ELSEVIER

Computer Communications 24 (2001) 422–433

computer communications

www.elsevier.com/locate/comcom

A reversing traversal algorithm to predict deleting node for the optimal *k*-node set reliability with capacity constraint of distributed systems

Yi-Shiung Yeh*, Chin-Ching Chiu

Institute of Computer Science and Information Engineering, National Chiao Tung University,1001 Ta Hsueh Road, Hsinchu 300, Taiwan, ROC

Received 5 January 2000; accepted 26 May 2000

Abstract

A *k*-node set reliability with capacity constraint is defined as the probability that a set, *K*, of nodes is connected in a distributed system and the total capacity of the nodes in *K* is sufficient under a given capacity. This is generally an NP-hard problem. For reducing computational time, a reasonable *k*-node set within a given capacity constraint must be determined by an efficient algorithm. In this work, we propose a reversing traversal method to derive a *k*-node set under capacity constraint having an approximate solution. Initially, the set *K* is assigned to all the nodes in a system. The proposed algorithm uses an objective function to evaluate the fitness value of each node in *K* and predict a deleting node, which is not a critical node, in *K* with minimal fitness value. After deleting the node, the fitness value of each node that is adjacent to the deleted node is tuned. The above two processes are repeated until the total capacity of the nodes in each subset of the set *K* does not satisfy the capacity constraint. In our simulation, the proposed method can obtain an exact solution above 90%. When a sub-optimal solution is obtained, the average deviation from an exact solution is under 0.0033. Computational results demonstrate that the proposed algorithm is efficient in execution time and effective for obtaining an optimal *k*-node set with capacity constraint.

Keywords: Reversing traversal algorithm; Distributed systems; Reliability optimization; *k*-node set reliability

1. Introduction

Distributed systems provide cost-effective ways for improving a computer system's performance such as throughput, fault-tolerance, and reliability optimization [1,2]. The reliability optimization of a DS has become a critical issue. Many computers networked over several universities, are such a distributed system. Those schools are willing to offer their computer resources but with resource constraints (e.g. CPU time, memory space, disk quota). Therefore, the data-files of each sever are distributed among several computers under the resource constraints on each computer. The schools are requested to list what services their computers can afford, so that the entire information system can be planned and the reliability of each computer or communication link evaluated according to the ratio of time periods individual computers will be unavailable [3]. When some data files are allocated into a DS, a set *K* of nodes in the DS must be selected to allocate the data files such that a *k*-node set reliability (KNR) is adequate under the constraints.

Computing the reliability of a distributed system is generally an NP-hard problem [4,5]. Much research has been performed to provide reliability optimization models of DS that optimize source to destination reliability, *k*-out-of-*n* systems reliability and overall system reliability [6–12]. An exact method (EM) [13] and a *k*-tree reduction method [14] have been used to examine KNR optimization with capacity constraint. The EM is based on the partial order relation and propose some rules to indicate the conditions under which certain vectors in the numerical ordering that do not satisfy the capacity constraints can be skipped over. The *k*-tree reduction method is a greedy algorithm which can reduce immediately disjoint terms and avoids unnecessary spanning tree generation.

The EM can obtain an optimal solution, but cannot effectively reduce the problem space. Moreover, an increasing number of nodes causes the EM execution time to exponentially grow. Occasionally, an application requiring an efficient algorithm with an approximate solution is highly attractive. The *k*-tree reduction method is one method that obtains a *k*-node set using a heuristic search. It serves the first node v_1 as the starting node and adds a node to the *k*-node set depending on the maximum the reliability product of a current selected set and the total capacity of nodes of

* Corresponding author. Tel.: +886-3-5724176; fax: +886-3-5712121.
E-mail address: ysyeh@csie.natu.edu.tw (Y.-S. Yeh).

0140-3664/01/$ - see front matter
PII: S0140-3664(00)00242-5

that set. The product of the above expression relies heavily on the summation capacity of each selected node but depends slightly on k-node reliability. Therefore, its computational complexity is still high and the average derivation from the exact solution is not ideal.

In summary, the previous methods are all extremely inefficient. The objective of this paper is to propose a reversing traversal algorithm for obtaining a subset of network nodes such that the reliability is maximum or nearly maximum and the specified capacity constraint is satisfied.

The rest of this paper is organized as follows. Section 2 defines the KNR problem. Section 3 discusses the predicting a removal node concept, treats an algorithm for KNR and illustrates an example. Section 4 describes the environment and parameters for verifying the algorithm. Section 5 provides the results of various parameters specified in Section 4 and analyzes the time complexity and deviation from the exact solution. Section 6 presents the concluding remarks.

2. Problem description

In this section, we described the problem addressed herein for convenience and clarification our research objectives.

2.1. Notations and definitions

The following notations, and definitions will be used.

Notations

$G = (V, E)$ an undirected DS graph where V denotes a set of processing elements, and E represents a set of communication links

n the number of nodes in G, $n = |V|$

v_i the ith processing element or the ith node

$c(v_i)$ the capacity of the ith node

e the number of links in G, $e = |E|$

$e_{i,j}$ an edge represents a communication link between v_i and v_j

$p_{i,j}$ the probability of success of link $e_{i,j}$

$q_{i,j}$ the probability of failure of link $e_{i,j}$

G_k the graph G with the set k of nodes specified, and $|k| \geq 2$

$R(G_k)$ the reliability of k-node set solution of a DS graph G

$c(G_k)$ the sum of capacity of k-node set of a DS graph G

C_{need} total capacity constraint in a DS

$w(G_k)$ object function for evaluating the weight of G_k

$d(v_i)$ the number of links connected to the node v_i

$w_{\text{fast}}(v_i)$ the approximate weight of the ith node

$w(e_{i,j})$ the weight of the link $e_{i,j}$

$w_{\text{optimal}}(v_i)$ the optimal weight of the ith node

X $\{v_i | c(G_k) - c(v_i) \geq C_{\text{need}},\ v_i \in G_k\}$

Y $\{v_i | v_i \in X$ and $G_k - \{v_i\}$ is connected without through a node in $V - \{G_k \cup \{v_i\}\}$

d_{same} $= 1$, if all $d(v_i)$ is same each other $= 0$, otherwise

$\text{FT}(v_i)$ the fitness value of the ith node

V_{visit} a set of nodes which are connected by some specified links

DSR_{app} an approximate solution which obtained by running heuristic algorithm

DSR_{opt} optimal solution which obtained by running exhaustive search algorithm

R_{err} the value of relative error which is equal to $1 - (\text{DSR}_{\text{opt}}/\text{DSR}_{\text{app}})$

c_d the maximal value of $\max(c(v_i))/\min(c(v_i))$ for tuning the range of each node's capacity

C_D a value of $C_{\text{need}}/([\sum_{i=1}^{n} c(v_i)]/n)$ for tuning the rang of capacity constraint, i.e. $C_{\text{need}} = [(\sum_{i=1}^{n} c(v_i))/n] \times C_D$

Definition 1. A KNR is defined as the probability that a specified set K of nodes is connected (where K denotes a subset of the set of processing elements).

Definition 2. A critical node is defined as a node, say v_i, in G, if G' represents v_i and all the links which are incidence with v_i deleted in G, G' with at least two connected components.

Definition 3. Capacity constraint is defined as the total memory size required when some files are loaded into the system.

