertices in K are connected, which we call *1-pruning*, or it reaches 0-pruned partition, which we call *0-pruning*. The reliability $R_{G,p}(K)$ can be computed by a top-down dynamic programming (DP) along this search diagram.

B. Proposed Method

The proposed method also builds the search diagram among partitions by deciding the state of each edge one by one in a predefined order and then performs the DP calculation. However, it is different in the following points. First, in building the search diagram, the blocks are marked with respect to sources T instead of $K = T \cup \{v\}$, and 1-pruning is not performed. For example, the search diagram of our method is presented in Fig. 3, which is built upon the graph of Fig. 2 given $T = \{2, 4\}$. Second, as with the top-down DP, the bottom-up DP among blocks of partitions is performed.

The principle behind our method is to divide the process of computing $R_{G,p}(T \cup \{v\})$ into that involving T and that involving v. The former corresponds to the construction of a search diagram and the top-down DP. The latter corresponds to the bottom-up DP; by fully using the blocks of partitions, the computation for each v is translated into the one-pass DP. In this way, we can compute reliability for any number of destinations by scanning the search diagram only twice. In addition, since our method focuses on only T vertices instead of $T \cup \{v\}$, we can achieve better time complexity than the HH method even for k-NR when k is small. Since the procedure of constructing search diagram is the same as that in the HH method except for the points described above, we now describe how to compute reliability $R_{G,p}(T \cup \{v\})$ using the diagram.

1) Level-wise Reliability Computation on Search Diagram: Let us focus on the i-th level partitions L_i such that $v \in F_i$. Note that such i always exists for any $v \in V$ if all vertices in G have degree ≥ 2; e.g., $v \in F_{i'}$ such that $e_{i'-1}$ is the first edge containing v within the edge order. Let $p(P)$ be the probability such that the state of $E_{<i}$ leads to the i-th level partition P. Then we can rewrite $R_{G,p}(T \cup \{v\})$ as follows:

$$\begin{aligned} R_{G,p}(T \cup \{v\}) &= p(v \text{ is connected with all } v' \in T) \\ &= \textstyle\sum_{P \in L_i} p(P) p(v \text{ is connected to all } v' \in T \mid P), \end{aligned} \tag{4}$$

where the second term is the conditional probability given that the state of $E_{<i}$ leads to partition P.

With regard to the second term, we define the probability $q(P, b)$ that, given P, a block b of P is connected to all T vertices with respect to the state $X_{\geq i}$ of $E_{\geq i}$. Such probability can be consistently defined due to the following observation. Given states $X^1_{<i}, X^2_{<i}$ of $E_{<i}$ whose corresponding i-th level partitions are identical, for every state $X_{\geq i}$ of $E_{\geq i}$, $G^{X^1_{<i} \cup X_{\geq i}}$ and $G^{X^2_{<i} \cup X_{\geq i}}$ are equivalent in whether v is either connected to all T sources. By representing the block of P containing v as b^P_v, this can be verified by imaginarily marking block b^P_v of P and then applying the HH method [2], [3]. Moreover, based on the above argument, the probability of the second term is the same even if v is replaced with any other vertex in b^P_v.

Using this, (4) can be rewritten as

$$R_{G,p}(T \cup \{v\}) = \textstyle\sum_{P \in L_i} p(P) \cdot q(P, b^P_v) \quad (v \in F_i). \tag{5}$$

2) Top-Down and Bottom-Up Dynamic Programming: The remaining discussion is how to compute $p(P)$ and $q(P, b)$. The former can be computed via the following equation:

$$\begin{aligned} p(P) = (1 - p_i) \cdot \textstyle\sum_{Q \in L_i : Q^{\mathrm{LO}} = P} p(Q) \\ + p_i \cdot \textstyle\sum_{Q \in L_i : Q^{\mathrm{HI}} = P} p(Q) \quad (P \in L_{i+1}). \end{aligned} \tag{6}$$

By starting with $p(\emptyset) = 1$ for the 1st level empty partition, all $p(P)$ can be computed using (6) in a top-down manner.

As for the latter, we formally consider the correspondence between blocks of successive partitions.

