Malaysian Journal of Computer Science, Vol. 15 No. 1, June 2002, pp. 45-56

OPTIMAL K-NODE SET RELIABILITY WITH CAPACITY CONSTRAINT OF DISTRIBUTED SYSTEMS, VIA A HEURISTIC ALGORITHM

Chin-Ching Chiu
Department of Management Information System
Private Takming College, Taipei, Taiwan, R.O.C.
Tel. : 886-2-26585801 ext 393
email: chiu@imd.takming.edu.tw

Yi-Shiung Yeh
Institute of Computer Science and Information Engineering
National Chiao-Tung University, Hsinchu, Taiwan R.O.C.

ABSTRACT

In this work, we present a heuristic method to reduce the computational time and the absolute error from the exact solution for obtaining a k-node set with capacity constraint. This method uses an efficient objective function to select an adequate node when deriving the k-node set process. Reliability computation is performed only once, thereby spending less time to compute the reliability. Moreover, the absolute error of the proposed algorithm from exact solution is smaller than that of k-tree reduction method. Computational results demonstrate that the proposed algorithm is a more efficient solution for a large distributed system then conventional ones.

Keywords: Heuristics; Distributed systems; Reliability optimisation

1.0 INTRODUCTION

The network reliability problem with respect to a network with a general structure is NP-hard [1, 2]. Efficient algorithms easily implemented on a computer are needed to analyse the reliability of large networks. In addition, such algorithms should yield good approximations of the reliability when the networks are so large that the computational time becomes prohibitive.

The topology of a network can be characterised by a linear graph. These network topologies can be characterised by their network reliability, message-delay, or network capacity. These performance characteristics depend on many properties of linear graphs that represent the network topology [3-7]: the number of ports at each node (degree of a node), and the number of links. Notably, the number of links directly impacts the system reliability.

This work largely focuses on how to compute nearly maximum system reliability subject to the capacity constraint. In the *k*-tree reduction method [8], the starting node is the first node v_1. To select other adequate nodes in a sequential manner depends on the maximum product of reliability by capacity of the *k*-node set with another node until the capacity constraint is satisfied. The number of reliability computation is still large. In addition, the above product heavily relies on the total capacity of each node but only slightly depends on the *k*-node set reliability; therefore, it barely derives the optimal solution.

In light of the above discussion, this work presents a heuristic algorithm by carefully selecting the starting node according to a node's weight. Before assigning a node to the selected set, the proposed algorithm probes those nodes that are adjacent to any node of a selected node except for the selected nodes. After obtaining the *k*-node set, SYREL [9] is applied to compute the reliability. For a large distributed system (DS) on various DS topologies, our results demonstrate that the proposed algorithm is more reliable and efficient than conventional algorithms, the exact method (EM) [10] and the *k*-tree reduction method [8] in terms of execution time.

2.0 PROBLEM DESCRIPTION

In this section, we describe the problem addressed herein to clarify our research objectives.

2.1 Notations and Definitions

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, $n = \|V\|$.
		$c(G_k)$	the sum of capacity of k-node set of a DS graph G.
v_i	the i^{th} processing element or the i^{th} node.	$w(G_k)$	the weight of G_k obtained by the object function.
$c(v_i)$	the capacity of the i^{th} node.	$d(v_i)$	the number of links connected to the node v_i.
e	the number of links in G, $e = \|E\|$.	$w(v_i)$	the weight of the i^{th} node.
$e_{i,j}$	an edge represents a communication link between v_i and v_j.	$w(e_{i,j})$	the weight of the link $e_{i,j}$.
		$V_{adj(G_k)}$	a set of nodes which are adjacent to any node of G_k.
$p_{i,j}$	the probability of success of link $e_{i,j}$.		
$q_{i,j}$	the probability of failure of link $e_{i,j}$.	V_{G_k}	a set of nodes of G_k.
P	the link reliability matrix where	n_{G_k}	the number of nodes of V_{G_k}.
	$P(i,j) = P(j,i) = p_{i,j}$, if $e_{i,j}$ exists in G,	$w(V_{G_k})$	$\sum_{j=0}^{n_{G_k}} w(v_j)$, where $v_j \in V_{G_k}$.
	$P(i,j) = P(j,i) = 0$, otherwise	E_{G_k}	a set of direct links between any two nodes in G_k.
	for $i, j = 1, 2, \ldots, n$.	e_{G_k}	the number of links of E_{G_k}.
C_{limit}	total capacity constraint in a DS.	$w(E_{G_k})$	$\sum w(e_{i,j})$, where $e_{i,j} \in E_{G_k}$.
G_k	the graph G with the set K of nodes specified, and $\|K\| \geq 2$.	v_s	a starting node for deriving a k-node set.
		$\varepsilon_{s,j}$	the direct link $e_{s,j}$ does not exist, but there are at least two paths whose length is two between v_s and v_j.
$R(G_k)$	the reliability of k-node set solution of a DS graph G.		
$y_{i,j}$	the number of paths whose length is two between v_i and v_j.	$V_{\varepsilon(G_k)}$	a set of nodes which $\varepsilon_{s,j}$ exist in G_k.
		$E_{\varepsilon(G_k)}$	a set of $\varepsilon_{s,j}$.
C_D	a value of $C_{limit} / ([\sum_{i=1}^{n} c(v_i)]/n)$ for tuning the rang of capacity constraint , i.e., $C_{limit} = [(\sum_{i=1}^{n} c(v_i))/n] \times C_D$	$w(\varepsilon_{s,j})$	the weight of $\varepsilon_{s,j}$.
		c_d	the maximal value of max($c(v_i)$) / *min*($c(v_i)$) for tuning the range of each node's capacity.

