The Study of Optimal File Allocation in Star Distributed Computing Systems based on Grouping Genetic Algorithm

基於分群基因演算法處理星狀分散式系統下最佳化檔案配置之研究

陳永瑞
國立台北科技大學電機系
s2319005@ntut.edu.tw

林敏勝
國立台北科技大學電機系
mslin@ee.ntut.edu.tw

Abstract

In the distributed system, the system reliability problem is always an important and interesting topic for discussion. System reliability and allocation of data are closely related — the well is files assignment, the higher is system reliability. The theme of this thesis is how to find an optimal file allocation in a star-topology network.

In this paper, we have shown an optimal file allocation in a star-topology network problem is NP-hard problem. With related knowledge and operations of grouping genetic algorithm, we eliminate worse file assignment through competition and produce well; lead to find out an optimal file assignment in amount of finite computing time. Finally, we compared our proposed algorithms with exact and heuristic algorithms. It is clear to see that our proposed algorithms could find an exact or a better file allocation.

Keywords： file allocation, grouping genetic algorithm, star-topology, reliability, distributed system.

摘要

分散式系統中，系統可靠度問題一直是一個重要和有趣的議題，系統的可靠度與資料之間分佈情形有著密不可分的關係 — 一個較好的檔案配置有著一個較高的系統可靠度，本論文研究的主題是如何在星狀網路下得到一個最佳化檔案配置。

在論文中，我們證明了星狀網路下最佳化檔案配置問題是一個 NP-hard 問題，透過相關的知識與分群基因演算法的操作，淘汰較差的檔案配置，產生較佳的檔案配置，以便在有限的時間內得到檔案的最佳配置，最後，再與正確解和其他的啟發式演算法做比較，可以清楚發現本論文所提出的演算法可以得到一個正確或一個較佳的檔案配置。

關鍵詞：檔案配置、分群基因演算法、星狀網路、可靠度、分散式系統。

1. Introduction

In the engineering, one of the most considerable problems is the reliability. It also occurs in the design of the distributed computing systems (DCSs). A DCS is made up of processing nodes and communication links. In a DCS, we can use the redundancy of the resource to increase its reliability. How to design a DCS with the best reliability is significant to the computer staff.

System performance can be described with different measures. One of them is reliability, which is the probability of successful operations. In the past few decades, numerous efficient reliability computing algorithms had been developed [2-5]. M. S. Chang, et al [2] proposed a polynomial algorithm to calculate distributed program reliability (DPR) on a star topology with some additional restricted file distribution, and a polynomial algorithm for computing DPR with approximate solution. In addition, much research has been done for the optimization of system reliability [6-13] with genetic algorithm.

A survey of reliability design with genetic algorithm based is presented in [6]. Y. S. Yeh, et al [11] for K-node set reliability optimization with capacity constraint of a distributed system. In [13], A. Kumar, et al achieved various files were allocated to different nodes of a distributed computing system so that the reliability of a executing a program was maximized.

In daily life, people usually connect their computer to a hub through communication lines for communicating with others no mater where they are. It is clear to see that most computers are centralized in a hub. In graphic, we can present this connection schema as star topology. We are interested in that how to allocate a set of various files such that the optimal reliability of a star DCS with given the reliability of communication and perfect processing nodes. In this paper, we took advantage of the concept of grouping genetic algorithm to handle it. It would find an exact or near-optimal file allocation in finite execution time.

2. Problem statements and Mathematical model

Before this file allocation problem proceeded, we firstly introduce the simple alike case, and then give a full description of ours.

2.1 Bin packing problem

The bin packing problem (BPP) is a combinatorial optimization problem and defined as below [15]:

Given a finite set of the *N* objects with sizes s_i and the identical bins with capacity *C*, such that each $s_i \le C$. Find a partition of the objects into the bins such that the number of bins, *K*, is minimized.

