Proc. Natl. Sci. Counc. ROC(A)
Vol. 25, No. 1, 2001. pp. 27-34

A Genetic Algorithm for *K*-node Set Reliability Optimization with Capacity Constraint of a Distributed System

YI-SHIUNG YEH*, CHIN CHING CHIU*, AND RUEY-SHUN CHEN**

*Institute of Computer Science and Information Engineering
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.
**Institute of Information Management
National Chiao Tung University
Hsinchu, Taiwan, R.O.C.

(Received October 12, 1999; Accepted May 30, 2000)

ABSTRACT

In the reliability analysis of a distributed system (DS), *k*-node set reliability is defined as the probabilities that all the nodes in *K* are connected, where *K* denotes a subset of a set of processing elements. A *k*-node set reliability computation can be very difficult to perform with exponential in many cases. A *k*-node set reliability optimization with a capacity constraint problem is to select a *k*-node set of nodes in a distributed system such that the *k*-node set reliability is maximal and possesses sufficient node capacity. It is evident that this is an NP-hard problem. Relative investigation, namely exact method, has examined *k*-node set reliability optimization with capacity constraint. Although the exact method can obtain an optimal solution, it cannot reduce the computational time. Occasionally, an efficiency algorithm with an exact or nearly exact solution is attractive. Thus, in this paper, we present a method based on a genetic algorithm used to evolve the best *k*-node sets. Because the final number of best *k*-node sets is only one, less time is needed to compute the reliability of the *k*-node set. The exact solution can be obtained in most cases using the proposed method. When it fails to give an exact solution, the deviation from the exact solution is very small. In addition, the proposed algorithm is compared with the exact method for various DS topologies. The results demonstrate that the proposed algorithm is more efficient in execution time. In summary, the proposed algorithm can efficiently obtain the maximal or near maximal *k*-node set reliability with a capacity constraint.

Key Words: distributed system, *k*-node set reliability, genetic algorithm

I. Introduction

A distributed system (DS) is a collection of processor-memory pairs connected by a communication link and logically integrated by means of a distributed communication network (Lin *et al*., 1999). The numerous merits of using a DS include more effective resource sharing, better fault tolerance, and higher reliability. A DS focuses on providing efficient communication among various nodes, thereby increasing their reliability and making their service available to more users (Stankovic, 1984). In designing such systems, system reliability must be considered, which heavily relies on the topological layout of the communication links (Aziz, 1997).

The topology of a network can be characterized by a linear graph. The network topologies can be characterized by their network reliability, message-delay, or network capacity. The performance characteristics depend on many linear graph properties which represent the network topology, such as the number of ports at each node (the degree of a node), and the number of links. The number of links directly impacts the system reliability: reliability decreases with a decrease in the number of links (Hwang and Chang, 1999; Irani and Khabbaz, 1982; Lin and Chen, 1992; Tom and Murthy, 1998). The literature includes some reliability optimization DS models which provide source to destination reliability optimization (Aggarwal *et al*., 1982), *k*-out-of-*n* system reliability optimization (Rai *et al*., 1987), overall system reliability optimization (Aggarwal and Rai, 1981; Torrieri, 1994) etc.

Genetic algorithms (GA) (Holland, 1973) can be applied to search large, multimodal, complex problem spaces (Holland, 1973; Miller and Todd, 1989). The main GA steps are reproduction, selection, crossover, and mutation. The selection, crossover, and mutation processes are repeated until the termination condition is satisfied.

When some data files are allocated onto a DS (Hwang and Tseng, 1993), a set, *K*, of nodes is selected in the DS to allocate data files such that the *k*-node set reliability is maximal under the capacity constraint. An exact method (EM) (Chen *et al*., 1995) examines *k*-node set reliability optimization with a capacity constraint for a DS. EM can obtain an optimal solution but can not effectively reduce the problem space.

Moreover, EM spends more execution time with a large DS. In fact, most distributed systems are large and an increase in the number of nodes causes the EM execution time to grow exponentially. Occasionally, an application requiring an efficient algorithm with an approximate solution is highly attractive.

The objective of this study is to compute the reliability of a subset of network nodes such that the reliability is maximized and the specified capacity constraint is satisfied. The fact that computing the reliability of a DS is an NP-hard problem accounts for why a reasonable solution must be derived in a very short time. Of primary concern here is to finding a nearly exact solution. There is good potential for obtaining optimal or near optimal results using GA for the *k*-node set reliability problem (Coit and Smith, 1996; Kumar and Agrawal, 1993). Therefore, a *k*-node set reliability optimization using a genetic algorithm was developed to solve this problem.