Definition 2 Given $P \in L_i$ and $f \in \{\mathrm{LO}, \mathrm{HI}\}$, assume that $P^f \in L_{i+1}$ is not 0-pruned. For a block $b \in P$, let $b^f \in P^f$ be a block defined as follows: (1) If b contains a vertex v in F_{i+1}, $b^f = b^{P^f}_v$, i.e., the block of P^f containing v. (2) If no such vertex exists, $b^{\mathrm{LO}} = \emptyset$. For b^{HI}, let $e_i = \{u, w\}$. If $w \in F_{i+1}$ and $u \in b$, $b^{\mathrm{HI}} = b^{P^{\mathrm{HI}}}_w$. If $u \in F_{i+1}$ and $w \in b$, $b^{\mathrm{HI}} = b^{P^{\mathrm{HI}}}_u$. Otherwise, $b^{\mathrm{HI}} = \emptyset$.

For example, let us focus on the 5th level partition $P =$ [3][4]* in Fig. 3. For $b =$ [4]*, $b^{\mathrm{LO}} = b^{\mathrm{HI}} =$ [4]* of $P^{\mathrm{LO}} = P^{\mathrm{HI}} =$ [4]*[5]. For $b =$ [3], $b^{\mathrm{LO}} = \emptyset$ since vertex 3 is not in F_6. However, $b^{\mathrm{HI}} =$ [5] because $e_5 = \{3, 5\}$ and vertex 5 is in F_6. Now we observe that if e_i is failing, a block b of the i-th level partition is connected to all T vertices iff b^{LO} is connected to all T vertices. If e_i is working, b^{LO} of the above argument is replaced with b^{HI}. Thus, $q(P, b)$ can be written as

$$q(P, b) = (1 - p_i) \cdot q(P^{\mathrm{LO}}, b^{\mathrm{LO}}) + p_i \cdot q(P^{\mathrm{HI}}, b^{\mathrm{HI}}), \tag{7}$$

if both P^{LO} and P^{HI} are not 0-pruned; here, we let $q(\cdot, \emptyset) = 0$. This enables us to compute $q(P, b)$ in a bottom-up manner.

If P^f ($f \in \{\mathrm{LO}, \mathrm{HI}\}$) is 0-pruned, $q(P^f, b^f)$ in (7) is replaced with 0 except for the case when all T vertices are in

$E_{<i}$ and b is the only marked block in P. In such a case, all T vertices are already connected to b and thus $q(P^f, b^f)$ is replaced with 1. There are corner cases for computing $q(P, b)$ when P^{HI} is 0-pruned and neither endpoint of e_i is in F_{i+1}. Let $e_i = \{u, w\}$ and assume that all T vertices are in $E_{<i}$ and that $b_u^P \neq b_w^P$. First, if b_u^P is the only marked block, $q(P, b_u^P) = 1$ and $q(P, b_w^P) = p_i$. Second, if b_u^P and b_w^P are the only two marked blocks, then $q(P, b_u^P) = q(P, b_w^P) = p_i$.

Fig. 3 depicts $p(P)$ and $q(P, b)$ for all partitions and blocks when $p_i = 0.8$ for all edges. The shaded box in the upper-right corner indicates $p(P)$, and $q(P, b)$ is written inside the white box. By focusing on L_5, we can compute $R_{G,p}(T \cup \{3\})$ as $.288 \cdot .9024 + .128 \cdot .9024 + .512 \cdot 1.0 = .8874$. Other reliability values can be computed with different levels. Note that even if there are multiple levels whose boundary vertices contain v, we can choose any one of them to compute $R_{G,p}(T \cup \{v\})$ because the computed value will be identical. We also note that (5) holds even for $v \in T$, which yields the value of $R_{G,p}(T)$.

C. Pseudocodes

Algs. 1 and 2 are the pseudocodes for search diagram generation and successive DP computations, respectively. In them, the serif characters indicate an associative array, i.e., a map. Here, an i-th level partition P is represented as a pair of a map $\mathsf{c} : F_i \to \mathbb{Z}_+$ and an integer set $A \subseteq \mathbb{Z}_+$. The map c satisfies the condition that, for every $u, w \in F_i$, $\mathsf{c}[u] = \mathsf{c}[w]$ iff u and w are in the same block of P; c can be seen as the numbering of the blocks. A is defined as $\{\mathsf{c}[v] \mid b_v^P \text{ is marked}\}$, i.e., the set of IDs of marked blocks. For example, $P = $ [3][4]* is represented by $\mathsf{c} = \{3 \mapsto 1, 4 \mapsto 2\}$ and $A = \{2\}$.