Definition 4. Absolute error is defined as the value of subtracting an approximate solution from an exact solution of KNR.

Definition 5. Relative error is defined as the value of dividing an exact solution into the absolute error.

Definition 6. The ratio of average relative error is defined as the value of dividing the summation of relative error by the number of the total simulation cases under consideration.

2.2. Problem statements

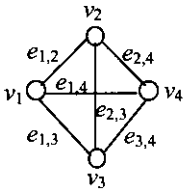
A set K of nodes can be derived from the given set V that constitutes a DS in that KNR is adequate and the total capacity satisfies the capacity constraint. Consider a DS of n nodes and e links. The capacity constraint is C_{need}, where its optimal DS topology is the set K of nodes. Restated, the set K of nodes has the maximum reliability and its total capacity at least is as large as the capacity constraint C_{need}. The main problem can be mathematically stated as follows:

Object: Maximize $R(G_k)$
subject to: $\sum_{v_i \in G_k} c(v_i) \geq C_{\text{need}}$
where $R(G_k)$, $c(v_i)$, C_{need} are defined in Section 2.1.

Obviously, the problem for a large DS, as in a metropolitan area network, requires a large execution time.

3. Reversing traversal algorithm for searching an optimal k-node set

In this section, we present an algorithm to obtain an



$p_{1,2}$=0.74 $p_{1,3}$=0.94 $p_{1,4}$=0.83
$p_{2,3}$=0.68 $p_{2,4}$=0.57 $p_{3,4}$=0.46

Fig. 1. A DS with four nodes and six links.

adequate k-node set with capacity constraint. The analysis performed herein assumes that all the nodes are perfect and all the links are unreliable.

3.1. Computing KNR

Bi-directional communication channels operate between processing elements. A distributed computing system can be modeled using a simple undirected graph. For a topology of the DS with n nodes and e links, there are 2^n subsets of nodes. A set, K, is a subset of the DS which includes some nodes of the given node set V. The KNR is the probability that a specified set K of nodes is connected, where K denotes a subset of the set of processing elements. Fig. 1 depicts the bridge topology of a DS with four nodes and six links.

For example, when $K = \{v_1, v_4\}$ is selected in the DS with bridge topology, then the reliability of the set, K, can be computed using a factoring algorithm as follows

$$\begin{aligned}R(G_k) = {} & p_{1,2}\{p_{3,4}[(p_{1,4} + p_{2,4} - p_{1,4}p_{2,4}) \\ & + (p_{1,3} + p_{2,3} - p_{1,3}p_{2,3}) - (p_{1,4} + p_{2,4} - p_{1,4}p_{2,4}) \\ & \times (p_{1,3} + p_{2,3} - p_{1,3}p_{2,3})] \\ & + q_{3,4}(p_{1,4} + p_{2,4} - p_{1,4}p_{2,4})\} \\ & + p_{1,2}[p_{3,4}(p_{1,3} + p_{1,4} - p_{1,3}p_{1,4}) \\ & + q_{3,4}(p_{1,4} + p_{1,3}p_{2,3}p_{2,4} - p_{1,4}p_{1,3}p_{2,3}p_{2,4})].\end{aligned}$$

The reliability of the set, K, can also be computed by means of a sum of mutually disjoint terms [15]

$$\begin{aligned}R(G_k) = {} & p_{1,2}q_{1,3}p_{1,4}p_{2,3}q_{2,4}q_{3,4} + p_{1,2}q_{1,3}p_{1,4}q_{2,3}q_{2,4} \\ & + q_{1,2}q_{1,3}p_{1,4} + p_{1,2}p_{1,3}p_{1,4}q_{2,3}q_{2,4}q_{3,4} \\ & + q_{1,2}p_{1,3}p_{1,4}q_{2,3}q_{3,4} + q_{1,2}p_{1,3}p_{1,4}p_{2,3}q_{2,4}q_{3,4} \\ & + p_{1,2}p_{1,3}q_{2,3}p_{2,4}q_{3,4} + p_{1,2}q_{1,3}q_{2,3}p_{2,4} \\ & + p_{1,2}q_{1,3}p_{2,3}p_{2,4}q_{3,4} + p_{1,3}q_{2,3}p_{3,4} \\ & + q_{1,2}p_{1,3}p_{2,3}q_{2,4}p_{3,4} + p_{1,2}p_{1,3}p_{2,3}q_{2,4}p_{3,4} \\ & + p_{1,2}q_{1,3}p_{2,3}p_{3,4} + p_{1,3}p_{2,3}p_{2,4}\end{aligned}$$

Assume that the probability of $p_{1,2}$, $p_{1,3}$, $p_{1,4}$, $p_{2,3}$, $p_{2,4}$ and $p_{3,4}$ is 0.74, 0.94, 0.83, 0.68, 0.57 and 0.46, respectively. We obtain $R(G_k) = 0.9539197$ using the above two methods.

3.2. The concept of proposed algorithm

The number of ports at each node (degree of a node), and the number of links directly impact the system reliability, while reliability decreases with a decrease in the number of links [2,16]. Therefore, we can quickly obtain the weight of each link and node according to the DS topology and the reliability of each link. We can serve V as k-node set. Then, we try to find an objective function to evaluate the fitness value of each node and select a node with the minimal fitness value and that is not a critical node for deletion. After deleting the node, the fitness value of each node is tuned which is adjacent to the deleted node. The latter two steps are repeated until no node can be deleted.

3.2.1. Initializing the k-node set

The algorithm initializes the k-node set as all the processing elements in the DS.

3.2.2. Evaluating the approximate weight of a node

The reliability of connecting the selected nodes is dependent on the degree of links and the link reliability. For a given node, the node degree can affect the information transformation between two nodes. The following formula is used to compute the approximate weight of node v_i

$$w_{\text{fast}}(v_i) = \max\{p_{ij} | e_{ij} \text{ is incident with } v_i\} \tag{1}$$

3.2.3. Evaluating the weight of a link

In the network, two nodes may contain many paths between them. The length of a path is between one and $n-1$. To reduce the computational time, the path in which the length is not greater than two is considered. The following formula is used to evaluate the weight of link $e_{i,j}$

$$w(e_{i,j}) = 1 - q_{i,j}\prod_{z=1}^{y_{i,j}}(q_{i,k_z}q_{k_z,j}) \tag{2}$$

where $y_{i,j}$ denotes the number of paths with a length two between v_i and v_j. In addition, the value of $y_{i,j}$ is not greater than $n-2$. This formula can easily edit a program and reduces many unnecessary multiplication operations. The weight of $e_{i,j}$ can be computed in one subtraction $2y_{i,j}-1$ multiplication. Thus, in the worst case, when the graph is a complete graph, we can obtain all the weights of each link in e addition and $2ne-5e$ multiplication.

3.2.4. Evaluating the optimal weight of a node

We employed a simple method for computing the node value, which takes less time and can quickly compute the weight of every node. The following formula is used to

compute the weight of node v_i

$$W_{\text{optimal}}(v_i) = 1 - \prod_{z=1}^{d(v_i)} (1 - w(e_{i,k_z})) \tag{3}$$

The above formula is easy to program and reduces many multiplicative operations. If the degree of v_i is $d(v_i)$, the weight of v_i can be computed in one subtract and $d(v_i) - 1$ multiplication. Thus, we can obtain the weight of every node in n addition and $2e$ multiplication.