Definitions

Definition 1. A k-node set reliability (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 node v_i is directly connected to a set V_{Gk} of nodes if and only if there is a link between v_i and a node in V_{Gk}.

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

Definition 4. An number of reliability computations (NRC) is the number of computations of a k-node set reliability $R(G_k)$ that the total capacity of G_k are sufficient the capacity constraint.

Definition 5. Absolute error is defined as the value of subtracting an approximate solution from an exact solution of KNR.

Definition 6. Relative error is defined as the value of dividing an exact solution into the absolute error.

Definition 7. 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 Statement

Bi-directional communication channels operate between processing elements. A distributed system can be modeled by a simple undirected graph. A *k*-node set reliability can be obtained using the sum of mutually disjoint events [11].

A set, *K*, of nodes can be derived from the given set *V* that constitutes a DS in that *k*-node set reliability is adequate and the total capacity satisfies the capacity constraint. The main problem can be mathematically stated as follows:

Object: Maximize $R(G_k)$

subject to: $\sum_{v_i \in G_k} c(v_i) \geq C_{\lim it}$

where $R(G_k)$, $c(v_i)$, C_{limit} are defined in section 2.1.

Obviously, the problem for a large DS, as in a metropolitan area network, requires a large execution time. Herein, we develop an efficient method that allows the *k*-node set reliability optimisation in the DS to achieve the desired performance. Owing to its computational advantages, the proposed method may be preferred to the EM and the *k*-tree reduction method when the DS is large.

3.0 HEURISTIC ALGORITHM FOR K-NODE SET RELIABILITY

In this section, we present a heuristic algorithm to maximise system reliability. The analyses performed herein assumes that all of the nodes are perfect and the links are unreliable.

3.1 The Concept of Proposed Algorithm

As generally known, the EM [6] spends long execution time in a large DS. The EM, an optimal solution, cannot effectively reduce the problem space. Occasionally, an application requires an efficient algorithm to compute the reliability due to its resource considerations. Under this circumstance, achieving optimal reliability may not be desirable. Instead, an efficient algorithm with an approximate reliability computation algorithm is highly attractive. The topologies of most DS are large and an increasing number of nodes causes the execution time for a solution to exponentially grow. Although capable of reducing computational time, the *k*-tree reduction method barely derives the optimal solution. Therefore, this work presents an algorithm capable of reducing the total execution time to achieve the sub-optimal KNR of DS.

Consider a DS with *n* nodes and *e* links. The capacity constraint is C_{limit}, 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 exceeds the capacity constraint C_{limit}.

The reliability of a set of selected nodes depends on their links and the link reliability. For any node, the degree of that node affects the number of paths of the information that can be transferred from others' nodes. Therefore, in this work, we employed a simple means of computing the node weight, 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.

$$w(v_i) = 1 - \prod_{z=1}^{d(v_i)} q_i k_z \tag{1}$$

The above formula is easily programmed 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 subtraction and $d(v_i)$ multiplication. Thus, we can obtain the weight of every node in *n* subtractions and $2e$ multiplication.

In the network, two nodes may contain many paths between them. A path's length is between one and n-1. To reduce the computational time, we consider the path in which the length is not greater than two. The following formula is used to evaluate the weight of link $e_{i,j}$. Where $y_{i,j}$ denotes the number in which the length of a path between v_i are v_j is two. In addition, $y_{i,j}$ is not greater than n-2. The weight of $e_{i,j}$ can be computed in one subtraction and $2(y_{i,j}+1)$ multiplication. Thus, in the worst case, when the graph is a complete graph, we can obtain all of the weights of each link in $n\times(n\text{-}1)/2$ subtractions and $n\times(n\text{-}1)\times(n\text{-}2)/2$ multiplication.

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

In the same manner, if no direct link exits between v_i and v_j, the following formula is used to evaluate the weight of $\varepsilon_{i,j}$ whose path's length is two.

$$w(\varepsilon_{i,j}) = 1 - \prod_{z=1}^{y_{i,j}} (q_{i,k_z} p_{k_z,j}) \tag{3}$$

The following observations can be made on how to reduce the order of a k-node set. For a given selected k-node set, the reliability of this k-node set is not greater than the reliability of its subset. Thus, during the reliability evaluation process, if the total capacity of the subset of the k-node set satisfies the capacity constraint, the k-node set should be replaced by its subset.