In Alg. 1, for the i-th level partition P and $f \in \{\mathrm{LO}, \mathrm{HI}\}$, $\texttt{Generate}(\mathsf{c}, i, f)$ in line 5 returns $\mathsf{c}^{\mathrm{new}}$ representing the blocks of the next partition P^f and the map Cor^f representing the correspondence between blocks of P and P^f. Since a procedure like $\texttt{Generate}(\mathsf{c}, i, f)$ is also used in the HH method, we omit a detailed explanation for this. Based on the above, the 0-pruning is judged first (lines 6–7), and the set of new marked blocks A^{new} is computed (line 9). Finally, P^{new} is stored as $P^f = \mathsf{Next}^f[P]$ in the $i+1$-th level partitions.

In Alg. 2, for the i-th level partition $P = (\mathsf{c}, A)$, $p(P)$ is represented as $\mathsf{p}[(i, P)]$ and $q(P, b)$ as $\mathsf{q}[(i, P)][j]$, where $j = \mathsf{c}[v],\ v \in b$. Lines 1–8 compute $p(P)$ via top-down DP and lines 9–20 compute $q(P, b)$ via bottom-up DP. The reliability computation is performed in lines 21–26. Finally, $\mathsf{r}[v]$ stores the value $R_{G,p}(T \cup \{v\})$ for each destination v.

D. Complexity

We first consider the size of the search diagram and then analyze the complexity of our algorithm. Let $F_{\max} = \max_i |F_i|$ be the maximum number of boundary vertices.

Lemma 3 *The number of partitions in the overall search diagram is at most $O(m \cdot 2^{F_{\max}} \cdot B_{F_{\max}})$, where B_l is the l-th Bell number.*

Proof. At level i, there are at most $B_{|F_i|}$ patterns of the blocks and $2^{|F_i|}$ patterns of the marks. Thus, the number of i-th level partitions is at most $2^{|F_i|} \cdot B_{|F_i|}$. The overall size is bounded by $\sum_{i=1}^{m} 2^{|F_i|} \cdot B_{|F_i|} = O(m \cdot 2^{F_{\max}} \cdot B_{F_{\max}})$. □

Algorithm 1: Generating search diagram.

```
1  R ← (∅, ∅), L_1 ← {R}
2  for i ← 1 to m do                          // Build from top to bottom
3      foreach P = (c, A) ∈ L_i do
4          foreach f ∈ {LO, HI} do
5              (c^new, Cor^f[P]) ← Generate(c, i, f)
6              if ∃a ∈ A s.t. Cor^f[P][a] does not exist then
7                  Next^f[P] ← ⊥              // indicates 0-pruning
8              else
9                  A^new ← {Cor^f[P][a] | a ∈ A} ∪ {c^new[v] | v ∈
                     F_{i+1} ∩ T}             // New marked position
10                 P^new ← (c^new, A^new)     // New partition
11                 if P^new ∉ L_{i+1} then L_{i+1} ← L_{i+1} ∪ {P^new}
12                 Next^f[P] ← P^new          // P^f = Next^f[P]
13 procedure Generate(c, i, f):
14     c' ← c, Cor ← ∅
15     foreach v ∈ F_{i+1} \ F_i do
16         c'[v] ← max(c') + 1                // max(c) = max_{v∈F_i} c[v]
17     if f = HI then
18         tmp ← c'[w]                        // e_i = {u, w}
19         foreach v ∈ F_i ∪ F_{i+1} do
20             if c'[v] = tmp then c'[v] ← c'[u]
21     c^new ← c'
22     foreach v ∈ F_i \ F_{i+1} do Remove entry c^new[v]
23     (c^new, Renum) ← ReNumber(c^new, i)
24     foreach v ∈ F_i do Cor[c[v]] ← c'[v]
25     for j ← 1 to max(c) do
26         if Renum[j] exists then Cor[j] ← Renum[Cor[j]]
27         else Remove entry Cor[j]
28     return (c^new, Cor)
29 procedure ReNumber(c, i):
30     cnt ← 0, c' ← ∅, Renum ← ∅
31     foreach v ∈ F_i in predefined order do
32         if Renum[c[v]] does not exist then
33             cnt += 1, Renum[c[v]] ← cnt
34         c'[v] ← Renum[c[v]]
35     return (c', Renum)
```