II. Computing Optimal Reliability

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

Bi-directional communication channels operate between processing elements. A distributed computing system can be modeled using a simple undirected graph. For a topology of a DS with four nodes, say $V = \{v_1, v_2, v_3, v_4\}$, and five links, say $E = \{e_{1,2}, e_{1,3}, e_{2,3}, e_{2,4}, e_{3,4}\}$, there are many 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. For example, $K = \{v_1, v_2, v_3\}$ is selected in a DS with bridge topology. The reliability of the set, *K*, can be computed by summing the mutually disjoint terms (Hariri and Raghavendra, 1987):

$$R(G_k)=a_{1,2}a_{1,3}b_{2,3}b_{2,4}+a_{1,2}a_{1,3}b_{2,3}a_{2,4}b_{3,4}+a_{1,2}b_{1,3}a_{2,3}b_{2,4}$$
$$+a_{1,2}b_{1,3}a_{2,3}a_{2,4}b_{3,4}+a_{1,3}a_{2,3}b_{2,4}+a_{1,3}a_{2,3}a_{2,4}b_{3,4}$$
$$+a_{1,2}b_{1,3}a_{2,4}a_{3,4}+a_{1,3}a_{2,4}a_{3,4}.$$

Assume that the probability of $a_{1,2}$, $a_{1,3}$, $a_{2,3}$, $a_{2,4}$, and $a_{3,4}$ is 0.95, 0.94, 0.93, 0.92, and 0.91, respectively. Then, $R(G_k)$ = 0.9958148.

A KNR problem can be characterized as follows:

Given

The topology of a DS.

The reliability of each communication link.

The capacity of each node.

A set of data files.

Assumption

Each node is perfectly reliable.

Each link is either in the working (ON) state or failed (OFF) state.

Constraint

The total data file capacity to be allocated.

Goal

To select a set, *K*, of nodes in a DS in order to allocate data files; by doing so, KNR is made optimal under a capacity constraint.

Reliability optimization can be defined as the maximum reliability when computing a given task under some constraints. For a given task, reliability can be computed as R_1, R_2, ..., R_x for *x* situations, where *x* may be an astronomically large. In doing so, the reliability optimization for the task is the maximal reliability in R_1, R_2, ..., R_x. The heuristic algorithm involves obtaining an approximate solution which is close to the maximal reliability in R_1, R_2, ..., R_x. Restated, a set *K* of nodes is found from the given node set *V* of a DS such that the *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: $\sum_{v_i \in G_k} c(v_i) \geq C_{constraint}$,

where $R(G_k)$, $c(v_i)$, and $C_{constraint}$ are defined in nomenclature.

Obviously, the problem is that a large DS, as in a metropolitan area network, requires a large amount of execution time. Herein, we develop an efficient method that allows KNR optimization in a DS to achieve the desired performance. Owing to its computational advantages, the proposed method may be preferred to an EM when the DS is large.

III. Genetic Algorithm Based *K*-node Reliability Methodology

The search space in GA is composed of possible solutions (chromosomes) to the problem. Each chromosome has an associated objective function value, called a fitness value, which denotes its strength. A set of chromosomes and the associated fitness values are called the population. This population at a given GA stage is referred to as a generation.

1. Development of GAKNR

The GAKNR development is described in the following subsections.

A. Chromosomal-Coding Scheme

Our problem involves a capacity constraint. The network topology is fixed. The size of every node is also fixed. We use a coding scheme with binary numbers. The length

of a chromosome is equal to the number of network nodes of the following types:

$$x_n x_{n-1} \ldots x_i x_{i-1} \ldots x_3 x_2 x_1.$$

Each bit indicates whether the node is selected for the k-node set, where $x_i = 1$ if v_i is selected; otherwise, $x_i = 0$. For example,

x_6	x_5	x_4	x_3	x_2	x_1
0	1	0	1	1	0

This chromosome shows that nodes v_5, v_3 and v_2 are selected.

B. Initialization Approach

The reliability of a set of selected nodes is related to their links and the link reliability. For any node, the degree of that node affects the number of information paths that can be transferred from other nodes. Therefore, we employ a simple method of computing the weight of a node, which takes less time and can quickly compute the weight of every node. For each node v_i, if the degree of v_i is $d(v_i)$, and if the links e_{i,k_1}, e_{i,k_2}, ..., $e_{i,k_{d(v_i)}}$ are adjacent to v_i, then we can construct the following formula to compute its weight:

$$w(v_i)=a_{i,k_1}+b_{i,k_1}\times(a_{i,k_2}+(b_{i,k_2}\times(a_{i,k_3}+(\ldots(b_{i,k_{d(v_i)-1}}\times a_{i,k_{d(v_i)}}))\ldots)), \quad \textbf{(1)}$$

where $i, k_1, k_2, \ldots, k_{d(v_i)} \in \{1, \ldots, n\}$.

