Reliability Analysis of Replicated And–Or Graphs

De-Ron Liang,[1] Rong-Hong Jan,[2] Satish K. Tripathi[3]

[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan, Republic of China

[2] Department of Information and Computer Science, National Chiao Tung University Hsinchu 30050, Taiwan, Republic of China

[3] Department of Computer Science, University of Maryland, College Park, Maryland 20742

Received 9 March 1994; accepted 6 July 1995

Abstract: A computation task running in distributed systems can be represented as a directed graph $H(V, E)$ whose vertices and edges may fail with known probabilities. In this paper, we introduce a reliability measure, called the distributed task reliability, to model the reliability of such computation tasks. The distributed task reliability is defined as the probability that the task can be successfully executed. Due to the *and-fork*/*and-join* constraint, the traditional network reliability problem is a special case of the distributed task reliability problem, where the former is known to be *NP*-hard in general graphs. For two-terminal *and–or series-parallel* (AOSP) graphs, the distributed task reliability can be computed in polynomial time. We consider a graph $H^k(\hat{V}, \hat{E})$, named a *k-replicated and–or series-parallel* (RAOSP) graph, which is obtained from an AOSP graph $H(V, E)$ by adding $(k - 1)$ replications to each vertex and adding proper edges between two vertices. It can be shown that the RAOSP graphs are not AOSP graphs; thus, the existing polynomial algorithm does not apply. Previously, only exponential time algorithms as used in general graphs are known for computing the reliability of $H^k(\hat{V}, \hat{E})$. In this paper, we present a linear time algorithm with $O(K(|V| + |E|))$ complexity to evaluate the reliability of the graph $H^k(\hat{V}, \hat{E})$, where $K = \max\{k^2 2^{2k}, 2^{3k}\}$. Networks **29:** 195–203, 1997

1. INTRODUCTION

In the past decade, distributed processing systems have become increasingly popular because they provide a potential increase in reliability, throughput, fault tolerance, and resource utilization. Usually, the computation task of a distributed processing system can be partitioned into a set of software modules (or simply, modules) and then modeled as a directed graph $H(V, E)$, called a *task graph.* In the task graph $H(V, E)$, V is the set of vertices which represents modules and E is the set of edges which represents the messages passing links between two modules.

To increase the survival rate of the task, a straightforward method is to replicate the complete task several times and to execute it independently on distinct computers. The primary site approach [2] is one such example. The disadvantage of this approach is that the system cannot tolerate more than one fault in each replicated task. Recently, the replication of software modules was proposed and implemented, such as in *Maruti* [8] and *Delta-4* [10]. The idea behind this approach can be illustrated

Correspondence to: D.-R. Liang

CCC 0028-3045/97/040195-09

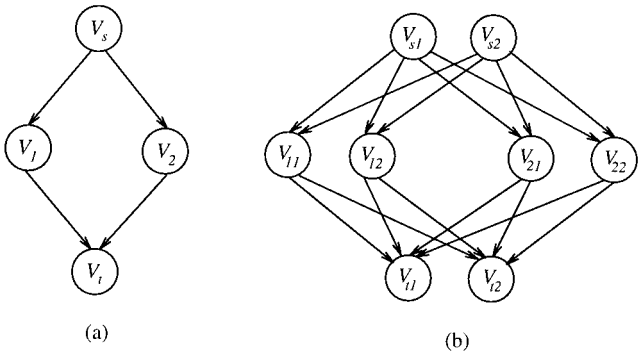

Fig. 1. (a) A fork-join task graph; (b) its replicated graph.

in the following example: Consider a simple application modeled by an and-fork/and-join graph as shown in Figure 1(a). (By convention, this task operates only if all the modules as well as the links operate.) Suppose that the application is implemented with an extra replication. In this approach, each module receives messages not only from its predecessors in the same replica, but also from the corresponding predecessors in the other replica. Figure 1(b) shows one such implementation. Thus, a task finishes successfully only if there is a set of modules which forms this application, and their associated communication links are operational. Obviously, this application may tolerate more than one fault in each task replication, depending on the fault patterns. Figure 2 shows a few examples where the task in Figure 1(b) is operational.

The modules and the communication links may fail due to two main factors: *software failure* and *hardware failure*. The software failures are mainly due to the design faults or implementation faults. The reliability can be increased when the software components are replicated with the N-version programming approach [5]. The hardware failures are due to either *transient failures* or *permanent failures* [10]. It has been reported that most of the hardware failures in computer systems are transient failures [8]. The random events of failures in modules or communication links can be considered as independent, provided that the software components are replicated with the N-version programming technique and the hardware failures are assumed to be transient.

Suppose that the modules and the communication links have a certain probability of being operational. Then, there is a certain probability, called the *distributed task reliability*, associated with the event that a task completes successfully. This measure accurately models the reliability of a task running in distributed systems. Due to the *and-fork/and-join* constraint of the task graph, the traditional network reliability problem [1, 3, 4, 6, 7, 11, 12] is a special case of the distributed task reliability problem, where the former is known to be *NP*-hard for general graphs. For the two-terminal series-parallel (TTSP) graphs, the distributed task reliability can be found in polynomial time using the technique developed in [13]. For the two-terminal and–or series-parallel (AOSP) graphs, we will show in Section 2 that their distributed task reliability can also be calculated in polynomial time using the same technique [13]. In this paper, we consider a graph $H^k(\hat{V}, \hat{E})$, named the *k-replicated and–or series-parallel* graph (*k*-replicated AOSP graph or, more, simply RAOSP graph), which is obtained from an AOSP graph $H(V, E)$ by adding $(k - 1)$ replications to each vertex and adding proper links between vertices. The main contribution of this paper is the design of a linear time algorithm to calculate the reliability of the RAOSP graph $H^k(\hat{V}, \hat{E})$, given the base AOSP graph $H(V, E)$ and the replication degree k. It can be shown that the RAOSP graphs are not AOSP graphs; thus, the existing polynomial algorithm does not apply. Previously, only exponential time algorithms, as used in general graphs, are known