Assume not only that we have a selected set G_k of nodes with reliability $R(G_k)$, but also that the nodes in G_k are all directly connected. If another set G'_k of nodes exists in which just one node is different from G_k and G'_k has one node which is not directly connected with other nodes in G'_k, then we say that $R(G_k) \geq R(G'_k)$ for $R(G'_k)$ the reliability of set G'_k.

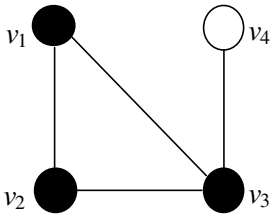


Fig. 1: G_k ={ v_1, v_2, v_3}

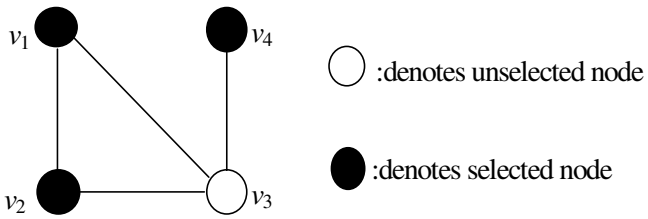


Fig. 2: G'_k={ v_1, v_2, v_4}

By assuming that the 2-terminal reliability between v_1 and v_2 is R_1, this relation can be represented as $R(\{v_1, v_2\}) = R_1$, and $R(\{v_1, v_3\}) = R_2$, $R(\{v_2, v_3\}) = R_3$, $R(\{v_3, v_4\}) = R_4$. In Fig. 1,we select nodes v_1, v_2 and v_3. Therefore, $R(\{v_1, v_2\}) = R_1$, $R(\{v_1, v_3\}) = R_2$, $R(\{v_2, v_3\}) = R_3$. According to Fig. 2, we select nodes v_1, v_2 and v_4. Therefore, $R(\{v_1, v_2\}) = R_1$, $R(\{v_1, v_4\}) = R_2\times R_4$ and $R(\{v_2, v_4\}) = R_3\times R_4$. Because $R_2 \leq 1$, $R_3 \leq 1$ and $R_4 \leq 1$, $R_2\times R_4 \leq R_2$ and $R_3\times R_4 \leq R_3$, the reliability of node v_1, v_2, $v_3 \geq$ the reliability of node v_1, v_2, v_4. Restated, $R(\{v_1, v_2, v_3\}) \geq R(\{v_1, v_2, v_4\})$. However, this assumption is not always true if (a) a path exists between v_4 and v_1 or between v_4 and v_2, and (b) the reliability of the path is larger than the reliability between v_3 and v_1 and between v_3 and v_2. For this reason, in some cases, the maximum reliability cannot be achieved using the proposed method.

This assumption is true if the reliability of any path between X and K is less than that of links between the set K of nodes. Restated, the proposed method can be used to achieve maximum reliability.

In each set of nodes, if the number of members of a set is k, the following formula can be used to compute its weight value.

$$w(G_k) = \{ [w(E_{G_k}) + w(\varepsilon_{s,j})] / [k(k\text{-}1)/2] + w(V_{G_k}) / [(n\text{-}1)\times k] \} \Big/ \sqrt{k+2} \tag{4}$$

According to $w(E_{G_k})$ and $w(V_{G_k})$ in formula (4), only the sum of the weight of the links between v_i and G_k and the weight of node v_i should be derived. Therefore, the weight of the k-node set with another node, say v_i, can be obtained easily and efficiently using the following formula (5).

$$w(G_k \bigcup \{v_i\}) = \{ [w(E_{G_k}) + w(\varepsilon_{s,j}) + \sum_{v_j \in V_{G_k}, v_i \notin V_{G_k}, e_{i,j} \notin E_{G_k}} w(e_{i,j})] / [((k+1) k) / 2] + [w(V_{G_k}) + w(v_i)] / [(n-1) \times (k+1)] \} \Big/ \sqrt{k+3} \quad (5)$$

The function of divisor $\sqrt{k+2}$ in formula (4) is to ensure that the weight of a k-node set is less than the weight of its subsets. Because the k-node set is appended in a sequential manner, the divisor $\sqrt{k+2}$ can be omitted without making a mistake. Therefore, we use formula (6) instead of formula (4).

$$w(G_k) = \{ [w(E_{G_k}) + w(\varepsilon_{s,j})] / [k(k-1)/2] + w(V_{G_k}) / [(n-1) \times k] \} \quad (6)$$

Same as the function of divisor $\sqrt{k+3}$ in formula (5). we use formula (7) instead of formula (5).

$$w(G_k \bigcup \{v_i\}) = \{ [w(E_{G_k}) + w(\varepsilon_{s,j}) + \sum_{v_j \in V_{G_k}, v_i \notin V_{G_k}, e_{i,j} \notin E_{G_k}} w(e_{i,j})] / [((k+1)k)/2] + [w(V_{G_k}) + w(v_i)] / [(n-1) \times (k+1)] \} \quad (7)$$

Herein, a node of the heaviest weight is selected and serves as the starting node for deriving an adequate k-node set. Before assigning one node to a selected set, the k-tree reduction method must probe all nodes except for the selected nodes. To reduce computation time, the proposed algorithm only inspects those nodes that are adjacent to any node of the selected node. In the first node assigned to a selected set, the proposed algorithm also probes nodes in $V_{\varepsilon(G_k)}$. In addition, the proposed algorithm selects an adequate node according to maximum $[w(G_k \bigcup \{v_i\})]$ instead of maximum $[R(G_k) \times \sum_{v_i \in G_k} c(v_i)]$ of the k-tree reduction method. The limitation of the latter expression is that the product is more sensitive by the total capacity than the k-node set reliability. Therefore, it barely obtain the optimal solution.