The above formula is easy to program and eliminates many multiplication operations. The weight of v_i can be computed in $d(v_i) - 1$ additions and $d(v_i) - 1$ multiplications. Thus, we can obtain the weight of every node in $2 \times e$ additions and $2 \times e$ multiplications.

The initial population can be randomly created or well-adapted (Liepins and Hilliard, 1989). Our algorithm uses some criteria to generate a biased or adapted population at initialization. First, we compute the weight of every node by means of Eq. (1) and select the optimal weight node. This node is then added to every chromosome in the initial population. When one chromosome is generated, we also compare its k-node set size summation with the capacity constraint. If it satisfies our requirement, then it is appended to the population; otherwise, it is discarded.

C. The Object Function

The number of ports at each node (the 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 (Irani and Khabbaz, 1982; Makri and Psillakis, 1997; Torrieri, 1994).

Assume that we have selected a set G_k of nodes with reliability $R(G_k)$, and that the nodes in G_k are all directly connected. If another set G_k' of nodes exists which is just one node (say v_x) different from G_k, if the node v_x is not directly connected to the other nodes in G_k', and if the reliability of any path between node v_x and the set G_k is less than any links between each two nodes of the set G_k of nodes, then we say that $R(G_k) \geq R(G_k')$, for $R(G_k')$ the reliability of set G_k'.

The following observations can be made on how the fitness value of a chromosome can be tuned. For a given selected k-node set, the subset reliability is at least as large as the reliability of the k-node set. Thus, during the fitness value evaluation process, if the total capacity of the subset of a k-node set satisfies the capacity constraint, then its fitness value should be at least as large as the fitness value of this k-node set. The value $\sqrt{|k|+2}$ used to tune the fitness value of a k-node set is less than that of its subset. We can also use the value $\log_2|k|$ instead of the value $\sqrt{|k|+2}$ to tune the fitness value.

Therefore, we can construct the objective function as follows:

$$fv_{i,j} = (ratioprdl_k + ratiodeg_k)/\sqrt{|k|+2}$$

$$1 \leq i \leq ps,\ 0 \leq j \leq tng,\ 2 \leq |k| \leq n, \quad \textbf{(2)}$$

where $ratioprdl_k$ and $ratiodeg_k$ are described in nomenclature.

The objective function is used to compute the fitness value of each chromosome i in generation j. The fitness values indicate which chromosomes are to be carried on to the next generation.

D. Genetic Reproduction and Selection

This is achieved by assigning a proportionately higher fitness value (Davis, 1987; Goldberg, 1989). A "biased" roulette wheel (Liepins and Hilliard, 1989) is used for chromosome selection in the mating pool.

E. Genetic Crossover Operators

A crossover is performed at the boundaries of the node bits. First, two chromosomes are randomly selected from the mating pool. Next, using a random number generator, an integer is generated in the range $(0, n - 1)$. This number is used as the crossover site. The result produces two new chromosomes with information from their parents. For example,

	x_6	x_5	x_4	|	x_3	x_2	x_1
String1	0	1	0	|	1	1	0
String2	0	1	1	|	0	1	1.

If the crossover site is 3, the information exchange occurs as

	x_6	x_5	x_4		x_3	x_2	x_1
Child1	0	1	0	|	0	1	1
	< string1>				<string2>		
Child2	0	1	1	|	1	1	0
	<string2>				<string1>		

The crossover operator sometimes generates a chromosome which does not satisfy the capacity constraint. For example, if the size of v_6, v_5, ..., and v_1 is 18, 20, 15, 12, 13, and 10, respectively, then the capacity constraint is 45. The size summation of child1 does not satisfy our requirement. This anomaly is simply discarded.

F. Genetic Mutation Operator

This operator is performed to improve the global optimal solution if it is appreciably reduced by the crossover operation. First, using a random number generator, an integer (say y) is generated in the range $(0, n \times ps - 1)$. Next, integer division (e.g., y/n) is used to find the mutation chromosomes and an integer module (e.g., y mod n) is used to find the mutation node. The mutation operator sometimes generates a chromosome which does not represent a valid capacity constraint. When this situation occurs, we reserve the original chromosome and select another chromosome for mutation.