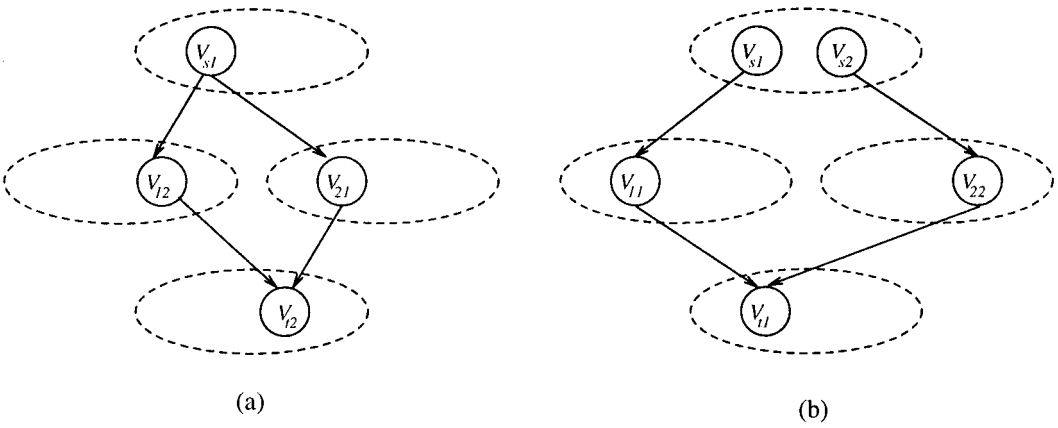

Fig. 2. Examples of an operational replicated task graph.

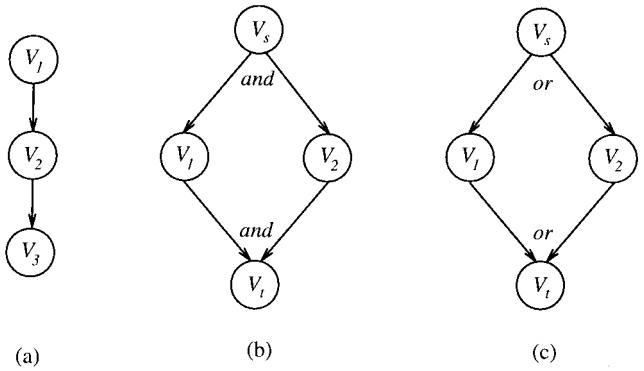

Fig. 3. (a) Sequential graph; (b) and-fork/and-join graph; (c) or-fork/or-join graph.

to find the reliability of $H^k(\hat{V}, \hat{E})$. In this paper, we present a linear time algorithm with $O(K(|V| + |E|))$ complexity to compute the reliability of the graph $H^k(\hat{V}, \hat{E})$, where $K = \max\{k^2 2^{2k}, 2^{3k}\}$.

The rest of the paper is organized as follows: In the next section, the definition of AOSP graphs is given and the reliability evaluation for AOSP graphs is discussed. In Section 3, we define the *k*-replicated AOSP graph and its reliability. An algorithm for computing the reliability of a *k*-replicated AOSP graph $H^k(\hat{V}, \hat{E})$ is developed in Section 4. The algorithm is shown to have an $O(K(|V| + |E|))$ time complexity, where $K = \max\{k^2 2^{2k}, 2^{3k}\}$. A numerical example is given in Section 5. Section 6 presents an extension to the model and considers the cases of partial replication as well as unreliable vertices. Finally, concluding remarks are presented.

2. RELIABILITY EVALUATION OF AOSP GRAPHS

Consider a computation task graph $H(V, E)$ consisting of a set of software modules V and a set of communication links E, which represents message passings between software modules. According to the logical structures and precedence relationship among the modules, a large class of task graphs can be expressed by a combination of three common types of subgraphs [5, 9]: sequential, and-fork to and-join (AFAJ), and or-fork to or-join (OFOJ) [see Fig. 3(a)–(c)], where AFAJ and OFOJ subgraphs may consist of several sequential subgraphs in a parallel structure. In this paper, we restrict our task graphs to contain a combination of these three types of subgraphs. This type of the graph can be modeled as a *two-terminal AOSP* graph. To state the model, we make the following assumptions:

1. All modules in the task graph $H(V, E)$ are perfectly reliable.
2. Any communication link may fail with a known probability.
3. All failures are assumed to occur independently of each other.

We remind the reader that we will extend this model to consider the case of unreliable modules in Section 6.

Formally, a *two-terminal AOSP graph* of *type k*, where $k \in \{L, P_A, P_O, S\}$ (which means *leaf, parallel-and, parallel-or,* and *series,* respectively) is recursively defined as follows:

1. A single edge (s, t) comprises an AOSP graph of type L with terminals s and t. The system operates if that edge operates.
   Let H_i be an AOSP graph with terminals s_i and t_i for $i = 1, 2$.
2. The graph $H_1 \wedge H_2$ is an AOSP graph of type P_A with terminals s and t, where the graph associated with $H_1 \wedge H_2$ is the disjoint union of H_1 and H_2, with s_1 identified with s_2 and t_1 identified with t_2, and the system operates if *both* H_1 and H_2 operate.
3. The graph $H_1 \vee H_2$ is an AOSP graph of type P_O with terminals s and t, where the graph associated with $H_1 \vee H_2$ is the same as that associated with $H_1 \wedge H_2$, except that the system operates if *either* H_1 or H_2 operates.
4. The graph $H_1 * H_2$ is an AOSP graph of type S with terminals s_1 and t_2, where the graph associated with $H_1 * H_2$ is the disjoint union of H_1 and H_2 with t_1 identified with s_2, and the system operates if *both* H_1 and H_2 operate.

We note that the TTSP graphs [13] can be formulated recursively using only the operations 1, 3, and 4 of the AOSP graphs defined previously. In other words, the class of the TTSP graphs is a subclass of the AOSP graphs.