3.2 The Proposed Heuristic Algorithm

In the following, we present a heuristic algorithm to maximise K-node reliability optimal design of a DS under capacity constraint.

Algorithm KNR

Step 0. Initialise, read system parameters: n, e, C_{limit}, P, $c(v_i)$, $i = 1, \ldots, n$.

Step 1. Evaluate the weight of each node using formula (1) and choose the heaviest one as the starting node, say v_s, for deriving an adequate k-node set. Notably, G_k is initialised to $\{v_s\}$.

Step 2. Find each $\varepsilon_{s,j}$ and insert it into $E_{\varepsilon(G_k)}$.

Step 3. Evaluate the weight of each link using formula (2).
Evaluate the weight of each $\varepsilon_{s,j}$ in $E_{\varepsilon(G_k)}$ using formula (3).

Step 4. Let $V_{tmp} = V_{\varepsilon(G_k)}$. /* V_{tmp} denotes a set of nodes */
Let $w(V_{G_k}) = w(v_s)$.
Let $w(E_{G_k}) = 0$.

Step 5. Find v_i, in $(V_{adj(Gk)} \bigcup V_{tmp})$, so that $c(G_k) + c(v_i) \geq C_{limit}$.

Step 6. Switch v_i
case 1: No such v_i found
/* find an adequate v_i after evaluating each $w(G_k \bigcup \{v_i\})$ by formula (7)*/
Find v_i, such that $w(G_k \bigcup \{v_i\}) = \max\{w(G_k \bigcup \{v_i\}) \mid v_i \in (V_{adj(Gk)} \bigcup V_{tmp})\}$.
Let $G_k = G_k \bigcup \{v_i\}$.

Let $w(V_{G_k}) = w(V_{G_k}) + w(v_i)$.

Let $w(E_{G_k}) = w(E_{G_k}) + \sum_{v_j \in V_{G_k}, v_i \notin V_{G_k}, e_{i,j} \notin E_{G_k}} w(e_{i,j})$.

Let $V_{tmp} = \varnothing$.

$\forall\ \varepsilon_{s,j} \in E_{\varepsilon(G_k)}$, let $w(\varepsilon_{s,j}) = 0$.

Go to step 5.

case 2: Exact one v_i found

Let $G_k = G_k \bigcup \{v_i\}$.

Break.

case 3: Many v_i found

/* find an adequate v_i after evaluating each $w(G_k \bigcup \{v_i\})$ using formula (7)*/

Let $N = \{v_i \mid c(G_k) + c(v_i) \geq C_{limit}\} \subseteq (V_{adj(Gk)} \bigcup V_{tmp})$. /* N is a set of nodes*/

Find v_i, such that $w(G_k \bigcup \{v_i\}) = \max\{w(G_k \bigcup \{v_j\}) \mid v_j \in N\}$.

Let $G_k = G_k \bigcup \{v_i\}$.

Break.

Step 7. /* If the total capacity of a subset of the k-node set satisfies capacity constraint, the k-node set is replaced by the subset. */

Find v_i, such that $c(v_i) = \min\{c(v_i) \mid v_i \in G_k\}$.

Dowhile $((c(G_k) - C_{limit}) > c(v_i))$

Let $G_k = G_k - \{v_i\}$./*discard v_i from G_k*/

Let $c(G_k) = c(G_k) - c(v_i)$.

Find v_i, such that $c(v_i) = \min\{c(v_i) \mid v_i \in G_k\}$.

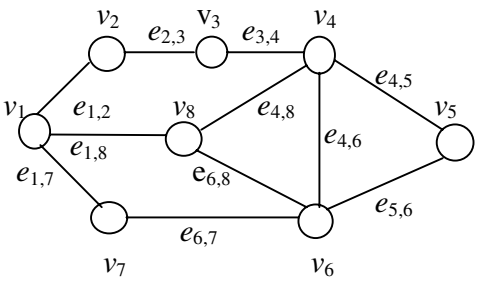
end dowhile

Step 8. Compute $R(G_k)$ using SYREL, output the k-node set G_k and its reliability.

End KNR

3.3 Illustrative Example

Fig. 3 illustrates the topology of a DS with eight nodes and eleven links. The problem involves determining a subset of the DS which includes some of the nodes v_1, v_2, …, v_8 whose total capacity exceeds the capacity constraints of one hundred units.