Theorem 4 *Given undirected graph $G = (V, E)$, working probabilities p and sources $T \subseteq V$, Algs. 1 and 2 can solve k-NR+ in $O(m \cdot F_{\max} \cdot 2^{F_{\max}} \cdot B_{F_{\max}})$ time.*

Proof. This follows from the fact that for each i-th level partition P, the processing of P in Alg. 1 is performed in $O(|F_i|)$ time, and the DP computations of $\mathsf{p}[(i, P)]$ and $\mathsf{q}[(i, P)][\cdot]$ are also in $O(|F_i|)$ time. The former holds because all operations in Alg. 1 can be performed in $O(|F_i|)$ time for P. The latter can be verified because there are at most $O(|F_i|)$ values ($\mathsf{p}[(i, P)]$ and $\mathsf{q}[(i, P)][\cdot]$) to be computed and they are each computed in constant time via (6) and (7). □

The complexity of Theorem 4 is the same as that of the HH method for solving k-NR. Regarding the k-NR+ problem, the proposed method is $O(n)$ times faster than the HH method. Moreover, when $k = |T| + 1$ is assumed to be a constant, a better time complexity can be achieved.

Theorem 5 *If $k = |T| + 1$ is assumed to be a constant, Algs. 1 and 2 can solve k-NR+ in $O(m \cdot F_{\max}^k \cdot B_{F_{\max}})$ time.*

Algorithm 2: DP and level-wise reliability calculation.

```
1  p[(1, R)] ← 1                                   // Start with 1st level
2  for i ← 2 to m do
3  |  foreach P ∈ L_i do p[(i, P)] ← 0                          // Init p
4  for i ← 1 to m do                                          // Top-down DP
5  |  foreach P ∈ L_i do
6  |  |  foreach f ∈ {LO, HI} do
7  |  |  |  if Next^f[P] ≠ ⊥ then
8  |  |  |  |  p[(i+1, Next^f[P])] += p_i^f · p[(i, P)]              // (6)
   |  |  |  |                  // here p_i^LO = 1 − p_i and p_i^HI = p_i
9  for i ← m to 2 do                                         // Bottom-up DP
10 |  foreach P = (c, A) ∈ L_i do
11 |  |  for j ← 1 to max(c) do q[(i, P)][j] ← 0                 // Init q
12 |  |  foreach f ∈ {LO, HI} do
13 |  |  |  if Next^f[P] = ⊥ then                          // P^f is 0-pruned
14 |  |  |  |  if (⋃_{k<i} e_k ⊇ T) ∧ (|A| = 1) then
15 |  |  |  |  |  q[(i, P)][j*] += p_i^f for j* s.t. A = {j*}
16 |  |  |  |  if (⋃_{k<i} e_k ⊇ T) ∧ (f = HI) ∧ (|F_i \ F_{i+1}| = 2)
   |  |  |  |  |  then ProcessCornerCase(i, P)
17 |  |  |  else                                          // P^f is not 0-pruned
18 |  |  |  |  for j ← 1 to max(c) do
19 |  |  |  |  |  if Cor^f[P][j] exists then
20 |  |  |  |  |  |  q[(i, P)][j] +=
   |  |  |  |  |  |    p_i^f · q[(i+1, Next^f[P])][Cor^f[P][j]]   // (7)
21 Choose I ⊆ {1, ..., m} s.t. ⋃_{i∈I} F_i ⊇ V
22 foreach i ∈ I do                                  // Level-wise Rel. Computation
23 |  foreach v ∈ F_i do
24 |  |  r[v] ← 0
25 |  |  foreach P = (c, A) ∈ L_i do
26 |  |  |  r[v] += p[(i, P)] · q[(i, P)][c[v]]                    // (5)
27 procedure ProcessCornerCase(i, P = (c, A)):
28 |  if c[u] ≠ c[w] then                                  // e_i = {u, w}
29 |  |  if (A = {c[u]}) ∨ (A = {c[u], c[w]}) then
30 |  |  |  q[(i, P)][c[w]] += p_i^HI
31 |  |  if (A = {c[w]}) ∨ (A = {c[u], c[w]}) then
32 |  |  |  q[(i, P)][c[u]] += p_i^HI
```