G. Replacement Strategy and Termination Rules

The most common replacement strategy is to probabilistically-replace the poorest performing chromosome in the previous generation (Liepins and Hilliard, 1989). On the other hand, the elitist strategy appends the best performing chromosome from the previous generation to the current population, thereby, ensuring that the chromosome with the best objective function value always survives to the next generation. Our GAKNR combines both of these concepts. Each offspring generated after crossover is added to the new generation if it has a better objective function value than both of its parents. If the objective function value of an offspring is better than that of only one of its parents, then a chromosome is randomly selected from the better of the two parents and the offspring. If the offspring is worse than both parents, then either one of the parents is selected at random for the next generation. This ensures that the best chromosome is carried on to the next generation while the worst is not.

GAKNR execution can be terminated when the average and maximum fitness string values in a generation become the same.

H. Genetic Algorithm Parameter

Four parameters affect the solution obtained from GAKNR: *cr*, *mr*, *ps*, and *ng*. The details of analyses of and experiments on choosing the appropriate parameter values are given in De Jong (1975) and Grefenstette (1989). The results of these parameter studies suggest that good GA performance requires high *cr*, very low *mr*, and moderate *ps*. Based on these observations (De Jong, 1975; Goldberg, 1989; Grefenstette, 1989), the appropriate values are $cr = 0.6$, $mr = 0.03$ and $ps = 30$. These values are valid for bit representation chromosomes.

2. Complete Algorithm of GAKNR

The algorithm begins with an initial valid chromosome generation, which satisfies the problem type constraint. This initial generation contains a finite number of valid strings selected by well-adapted. The number of strings in any generation, the population size, is kept to an even number to ease the crossover. The detailed steps in GAKNR are given in Appendix.

3. An Illustrate Example

The topology of the distributed system with eight nodes and eleven links is described as follows. $c(v_i)$ represents the capacity of node v_i, and $a_{i,j}$ represents the reliability of link $e_{i,j}$.

In step 0, after evaluating each node's weight is evaluated using Eq. (1), the weights of v_1, v_3, ..., 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 heaviest node and is added to each chromosome of the initial population. The first column of Table 1 lists the initial population chromosomes.

In step 1, each chromosome's fitness value, fitness value ratio, and roulette-wheel area are derived. The results are displayed in columns 2, 3 and 4 of Table 1. The average fitness value is 0.2708945.

The algorithm executes statements between steps 2 and 5. They are reproduction and selection for the mating pool, crossover and mutation for the next generation, replacement and creation of a new generation, and testing for the terminating condition. In addition, step 1 is executed again. The right side of Table 1 lists the results of generation 1. The average fitness value is 0.3191933.

In the same way, the average fitness values for generations 2, 3, and 4 are 0.3875577, 0.5026517, and 0.6484991, respectively. In generation 5, all the chromosomes are 00101000, and the fitness value is 0.665. The average fitness value is 0.665. Figure 2 illustrates the average fitness value of every generation. Because the terminating condition is satisfied, the algorithm goes on to step 6.

In step 6, the k-node set in the population chromosome is $\{v_4, v_6\}$. The algorithm computes the reliability using SYREL and outputs the k-node set, which is the highest

Table 1. The Results of Initial Generation and Generation 1

Initial generation *Gen*(0)				generation *Gen*(1)			
$x_8 \ldots x_2x_1$	$fv_{i,0}$	$p_{i,0} = fv_{i,0}/F_0$	$q_{i,0} = \sum_{k=1}^{i} p_{k,0}$	$x_8 \ldots x_2x_1$	$fv_{i,1}$	$p_{i,1} = fv_{i,1}/F_1$	$q_{i,1} = \sum_{k=1}^{i} p_{k,1}$
10011010	0.2614490	0.0965132	0.0965132	00111000	0.5619984	0.1760683	0.1760683
01001001	0.2884528	0.1064816	0.2029948	01001001	0.2884528	0.0903693	0.2664376
01111100	0.2713785	0.1001787	0.3031735	01001001	0.2884528	0.0903693	0.358069
11011010	0.2936748	0.1084106	0.4115841	00111101	0.2713785	0.0850201	0.4418270
01001101	0.1901161	0.0701809	0.4817649	01111100	0.2936785	0.0920064	0.5338335
11101110	0.2573665	0.0950062	0.5767710	01001101	0.2573665	0.0806303	0.6144637
11101111	0.2494319	0.0920771	0.6688483	11101110	0.2494319	0.0781445	0.6926802
11011010	0.2444444	0.0902360	0.7590843	11101110	0.2494319	0.0781445	0.7707527
10111000	0.1901161	0.0701809	0.8292656	11111100	0.2692309	0.0843473	0.8551000
10111000	0.4625113	0.1707348	1.0000000	10111000	0.4625113	0.1449000	1.0000000
$avgfv = F_0/ps$ = 0.2708945 (average fitness value)				$avgfv = F_1/ps$ = 0.3191933 (average fitness value)			

Note: The means of each notation is described in nomenclature.