$c(v_1)$=39 $c(v_2)$=45 $c(v_3)$=38 $c(v_4)$=53
$c(v_5)$=47 $c(v_6)$=49 $c(v_7)$=51 $c(v_8)$=41

$p_{1,2}$=0.89 $p_{1,7}$=0.81 $p_{1,8}$=0.93 $p_{2,3}$=0.85
$p_{2,4}$=0.91 $p_{4,5}$=0.82 $p_{4,6}$=0.83 $p_{4,8}$=0.96
$p_{5,6}$=0.87 $p_{6,7}$=0.84 $p_{6,8}$=0.88

$C_{limit} \geq 100$

Fig. 3: The DS with eight nodes and eleven links

In step 1, each node's weight is evaluated using formula (1). The weight of v_1, v_2, …, and v_8 are 0.998537, 0.9835, 0.9865, 0.9998898, 0.9766, 0.9995756, 0.9696 and 0.999664, respectively. Therefore, v_4 is the node with maximal weight and is served as starting node for obtaining an adequate k-node set. Notably, G_k is $\{v_4\}$.

In step 2, because $\varepsilon_{s,j}$ does not exist, set $E_{\varepsilon(G_k)}$ to empty.

In step 3, each link's weight is evaluated using formula (2).

In step 4, let $V_{tmp} = V_{\varepsilon(G_k)} = \varnothing$, $w(V_{G_k}) = w(v_s)$, $w(E_{G_k}) = 0$.

In step 5, for the set of nodes, $(V_{adj(Gk)} \bigcup V_{tmp})=\{v_3, v_5, v_6, v_8\}$, find v_i, in $(V_{adj(Gk)} \bigcup V_{tmp})$. By doing so, $c(G_k) + c(v_i) \geq C_{limit}$. Consequently, $c(v_4) + c(v_5) \geq 100$ and $c(v_4) + c(v_6) \geq 100$ are obtained.
In step 6, switch v_i is described in step 5. Because many v_i are found, case 3 is executed. The set of nodes, $N = \{v_5, v_6\}$ satisfy $\{v_i \mid c(G_k) + c(v_i) \geq C_{limit}\} \subseteq (V_{adj(Gk)} \bigcup V_{tmp})$. Using formula (7) to evaluate weight, we have $w(\{v_4, v_5\}) = 0.5455779$ and $w(\{v_4, v_6\}) = 0.5491064$, respectively. Therefore, v_6 is selected and $G_k = G_k \bigcup \{v_6\}$.
In step 7, because $c(G_k) = 102$, $C_{limit} = 100$, $c(G_k) - C_{limit} = 2 < 49 = c(v_6) = \min\{c(v_i) \mid v_i \in G_k\}$, the k-node set $\{v_4, v_6\}$ can not be replaced by its set.
In step 8, the reliability of the k-node set $\{v_4, v_6\}$ is computed using SYREL. We have $R(\{v_4, v_6\}) = 0.9974378$ which has the maximum reliability under the capacity constraint. The number of reliability computation is exactly one.
The result is the same as in the k-node set, which is derived by an exhaustive method.

4.0 COMPARISON AND DISCUSSION

Results obtained from our algorithm were compared with those of EM and k-tree reduction method. Although capable of yielding the optimal solution, conventional techniques such as EM cannot effectively reduce the reliability count. An application occasionally requires an efficient algorithm to compute reliability owing to resource considerations. Under this circumstance, deriving the optimal reliability may not be feasible. Instead, an efficient algorithm yielding approximate reliability is preferred. Although the k-tree reduction method can reduce computational time in a moderate DS, the error from an exact solution is relatively high.

In contrast to the computer reliability problem, which is static-oriented, the KNR problems in the DS are dynamic-oriented since many factors, e.g. node capacity, DS topology, link reliability, and the number of paths between each node, can significantly affect the efficiency of the algorithm [12]. Thus, quantifying the time complexity exactly is extremely difficult. Next, the accuracy and efficiency of the proposed algorithm are verified by implementing simulation programs in C language that are 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.

Table 1: Comparison with other methods

Size		**Exhaustive Method**			**EM**	**KM**		**Proposed Method**	
n	***e***	**Max_Rel**	***k*-node set**	**NRC**	**NRC**	**NRC**	**err**	**NRC**	**err**
5	6	0.9462500	1,2,3	32	10	7	0.0122881	1	0
6	8	0.9383065	4,5,6	64	15	9	0.0444220	1	0
6	9	0.9950069	1,3,5	64	20	9	0.0280687	1	0
7	8	0.9187206	1,2,4	128	35	11	0	1	0
7	11	0.9967785	1,2,3	128	35	11	0.0200188	1	0
8	10	0.9894613	6,7	256	45	13	0.0390236	1	0
8	11	0.9974378	4,6	256	44	13	0.0361958	1	0
10	13	0.9347952	1,7,8,9,10	1024	255	24	0.2155169	1	0
10	17	0.9994068	2,8,9	1024	119	17	0.0074441	1	0
10	19	0.9995282	1,5,6	1024	150	17	0.0019527	1	0
11	17	0.9974023	1,10,11	2048	135	19	0.0120129	1	0
12	18	0.9858263	3,4,5,6	4096	538	30	0.0299250	1	0
12	21	0.9990777	1,3,5,6	4096	537	30	0.0056617	1	0.0006069
13	20	0.9978402	4,6	8192	246	45	0.0189070	1	0
19	31	0.9979870	6,8,9	524288	2369	37	0.0856661	1	0
Average					303.53	19.46	0.0371401	1	0.0000404

n: the number of nodes in G, $n = |V|$
e: the number of links in G, $e = |E|$
NRC: the number of reliability computation
k-node set: the nodes we selected
Max_Rel: maximum reliability satisfies our constraints
KM: k-tree reduction method