3.2.5. Reducing the order of k-node set

The following observations can be made on how to reduce a k-node set. For a given selected k-node set, the reliability of this k-node set is less than the reliability of its proper subset. Thus, during the reliability evaluation process, if the total subset capacity of selected set is at least as large as capacity constraints, then the selected set should be replaced by its subset.

3.2.6. Objective function for obtaining fitness value of a node

We can obtain following simple objective function to evaluate each node's fitness value of a k-node set before discarding a node

$$\text{FT}(v_i) = \begin{cases} W_{\text{fast}}(v_i), & \text{if } n = e \\ W_{\text{optimal}}(v_i), & \text{if } \forall \deg(v_i) \text{ is same each other and } n \neq e \\ W_{\text{optimal}}(v_i) \times W_{\text{fast}}(v_i), & \text{otherwise,} \end{cases} \tag{4}$$

where $\text{FT}(v_i)$, $w_{\text{fast}}(v_i)$, $w_{\text{optimal}}(v_i)$ is defined in Section 2.1.

3.2.7. Predicting a removal node

After obtaining each node's fitness value, a node is selected for deletion, say v_i, which has the minimal fitness value and all the other nodes in the k-node set are connected without through any nodes which are not in k-node set or the node v_i.

3.2.8. Tuning the weight of a node

After discarding a node, say v_i, the weight of each node is tuned, say $w_{\text{optimal}}(v_j)$, which is adjacent to v_i as follows:

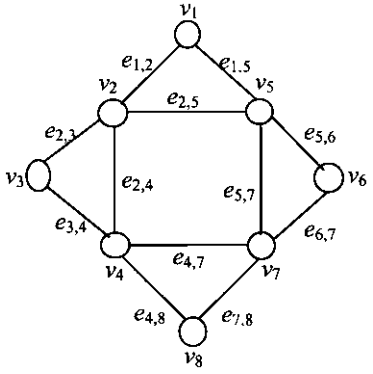
$$W_{\text{optimal}}(v_j) = 1 - \prod_{z=1, k_z \neq j}^{d(v_i)-1} (1 - w(e_{i,k_z})) \tag{5}$$

3.3. Proposed algorithm

A heuristic algorithm is presented to obtain an optimal k-node set with sufficient capacity in a DS. The algorithm is based on an objective function to evaluate the fitness value of each node in K and then discard the node which is not critical and has the minimum fitness value. This process will be performed until the total capacity of proper set is less than the capacity constraint.

```
Algorithm reverse_traversal_KNR
Step 0 Read a DS topology: n, e, V, E.
  Generate p_{i,j} of each link, c(v_i) of each node and C_need
  using random number generator.
Step 1 /*evaluate the approximate weight of each node
using Eq. (1) and inspect whether the degree of each node
is equal to each other.*/
  Let E_tmp = E.
  dowhile (E_tmp! = ∅)
    choose a link, say e_{i,j}, from E_tmp.
    if (p_{i,j} > w_fast(v_i))
      w_fast(v_i) = p_{i,j}.
    end if
    if (p_{i,j} > w_fast(v_i))
      w_fast(v_j) = p_{i,j}.
    end if
    Let d(v_i) = d(v_i) + 1.
    Let d(v_j) = d(v_j) + 1.
    Let E_tmp = E_tmp − {e_{i,j}}./*discard e_{i,j} from E_tmp*/
  end dowhile
  Let d_same = 1 /*set d_same to TRUE.*/
  Let d_tmp = d(v_i).
  Let V_tmp = V.
  dowhile (V_tmp! = ∅)
    choose a node from V_tmp.
    if (d(v_i)! = d_tmp)
      Let d_same = 0./* set d_same to FALSE.*/
      break.
    end if
    Let V_tmp = V_tmp − {v_i}.
  end while
Step 2 /*Evaluate the weight of each link using Eq. (2).*/
  Let E_tmp = E.
  dowhile (E_tmp! = ∅)
    choose a link, say e_{i,j}, from E_tmp.
    Let w(e_{i,j}) = q_{i,j},
    Let k = 1.
    Dowhile (k <= n)
      if (link e_{i,k} and e_{k,j} are belong to E)
        Let w(e_{i,j}) = w(e_{i,j})*q_{i,k}*q_{k,j}.
      end if
      Let k = k + 1.
    end dowhile
    Let w(e_{i,j}) = 1 − w(e_{i,j}).
    Let E_tmp = E_tmp − {e_{i,j}}./*discard e_{i,j} from E_tmp*/
  end dowhile
Step 3 /*Evaluate the optimal weight of each node using
Eq. (3).*/
  Let w_optimal(v_i) = 0,  for i = 1, ..., n.
  Let E_tmp = E.
  dowhile (E_tmp! = ∅)
    choose a link, say e_{i,j}, from E_tmp.
    if (w_optimal(v_i) = 0)
```

Let ($w_{optimal}(v_i) = w(e_{i,j})$.
else
Let ($w_{optimal}(v_i) = w_{optimal}(v_i)$*(1 − $w_{optimal}(v_i)$) *$w(e_{i,j})$.
end if
if ($w_{optimal}(v_j) = 0$)
Let ($w_{optimal}(v_j) = w(e_{i,j})$.
else
Let ($w_{optimal}(v_j) = w_{optimal}(v_j)$*(1 − $w_{optimal}(v_j)$) *$w(e_{i,j})$.
end if
Let $E_{tmp} = E_{tmp} - \{e_{i,j}\}$./*discard $e_{i,j}$ from E_{tmp}*/
end dowhile
Step 4 /*evaluate the fitness value using Eq. (4).*/
Let $V_{tmp} = V$.
dowhile ($V_{tmp}! = \varnothing$)
choose a node from V_{tmp}.
if ($n = e$)
Let $FT(v_i) = w_{fast}(v_i)$.
else if ($d_{same} = 1$)
Let $FT(v_i) = w_{optimal}(v_i)$.
else
Let $FT(v_i) = w_{fast}(v_i)$*$w_{optimal}(v_i)$, for $i = 1, \ldots, n$.
end if
end if
Let $V_{tmp} = V_{tmp}$.
end while
Step 5 Let $G_k = V$. /* V is all the nodes in the DS */
Step 6 /*Find each v_i, in G_k, so that $c(G_k) - c(v_i) \geq C_{need}$.*/
Let $X = \{\ \}$.
Let $V_{tmp} = G_k$.
dowhile ($V_{tmp}! = \varnothing$)
choose a node, say v_i, from V_{tmp}.
if ($c(G_k) - c(v_i) \geq C_{need}$)
Let $X = X \cup \{v_i\}$.
end if
Let $V_{tmp} = V_{tmp} - \{v_i\}$.
end dowhile
if ($X = \varnothing$)
go to step 9.
end if
Step 7 /* find each v_i, in X, so that $G_k - \{v_i\}$ is connected without through a node in $V - (G_k \cup v_i)$.*/
Let $Y = \{\ \}$.
dowhile ($X! = \varnothing$)
Let $V_{visit} = \{\ \}$.
choose a node, say v_i, from X.
Let $E_{tmp} = E$.
dowhile ($E_{tmp}! = \varnothing$ or $V_{visit}! = G_k - \{v_i\}$)
choose a link, say $e_{s,t}$, from E_{tmp}.
if ($v_s \in G_k - \{v_i\}$ and $v_t \in G_k - \{v_i\}$)
Let $V_{visit} = V_{visit} \cup \{v_s, v_t\}$.
end if
Let $E_{tmp} = E_{tmp} - \{e_{s,t}\}$./*discard $e_{s,t}$ from E_{tmp}*/
end dowhile
if ($V_{visit} = G_k - \{v_i\}$)
Let $Y = Y \cup \{v_i\}$.
end if
Let $X = X - \{v_i\}$.
end dowhile
if ($Y = \varnothing$)
go to step 9.
end if
Step 8 /*Find a v_i, in Y, so that $v_i = \min\{FT(v_i) | v_i \in Y\}$.*/
Let $w_{tmp} = 1$.
dowhile ($Y! = \varnothing$) /* find a node, in Y, with a minimal fitness value.*/
choose a node, say v_i, from Y.
if ($FT(v_i) < w_{tmp}$)
Let $v_x = v_i$.
Let $w_{tmp} = FT(v_i)$.
end if
Let $Y = Y - \{v_i\}$.
end dowhile
Let $G_k = G_k - \{v_x\}$.
Let $c(G_k) = c(G_k) - c(v_x)$.
if ($n! = e$ and $d_{same} = 0$)
/*tune the fitness value of nodes which is adjacent to the deleted node v_x.*/