Table 2. The Results Obtained by Three Different Methods for Various DS Topologies

Size		Optimal solution		Exhaust	Exact method		Proposed method				
n	*e*	Max_rel	*k*-node set	NRC	time (Sec)	NRC	time (Sec)	NRC	*ng*	*ps*	absolute err
5	6	0.9462500	1,2,3	32	0.11	10	0.11	1	5	10	0
6	8	0.9383035	4,5,6	64	0.16	15	0.11	1	9	10	0
6	9	0.9950069	1,3,5	64	0.22	20	0.11	1	10	10	0
7	8	0.9187206	1,2,4	128	0.27	35	0.11	1	6	10	0
7	11	0.9967785	1,2,3	128	0.33	35	0.11	1	7	10	0
8	11	0.9974378	4,6	256	0.61	44	0.33	1	5	10	0
10	13	0.9347952	1,7,8,9,10	1024	34.05	255	0.38	1	8	25	0
10	17	0.9994068	2,8,9	1024	10.98	119	0.38	1	8	10	0
10	19	0.9995282	1,5,6	1024	58.88	150	0.44	1	13	40	0
12	18	0.9858263	3,4,5,6	4096	52.07	538	0.49	1	37	80	0
12	21	0.9990777	1,3,5,6	4096	679.04	537	0.44	1	24	80	0.0006069
13	20	0.9978402	4,6	8192	125.23	246	0.44	1	19	40	0.0006842
19	31	0.9979870	6,8,9	524288	761.04	2369	0.82	1	45	80	0

Note: *n*: the number of nodes in *G*; *e*: the number of links in *G*; *k*-node set: selected nodes; NRC: the number of reliability computations; Exhaust: the exhaustive method; Max_rel: the maximum reliability satisfies our constraint.

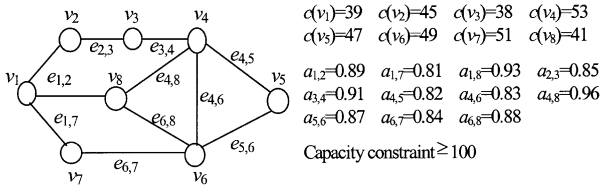

Fig. 1. The DS with eight nodes and eleven links.

reliability *k*-node set, and $R(\{v_4, v_6\}) = 0.9974378$. The reliability count is equal to 1.

IV. Results and Discussion

Table 2 presents the data on the results obtained using three different methods under various DS topologies. With the exhaustive method and EM, the number of reliability computations grows rapidly when the size of the DS topology is increased. The number of the computations with the proposed method, however, is constant and is independent of the DS topology size. The deviation is very small when the proposed method cannot obtain an optimal solution. These data show that the proposed method could be more effective than the conventional method.

In this paper, we have proposed a new technique for solving the *k*-node set reliability problem. The EM complexity is $O(2^e \times 2^n)$, where e denotes the number of edges and n represents the number of nodes. In our proposed algorithm, in the worst case, the complexity of evaluating the weight of each node is $O(e)$, that of selecting the heaviest node is $O(n)$, and that of computing the reliability of the *k*-node set is $O(m^2)$, where m represents the number of paths of the selected

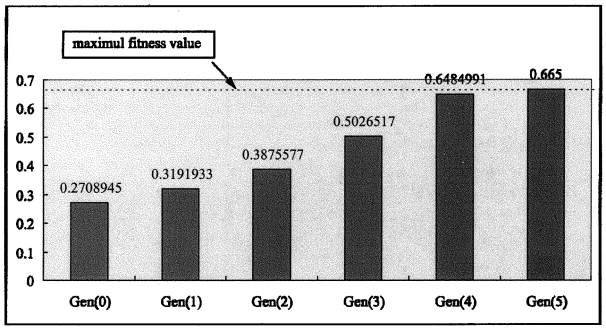

Fig. 2. The average fitness value of every generation of the DS.

k-node set (Aziz, 1997; Hariri and Raghavendra, 1987). Therefore, the complexity of the proposed algorithm is $O(n \times ps \times tng + m^2)$. Results obtained using our algorithm have been compared with those obtained using the exhaustive method and EM. Although conventional techniques such as EM can yield an optimal solution, they cannot effectively reduce the number of reliability computations. An application occasionally requires an efficient algorithm for computing the reliability owing to resource considerations. Under this circumstance, deriving the optimal reliability may not be a promising option. Instead, an efficient algorithm yielding approximate reliability is preferred.