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

T(s)	Lr	c_d	C_D	AES	HitR	ARErrR	UpErrBnd	UpErrBndR	ARErrRlnk	ARErrRT
Ring (n8e8)	0.0~1.0	4	6	0.354597	20	0.334339	0.288172	0.744694		
		3	4	0.444389	40	0.344601	0.610110	0.918607		
		2	3	0.797079	20	0.206886	0.606823	0.663967	0.295275	
	0.5~1.0	4	6	0.663204	30	0.218068	0.375427	0.422088		
		3	4	0.715659	10	0.260751	0.350206	0.328726		
		2	3	0.950087	40	0.139155	0.467417	0.547818	0.205991	
	0.8~1.0	4	6	0.885318	20	0.051238	0.087854	0.100641		
		3	4	0.961860	10	0.067056	0.185605	0.188950		
		2	3	0.983184	10	0.044877	0.083906	0.085232	0.054390	0.185219
Bridge (n8e12)(Fig. 3.)	0.0~1.0	4	6	0.488085	0	0.227047	0.337566	0.588117		
		3	4	0.789343	10	0.295517	0.473265	0.514837		
		2	3	0.921606	20	0.281299	0.513867	0.582140	0.267954	
	0.5~1.0	4	6	0.957204	0	0.063099	0.124063	0.130684		
		3	4	0.961134	0	0.095304	0.185281	0.186788		
		2	3	0.993096	40	0.024850	0.075325	0.076523	0.061084	
	0.8~1.0	4	6	0.992293	30	0.007936	0.012622	0.012670		
		3	4	0.998827	0	0.024049	0.026125	0.026133		
		2	3	0.998632	0	0.003233	0.003307	0.003321	0.011739	0.113593
Hypercube (n8e12)	0.0~1.0	4	6	0.542984	0	0.155424	0.526773	0.233754		
		3	4	0.824934	0	0.375674	0.490780	0.643362		
		2	3	0.856004	0	0.123988	0.139418	0.185177	0.218362	
	0.5~1.0	4	6	0.930334	0	0.042077	0.058103	0.063987		
		3	4	0.960704	0	0.044191	0.060399	0.064188		
		2	3	0.993787	0	0.035937	0.059887	0.059991	0.040735	
	0.8~1.0	4	6	0.996169	30	0.000639	0.002053	0.002058		
		3	4	0.998619	30	0.000473	0.001379	0.001381		
		2	3	0.999448	0	0.000464	0.000735	0.000735	0.000525	0.086541
Average					13.3	0.128451	(0.227647)	(0.273431)	0.128451	0.128451

T(s): represents the topology (size) of a DS.
Lr : the range of link's reliability, the reliability is obtained by random generator.
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 : the value of $C_{limit}/([\Sigma_{i=1}^{n} c(v_i)]/n)$ for tuning the rang of capacity constraint, i.e.

$C_{limit} = [(\Sigma_{i=1}^{n} c(v_i))/n] \times C_D$

AES : the value of average exact solution whose value is $[\Sigma(DSR_{opt}]/(total\,simulation\,cases)$.
HitR : the ratio of obtaining exact solution.
ARErrR : the value of (Σ[1-(DSR_{app} / DSR_{opt})])/(total simulation cases).
UpErrBnd : the upper error bound whose value is the max(DSR_{opt}-DSR_{app}) in total simulation cases.
UpErrBndR : the value of the max[(DSR_{opt}-DSR_{app})/ DSR_{opt}] in total simulation cases.
ARErrRlnk : the value of average of ARErrR which are in same Lr and T(s).
ARErrRT : the value of average of ARErrR which are in same T(s).
(.) : denotes the value is just for reference
DSR_{app} : approximation solution which obtained by running heuristic algorithm
DSR_{opt} : optimal solution which obtained by running exhaustive search algorithm

Table 3: The results obtained by the proposed method for three DS topologies with eight nodes