The distributed task reliability of task H, denoted by $R(H)$, is defined as the probability that the task H operates. For example, if H contains only one edge e of type L, $R(H) = r(e)$, where $r(e)$ is the reliability of edge e. If H consists of two AOSP graphs H_1 and H_2, then

$$R(H) = \begin{cases} R(H_1)R(H_2) & \text{if } H = H_1 \wedge H_2 \\ & \text{or } H = H_1 * H_2 \\ 1 - (1 - R(H_1))(1 - R(H_2)) & \text{if } H = H_1 \vee H_2. \end{cases} \quad (1)$$

To compute $R(H)$, we first describe an AOSP graph H by a binary tree structure, $T(H)$, called the *parsing*

tree of H. For example, Figure 4(b) depicts a parsing tree of the AOSP graph H in Figure 4(a). The nodes in the parsing tree are numbered (at the upper right corner) according to their postorder sequence. Each leaf node in $T(H)$ corresponds to an AOSP subgraph of type L, i.e., a single-edge AOSP graph in H. In Figure 4, for example, $H_1 = e_1$, $H_2 = e_2$, $H_4 = e_3$, and $H_5 = e_4$. Each internal node is labeled by S, P_O, or P_A according to the type of that AOSP subgraph. An internal node numbered x in $T(H)$ along with all of its descendant nodes induce a subtree T_x which also describes an AOSP subgraph H_x in H. For example, subtree T_3 describes the AOSP graph $H_3 = H_1 * H_2$, in Figure 4(b). Given the parsing tree $T(H)$, we can obtain $R(H)$ using Eq. (1) to compute the $R(H_x)$ level by level for every node x in the parsing tree $T(H)$. For example, we first find $R(H_1)$ and $R(H_2)$ in Figure 4(b), where $H_1 = e_1$ and $H_2 = e_2$. Next, we consider subtrees T_3 and compute $R(H_3) = R(H_1)R(H_2)$. Finally, we can determine $R(H) = R(H_7) = R(H_3 \wedge H_6) = R(H_3)R(H_6)$.

Note that an AOSP graph $H(V, E)$ can be translated into its parsing tree in $O(|E|)$ time using the algorithm proposed by Valdes et al. [13]. In other words, $R(H)$ can be found in $O(|E|)$ time.

3. TASKS WITH REPLICATION

To increase the reliability of a task, we replicate the modules and the message passing links of the task. The *k-replicated task graph* $H^k(\hat{V}, \hat{E})$ of $H(V, E)$ is created by replicating each vertex in $V(k-1)$ times and letting the edges in H^k be established in such a way that each vertex is not only descendant of its predecessors in the same replica, but is also descendant to the corresponding predecessors in the other replicas. For example, Figure 5(b) shows a k-replicated task graph created from an AOSP graph of type L in Figure 5(a).

Imagine that each edge in H represents a thread of execution; then, a replicated edge in H^k represents a replica thread of the corresponding thread in H [such as (s_i, t_j), $1 \le i, j \le k$, are a replica of (s, t) in Fig. 5(b).] For clarity, we introduce the concept of ''*correct value at terminal.*'' Without loss of generality, the source terminal of an edge is either offered a valid input value or a nil value. Suppose that a valid input value is offered at the source terminal s of an AOSP graph H; then, the sink terminal t of H is said to have a ''*correct*'' value *with respect to the valid input value* if H operates. Furthermore, the sink terminal t has a *nil* value if either the source terminal is offered as a valid input value but H fails to operate or the source is offered a nil value regardless of the operation of H.*

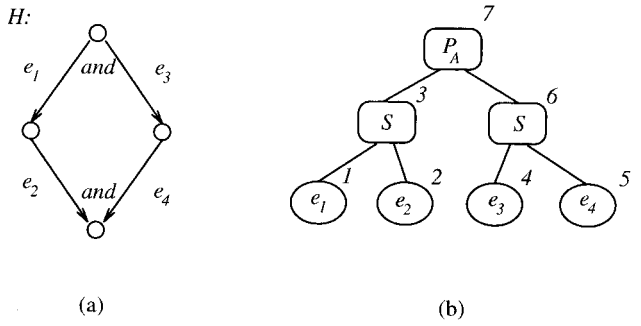

Fig. 4. (a) A fork-join AOSP graph; (b) the corresponding parsing tree.

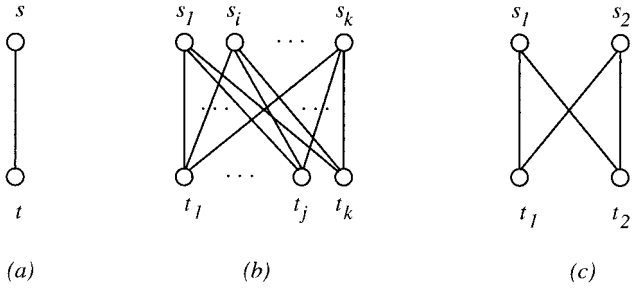

Fig. 5. (a) A type L AOSP graph; (b) the corresponding k-replicated RAOSP; (c) the 2-replicated RAOSP.

Consider a k-replicated AOSP graph H^k. We first notice that it has k source terminals and k sink terminals. Without loss of generality, we assume that those source terminals of H^k which are offered valid input have the same input value. It is obvious that whether a sink terminal of H^k has a correct output depends not only on which source terminals of H^k are offered valid input values but also on the execution of H^k. Let S and T be the sets of source and sink terminals of H^k, respectively. Let A be the set of source terminals offered with the same valid input value and B be the set of sink terminals with correct outputs; $A \subseteq S$, $B \subseteq T$. H^k is said to *operate w.r.t.* (A, B) if $\forall t_j \in B$, t_j has a correct value and $\forall t_j' \in T \backslash B$, t_j' has nil value, given that $\forall s_i \in A$, s_i are offered with a valid input value and other source terminals are offered a nil value. Now, we are ready to formally define the k-replicated AOSP graph:

1. (L): A k-replicated AOSP structure of type L consists of a set of k source terminals, $S = \{s_1, \ldots, s_k\}$, a set of k sink terminals, $T = \{t_1, \ldots, t_k\}$, and k^2 edges, (s_i, t_j), $\forall i, j$. Let $A \subseteq S$, $B \subseteq T$. The system operates w.r.t. (A, B) iff $\forall t_j \in B$, $\exists s_i \in A$, such that (s_i, t_j)

* This is known as *fail-stop* model in fault-tolerant computing.

operates and $\forall t'_j \in T\backslash B$, $\exists s_i \in A$ such that (s_i, t'_j) operates.