2.2 File allocation in a star network problem

File allocation in a star network problem (FASP) can be described as follows: given a set of files of various sizes and copies, subject to physical environment constraints (i.e. reliability and memory of processing nodes, reliability of distributed systems and communication links, cost and so on) , how should be the files distributed among the processing nodes in a star network, so that total size of the files within each processing node are no larger than its capacity and the redundant files do not in the same processing node, as to obtain the best distributed system reliability ?

2.3 Mathematical model

An optimal file allocation in a star topology problem is described above can be formulated as follows:

Maximize:

$$DSR(S) \tag{1}$$

Subject to:

$$\sum_j x_{ij} s_j \le c_i \ , \tag{2}$$

$$\sum_i x_{ij} = \left| f_j \right| \tag{3}$$

$$i = 1,2\cdots, n;\ j = 1,2,\cdots, m \tag{4}$$

where

DSR(*S*) is the distributed system reliability of a star network *S*.

x_{ij} represents that file *j* is allocated to node *i*.

$$x_{ij} = \begin{cases} 1 & \text{if file } j \text{ is allocated to node } i \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

s_j is the size of file j.

c_i is the capacity of processing node i.

$\left| f_j \right|$ is the copies of file j.

n is the number of processing nodes in *S*.

m is the number of distinct files in the file set.

The following list the relevant assumptions of this problem:

1. Both the processing nodes and the central node (hub) in a star network are perfect.
2. Failure of a communication link in a star network is independent of failures of others.
3. At most one copy of each file can be allocated in a processing node.

2.4 File allocation in a star network problem is NP-hard

In this section, we assume that the reader is familiar with the basic notions of NP-hard [15]. The BPP [15] is a well-known NP-hard problem. Therefore, we show that the BPP can be reduced to the FASP, and the FASP is NP-hard as well. The statements of the proof are delineated as below:

Proof:

Given any instance *I* of the BPP, which has identical bins of capacity *C* and a finite set of object of sizes s_i such that each $s_i \le C$. Trivially, there is a direct mapping can make the

BPP be reduced to the FASP easily. The direct mapping maps the objects and the bins in the BPP to the files and the processing nodes in the FASP respectively. Here, we assume the mapped star network *S* in the FASP has the same reliability *r* (<1) of communication links and each file has only one copy. This *S* is the simplest special case in the FASP. In addition, we assume the maximal distributed system reliability *R* of the *S* can be computed in amount of finite time. That is, $R = r^k$ where the parameter *k* is the minimal number of the nodes to which the files allocated. The parameter *k* is able to draw as $\frac{\log R}{\log r}$ and also the minimal number of bins in the BPP. Owing to NP-hard problem, there is no known optimal algorithm for BPP running in polynomial time; in other words, the minimal number of bins, *k*, can't be computed in amount of finite time. The fact contradicts the assumption. So, the FASP is NP-hard as well ■.

The FASP has been shown as NP-hard. Thus, as either the number of processing nodes or of the distinct files, or both increase, it would take a large amount of execution time to find an optimal file allocation in a star network.

3. Solution algorithm based on grouping genetic algorithm

In the industrial, many optimal problems are quite hard to solve by conventional optimization techniques. A powerful method originated from natural genetics is called genetic algorithm (GA) which has been developed by John Holland in 1975 can provide a means to handle various optimal problems [16-18]. In GA, a chromosome represents a solution to the problem at hand. Each chromosome is composed of genes. A gene is the basic unit of information. Each gene represents a trait and will influence the future individual. The basic GA operations include crossover, mutation, selection, and fitness evaluation. In [20], E. Falkenauer modified GA for grouping problems. The modified GA is called grouping genetic algorithm (GGA). The main steps in GGA are the same as GA. The differences in GGA are that crossover operator transfers groups from parents to offspring and mutation operator works with groups rather than items. The steps in our solution algorithms are described as the following sections.