Proof. For this scenario, we can say that the size of the search diagram is at most $O(m \cdot F_{\max}^{k-1} \cdot B_{F_{\max}})$. This is because, at level i, the number of patterns of the marks is at most $\sum_{t=0}^{\min\{k-1,|F_i|\}} \binom{|F_i|}{t} = O(F_{\max}^{k-1})$. Following the proof of Theorem 4, the overall complexity is $O(m \cdot F_{\max}^{k} \cdot B_{F_{\max}})$. □

Theorem 5 includes the computation of two-terminal reliabilities for multiple destinations as a special case ($k = 1$). If k is constant, the HH method [2], [3] could solve k-NR in $O(m \cdot F_{\max}^{k+1} \cdot B_{F_{\max}})$ time. This means that, when k is small, although our method can compute reliabilities for $n - k + 1$ destinations, the time complexity is $O(F_{\max})$ times faster than the single reliability computation with the HH method. This difference comes from the fact that our method marks the blocks with T, while the HH method marks them with $T \cup \{v\}$.

With regard to the value of $F_{\max}$, it is known that we can construct edge ordering with $F_{\max} =$ (pathwidth of G) from the path decomposition of G [8]. Therefore, for graphs with a constant pathwidth, our method runs in $O(m)$ time.

IV. Experiments

We compared our method and the HH method [2], [3] with respect to the consumed time for solving k-NR+. We also compared the performance for solving k-NR to examine the effect of complexity results in Section III-D. Our method is implemented in C++11. For HH method, we used TdZdd (available at https://github.com/kunisura/TdZdd), which implements the improved variant [3]. All codes were compiled by g++-9.1.0 with `-O3 -DNDEBUG` options. Experiments were conducted on a single thread of a Linux machine with Intel Xeon E7-8880 2.20 GHz CPU and 3072 GB RAM; note that, as described later, both methods used less than 65 GB of memory, except for one instance. Our implementation and all data used in this section are available at https://github.com/nttcslab-alg/cs-rel.

We used both synthetic and real graphs as tested instances. The synthetic graphs are the grid graphs; Grid-wxh denotes a grid graph with $w \times h$ vertices. For these, we used the edge ordering of Iwashita et al. [9], which is known to be better for DP on grid graphs. The real graphs are from the Internet Topology Zoo [10] and Rocketfuel [11] datasets. For these, we extracted the largest connected component, removed self-loops, and recursively removed the degree 1 vertices. We decided the edge ordering according to the beam-search based method [8]. For each graph, we computed the betweenness centrality [12] of each vertex and chose $|T| = 1, 5, 10$ vertices with higher centrality as sources T of the k-NR+ problem, because sources should be deployed in an easily accessible "center" of the network. For the k-NR problem, we chose $k = 2, 6, 11$ vertices with higher betweenness centrality as terminals K. Since the original data [10], [11] do not include the working probability p_i of each e_i, it was chosen uniformly at random from $[0.9, 0.95]$ according to the literature [6], [7], [13]–[15]. We set the time limit of each run to 1 hour.

Table I shows the results. The results demonstrate that our method can solve k-NR+ 30–400 times faster than the existing method. Notably, our method successfully computed all reliability measures within 1 hour even for larger graphs, e.g., those having more than 200 edges and $F_{\max} = 12, 13$. Although the k-NR problem on Rocketfuel graphs with more than 200 edges has not yet been solved to our knowledge, our method can solve more difficult k-NR+ problems in a reasonable time. Regarding the results of Grid-7xh, the computation time is roughly $O(h)$ for our method and $O(h^2)$ for TdZdd. This is because the complexity for a graph with constant $F_{\max}$ is $O(m)$ for our method and $O(mn)$ for the HH method. As for the memory usage, in solving a Grid-12x12 ($|T| = 10$) instance, our method used 341.2 GB while TdZdd used 177.3 GB when the computation expired. The second largest peak memory usage was recorded for Rocketfuel-3257, in which our method used 64.2 GB and TdZdd used 61.3 GB.