T(s)	Lr	c_d	C_D	AES	HitR	ARErrR	UpErrBnd	UpErrBndR	ARErrRlnk	ARErrRT
Ring (n8e8)	0.0~1.0	4	6	0.354597	80	0.035301	0.030782	0.272066		
		3	4	0.444389	70	0.016145	0.058998	0.015123		
		2	3	0.797079	80	0.000818	0.004899	0.006532	0.017421	
	0.5~1.0	4	6	0.663204	90	0.010678	0.045114	0.106777		
		3	4	0.715659	60	0.012166	0.036455	0.055116		
		2	3	0.950087	100	0.0	0.0	0.0	0.007615	
	0.8~1.0	4	6	0.885318	30	0.014791	0.037574	0.033226		
		3	4	0.961860	90	0.005105	0.049394	0.051054		
		2	3	0.983184	100	0.0	0.0	0.0	0.006632	0.010556
Bridge (n8e12)(Fig. 3.)	0.0~1.0	4	6	0.488085	60	0.018571	0.025672	0.068629		
		3	4	0.789343	80	0.002145	0.016444	0.018691		
		2	3	0.921606	80	0.005562	0.027008	0.030916	0.008759	
	0.5~1.0	4	6	0.957204	70	0.006989	0.029705	0.030067		
		3	4	0.961134	30	0.005784	0.015228	0.016305		
		2	3	0.993096	100	0.0	0.0	0.0	0.004258	
	0.8~1.0	4	6	0.992293	70	0.003717	0.012292	0.012391		
		3	4	0.998827	40	0.000954	0.001759	0.001763		
		2	3	0.998632	100	0.0	0.0	0.0	0.001557	0.004858
Hypercube (n8e12)	0.0~1.0	4	6	0.542984	60	0.045530	0.087907	0.138565		
		3	4	0.824934	70	0.005258	0.017527	0.019959		
		2	3	0.856004	100	0.0	0.0	0.0	0.016929	
	0.5~1.0	4	6	0.930334	60	0.005276	0.023076	0.024289		
		3	4	0.960704	70	0.019345	0.064484	0.074382		
		2	3	0.993787	70	0.004912	0.016157	0.016374	0.009844	
	0.8~1.0	4	6	0.996169	40	0.000493	0.000917	0.000919		
		3	4	0.998619	40	0.000394	0.000895	0.000896		
		2	3	0.999448	100	0.0	0.0	0.0	0.000296	0.009023
Average					71.8	0.008146	(0.022307)	(0.042274)	0.008146	0.008146

The mean of notations is described in footnote of Table 2.
The value of AES is same as Table 2.

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, …, 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 $C_{\lim t}/([\Sigma_{i=1}^{n} c(v_i)]/n)$ to be not greater than 6, 4 and 3, respectively. 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 number of reliability computations grew rapidly when the DS topology size is increased. Tables 2, 3 and 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. 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 loss of generality, the ratio of the average relative error was negatively correlated with the link reliability range and the number of links. The complexity of EM is $O(2^e \times 2^n)$ [10], where e denotes the number of edges and n represents the number of nodes. The complexity of the *k*-tree reduction method is $O(2^e \times n^2)$ [8]. In the proposed algorithm, in the worst case, the complexity of evaluating the weight of each node is $O(e)$ and each link is $O(e \times n)$, selecting an adequate *k*-node set is $O(n^3)$, and computing the reliability of the *k*-node set using SYREL is $O(m^2)$ [9].

Table 4: The average *k*-tree reduction method execution time and the proposed method for three DS topologies with eight nodes.

				k-tree reduction metohd			the proposed method		
T(s)	Lr	c_d	C_D	AT(sec)	ATlnk(sec)	ATT(sec)	AT(sec)	ATlnk(sec)	ATT(sec)
Ring (n8e8)	0.0~1.0	4	6	0.189			0.033		
		3	4	0.098			0.027		
		2	3	0.058	0.1154		0.017	0.0256	
	0.5~1.0	4	6	0.140			0.027		
		3	4	0.098			0.055		
		2	3	0.049	0.0961		0.011	0.0311	
	0.8~1.0	4	6	0.124			0.044		
		3	4	0.098			0.038		
		2	3	0.098	0.1071	0.1062	0.027	0.0366	0.0311
Bridge (n8e11)(Fig. 3.)	0.0~1.0	4	6	0.115			0.055		
		3	4	0.115			0.038		
		2	3	0.041	0.0906		0.022	0.0384	
	0.5~1.0	4	6	0.107			0.033		
		3	4	0.115			0.022		
		2	3	0.065	0.0962		0.027	0.0274	
	0.8~1.0	4	6	0.123			0.038		
		3	4	0.091			0.044		
		2	3	0.082	0.0989	0.0952	0.027	0.0366	0.0342
Hypercube (n8e12)	0.0~1.0	4	6	0.412			0.098		
		3	4	0.247			0.061		
		2	3	0.223	0.2939		0.076	0.0787	
	0.5~1.0	4	6	0.395			0.121		
		3	4	0.263			0.098		
		2	3	0.148	0.2692		0.055	0.0915	
	0.8~1.0	4	6	0.330			0.109		
		3	4	0.272			0.071		
		2	3	0.247	0.2830	0.2820	0.066	0.0824	0.0842
	Average			0.1612	0.1612	0.1612	0.0498	0.0498	0.0498

AT : the seconds of average execution time,
AT = (Σ (execution time)) / (total simulation time.
ATlnk : the seconds of average execution time of same Lr and T(s).
ATT : the seconds of average execution time of same T(s).
The mean of other notations is described in footnote of Table 2.