$c(v_1)$=9 $c(v_2)$=4 $c(v_3)$=11 $c(v_4)$=8
$c(v_5)$=6 $c(v_6)$=10 $c(v_7)$=4 $c(v_8)$=5
$p_{1,2}$=0.742637 $p_{1,5}$=0.577502 $p_{2,3}$=0.775323
$p_{2,4}$=0.989624 $p_{2,5}$=0.239509 $p_{3,4}$=0.904599
$p_{4,7}$=0.703879 $p_{4,8}$=0.276681 $p_{5,6}$=0.990020
$p_{5,7}$=0.230567 $p_{6,7}$=0.910642 $p_{7,8}$=0.758843
Capacity constraints ≥ 31

Fig. 2. A DS with eight nodes and twelve links.

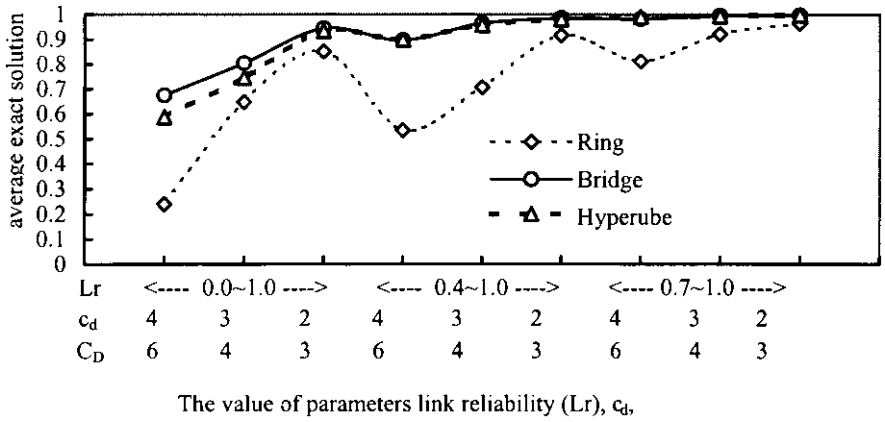


Fig. 3. The results of the average exact solution of KNR using exhaustive method on different topologies and parameters considerations.

For each v_i which is adjacent to v_x,
Let $FT(v_i) = FT(v_i)/w_{fast}(v_i)$.
Let $w_{optimal}(v_i) = 1 - ([1 - w_{optimal}(v_i)]/[1 - w(e_{x,i})])$.
Let $FT(v_i) = FT(v_i)*w_{fast}(v_i)$.
end if
go to step 5.
Step 9 /* reduce the order of the k-node set as much as possible, find a v_i, in G_k, so that $v_i = \min\{c(v_i)|v_i \in G_k\}$.*/
dowhile ($c(G_k) >= C_{need}$)
choose a node, say v_i, from G_k.
Let $c_{tmp} = c(v_i)$.
dowhile ($G_k ! = \varnothing$) /* find a node, in G_k, with a minimal capacity.*/
choose a node, say v_i, from $G_k - \{v_i\}$.
if ($c(v_i) < c_{tmp}$)
Let $v_x = v_i$.
Let $c_{tmp} = c(v_i)$.
end if
end dowhile
if ($c(G_k) - c(v_x) >= C_{need}$)
Let $G_k = G_k - \{v_x\}$.
Let $c(G_k) = c(G_k) - c(v_x)$.
end if
end dowhile
Step 10 Compute $R(G_k)$ using SYREL [15], output the k-node set G_k and its reliability.

Table 1
The results obtained using different methods for various DS topologies

Size		Optimal solution		EM	k-tree reduction method		Proposed method	
n[a]	e[b]	DSR_{opt}[c]	k-node set[d]	Time (s)[e]	R_{err}[f]	Time (s)	R_{err}	Time (s)
10	17	0.9994068	2,8,9	10.98	0.0074441	8.69	0.0	0.64
10	19	0.9995282	1,5,6	58.88	0.0019527	15.06	0.0	2.37
11	16	0.9911286	1,6,11	10.88	0.0378511	4.97	0.0	1.45
11	17	0.9974023	1,10,11	29.55	0.0120129	7.54	0.0	1.95
12	18	0.9858263	3,4,5,6	52.07	0.0299250	6.22	0.0	2.92
12	21	0.9990777	1,3,5,6	679.04	0.0056617	72.00	0.0	4.73
13	20	0.9978402	4,6	125.23	0.0189070	102.09	0.0006842	3.04
17	20	0.7644486	2,3,4,5	219.21	0.2224275	8.86	0.0	3.11
19	31	0.9979870	6,8,9	761.24	0.0856661	507.62	0.0031965	4.88
30	30	0.9856600	2,3,4	404.91	0.0706510	14.30	0.0	1.13
32	33	0.9937695	2,3,4	964.58	0.1050981	895.70	0.0	7.31
60	60	0.9530911[g]	5,6	–[h]	0.1408611	199.17	0.0	5.63
120	120	0.9800450[g]	2,3	–[h]	0.1812755	2042.94	0.0	29.34

[a] n: the numbger of nodes in G.
[b] e: the number of links in G.
[c] DSR_{opt}: optimal solution which obtained by running exhaustive search algorithm; DSR_{app}: approximation solution which obtained by running heuristic algorithm.
[d] k-node set: selected nodes.
[e] Time: seconds for obtaining k-node set whose reliability is maximal.
[f] R_{err}: the value of relative error which is equal to $[1 - (DSR_{opt}/DSR_{app})]$.
[g] Value obtained by manual calculation.
[h] Value barely obtained.