3.1 Chromosome encoding

We use a binary number based coding scheme. The length of a chromosome, *lchrom*, depends on *n* and *m*, and is equal to $m \times n$, where *m* is the number of distinct files and *n* is the number of processing node. An illustration of a chromosome scheme is given in Figure 1, where x_{ij} = 1 (0) if file *j* is (not) allocated to node *i*.

3.2 Evaluation of chromosome

Because computing the reliability of a star DCS has been shown NP-hard [2], it is hard to calculate exact reliability of a DCS. Therefore, we adopt approximate computing methods [2, 3] to be our evaluation of chromosomes. There are two bound for our problem, and they are described respectively as follows:

Procedure : L_BOUND (v) /* lower bound */
  For $k = 1$ to m do
    $F_k^{'} = P_k$;
    $O_k^{'} = \phi$;
    For $i = 1$ to k - 1 do
      if $|P_i - P_k| = 1$ then
        $O_k^{'} = O_k^{'} \cup (P_i - P_k)$;
    End.
  End.
  return $1 - \sum_{k=1}^{m} \prod_{j \in F_k^{'}} q_j \prod_{j \in O_k^{'}} p_j$; (6)
End Procedure.

← node 1 →			← node 2 →			← node 3 →			← node 4 →			← node 5 →		
x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	x_{31}	x_{32}	x_{33}	x_{41}	x_{42}	x_{43}	x_{51}	x_{52}	x_{53}

Figure 1 Illustration of a chromosome scheme with *n* = 5 and *m* = 3 (15 genes).

Procedure : U_BOUND(v) /* upper bound */

For $k = 1$ to m do

$F_k'' = P_k$;

$O_k'' = T - P_k$;

For $j \in T - P_k$ do

if $P_i \cap \left(O_k'' - \{j\}\right) \neq \phi$ *for* $i = 1,\ldots,k-1$

then $O_k'' = O_k'' - \{j\}$;

End.

End.

$$\textit{return} \quad 1 - \sum_{k=1}^{m} \prod_{j \in F_k''} q_j \prod_{j \in O_k''} p_j \; ; \qquad (7)$$

End Procedure.

where

P_k is the kth file cutest in the set of file distributed in a star network.

T is the set of distinct files distributed in a star network.

O_k' , O_k'' are the sets of nodes that do not contain file k.

F_k' , F_k'' are the set of nodes that contain file k.

p_j is the working probability of link associate with the node j in O_k' or O_k''.

q_j is the failure probability of link associate with the node j in F_k' or F_k''.

The computational complexity of these two approximate methods are $O(mn^2)$ [3].

We take conveniently the mean of the upper and the lower bound as the fitness of a chromosome. That is as follows:

$$\textit{fitness}(v) = \frac{\text{U\_BOUND}(v) + \text{L\_BOUND}(v)}{2} \qquad (8)$$

3.3 Initial population

In this stage, we use a combinatorial method of random and worst fit (WF) heuristic to produce a chromosome that satisfies the node's constraints and the file's constraints. Provided that the selected random node does not contain the file and its capacity is large enough to put, we can put it into the selected random node. If not, require another node. Putting the file into the founded node that with largest capacity and does not yet contain it yields the worst fit heuristic.

3.4 Operators

The roulette wheel strategy [16, 17], fitness-proportional selection, is used. Crossover operator transfers some node section from parents to offspring [20]. Mutation operator works with a random node rather than a random bit [20]. Repair operation used to tune those infeasible chromosomes to feasible ones.

Owing to the fact that exchanges a section of the chromosome in the crossover stage, the file constraints could be violated. We call the violation of the file constraints as defect phenomenon. It can be divided two case further — positive defects and negative defects. Positive defects refer to the copies of files over than the file constraints; on the contrary, refer as negative defects. In order to tune those infeasible chromosomes to feasible ones, we adopt a pair of repairing mechanisms. The repair procedure is listed as bellows, where RC' and RC'' are the remaining capacity factor for positive and negative defects respectively.

Procedure : REPAIR

Calculate the current copies of each file and

the current remaining capacity of each node in a chromosome.