In contrast to the computer reliability problem, which is static-oriented, the KNR problems in a DS are dynamic-oriented since many factors, such as the node capacity, DS topology, link reliability, and the number of paths between each node, can significantly affect the efficiency of the algorithm (Aziz, 1997; Hariri and Raghavendra, 1987). Thus, quantifying the time complexity exactly is extremely difficult. The accuracy and efficiency of the proposed algorithm has been verified by implementing simulation C language programs executed on a Pentium 100 with 16M-DRAM on MS-Windows 95. In our simulation, the number of reliability computations for the proposed algorithm was constant. The exact solution that could be obtained was above 84%, and the average deviation from the exact solution was under 0.001% (Makri and Psillakis, 1997). Because the proposed algorithm uses an adapted population at the initialization stage and uses the elitist strategy at the replacement stage, in a few cases, we could not obtain the exact solution.

V. Conclusion

This paper has presented a genetic algorithm based k-node reliability methodology (GAKNR) for solving the k-node set reliability problem. The proposed algorithm uses a genetic algorithm to select a k-node set that has maximal or nearly maximal system reliability. Our numerical results show that the proposed algorithm can obtain the exact solution in most cases, and the computation time seems to be significantly shorter than that needed for the exact method. When the proposed method fails to give an exact solution, the deviation from the exact solution is very small. The technique presented in this paper might help readers understand the relationship between k-node set reliability and the DS topology.

Acknowledgment

The authors would like to thank the National Science Council of the Republic of China for financially supporting this research under contract NSC 86-2213-E-009-012.

Nomenclature

The following notations and definitions are used herein.

A. 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	an i node which represents the i processing element
$c(v_i)$	the capacity of the i node
e	the number of links in G, $e = \|E\|$
$e_{i,j}$	an edge which represents a communication link between v_i and v_j
$a_{i,j}$	the probability of success for link $e_{i,j}$
$b_{i,j}$	the probability of failure for link $e_{i,j}$
$C_{constraint}$	the total capacity constraint in a DS
$w(v_i)$	the weight of the i node
$d(v_i)$	the number of links connected to the node v_i
G_k, G'_k	the graph G with the set K of nodes specified, and $\|K\| \geq 2$
$R(G_k)$	the reliability of the k-node set
$R(G'_k)$	the solution of a DS graph G
$sumdlpr_k$	the sum of $a_{i,j}$ of $e_{i,j}$, which is one of the direct links between the nodes G_k, i.e., $sumdlpr_k = (\sum_{v_i \in G_k} \sum_{v_j \in G_k} a_{i,j})/2$
$avgfv$	the average fitness value, i.e., $avgfv = F_j/ps$
$ratioprdl_k$	the ratio of $sumdlpr_k$ to $k(k-1)/2$, where $k(k-1)/2$ represents the number of links if there is a link between every two nodes of the k-node set.
$ratiodeg_k$	denotes $(\sum_{v_i \in G_k} d(v_i))/((n-1)k)$, where $((n-1)k)$ is the maximal degree of the k-node set if there is a link between each two nodes of V, $v_i \in G_k$
$Gen(i)$	the ith generation of GA, for $i = 0, ..., ng$
mr, cr	mutation rate, crossover rate
ps	the population size representing the total number of strings in a generation
ng	the number of generations when GA end
$x_n, ..., x_1$	a binary string of length n representing a chromosome, where $x_i = 1$ if v_i is selected; otherwise, $x_i = 0$, for $i = 1, ..., n$
$fv_{i,j}$	the actual fitness value of a chromosome i in generation j
F_j	the total fitness value of all the chromosomes in generation j, i.e., $F_j = \sum_{i=1}^{ps} fv_{i,j}$
$p_{i,j}$	a proportion of a roulette-wheel slot-sized chromosome i in generation j, i.e., $p_{i,j} = fv_{i,j}/F_j$
$q_{i,j}$	the accumulation of $p_{i,j}$, i.e., $q_{i,j} = \sum_{k=1}^{i} p_{k,j}$.

B. Definitions

Definition 1. k-node set reliability (KNR) is defined as the probability that

a specified set of nodes, *K*, 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, *K*, of nodes if and only if there is a link between v_i and *K*.

Appendix

This appendix describes in detail the steps in GAKNR.