Let H_i^k be a k-replicated AOSP graph with terminals set S_i and T_i, and let $A_i \subseteq S_i$, $B_i \subseteq T_i$, for $i = 1, 2$.

2. (P_A): The graph $H_1^k \wedge H_2^k$ is a k-replicated AOSP graph of type P_A with terminals set S and T, where the graph associated with $H_1^k \wedge H_2^k$ is the disjoint union of H_1^k and H_2^k, with S_1 identified with S_2 (i.e., $s_{1i} \in S_1$ identified with $s_{2i} \in S_2$, $1 \le i \le k$) and T_1 identified with T_2. Furthermore, $S = S_1 = S_2$ and $T = T_1 = T_2$. Suppose that $A \subseteq S$ and $B \subseteq T$. The system operates w.r.t. (A, B) iff H_1^k operates w.r.t. (A, B_1), H_2^k operates w.r.t. (A, B_2), and $B_1 \cap B_2 = B$.
3. (P_O): The graph $H_1^k \vee H_2^k$ is a k-replicated AOSP graph of type P_O with terminals set S and T, where the graph associated with $H_1^k \vee H_2^k$ is the disjoint union of H_1^k and H_2^k, with S_1 identified with S_2 and T_1 identified with T_2. Let $S = S_1 = S_2$, and $T = T_1 = T_2$. Suppose that $A \subseteq S$ and $B \subseteq T$. The system operates w.r.t. (A, B) iff H_1^k operates w.r.t. (A, B_1), H_2^k operates w.r.t. (A, B_2), and $B_1 \cup B_2 = B$.
4. (S): The graph $H_1^k * H_2^k$ is a k-replicated AOSP graph of type S with terminals sets $S = S_1$ and $T = T_2$, where the graph associated with $H_1^k * H_2^k$ is the disjoint union of H_1^k and H_2^k with T_1 identified with S_2. Suppose that $A \subseteq S$ and $B \subseteq T$. The system operates w.r.t. (A, B) iff H_1^k operates w.r.t. (A, B_1), H_2^k operates w.r.t. (A_2, B), and $B_1 = A_2$.

Let $E_{AB}(H^k)$ be a probability event that H^k operates w.r.t. terminal sets (A, B). We denote $p_{AB}(H^k) = Pr\{E_{AB}(H^k)\}$. We notice that for any set $A \subseteq S$, $\Sigma_{B\subseteq T} p_{AB}(H^k) = 1$. Given that all the source terminals are offered with the valid input value initially, i.e., $A = S$, the reliability of H^k is the same as the probability that at least one sink terminal has a correct value. Thus, the reliability of H^k is defined as

$$\begin{aligned} R(H^k) &= Pr\{\bigcup_{B\subseteq T, B\neq\emptyset} E_{SB}(H^k)\}, \\ &= \sum_{\emptyset\neq B\subseteq T} p_{SB}(H^k). \end{aligned} \tag{2}$$

A straightforward way to compute $R(H^k)$ is to enumerate the execution outcomes of all the edges in $H^k(\hat{E}, \hat{V})$, where $|\hat{E}| = k^2 \cdot |E|$ and set E is an edge set in H. However, this method takes $O(2^{k^2|E|})$. In next section, a linear time algorithm for computing $R(H^k)$ will be presented.

4. RELIABILITY EVALUATION OF RAOSP GRAPHS

In this section, we present an algorithm to compute the reliability of a k-replicated task $H^k(\hat{V}, \hat{E})$ in $O(K|E|)$ time. Note that $K = \max\{k^2 2^{2k}, 2^{3k}\}$. We first consider the k-replicated task graph of type L. [See Fig. 5(b) for an example of H^k.] Suppose that $A \subseteq S$ and $B \subseteq T$. For any $t \in B$, the probability that at least one edge (s, t), $s \in A$, operates is $(1 - \prod_{s\in A} Pr[(s, t) \text{ fails}])$. Thus, for the k-replicated AOSP graph H^k of type L, we have

$$\begin{aligned} (L)\colon p_{AB}(H^k) = \{\prod_{t\in B}(1 - \prod_{s\in A} \Pr[(s, t) \text{ fails}])\} \\ \times \{\prod_{t\in T\backslash B}\prod_{s\in A} \Pr[(s, t) \text{ fails}]\}. \end{aligned} \tag{3}$$

For convenience, let M denote the matrix of $p_{AB}(H^k)$, so that $M = [p_{AB}(H^k)]_{A\subseteq S, B\subseteq T}$ with dimension $2^k \times 2^k$. For example, a 2-replicated AOSP graph of type L, $H^2(\hat{V}, \hat{E})$ is obtained from edge (s, t), where $\hat{V} = \{s_1, s_2, t_1, t_2\}$ and $\hat{E} = \{(s_1, t_1), (s_1, t_2), (s_2, t_1), (s_2, t_2)\}$. Let probabilities r_1, r_2, r_3, and r_4 ($\bar{r}_1$, $\bar{r}_2$, $\bar{r}_3$, and $\bar{r}_4$) be the reliabilities (unreliabilities) of edges (s_1, t_1), (s_2, t_1), (s_1, t_2), and (s_2, t_2), respectively. Then, the matrix M of H^2 is

$$\begin{aligned} M &= \begin{bmatrix} p_{\{s_1,s_2\},\{t_1,t_2\}}(H^2) & p_{\{s_1,s_2\},\{t_1\}}(H^2) & p_{\{s_1,s_2\},\{t_2\}}(H^2) & p_{\{s_1,s_2\},\{\}}(H^2) \\ p_{\{s_1\},\{t_1,t_2\}}(H^2) & p_{\{s_1\},\{t_1\}}(H^2) & p_{\{s_1\},\{t_2\}}(H^2) & p_{\{s_1\},\{\}}(H^2) \\ p_{\{s_2\},\{t_1,t_2\}}(H^2) & p_{\{s_2\},\{t_1\}}(H^2) & p_{\{s_2\},\{t_2\}}(H^2) & p_{\{s_2\},\{\}}(H^2) \\ p_{\{\},\{t_1,t_2\}}(H^2) & p_{\{\},\{t_1\}}(H^2) & p_{\{\},\{t_2\}}(H^2) & p_{\{\},\{\}}(H^2) \end{bmatrix} \\ &= \begin{bmatrix} (1-\bar{r}_1\bar{r}_2)(1-\bar{r}_3\bar{r}_4) & (1-\bar{r}_1\bar{r}_2)\bar{r}_3\bar{r}_4 & (1-\bar{r}_3\bar{r}_4)\bar{r}_1\bar{r}_2 & \bar{r}_1\bar{r}_2\bar{r}_3\bar{r}_4 \\ r_1r_3 & r_1\bar{r}_3 & r_3\bar{r}_1 & \bar{r}_1\bar{r}_3 \\ r_2r_4 & r_2\bar{r}_4 & r_4\bar{r}_2 & \bar{r}_2\bar{r}_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{4}$$

where {} represents the null set.