Table 2
The results obtained using k-tree reduction method for three DS topologies with eight nodes

T(s)[a]	Lr[b]	c_d[c]	C_D[d]	AES[e]	HitR[f]	ARErrR[g]	UpErrBnd[h]	UpErrBndR[i]	ARErrRlnk[j]	ARErrRT[k]
Ring (n8e8)	0.0–1	4	6	0.239511	30	0.164248	0.293497	0.74322		
		3	4	0.647318	40	0.234192	0.414967	0.87533		
		2	3	0.850519	30	0.295076	0.573058	0.671024	0.231172	
	0.4–1.0	4	6	0.534633	0	0.37804	0.576078	0.82229		
		3	4	0.706684	10	0.289142	0.528725	0.618443		
		2	3	0.914725	0	0.17879	0.226831	0.315245	0.281991	
	0.7–10	4	6	0.811063	0	0.084463	0.140802	0.153146		
		3	4	0.920673	0	0.140077	0.168355	0.179859		
		2	3	0.963919	10	0.059963	0.145553	0.157428	0.094834	0.202666
Bridge (n8e12)(Fig. 3)	0.0–1.0	4	6	0.673502	30	0.193834	0.449903	0.636683		
		3	4	0.803006	0	0.359327	0.693904	0.845069		
		2	3	0.943866	0	0.281859	0.397201	0.518118	0.27834	
	0.4–1.0	4	6	0.895392	0	0.158996	0.275139	0.318859		
		3	4	0.963701	0	0.098579	0.196163	0.206595		
		2	3	0.984607	10	0.122654	0.347362	0.348134	0.126743	
	0.7–1.0	4	6	0.981713	0	0.040552	0.075066	0.076287		
		3	4	0.994233	10	0.020554	0.044664	0.045001		
		2	3	0.997221	0	0.022574	0.052358	0.052402	0.027893	0.144325
Hypercube (n8e12)	0.0–1.0	4	6	0.586086	10	0.256013	0.358596	0.308105		
		3	4	0.744266	10	0.248013	0.398589	0.547448		
		2	3	0.933596	0	0.300219	0.815804	0.822601	0.268082	
	0.4–1.0	4	6	0.898183	0	0.046946	0.112462	0.129018		
		3	4	0.957629	0	0.04992	0.117201	0.123205		
		2	3	0.980899	10	0.048929	0.11568	0.116619	0.048598	
	0.7–1.0	4	6	0.990639	0	0.007818	0.013755	0.013854		
		3	4	0.993888	20	0.005746	0.020516	0.006131		
		2	3	0.998169	20	0.00677	0.024302	0.024318	0.006778	0.107819
Average					8.9	0.151603	(0.280612)[l]	(0.358312)[l]	0.151603	0.151603

[a] T(s): represents the topology (size) of a DS.
[b] Lr: the range of link's reliability, the reliability is obtained by random generator.
[c] c_d: the maximal value of $\max(c(v_i))/\min(c(v_i))$ for tuning the range of each node's capacity.
[d] C_D: the value of $C_{\text{need}}/([\sum_{i=1}^{n} c(v_i)]/n)$ for tuning the range of capacity constraint, i.e. $C_{\text{need}} = [(\sum_{i=1}^{n} c(v_i))/n] \times C_D$.
[e] AES: the value of average exact solution whose value is $[\sum(\text{DSR}_{\text{opt}}]/$(total simulation cases).
[f] HitR: the ratio of obtaining exact solution.
[g] ARErrR: the value of $(\sum[1 - (\text{DSR}_{\text{opt}}/\text{DSR}_{\text{app}})])/$(total simulation cases).
[h] UpErrBnd: the upper error bound whose value is the $\max(\text{DSR}_{\text{opt}} - \text{DSR}_{\text{app}})$ in total simulation cases.
[i] UpErrBndR: the value of the $\max[(\text{DSR}_{\text{opt}} - \text{DSR}_{\text{app}})/\text{DSR}_{\text{opt}}]$ in total simulation cases.
[j] ARErrRlnk: the value of average of ARErrR which are in same Lr and T(s).
[k] ARErrRT: the value of average of ARErrR which are in same T(s).
[l] The value is just for reference.

End reverse_traversal_KNR

3.4. An illustrative example

Fig. 2 illustrates the topology of a DS with eight nodes and twelve links. The problem involves determining a subset of the DS that includes some of the nodes v_1, v_2, …, v_7, v_8 whose total capacity is at least as large as thirty-one units.

In step 0, read the DS topology and randomly generate the reliability of each link and the capacity of each node and the capacity constraint, as displayed in Fig. 3.

In Step 1, obtain $w_{\text{fast}}(v_1)$, $w_{\text{fast}}(v_2)$, …, and $w_{\text{fast}}(v_8)$ are 0.742637, 0.989624, 0.904599, 0.989624, 0.990020, 0.990020, 0.910642 and 0.758843, respectively.

In Step 2, obtain the weight of $e_{1,2}$, $e_{1,5}$, …, $e_{7,8}$ are 0.778235, 0.652651, 0.976457, 0.996901, 0.565664, 0.977798, 0.766052, 0.663030, 0.992116, 0.924252, 0.931039 and 0.805808, respectively.

In Step 3, obtain $w_{\text{optimal}}(v_1)$, $w_{\text{optimal}}(v_2)$, …, $w_{\text{optimal}}(v_8)$ are 0.92297, 0.999993, 0.999477, 0.999995, 0.999910, 0.999456, 0.999763 and 0.934563, respectively.

In Step 4, obtain the fitness value of $\text{FT}(v_1)$, $\text{FT}(v_2)$, …, $\text{FT}(v_8)$ are 0.685432, 0.989617, 0.90416, 0.989618, 0.989931, 0.989482, 0.910426 and 0.709186, respectively.

In step 5, $G_k = V = \{v_{1,2}, v_3, v_4, v_5, v_6, v_7, v_8\}$.

In Step 6, obtain the set $X = \{v_i | c(G_k) - c(v_i) \geq C_{\text{need}}\} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$.

In Step 7, obtain the set $Y = \{v_i | v_i \in X \text{ and } G_k - \{v_i\}\}$ is connected without through any node in $V - \{G_k \cup \{v_i\}\} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$.

In Step 8, because the value of $w(v_1)$ is minimum, v_1 is

Table 3
The results obtained using the proposed method for three DS topologies with eight nodes (the mean of notations is described in the footnotes of Table 2; the value of AES is same as Table 2)