Algorithm GAKNR

step 0 /*Read the system parameters and initialize of GA*/
Read DS parameters, such as the number of nodes, the number of links, each link's reliability, each node's capacity, capacity constraint (say n, e, $a_{i,j}$, $c(v_i)$, $C_{constraint}$) etc.
Read genetic algorithm parameters, such as the population size, crossover rate, mutation rate, terminate generation number (say ps, cr, mr, tng) etc.
Compute the weight of each node using Eq. (1) and select the heaviest node.
/* Generate each chromosome of the initial population by well-adapted; that is, every chromosome includes the heaviest node and satisfies the capacity constraint. */
$i = 0$.
Dowhile (*number_of_chromosome* < ps)
Randomly generate chromosome (x_n, ..., x_1).
Add the heaviest node to the chromosome.
Compute the nodes capacity that includes this chromosome.
If the sum of the capacity of this chromosome satisfies $C_{constraint}$, then append the chromosome to the population and add one to the *number_of_ chromosome*.
Enddowhile
step 1 /*Compute the fitness value for each chromosome */
Compute the fitness value of each chromosome in generation *Gen*(i) using the objective function (Eq. (2)) described in Subsection III.1.C.
Compute the fitness value sum and the fitness value ratio of each chromosome, and generate the roulette-wheel area.
step 2 /*Reproduction/Selection for the mating pool*/
/*Perform selection on *Gen*(i) to form the mating pool.*/
Dowhile (*number_of_chromosome* < ps)
Generate a random number between 0.0 and 1.0.
According to the random number and q_i, select a chromosome for the mating pool.
Increment the *number_of_chromosome*.
Enddowhile
step 3 /*Crossover and mutation for the next generation*/
crossover_count = $\lceil cr \times ps/2 \rceil$.
Dowhile (*crossover_count* > 0)
Generate two random integers between 1 and ps to select two chromosomes in the mating pool as the parent string for crossover.
Generate one random number between 1 and $n-1$ for the crossover position.
According to crossover position, generate two children.
If the capacity constraint is satisfied and the fitness value is better than the parent, then replace the old string; otherwise discard the string.
Decrement the *crossover_count*.
Enddowhile
/* Select a chromosome in the mating pool for mutation and generate a new chromosome.*/
mutation_count = $\lceil mr \times ps \rceil$.
Dowhile (*mutation_count* > 0)
Generate one random number (say y) between 1 and ($n \times ps - 1$).
Use the integer division (e.g., y/n) to find the mutation chromosome.
Use the integer module (e.g., y mod n) to find the mutation node.
If the capacity constraints are satisfied and the fitness value is better than the old string, then replace it; otherwise, discard the string.
Decrement the *mutation_count*.
Enddowhile
step 4 /*Replacement and new generation creation */
Replace the population of *Gen*(i) with the mating pool to create the population of the new generation *Gen*(i+1).
step 5 /*Test for terminating condition*/
$i = i+1$.
If ($i \le ng$ and some fitness value $\neq$ the average fitness value), then go to step 1.
step 6 /* Compute the reliability of the k-node set and output the best k-node set.*/
Compute the reliability $R(G_k)$ of the final k-node set result in the chromosome of the population using SYREL (Hariri and Raghavendra, 1987) and Output the k-node set.

End GAKNR

References

Aggarwal, K. K. and S. Rai (1981) Reliability evaluation in computer communication networks. *IEEE Transactions on Reliability*, **R-30**(1), 32-35.

Aggarwal, K. K., Y. C. Chopra, and J. S. Bajwa (1982) Topological layout of links for optimizing the S-T reliability in a computer communication system. *Microelectronics and Reliability*, **22**(3), 341-345.

Aziz, M. A. (1997) Pathset enumeration of directed graphs by the set theoretic method. *Microelectronics and Reliability*, **37**(5), 809-814.

Chen, R. S., D. J. Chen, and Y. S. Yeh (1995) Reliability optimization of distributed computing systems subject to capacity constraint. *Journal of Computers and Mathematics with Applications*, **29**(4), 93-99.

Coit, D. W. and A. E. Smith (1996) Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transaction on Reliability*, **45**(2), 254-260.

Davis, L. (1987) *Genetic Algorithms and Simulated Annealing: an Overview*, *Genetic Algorithms and Simulated Annealing*. Morgan Kaufman, Los Altos, CA, U.S.A.

De Jong, K. A. (1975) *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. Dissertation. University of Michigan, Ann Arbor, MI, U.S.A.

Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, pp. 412-415. Addison-Wesley, New York, NY, U.S.A.