/* process for positive defects */

For $i = 1$ to m do

times = the current copies of f_i - the constraint of f_i.

if *times* > 0 then

For $j = 1$ to *times* do

For $k = 1$ to n do

Calculate the RC_k' and the selective weight $RC_k' \times q_k$ for node k.

End.

Select a node which contains f_i with largest selective weight

and then delete f_i and update the capacity of that node.

End.

End.

/* process for negative defects */
For $i = 1$ to m do
  times = the constraint of f_i - the current copies of f_i.
  if *times* > 0 then
    For $j = 1$ to *times* do
      For $k = 1$ to n do
        Calculate the RC_k'' and the selective weight $(1 - RC_k'') \times p_k$ for node k.
      End.
      Select a node with largest selective weight and then add f_i
      and update the capacity of that node.
    End.
  End.
End Procedure..

For the positive defects, the RC' of node i refers as follows:

$$RC_i' = \frac{\text{Remaining Capacity of node } i}{\sum_{j=1}^{n} \text{Remaining Capacity of node } j}, \quad (9)$$
$$i = 1, 2, \ldots, n$$

For the negative defects, the RC'' of node i refers as follows:

$$RC'' = \begin{cases} \dfrac{\text{Remaining Capacity of node } i}{\sum_{j=1}^{n} \text{Remaining Capacity of node } j}, \\ \text{if node i can accommodate the file.} \\ 0, \quad \textit{otherwise} \end{cases} \quad (10)$$
$$i = 1, 2, \ldots, n.$$

3.5 Replacement

In order to get better solutions, the worse individuals have to be thrown away in processing. Two replacement used here; one is full generational replacement and the other is elitist strategy [16-18]. The two worst chromosomes are replaced by their offspring soon after they give birth. This is called the full generational replacement.

3.6 Termination rule

We treat the upper bound of generations specified by the tester as the termination condition of the GGAFAS. When the number of generations exceeds the upper bound, the execution of the GGAFAS is terminated.

4. Experimental results

The experimental programs used in our experiments had been implemented in Borland C++, and listed as below:

1. Exact algorithm (EA) [21, 22]: find the exact reliability with the best file allocation.
2. Heuristic algorithm 1 (HA 1): assign files based on the reliability of a communication link.
3. Heuristic algorithm 2 (HA 2): assign files based on the product of a node capacity and the reliability of the communication link which incident on the node.
4. Heuristic algorithm 3 (HA 3): It's criterion of the candidate node selection as follows:

$$weight(i) = \sqrt[c_i]{(p_i)^{\sum_{j=1}^{m} s_j}}, \; i = 1, 2, ..., n, \quad (11)$$

where c_i is the capacity of node i, and s_j is the size of file j.

5. Grouping genetic algorithm 1 (GGAFAS 1): without ranking and replacement; that is, only crossover and mutation.
6. Grouping genetic algorithm 2 (GGAFAS 2): ranking [16-18] version.
7. Grouping genetic algorithm 3 (GGAFAS 3): full generational replacement version.
8. Grouping genetic algorithm 4 (GGAFAS 4): elitist [16-18] version.

In the implementation of our heuristics, we also consider the effect affected by the files assigned order. We divided this influence into two ways in detail; one is the smallest file size first and the other is the biggest first. At each pass, files relying on their sizes assigned in turn until satisfy the file requirements.

We used a set of star distributed computing systems with the numbers of various processing nodes, the unlike capacities of processing nodes, the number of distinct files, and the different copies of files as our experimental materials. The first number of the test file name represents that

the node number in a star DCS, and the second number represents that the distinct file number in a star DCS. For example, a 6-4 test file represents a star DCS consists of 6 different processing nodes and 4 distinct files.

Due to all of the heuristics have two file assignments, in Table 1, the column title "smallest" refers to the file with smallest size first assigned, and "biggest" refers to the file with biggest size first assigned. Symbol '—' means that the star DCS is too large to exactly calculate its reliability. The data met the exact values shown in Table 1 are underlined, and the most approximate is shown with asterisk (*).