T(s)	Lr	c_d	C_D	AES	HitR	ARErrR	UpErrBnd	UpErrBndR	ARErrRlnk	ARErrRT
Ring (n8e8)	0.0–1	4	6	0.239511	90	0.006996	0.002412	0.050938		
		3	4	0.647318	90	0.021723	0.158663	0.21723		
		2	3	0.850519	100	0.0	0.0	0.0	0.009573	
	0.4–1.0	4	6	0.534633	90	0.011021	0.073199	0.110212		
		3	4	0.706684	90	0.006193	0.028025	0.61933		
		2	3	0.914725	90	0.015327	0.011307	0.153269	0.010847	
	0.7–10	4	6	0.811063	90	0.00514	0.041604	0.051405		
		3	4	0.920673	90	0.000636	0.005934	0.006358		
		2	3	0.963919	100	0.0	0.0	0.0	0.001925	0.007448
Bridge (n8e12)(Fig. 3)	0.0–1.0	4	6	0.673502	90	0.001434	0.004985	0.14338		
		3	4	0.803006	90	0.000172	0.001151	0.001724		
		2	3	0.943866	100	0.0	0.0	0.0	0.000535	
	0.4–1.0	4	6	0.895392	80	0.001213	0.007462	0.008212		
		3	4	0.963701	80	0.005122	0.031653	0.033336		
		2	3	0.984607	90	0.001437	0.014128	0.01437	0.002591	
	0.7–1.0	4	6	0.981713	80	0.000306	0.002208	0.002276		
		3	4	0.994233	90	0.000179	0.001785	0.001786		
		2	3	0.997221	80	0.000965	0.009181	0.009189	0.000483	0.001203
Hypercube (n8e12)	0.0–1.0	4	6	0.586086	90	0.000919	0.008046	0.009193		
		3	4	0.744266	90	0.006647	0.021677	0.053438		
		2	3	0.933596	70	0.0	0.0	0.0	0.002522	
	0.4–1.0	4	6	0.898183	80	0.000191	0.001684	0.001913		
		3	4	0.957629	90	0.000237	0.00214	0.002366		
		2	3	0.980899	100	0.0	0.0	0.0	0.000143	
	0.7–1.0	4	6	0.990639	100	0.001158	0.008386	0.008397		
		3	4	0.993888	100	0.0	0.0	0.0		
		2	3	0.998169	100	0.0	0.0	0.0	0.000386	0.001017
Average					90	0.003223	(0.16134)	(0.034849)	0.003223	0.003223

deleted from G_k. Note that G_k is $\{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$. Both the v_2 and v_5 are adjacent to v_1 therefore, the weight of the two nodes is tuned and obtain FT($v_2 = 0.989592$, FT(v_5) = 0.989794 and go back step 6.

In step 6, obtain the set $X = \{v_i | c(G_k) - c(v_i) \geq C_{\text{need}}\} = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$.

In step 7, obtain the set $Y = \{v_i | v_i \in X \text{ and } G_k - \{v_i\}\}$ is connected without through any node in $V - \{G_k \cup \{v_i\}\} = \{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$.

In Step 8, because the value of $w(v_8)$ is minimum, v_8 is deleted from G_k. Note that G_k is $\{v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$. Both of v_4 and v_7 are adjacent to v_8, therefore, we tune the weight of the two nodes is tuned and obtain FT(v_4) = 0.989608, FT(v_7) = 0.909529 and go back step 6.

In step 6, obtain the set $X = \{v_i | c(G_k) - c(v_i) \geq C_{\text{need}}\} = \varnothing$ and go to step 9.

In step 9, obtain $c(v_7) = 4$ which is minimal capacity in G_k. Because $c(G_k) - c(v_7) < 31$, no nodes can be deleted in G_k.

In Step 10, the reliability of a k-node set $\{v_2, v_4, v_5, v_6, v_7\}$ is computed using SYREL. We have $R(\{v_2, v_4, v_5, v_6, v_7\}) = 0.8612462$ which has the maximum reliability under the capacity constraint. The number of reliability computation is one.

The result is the same as a k-node set, which is derived using an exhaustive method.

4. Simulation

The accuracy and efficiency of the proposed algorithm are verified by implementing a C language simulation program executed on a Pentium 100 with 16M-DRAM on MS-Windows 95. We use many network topologies and generated several hundreds of data for simulation. The reliability of each link, the capacity of each node and the total capacity requirement were generated using a random number generator. For verifying the sensitivity of our proposed algorithm, three data categories were given in different ranges. For the link reliability, we considered the following range: 0.0–1.0, 0.4–1.0 and 0.7–1.0. For the capacity of each node in the system, we consider that the quotient of $\max(c(v_i))/\min(c(v_i))$, $i = 1, \ldots, n$, to be not greater than 4, 3 and 2, respectively. For the total capacity requirement, which must be greater than $max(c(v_i))$, we considered the value of $C_{\text{need}}/([\sum_{i=1}^{n} C(v_i)]/n)$ to be not greater than 6, 4 and 3, respectively. When the number of nodes in the topology is very large, such as exceeding 60, obtaining the global optimal solution using the exhaustive method or EM is nearly impossible. For verifying the correctness of the proposed algorithm, when the topology is very large, we utilized a low connective topology.

Table 4
The average *k*-tree reduction method execution time and the proposed method for three DS topologies with eight notes (the mean of the other notations is described in the footnotes of Table 2)

				k-tree reduction method			The proposed method		
T(s)	Lr	c_d	C_D	AT (s)[a]	ATlnk (s)[b]	ATT (s)[c]	AT (s)	ATlnk (s)	ATT (s)
Ring (n8e8)	0.0–1	4	6	0.248			0.016		
		3	4	0.137			0.006		
		2	3	0.140	0.175		0.006	0.0093	
	0.4–1.0	4	6	0.258			0.011		
		3	4	0.226			0.011		
		2	3	0.054	0.152		0.006	0.0093	
	0.7–10	4	6	0.236			0.026		
		3	4	0.166			0.006		
		2	3	0.67	0.183	0.170	0.015	0.0156	0.0114
Bridge (n8e12)(Fig. 3)	0.0–1.0	4	6	0.33			0.05		
		3	4	0.323			0.026		
		2	3	0.207	0.2866		0.006	0.0273	
	0.4–1.0	4	6	0.431			0.057		
		3	4	0.259			0.033		
		2	3	0.187	0.2923		0.012	0.034	
	0.7–1.0	4	6	0.418			0.03		
		3	4	0.286			0.021		
		2	3	0.149	0.2843	0.2877	0.006	0.019	0.0267
Hypercube (n8e12)	0.0–1.0	4	6	0.631			0.364		
		3	4	0.399			0.191		
		2	3	0.126	0.3853		0.099	0.218	
	0.4–1.0	4	6	0.604			0.34		
		3	4	0.351			0.207		
		2	3	0.11	0.355		0.11	0.219	
	0.7–1.0	4	6	0.626			0.376		
		3	4	0.085			0.207		
		2	3	0.11	0.3403	0.3602	0.11	0.2316	0.2226
Average				0.2727	0.2727	0.2727	0.0870	0.0870	0.0870

[a] AT: the seconds of average execution time, AT = (∑(execution time))/(total simulation case).
[b] ATlnk: the seconds of average execution time of same Lr and T(s).
[c] ATT: the seconds of average execution time of same T(s).

5. Results and discussion

Table 1 presents the data on the results obtained using different methods for various DS topologies. In contrast to the EM and the *k*-tree reduction method, the execution time grew rapidly when the DS topology size is increased. The proposed method grew very slowly. The deviation was small when the proposed method could not obtain an

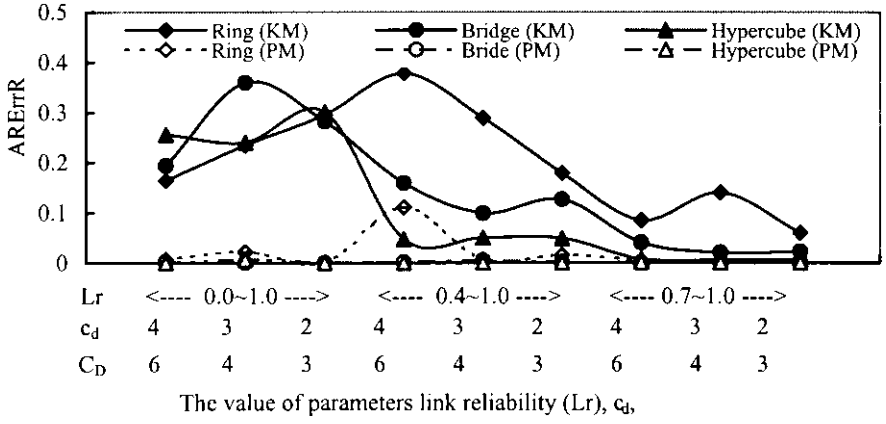


Fig. 4. The average relative error ratio from the average exact solution using *k*-tree reduction method (KM) and the proposed method (PM) on different topologies and parameters consideration.