Next, we consider $H^k = H_1^k \wedge H_2^k$. By definition, the terminal sets S and T of H^k are given as $S = S_1 = S_2$ and $T = T_1 = T_2$. Suppose that $A \subseteq S$, $B_1 \subseteq T_1$, and $B_2 \subseteq T_2$; then, the event $[E_{AB_1}(H_1^k)] \wedge [E_{AB_2}(H_2^k)]$ implies the event $E_{AB}(H_1^k \wedge H_2^k)$ if $B_1 \cap B_2 = B$. Thus,

$$E_{AB}(H_1^k \wedge H_2^k) = \bigcup_{\substack{B_1 \subseteq T_1, B_2 \subseteq T_2 \\ s.t. B_1 \cap B_2 = B}} \{E_{AB_1}(H_1^k) \wedge E_{AB_2}(H_2^k)\},$$

i.e.:

$$(P_A): \quad p_{AB}(H_1^k \wedge H_2^k) = \sum_{\substack{B_1 \subseteq T_1, B_2 \subseteq T_2 \\ s.t. B_1 \cap B_2 = B}} p_{AB_1}(H_1^k) \cdot p_{AB_2}(H_2^k). \tag{5}$$

For example, let H_i^2 be a 2-replicated AOSP graph with terminal sets S_i and T_i for $i = 1, 2$, respectively. Let $S_1 = \{s_{11}, s_{12}\}$, $S_2 = \{s_{21}, s_{22}\}$, $T_1 = \{t_{11}, t_{12}\}$, and $T_2 = \{t_{21}, t_{22}\}$. Let $(H_1^2 \wedge H_2^2)$ be the 2-replicated AOSP graph obtained from H_1^2 and H_2^2 with terminals (s_1, s_2) and (t_1, t_2). Note that $(s_1, s_2) = (s_{11}, s_{12}) = (s_{21}, s_{22})$ and $(t_1, t_2) = (t_{11}, t_{12}) = (t_{21}, t_{22})$.

By Eq. (5), for any $A \subseteq \{s_1, s_2\}$, we have

$$\begin{aligned}
p_{A,\{t_1,t_2\}}(H_1^2 \wedge H_2^2) &= p_{A,\{t_{11},t_{12}\}}(H_1^2) p_{A,\{t_{21},t_{22}\}}(H_2^2), \\
p_{A,\{t_1\}}(H_1^2 \wedge H_2^2) &= p_{A,\{t_{11},t_{12}\}}(H_1^2) p_{A,\{t_{21}\}}(H_2^2) \\
&\quad + p_{A,\{t_{11}\}}(H_1^2) p_{A,\{t_{21},t_{22}\}}(H_2^2) \\
&\quad + p_{A,\{t_{11}\}}(H_1^2) p_{A,\{t_{21}\}}(H_2^2), \\
p_{A,\{t_2\}}(H_1^2 \wedge H_2^2) &= p_{A,\{t_{11},t_{12}\}}(H_1^2) p_{A,\{t_{22}\}}(H_2^2) \\
&\quad + p_{A,\{t_{12}\}}(H_1^2) p_{A,\{t_{21},t_{22}\}}(H_2^2) \\
&\quad + p_{A,\{t_{12}\}}(H_1^2) p_{A,\{t_{22}\}}(H_2^2), \\
p_{A,\{\}}(H_1^2 \wedge H_2^2) &= 1 - [p_{A,\{t_1,t_2\}}(H_1^2 \wedge H_2^2) \\
&\quad + p_{A,\{t_1\}}(H_1^2 \wedge H_2^2) \\
&\quad + p_{A,\{t_2\}}(H_1^2 \wedge H_2^2)].
\end{aligned} \tag{6}$$

Thus, a $2^2 \times 2^2$ matrix $M = [p_{AB}(H^2)]_{A \subseteq S, B \subseteq T}$ for graph $H_1^2 \wedge H_2^2$ can be obtained using the above equations.

Similarly, let $H^k = H_1^k \vee H_2^k$ with terminal sets S and T, where $S = S_1 = S_2$ and $T = T_1 = T_2$. Then, for any $A \subseteq S$, $B_1 \subseteq T_1$, and $B_2 \subseteq T_2$, we have

$$(P_O): \quad p_{AB}(H_1^k \vee H_2^k) = \sum_{\substack{B_1 \subseteq T_1, B_2 \subseteq T_2 \\ s.t. B_1 \cup B_2 = B}} p_{AB_1}(H_1^k) \cdot p_{AB_2}(H_2^k). \tag{7}$$