As for the k-NR problem, our method is faster than TdZdd for all instances with $k = 2$ and graphs having large $F_{\max}$ with $k = 6, 11$. This reflects the complexity result of Theorem 5: When k is small, our method is $O(F_{\max})$ times faster than the HH method. On the other hand, since the HH method has pruning techniques that are not used in our methods, namely 1-pruning when all terminals are connected and 0-pruning when v is disconnected, the HH method is sometimes faster in solving k-NR when k is large or $F_{\max}$ is small.

TABLE I
CONSUMED TIME FOR SOLVING k-NR+ AND k-NR: PROPOSED METHOD VS. TDZDD, A MODERN IMPLEMENTATION OF THE HH METHOD.

				k-NR+ (sec.)						k-NR (sec.)					
				$k=2(\lvert T\rvert=1)$		$k=6(\lvert T\rvert=5)$		$k=11(\lvert T\rvert=10)$		$k=2$		$k=6$		$k=11$	
Instance	n	m	$F_{\max}$	Ours	TdZdd	Ours	TdZdd	Ours	TdZdd	Ours	TdZdd	Ours	TdZdd	Ours	TdZdd
Grid-7x14	98	175	7	**0.17**	17.69	**0.57**	22.42	**0.45**	18.95	**0.19**	0.33	0.51	**0.35**	0.39	**0.30**
Grid-7x28	196	357	7	**0.31**	90.84	**1.04**	124.18	**1.21**	114.14	**0.29**	0.49	0.95	**0.64**	1.03	**0.59**
Grid-7x42	294	539	7	**0.53**	221.00	**1.81**	324.94	**1.78**	298.82	**0.47**	0.79	1.66	**1.04**	1.59	**1.01**
Grid-8x8	64	112	8	**0.24**	22.89	**0.71**	24.23	**0.95**	19.71	**0.21**	0.44	0.65	**0.49**	0.80	**0.35**
Grid-10x10	100	180	10	**10.78**	>1h	**48.76**	>1h	**63.96**	>1h	**9.18**	42.56	**44.39**	56.23	57.97	**54.37**
Grid-12x12	144	264	12	**320.74**	>1h	**2309.55**	>1h	**3465.40**	>1h	**284.11**	1535.82	**2105.08**	>1h	**3139.19**	>1h
Interroute	102	138	7	**0.02**	2.12	**0.04**	2.13	**0.05**	1.88	**0.02**	0.05	**0.04**	**0.04**	0.04	**0.03**
TATA	136	177	6	**0.01**	0.99	**0.02**	1.11	**0.01**	0.51	**0.01**	0.02	0.02	**0.01**	**0.01**	**0.01**
Kentucky Datalink	680	821	12	**6.55**	>1h	**24.48**	>1h	**26.82**	>1h	**5.57**	164.55	**19.38**	248.36	**23.95**	247.99
Rocketfuel-1221	178	618	12	**15.38**	>1h	**12.52**	2032.51	**12.43**	2619.35	**12.55**	30.88	**10.83**	13.32	**11.06**	12.20
Rocketfuel-1755	146	355	12	**22.18**	>1h	**105.30**	>1h	**48.65**	>1h	**19.80**	116.32	87.36	**70.23**	44.85	**42.71**
Rocketfuel-3257	139	303	12	**69.08**	>1h	**400.97**	>1h	**611.06**	>1h	**56.70**	108.63	**360.49**	784.34	496.69	**224.91**
Rocketfuel-3967	166	399	13	**128.33**	>1h	**277.61**	>1h	**213.04**	>1h	**103.58**	346.81	**223.50**	399.75	**177.51**	242.20
Rocketfuel-6461	100	212	10	**6.21**	1565.17	**9.51**	733.77	**3.89**	182.64	**5.59**	19.07	7.56	**3.81**	3.33	**2.01**

V. RELATED WORK

Recalling two state-of-the-art k-NR methods, one [2] proposed a basic method, while the other [3] improved the equivalence decision of states, but their computational complexity is the same. Several reliability problems extending k-NR have been studied: e.g., reliability optimization [6], reliability for any k terminals of n vertices [16], reliability under routing constraints [17], and reliability under disjoint-paths constraints [7]. Despite that rich body of literature, no work has studied k-NR+.