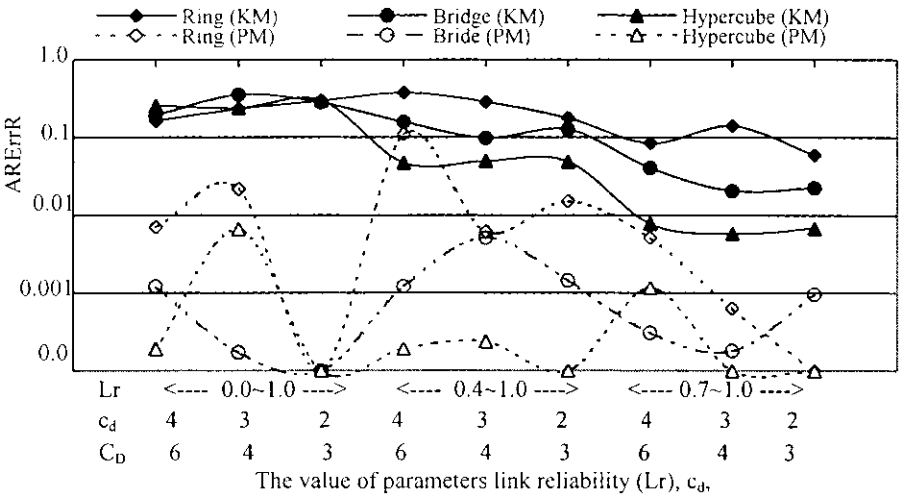


Fig. 5. The average relative error ratio from the average exact solution using *k*-tree reduction method (KM) and the proposed method (PM) on different topologies and parameters consideration. (use logarithm scale).

optimal solution. Tables 2–4 list the results obtained using the *k*-tree reduction and our proposed method for three different topologies (ring, bridge, hyper-cube) with eight nodes, respectively. These data show that the proposed method is more effective than the conventional method.

In contrast to the computer reliability problem, which is static oriented, an optimal *k*-node set with capacity constraint in the DS are dynamic-oriented has many factors such as node capacity, DS topology, link reliability, capacity constraint and the number of paths between each node can significantly affect the efficiency of the algorithm [2,5,17]. Thus, exactly quantifying the time complexity is extremely difficult. The complexity of the EM is $O(2^{(e+n)})$, where e denotes the number of links and n represents the number of processing elements. The complexity of the *k*-tree reduction method is $O(2^e n^2)$.

In our proposed algorithm, in the worst case, the complexity of steps $0, 1, \ldots, 10$ are $O(e+n), O(e+n), O(en), O(e), O(n), O(1), O((n-|k|)n), O((n-|k|)ne), O((n-|k|)n), O(n^2)$ and $O(m^2)$, respectively, where m denotes the number of paths of a selected *k*-node set [15], $|k|$ is the order of a selected set. Therefore, the complexity of the proposed algorithm is $O((n-|k|ne+m^2)$.

There is an investigation on the reliability error bound [18]. In our simulation cases, consequently, we obtained information as follows. According to Tables 2 and 3, the reliability of a *k*-node set was correlated with topology density and negatively correlated with c_d and C_D, therefore high connective DS lead to a high a KNR, low value c_d and C_D possess a high KNR. For example, Fig. 3 demonstrates that the average exact solution for KNR of hyper-cube and bridge topologies is better than that for the ring topology corresponding link reliability. When the DS topology and the link reliability are fixed, the parameters c_d and C_D affect the average exact KNR solution. For example, the average exact KNR solution when c_d is set to four and C_D is assigned to six is worse than when c_d is set to three or two and C_D is assigned to four or three. Without a loss of generality, the ratio of the average relative error was negatively correlated with the link reliability range and the number of links. For

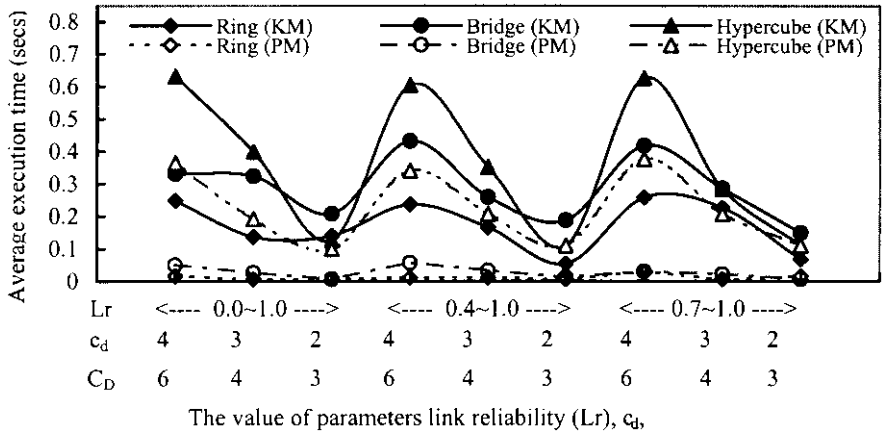


Fig. 6. The average *k*-tree reduction method (KM) execution time and the proposed method (PM) on different parameters consideration.

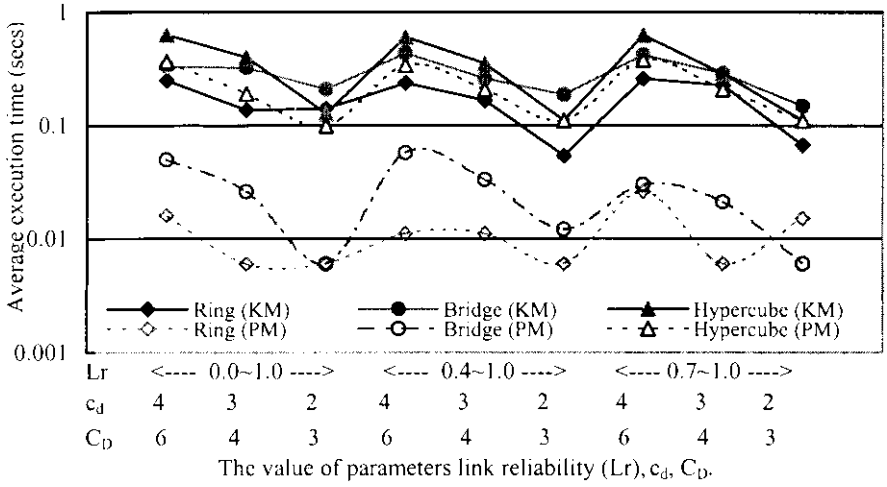


Fig. 7. The average *k*-tree reduction method (KM) execution time and the proposed method (PM) on the different parameters consideration (use logarithm scale).

example, Figs. 4 and 5 display that the average relative error ratio increased as the number of links decreased, that is, the ring topology lead to the highest average relative error ratio. Figs. 4 and 5 reveal that the proposed algorithm can improve the average relative error ratio in various topologies and parameters. In the *k*-tree reduction method, which obtains the exact solution about 9%, the average relative error ratio from average exact solution exceeds 0.1516. The maximal error bound average exceeds 0.280612. In proposed algorithm, the exact solution can be obtained about 90%, the average relative error ratio from exact solution is about 0.0032. The maximal error bound average is about 0.016134. According to Table 4, the execution time was independent of the link reliability, but it increased as the number of links, C_D, c_d increased. Figs. 6 and 7 illustrate the average *k*-tree reduction method execution time and the proposed method. The average execution time is dependent on not only the topology of DS, but also on c_d and C_D. In Fig. 6, the proposed algorithm is more efficient than the *k*-tree reduction method. The average *k*-tree reduction method execution time and the proposed method are 0.2727 and 0.087 s, respectively.