For example, let H_i^2 be a 2-replicated AOSP graph with terminal sets S_i and T_i for $i = 1, 2$, respectively. Let $S_1 = \{s_{11}, s_{12}\}$, $S_2 = \{s_{21}, s_{22}\}$, $T_1 = \{t_{11}, t_{12}\}$, and $T_2 = \{t_{21}, t_{22}\}$. Let $(H_1^2 \vee H_2^2)$ be the 2-replicated AOSP graph obtained from H_1^2 and H_2^2 with terminals (s_1, s_2) and (t_1, t_2). Note that $(s_1, s_2) = (s_{11}, s_{12}) = (s_{21}, s_{22})$ and $(t_1, t_2) = (t_{11}, t_{12}) = (t_{21}, t_{22})$. Then, for any $A \subseteq \{s_1, s_2\}$,

$$\begin{aligned}
p_{A,\{\}}(H_1^2 \vee H_2^2) &= p_{A,\{\}}(H_1^2) p_{A,\{\}}(H_2^2), \\
p_{A,\{t_1\}}(H_1^2 \vee H_2^2) &= p_{A,\{\}}(H_1^2) p_{A,\{t_{21}\}}(H_2^2) \\
&\quad + p_{A,\{t_{11}\}}(H_1^2) p_{A,\{\}}(H_2^2) \\
&\quad + p_{A,\{t_{11}\}}(H_1^2) p_{A,\{t_{21}\}}(H_2^2), \\
p_{A,\{t_2\}}(H_1^2 \vee H_2^2) &= p_{A,\{\}}(H_1^2) p_{A,\{t_{22}\}}(H_2^2) \\
&\quad + p_{A,\{t_{12}\}}(H_1^2) p_{A,\{\}}(H_2^2) \\
&\quad + p_{A,\{t_{12}\}}(H_1^2) p_{A,\{t_{22}\}}(H_2^2), \\
p_{A,\{t_1,t_2\}}(H_1^2 \vee H_2^2) &= 1 - [p_{A,\{\}}(H_1^2 \vee H_2^2) \\
&\quad + p_{A,\{t_1\}}(H_1^2 \vee H_2^2) \\
&\quad + p_{A,\{t_2\}}(H_1^2 \vee H_2^2)].
\end{aligned} \tag{8}$$

Finally, we consider $H^k = H_1^k * H_2^k$. By definition of the k-replicated AOSP graph of type S, the event $[E_{AB_1}(H_1^k) \cap E_{A_2B}(H_2^k)]$ implies the event $E_{AB}(H_1^k * H_2^k)$ if $B_1 = A_2$. So, we have

$$E_{AB}(H_1^k * H_2^k) = \bigcup_{\substack{B_1 \subseteq T_1, A_2 \subseteq S_2 \\ s.t. B_1 = A_2}} [E_{AB_1}(H_1^k) \wedge E_{A_2B}(H_2^k)].$$

Therefore,

$$(P_S): \quad p_{AB}(H_1^k * H_2^k) = \sum_{\substack{B_1 \subseteq T_1, A_2 \subseteq S_2 \\ s.t. B_1 = A_2}} p_{AB_1}(H_1^k) \cdot p_{A_2B}(H_2^k). \tag{9}$$

For example, let H_i^2 be a 2-replicated AOSP graph with terminal sets S_i and T_i for $i = 1, 2$, respectively. Let $S_1 = \{s_{11}, s_{12}\}$, $S_2 = \{s_{21}, s_{22}\}$, $T_1 = \{t_{11}, t_{12}\}$, and $T_2 = \{t_{21}, t_{22}\}$. Let $(H_1^2 * H_2^2)$ be the 2-replicated AOSP obtained from H_1^2 and H_2^2 with terminals (s_1, s_2) and (t_1, t_2). Note that $(s_1, s_2) = (s_{11}, s_{12})$, $(t_{11}, t_{12}) = (s_{21}, s_{22})$, and $(t_1, t_2) = (t_{21}, t_{22})$. Then,

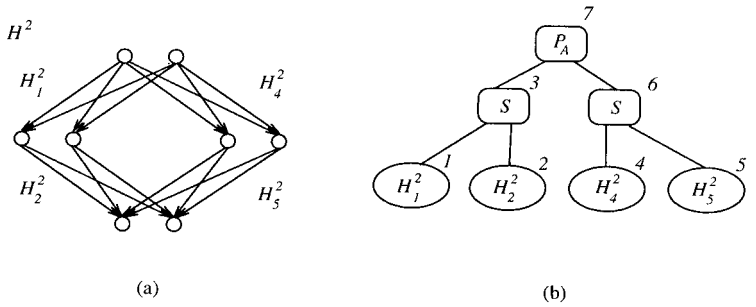

Fig. 6. (a) A numerical example; (b) its corresponding parsing tree.

$$\begin{aligned} p_{AB}(H_1^2 * H_2^2) &= p_{A,\{t_{11},t_{12}\}}(H_1^2)p_{\{s_{21},s_{22}\},B}(H_2^2) \\ &+ p_{A,\{t_{11}\}}(H_1^2)p_{\{s_{21}\},B}(H_2^2) \\ &+ p_{A,\{t_{12}\}}(H_1^2)p_{\{s_{22}\},B}(H_2^2) \\ &+ p_{A,\{\}}(H_1^2)p_{\{\},B}(H_2^2). \end{aligned} \tag{10}$$

Let $H^k(\hat{V}, \hat{E})$ be the k-replicated AOSP graph derived from $H(V, E)$. Since H^k is derived from H, the structure of the parsing tree of H, $T(H)$, is equivalent to the structure of the parsing tree of H^k, $T(H^k)$, i.e., $T(H) \simeq T(H^k)$. The only difference is that the leaf nodes in $T(H)$ are type L AOSP graphs, whereas the leaf nodes in $T(H^k)$ are type L RAOSP graphs with degree k. Therefore, $T(H^k)$ can be obtained by applying the algorithm [13] to its base AOSP graph H.

As discussed, each leaf node with the postorder sequence x in $T(H^k)$ corresponds to an RAOSP subgraph of type L, denoted as H_x^k. Every internal node x is labeled by S, P_O, or P_A according to the type of that RAOSP subgraph. Similar to the $T(H)$, every internal node x in $T(H^k)$, along with all its descendant nodes, induces a subtree T_x which describes a k-replicated AOSP subgraph H_x^k in H^k. Therefore, M_r for root node r can be determined by computing the M_x level by level for every node x in the parsing tree $T(H^k)$ using Eqs. (3), (5), (7), and (9). Finally, $R(H^k) = \Sigma_{\emptyset \neq B \subseteq T} p_{SB}(H^k)$ is emerged in the first row of matrix M_r.

We now present the algorithm:

Algorithm 1

STEP 1. *Find the parsing tree of the graph H^k, denoted as $T(H^k)$, by applying Valdes' algorithm to H* [13].