Therefore, the complexity of the proposed algorithm is max($O(n^3)$, $O(m^2)$), where m represents the number of paths of a selected k-node set [9]. In the k-tree reduction method, which obtains the exact solution below 13.3%, the average error from exact solution surpasses 0.12. In our simulation case, the reliability count for the proposed algorithm is exactly one. The exact solution can be obtained above 71.85%, in which the average error from exact solution is under 0.008. In a few cases, an adequate node which has arrived for selected node set through many paths and the length of a great number of those paths exceeds two, the node may be lost when using our formula for computing link's weight. Notably, the proposed algorithm cannot obtain the exact solution.

5.0 CONCLUSIONS

DS provides a cost-effective means of enhancing a computer system's performance in areas such as throughput, fault-tolerance, and reliability optimisation. Consequently, the reliability optimisation of a DS has become a critical issue. When some data files are allocated into DS, a specified set, K, of nodes in a DS must be selected to allocate the data files such that k-node set reliability is adequate under constraints.

Computing DS reliability is generally NP-hard. Because the *k*-tree reduction method derives a *k*-node set reliability according to the product of reliability by capacity, it barely obtains an optimal solution. In this work, we presented a heuristic algorithm to obtain a *k*-node set with sub-optimal reliability. The proposed algorithm is based on not only a simple method to compute each node's weight and each link's weight, but also an efficient and effective objective function to evaluate the weight of node sets. Before appending one node to *k*-node set, instead of computing the weight of all links and all nodes of set, only the weight of node v_i and links between v_i and G_k are accumulated. The proposed algorithm depends on the maximum weight to find an adequate node and assign it to *k*-node set in a sequential manner until the capacity constraint is satisfied. The reliability computation in our algorithm is only exactly one. The reduction processing is also performed for purifying an adequate *k*-node set. Therefore, KNR in the DS can provide the desired performance.

In addition, the algorithm proposed herein is compared with the EM and *k*-tree reduction method for various topologies. According to that comparison, the proposed algorithm is more efficient in terms of execution time for a large DS. When the proposed method fails to provide an exact solution, the error from the exact solution is only slight.

REFERENCES

[1] A. S. Tanenbaum, *Computer Networks*. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[2] W. Stallings, *Local Networks*. Macmillan Publ. Co., New York, 1984

[3] J. A. Stankovic, "A Perspective on Distributed Computer Systems". *IEEE Trans. Comput.,* Vol. 33, 1984, pp. 1102-1115.

[4] K. B. Irani and N. G. Khabbaz, "A Methodology for the Design of Communication Networks and the Distribution of Data in Distributed Supercomputer Systems". *IEEE Trans. Computers,* Vol. C-31, 1982, pp. 420-434.

[5] L. Kleinrock, "Analytic and Simulation Methods in Computer Network Design", in *Proc. Spring Joint Computer Conf,* 1970, pp. 569-579.

[6] H. Frank, W. Chou, "Topological Optimization of Computer Networks", in *Proc. IEEE 60*, 1972, pp. 1385-1397.

[7] R. S. Wilkov, "Analysis and Design of Reliable Computer Networks". *IEEE trans. Communications,* Vol. COM-20, 1970, pp. 660-678.

[8] Ruey-Shun Chen, D. J. Chen and Y. S. Yeh, "A New Heuristic Approach for Reliability Optimization of Distributed Computing Systems Subject to Capacity Constraints". *Journal of Computers Math. with Applic.,* Vol. 29 No. 3, pp. 1995, pp. 37-47.

[9] Salim Hariri, C. S. Raghavendra, "SYREL: A Symbolic Reliability Algorithm Based on Path and Cuset Methods". *IEEE Transactions on Computers*, Vol. C-36 No. 10, 1987, pp. 1224-1232.

[10] Ruey-Shun Chen, D. J. Chen and Y. S. Yeh, "Reliability Optimization of Distributed Computing Systems Subject to Capacity Constraint". *Journal of Computers Math. with Applic.,* Vol. 29 No. 4, 1995, pp. 93-99.

[11] Wesley W. Chu, "Optimal File Allocation in a Multiple Computer System". *IEEE Trans. on Computer,* Vol. C-18 No. 10, 1969.

[12] M. S. Lin and D. J. Chen, "New Reliability Evaluation Algorithms for Distributed Computing Systems". *Journal of Information Science and Engineering,* Vol. 8 No. 3, 1992.

BIOGRAPHY

Chin-Ching Chiu obtained his Ph.D. of Computer Science and Information Engineering from National Chiao-Tung University in 2000. Currently, he is an Associate Professor at Department of Information Management, Tak-Ming College. His research interest is reliability analysis, genetic algorithm, DNA computing.

Yi-Shiung Yeh obtained his Ph.D. of Computer Science from Of Wisconsin-Milwaukee University in 1985. Currently, he is an Associate Professor at Institute of CS & IE, National Chiao-Tung University. His research interest is data security and privacy, information theory, game theory, reliability and performance.