From Figs. 4–7, we observe the relationship between the average execution time and the average relation error ratio. The average execution time is highly affected not only by the parameters of the maximal value of $\max(c(v_i))/\min(c(v_i))$, e.g. c_d, and the capacity constraint, e.g. C_D, which we need, but also by the DS topology. The high c_d and C_D value may increase the average execution time. The average relation error ratio is strongly affected by the DS topology, but the c_d and C_D are not strong.

The proposed method was faster than the *k*-tree reduction method. When the DS is sufficiently large, the proposed algorithm can obtain a much better solution than the *k*-tree reduction method. Moreover, the proposed algorithm can reduce the computation time in a large DS. The proposed algorithm is in the worst case, when the $\max(c(v_i))/\min(c(v_i))$ quotient has a high value. In this case, the deviation from exact solution seems to increase. If the $\max(c(v_i))/\min(c(v_i))$ quotient is equal to one, the proposed method is in the best case.

6. Conclusions

This paper has presented a reversing traversal algorithm for obtaining an approximate KNR with capacity constraint in a DS. The accurate prediction hit ratio for a deleting node is very high and the reliability computation number is just one. Our numeral results show that the proposed algorithm may obtain a *k*-node set that has maximal or nearly maximal KNR in most cases and the computation time seems to be significantly shorter than that needed for the EM and the *k*-tree reduction method. When the proposed method fails to give an exact solution, the deviation from the exact solution appears very small. To sum up, the technique presented in this paper is an efficient means for obtaining an adequate *k*-node set with capacity constraint in a large DS.

References

[1] T. Tsuchiya, Y. Kakuda, T. Kikuno, Three-mode failure model for reliability analysis of distributed programs, IEICE Trans. Inf. Systems E80-D (1) (1997) 3–9.
[2] D.J. Chen, T.H. Huang, Reliability analysis of distributed systems based on a fast reliability algorithm, IEEE Trans. Parallel Distributed Systems 3 (2) (1992) 139–154.
[3] G.J. Hwang, S.S. Tseng, A. heuristic, A heuristic task assignment algorithm to maximize reliability of a distributed system, IEEE Trans. Reliability R-42 (3) (1993) 408–415.
[4] M.A. Aziz, Pathset, enumeration of directed graphs by the set theoretic method, Microelectron Reliab. 37 (5) (1997) 809–814.
[5] A. Kumar, D.P. Agrawal, A. generalized, A generalized algorithm for

evaluation distributed program reliability, IEEE Trans. Reliability 42 (3) (1993) 416–426.
[6] P. Tom, C.R. Murthy, Algorithms for reliability-oriented module allocation in distributed computing systems, J. Systems Software 40 (2) (1998) 125–138.
[7] D. Torreri, Calculation of node-pair reliability in large networks with unreliable nodes, IEEE Trans. Reliability 43 (3) (1994) 375–377.
[8] D.W. Coit, A.E. Smith, Reliability optimization of series — parallel systems using a genetic algorithm, IEEE Trans. Reliability 45 (2) (1996) 254–260.
[9] F. Altiparmak, B. Dengiz, A.E. Smith, Reliability optimization of computer communication networks using genetic algorithms, Proc. IEEE Int. Conf. Syst. Man Cybern. 5 (1998) 4676–4680.
[10] M.S. Lin, D.J. Chen, New reliability evaluation algorithms for distributed computing systems, J. Inf. Sci. Engng 8 (3) (1992).
[11] M.S. Lin, M.S. Chang, D.J. Chen, Efficient algorithms for reliability analysis of distributed computing systems, J. Inf. Sci. Engng 17 (1/2) (1999).
[12] L.C. Hwang, C.J. Chang, Analysis of a general limited scheduling mechanism for a distributed communication systems, Computer Network 31 (18) (1999) 1879–1889.
[13] R.S. Chen, D.J. Chen, Y.S. Yeh, Reliability optimization of distributed computing systems subject to capacity constraint, J. Computers Math. Appl. 29 (4) (2000) 93–99.
[14] R.S. Chen, D.J. Chen, Y.S. Yeh, A. new, heuristic approach for reliability optimization of distributed computing systems subject to capacity constraints, J. Computers Math. Appl. 29 (3) (1995) 37–47.
[15] S. Hariri, C.S. Raghavendra, SYREL: a symbolic reliability algorithm based on path and cuset methods, IEEE Trans. Computers C-36 (10) (1987) 1224–1232.
[16] D.J. Chen, R.S. Chen, T.H. Huang, Heuristic approach to generating file spanning trees fro reliability analysis of distributed computing systems, J. Computers Math. Appl. 34 (10) (1997) 115–131.
[17] M. Kafil, I. Ahmad, Optimal task assignment in heterogeneous distributed computing systems, IEEE Concurrency 6 (3) (1998) 42–51.
[18] F.S. Makri, Z.M. Psillakis, Bound for reliability of k-within connected-(v,s) out-of (m,n) failure systems, Microelectron Reliab. 37 (8) (1997) 1217–1224.

Yi-Shiung Yeh — Education: September 1981–December 1985 PhD in Computer Science, Department of EE & CS, University of Wisconsin-Milwaukee. September 1978–June 1980 MS in Computer Science, Department of EE & CS, University of Wisconsin-Milwaukee. Professional background: August 1988, Now Associate Professor, Institute of CS & IE, National Chiao-Tung University. July 1986–August 1988 Assistant Professor, Department of Computer & Information Science, Fordham University. July 1984–December 1984 Doctorate Intern, Johnson Controls, Inc. August 1980–October 1981 System Programmer, System Support Division, Milwaukee County Government Research interest: Data security & Privacy, Information & Coding Theory, Game Theory, Reliability and Performance.

Chin-Ching Chiu — Education: He is currently working towards his PhD degree in the Department of CS & IE, National Chiao-Tung University. September 1988–June 1991 MS in Computer Science, Department of EE & IE, Tamkang University. He received a BS degree in Computer Science from Soochow University. Professional background: September 1991 Present Instructor, Department of Information Management, Tak-Mi College. April 1984–August 1991 Computer Engineer in Data Communication Institute of Directorate General of Telecommunications. October 1982–April 1984 System Analyst of SYSCOM computer company. Research interest: Reliability Analysis of Network and Distributed System.