STEP 2. *Evaluate the matrix M_x for each node x in the parsing tree $T(H^k)$ by postorder traversal.*

STEP 3. *Compute $R(H^k) = \Sigma_{\subseteq \neq B \subseteq T} p_{SB}(H^k)$, where the terms $p_{SB}(H^k)$ can be found in the first row of matrix M_r at root node r in $T(H^k)$.*

It is known that Step 1 takes $O(|E|)$ [13]. In Step 2, if H_x^k is of type L, it takes $O(k^2)$ time to compute each entry in matrix M_x and, thus, $O(k^2 2^k)$ time in total for matrix M_x. If H_x^k is of type P_A or P_O, it takes $O(2^k \cdot 2^k)$ to compute each row in matrix M_x and, thus, $O(2^{3k})$ time for matrix M_x. If H_x^k is of type S (say $H_x^k = H_y^k * H_z^k$), matrix M_x is obtained from multiplying M_y by M_z and, thus, it takes $O(2^{3k})$ time to compute matrix M_x. Thus, the total time in Step 2 is $O(\max\{k^2 2^{2k}|E|, 2^{3k}|E|\})$. Hence the time complexity of Algorithm 1 is $O(K|E|)$, where $K = \max\{k^2 2^{2k}, 2^{3k}\}$.

5. A NUMERICAL EXAMPLE

We illustrate the calculation of the distributed task reliability in this section through an example: Consider a 2-replicated AOSP graph H^2 as shown in Figure 6(a), which is generated from the AOSP graph in Figure 4(a). It is readily seen that $H^2 = H_7^2 = (H_1^2 * H_2^2) \wedge (H_4^2 * H_5^2)$, where H_1^2, H_2^2, H_4^2, and H_5^2 are 2-replicated AOSP subgraphs of type L. [See Fig. 5(c) for $k = 2$.] In analogy to the parsing tree of H in Figure 4(b), the parsing tree for H^2 is given in Figure 6(b).

Suppose that the reliability of each link in H_i^2 is r_i, for $i = 1, 2, 4, 5$. Let $r_1 = 0.9$, $r_2 = 0.8$, $r_4 = 0.7$, and $r_5 = 0.7$. To calculate $R(H^2)$, we first calculate the M_i matrices for H_i^2, $i = 1, 2, 4, 5$. From Eq. (4), we have

$$M_1 = \begin{bmatrix} 0.9801 & 0.0099 & 0.0099 & 0.0001 \\ 0.81 & 0.09 & 0.09 & 0.01 \\ 0.81 & 0.09 & 0.09 & 0.01 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$M_2 = \begin{bmatrix} 0.9216 & 0.0384 & 0.0384 & 0.0016 \\ 0.64 & 0.16 & 0.16 & 0.04 \\ 0.64 & 0.16 & 0.16 & 0.04 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$M_4 = \begin{bmatrix} 0.8281 & 0.0819 & 0.0819 & 0.0081 \\ 0.49 & 0.21 & 0.21 & 0.09 \\ 0.49 & 0.21 & 0.21 & 0.09 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$M_5 = \begin{bmatrix} 0.8281 & 0.0819 & 0.0819 & 0.0081 \\ 0.49 & 0.21 & 0.21 & 0.09 \\ 0.49 & 0.21 & 0.21 & 0.09 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Next, we consider the subgraphs $(H_1^2 * H_2^2)$ and $(H_4^2 * H_5^2)$, i.e., the subtrees rooted at nodes 3 and 6 in Figure 6(b). Applying Eq. (10), we have

$$M_3 = M_1 * M_2$$

$$= \begin{bmatrix} 0.9159 & 0.0408 & 0.0408 & 0.0025 \\ 0.8617 & 0.0599 & 0.0599 & 0.0185 \\ 0.8617 & 0.0599 & 0.0599 & 0.0185 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$M_6 = M_4 * M_5$$

$$= \begin{bmatrix} 0.7660 & 0.1022 & 0.1022 & 0.0296 \\ 0.6116 & 0.1283 & 0.1283 & 0.1318 \\ 0.6116 & 0.1283 & 0.1283 & 0.1318 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Finally, we consider H^2. Given M_3 and M_6, we obtain the matrix $M = M_7$ for H^2 using Eq. (6). Thus,

$$M = M_7 = \begin{bmatrix} p_{\{s,s'\},\{t,t'\}}(H^2) & p_{\{s,s'\},\{t\}}(H^2) & p_{\{s,s'\},\{t'\}}(H^2) & p_{\{s,s'\},\{\}}(H^2) \\ p_{\{s\},\{t,t'\}}(H^2) & p_{\{s\},\{t\}}(H^2) & p_{\{s\},\{t'\}}(H^2) & p_{\{s\},\{\}}(H^2) \\ p_{\{s'\},\{t,t'\}}(H^2) & p_{\{s'\},\{t\}}(H^2) & p_{\{s'\},\{t'\}}(H^2) & p_{\{s'\},\{\}}(H^2) \\ p_{\{\},\{t,t'\}}(H^2) & p_{\{\},\{t\}}(H^2) & p_{\{\},\{t'\}}(H^2) & p_{\{\},\{\}}(H^2) \end{bmatrix}$$

$$= \begin{bmatrix} 0.7016 & 0.1290 & 0.1290 & 0.0404 \\ 0.5270 & 0.1549 & 0.1549 & 0.1632 \\ 0.5270 & 0.1549 & 0.1549 & 0.1632 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Finally, the reliability of H^2 can be obtained by Eq. (2):

$$R(H^2) = Pr\{\bigcup_{B \subseteq T, B \neq \emptyset} E_{SB}(H^2)\} = \sum_{\emptyset \neq B \subseteq T} p_{SB}(H^2)$$

$$= p_{\{s,s'\},\{t,t'\}}(H^2) + p_{\{s,s'\},\{t\}}(H^2) + p_{\{s,s'\},\{t'\}}(H^2)$$

$$= 0.7016 + 0.1290 + 0.1290 = 0.9596.$$

6. FURTHER DISCUSSION

In Section 4, we assumed that a k-replicated AOSP graph is fully replicated, i.e., each vertex is replicated exactly $(k-1)$ times and each edge is replicated $(k-1)^2$ times. Furthermore, we assumed that all vertices are perfectly reliable. In this section, we first extend our model to consider those cases where vertices and edges are partially replicated. Later on, we present a solution method to the problem that both vertices and edges can fail.