Grefenstette, J. J. (1989) Incorporating problem specific knowledge into genetic algorithm. *Annals of Operations Research*, **21**, 31-58.

Hariri, S. and C. S. Raghavendra (1987) SYREL: a symbolic reliability algorithm based on path and cuset methods. *IEEE Transactions on Computers*, **C-36**(10), 1224-1232.

Holland, J. H. (1973) Genetic algorithm and the optimal allocation of trials. *SIAM Journal on Computing*, **2**(2), 88-105.

Hwang, G. J. and S. S. Tseng (1993) A heuristic task assignment algorithm to maximize reliability of a distributed system. *IEEE Transaction on Reliability*, **R-42**(3), 408-415.

Hwang, L. C. and C. J. Chang (1999) Analysis of a general limited scheduling mechanism for a distributed communication systems. *Computer Network*, **31**(18), 1879-1889.

Irani, K. B. and N. G. Khabbaz (1982) A methodology for the design of communication networks and the distribution of data in distributed supercomputer systems. *IEEE Transaction on Computers*, **C-31**(5),

420-434.
Kumar, A. and D. P. Agrawal (1993) A generalized algorithm for evaluation distributed program reliability. *IEEE Transactions on Reliability*, **42** (3), 416-426.
Liepins,G. E. and M. R. Hilliard (1989) Genetic algorithms: foundations and applications. *Annals of Operations Research*, **21**, 31-58.
Lin, M. S. and D. J. Chen (1992) New reliability evaluation algorithms for distributed computing systems. *Journal of Information Science and Engineering*, **8**(3), 353-391.
Lin, M. S., M. S. Chang, and D. J. Chen (1999) Efficient algorithms for reliability analysis of distributed computing systems. *Journal of Information Science and Engineering*, **117**(1-2).
Makri, F. S. and Z. M. Psillakis (1997) Bound for reliability of k-within connected-(v,s) out-of (m,n) failure systems. *Microelectronics and Reliability*, **37**(8), 1217-1224.
Miller, G. F. and P. M. Todd (1989) Designing neural network using genetic algorithm. 3rd International Conference on Genetic Algorithm and Application, San Mateo, CA, U.S.A.
Rai, S., A. K. Sarje, E. V. Prasad, and A. Kumar (1987) Two recursive for computing the reliability of *k*-out-of-*n* systems. *IEEE Transactions on Reliability*, **R-36**(2), 261-265.
Stankovic, J. A. (1984) A perspective on distributed computer systems. *IEEE Transaction on Computer*, **C-33**(12), 1102-1115.
Tom, P. and C. R. Murthy (1998) Algorithms for reliability-oriented module allocation in distributed computing systems. *Journal of Systems and Software*, **40**(2), 125-138.
Torrieri, Don. (1994) Calculation of node-pair reliability in large networks with unreliable nodes. *IEEE Transactions on Reliability*, **43**(3), 375-377.

基因演算法求分散式系統中具有容量限制的 K 節點可靠度之最佳化

葉義雄[*] 邱錦清[*] 陳瑞順[**]

[*]國立交通大學資訊工程研究所
[**]國立交通大學資訊管理研究所

摘 要

在分散式系統的可靠度分析中，K節點的可靠度是所有在集合K（從分散式系統中所有的處理單元內，選取部分處理單元所組成的集合）中的處理單元皆連通的機率。計算K節點的可靠度有時是非常耗時的，且在許多情況是呈指數複雜度。具有容量限制的K節點可靠度之最佳化之問題，乃是在分散式系統的節點中，選出一組K節點集合，使得K節點集合可靠度爲最大，並且擁有足夠的節點容量。顯見這是一個NP-Hard問題。相關的研究中，一個稱之爲“精確方法”，已研究具有容量限制的K節點可靠度之最佳化。雖然“精確方法”可求得最佳解，卻不能減少計算時間，所以只適用於處理單元很少時。有時候，一個有效率的演算法且能求得精確解或近似精確解較吸引人。因此，本論文提出一個以基因演算法爲基礎的方法，使用目標函數，避免冗長繁多的各種處理單元組合的可靠度計算，推演出最佳的K節點。因爲最後最佳的K節點集合只有一組，因此只須花少許的時間計算此組的可靠度。本方法絕大部分情況皆可求得精確解，當所求得的不是精確解時，其誤差值也非常小。本方法與“精確方法”，藉由各種不同的分散式網路拓樸加以比較，結果顯示本方法可以獲得更有效的執行時間。總之，本方法可以很有效率求得具有容量限制的K節點之最佳或近似最佳可靠度。