To reduce the computation costs, approximate solutions like a Monte Carlo simulation have been studied [18], but other works [6], [7] showed that the deviation of the Monte Carlo approach is often more than double. One work [19] proposed F-Monte Carlo, which estimates the probability of rare events accurately, but depends on an unrealistic assumption, i.e., that all links are likely to fail with equal probability. Another study [20] approximately enumerated destinations whose reliability for any source exceeds a given threshold. The problem seems similar to ours, but reliability values are not given, so we cannot quantitatively assess the risk of each destination.

VI. CONCLUSION

The work in this paper made two contributions: (1) An efficient method was proposed for k-NR+, with a time complexity that is the same as the existing method for k-NR, that is, our method can solve k-NR+ $O(n)$ times faster than the existing method; and (2) We revealed the method's practical efficiency with real networks that have hundreds of edges. In future work, we will elaborate our method for specific infrastructures.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number JP20H05963.

REFERENCES

[1] F. T. Boesch, A. Satyanarayana, and C. L. Suffel, "A survey of some network reliability analysis and synthesis results," *Networks*, vol. 54, no. 2, pp. 99–107, 2009.

[2] G. Hardy, C. Lucet, and N. Limnios, "K-terminal network reliability measures with binary decision diagrams," *IEEE Trans. Rel.*, vol. 56, no. 3, pp. 506–515, 2007.

[3] J. Herrmann, "Improving reliability calculation with augmented binary decision diagrams," in *AINA*, 2010, pp. 328–333.

[4] J. Kawahara, K. Sonoda, T. Inoue, and S. Kasahara, "Efficient construction of binary decision diagrams for network reliability with imperfect vertices," *Reliab. Eng. Syst. Safety*, vol. 188, pp. 142–154, 2019.

[5] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM J. Comput.*, vol. 8, no. 3, pp. 410–421, 1979.

[6] M. Nishino, T. Inoue, N. Yasuda, S. Minato, and M. Nagata, "Optimizing network reliability via best-first search over decision diagrams," in *INFOCOM*, 2018, pp. 1817–1825.

[7] T. Inoue, "Reliability analysis for disjoint paths," *IEEE Trans. Rel.*, vol. 68, no. 3, pp. 985–998, 2019.

[8] Y. Inoue and S. Minato, "Acceleration of ZDD construction for subgraph enumeration via pathwidth optimization," Division of Computer Science, Hokkaido University, Tech. Rep. TCS-TR-A-16-80, 2016.

[9] H. Iwashita, Y. Nakazawa, J. Kawahara, T. Uno, and S. Minato, "Efficient computation of the number of paths in a grid graph with minimal perfect hash functions," Division of Computer Science, Hokkaido University, Tech. Rep. TCS-TR-A-13-64, 2013.

[10] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, pp. 1765–1775, 2011.

[11] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring isp topologies with rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, 2004.

[12] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, pp. 35–41, 1977.

[13] B. Elshqeirat, S. Soh, S. Rai, and M. Lazarescu, "Topology design with minimal cost subject to network reliability constraint," *IEEE Trans. Rel.*, vol. 64, no. 1, pp. 118–131, 2015.

[14] Z. Botev, P. L'Ecuyer, and B. Tuffin, "Dependent failures in highly reliable static networks," in *Simulation Conference*, 2012, pp. 1–12.

[15] Y. Xiao, X. Li, Y. Li, and S. Chen, "Evaluate reliability of wireless sensor networks with OBDD," in *ICC*, 2009, pp. 1–5.

[16] R. Li, N. Huang, and R. Kang, "A new parameter and its algorithm for network connection reliability: k/n-terminal reliability," in *ICFIN*, Oct 2009, pp. 259–262.

[17] M. Hayashi and T. Abe, "Evaluating reliability of telecommunications networks using traffic path information," *IEEE Trans. Rel.*, vol. 57, no. 2, pp. 283–294, June 2008.

[18] I. B. Gertsbakh and Y. Shpungin, *Models of network reliability: analysis, combinatorics, and Monte Carlo*. CRC press, 2009.

[19] E. Canale, F. Robledo, P. Romero, and P. Sartor, "Monte Carlo methods in diameter-constrained reliability," *Opt. Switch. Netw.*, vol. 14, pp. 134–148, 2014.

[20] A. Khan, F. Bonchi, A. Gionis, and F. Gullo, "Fast reliability search in uncertain graphs." in *EDBT*, 2014, pp. 535–546.