To calculate the reliability of partially replicated RAOSP graph, we first convert such a graph into a fully replicated RAOSP graph, then calculate its reliability using Algorithm 1. Suppose that $H^k = (\hat{V}, \hat{E})$ is an RAOSP graph of type L and is fully replicated. The RAOSP graph $H'^k = (\bar{V}, \bar{E})$ is partially replicated if $\bar{V} \subseteq \hat{V}$ and/or $\bar{E} \subseteq \hat{E}$. To compute $R(H^k)$, we first add those unreplicated vertices and edges (vertices and edges that would have been in a fully replicated H^k) into $\bar{V}$ and $\bar{E}$ and simply set the reliability of those edges to 0, i.e., $Pr\{(s_i, t_j)$ operates$\} = 0$, $\forall (s_i, t_j) \in \hat{E} \wedge (s_i, t_j) \notin \bar{E}$. After adding those edges, H'^k becomes fully replicated and $R(H'^k)$ can be obtained via Eq. (3). Now, we summarize the reliability analysis of the general RAOSP graph H^k with partial replication; we first find its parsing tree using the same algorithm in [13]. Then, we convert each RAOSP graph of type L in H^k into a fully replicated RAOSP graph of type L as shown above. Then, we apply Steps 2 and 3 in Algorithm 1 to obtain the reliability of H^k.

We next consider the RAOSP graphs with unreliable vertices. We begin the discussion with the AOSP graphs. To incorporate the unreliable vertices into our graph model, for each vertex (or terminal) t in an AOSP graph, we replace it by an edge (t, t') and assign the reliability of that edge to be the failure probability of that vertex. [See Fig. 7(b) as an example.] Similarly, to consider the unreliable vertices (or terminals) in an RAOSP graph H^k,

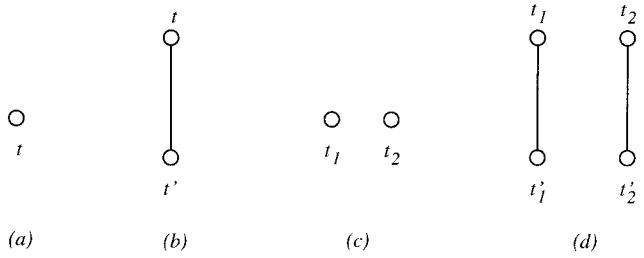

Fig. 7. (a) A single vertex; (b) its AOSP equivalent; (c) a vertex and its replica; (d) its RAOSP equivalent.

we can also replace each terminal in H^k by an edge. As shown in Figure 7(c) and (d), it is a terminal after being replicated once and each terminal being replaced by an edge. We notice that this structure is an RAOSP graph of type L with partial replication, and, thus, its reliability can be obtained using the method presented in the previous section.

7. CONCLUSIONS

This paper has focused on the design of an efficient algorithm to predict the reliability of tasks characterized by k-replicated and–or series-parallel (AOSP) graphs. A k-replicated AOSP graph is derived from an AOSP graph with vertex and edge replications. Conventional algorithms may apply to compute the reliability of a k-replicated AOSP graph. However, these algorithms take exponential time in the number of edges. We have presented an algorithm with time complexity $O(K(|V| + |E|))$, where $K = \max\{k^2 2^{2k}, 2^{3k}\}$ and $|V|$ and $|E|$ are the number of vertices and edges, respectively, in its corresponding AOSP graph. In real-life applications, the k is typically small whereas $|V| + |E|$ is much larger; thus, our algorithm is a significant improvement over the traditional approaches.

Many graph-related problems are NP-complete for general graphs but can be solved in polynomial-time for AOSP graphs. In this paper, we have shown that the distributed task reliability problem can also be solved in linear time for RAOSP graphs. It seems that for other graph problems there may also exist polynomial-time algorithms for RAOSP graphs provided there exist polynomial-time algorithms for AOSP graphs.

REFERENCES

[1] A. Agrawal and R. E. Barlow, A survey of network reliability and domination theory. *Oper. Res.* **32** (1984) 478–492.

[2] P. A. Alsberg and J. D. Day, A principle for resilient sharing of distributed resources. *Proceedings of the 2nd International Conference on Software Engineering* (Oct. 1976) 562–570.

[3] S. Arnborg and A. Proskuronski, On network reliability. *Discr. Appl. Math.* **23**(1) (1989) 11–24.

[4] M. O. Ball, Computational complexity of network reliability analysis: An overview. *IEEE Trans. Reliab.* **R-35**(3) (1986) 230–239.

[5] W. W. Chu and K. K. Leung, Module replication and assignment for real-time distributed processing systems. *Proceed. IEEE* **75** (1987) 547–562.

[6] C. J. Colbourn, *The Combinatorics of Network Reliability.* Oxford University Press, New York (1987).

[7] S. Hariri and C. S. Raghavendra, Syrel: A symbolic reliability algorithm based on path and cutset methods. *IEEE Trans. Comput.* **C-36** (1987) 1224–1232.

[8] S.-T. Levi, S. Tripathi, S. Carson, and A. Agrawala, The maruti hard real-time operating system. *ACM Oper. Syst. Rev.* **23** (1989) 90–105.

[9] V. W. Mak and S. F. Lundstrom, Predicting performance of parallel computations. *IEEE Trans. Parallel Distrib. Syst.* **1** (1990) 257–270.

[10] D. Powell, *Delta-4: Overall System Specification.* The Delta-4 Project Consortium (1988).

[11] C. S. Raghavendra, V. K. Prasanna, J. Kumar, and S. Hariri, Reliability analysis in distributed systems. *IEEE Trans. Comput.* **C-37** (1988) 352–358.

[12] A. Satyanarayana and A. Prabhakar, New topological formula and rapid algorithm for reliability analysis of complex networks. *IEEE Trans. Reliab.* **R-27**(2) (1979) 82–100.

[13] J. Valdes, R. Tarjan, and E. L. Lawler, The recognition of series parallel digraphs. *SIAM J. Comput.* **11** (1982) 298–313.