When the number of processing nods in a star DCS greater than 20, we calculated the average of upper bound and lower bound as its reliability. We could see that the HA1 performs well than other heuristic algorithms in almost all cases form Table 1.

Table 1 the experimental data of the exact algorithm and the heuristics.

		EA	HA 1		HA 2		HA 3	
n	m		smallest	biggest	smallest	biggest	smallest	biggest
6	4	0.85344	0.7576	0.82754*	0.721	0.784	0.7735	0.79856
7	3	0.89544	0.871192	0.8596	0.787	0.8176	0.872032*	0.8596
8	3	0.90272	0.89014*	0.8806	0.8575	0.8398	0.88804	0.8785
9	3	0.953907	0.953907	0.952828	0.841704	0.93761	0.840854	0.935636
10	3	0.945002	0.918492	0.945002	0.83814	0.933372	0.915672	0.926386
15	6	—	0.895647	0.908301	0.804284	0.864195	0.886023	0.916376*
20	13	—	0.912788	0.905985	0.87128	0.905356	0.919974*	0.909575
30	19	—	0.9106	0.919068	0.893928	0.882953	0.920838*	0.905563
50	22	—	0.918804*	0.902886	0.86129	0.856423	0.908406	0.899141
100	40	—	0.908864*	0.907111	0.806761	0.785517	0.885185	0.885948
200	25	—	0.968567*	0.96558	0.863665	0.850215	0.961191	0.965395

In our experiments for GGAFASs, each test file represents a star DCS. According to the number of processing nodes in a DCS, we could classify them into three sorts – small-scale $(n \le 10)$, middle-scale $(10 < n \le 50)$,and large-scale $(n \ge 50)$, where and n is the number of processing node. We chose appropriate population sizes for each sort. Small-scale would have 100 individuals. Middle-scale would have 300 individuals. Large-scale would have 500 individuals. We also found well performance for GGAFASs with crossover rate r_c = 1, mutation rate r_m = 0.05, and generation=30. The experimental data listed in Table 2**.**

In Table 2, symbol '—' indicates that the calculation of the final reliability is intractable when n > 20.The data met the exact values shown in Table 1 are underlined, and the most approximate is shown with asterisk (*). The meanings of columns are illustrated as below:

1. Exact_Max : the maximum of exact value at the last generation in 30 trials.
2. Max_FT : the maximum fitness at the last generation in 30 trials.
3. Min_FT : the minimum fitness at the last generation in 30 trials.
4. FT_Avg : the average fitness at the last generation for 30 trials.

We observed both GGAFAS 3 and GGAFAS 4 could find a global optimal in the small-scale, but GGAFAS 1 and GGAFAS 2 sometimes could find only a local optimal. In the middle-scale and large-scale, the GGAFAS 3 could performer best, the second is GGAFAS 4, the third is GGAFAS 1, and the last is GGAFAS 2. From Table 1, we observed the heuristic algorithm 1 could get better solutions, the heuristic 3 is the second, and the heuristic 2 is the poor. We compared GGAFASs with heuristics as Figure 3 shown; most of GGAFASs would get better solutions than heuristics besides GGAFAS 2. The performance of GGAFAS 3 and GGAFAS 4 are excellent in middle-scale and large-scale.

We presented the relative errors of all cases in small-scale in Table 3.

Table 2 the experimental data of GGAFASs under $r_c = 1$ and $r_m = 0.05$

		GGAFAS 1				GGAFAS 2			
n	m	Exact_Max	Max_FT	Min_FT	FT_Avg	Exact_Max	Max_FT	Min_FT	FT_Avg
6	4	0.85344	—	—	—	0.80948	—	—	—
7	3	0.89544	—	—	—	0.874752	—	—	—
8	3	0.90272	—	—	—	0.90272	—	—	—
9	3	0.951824	—	—	—	0.938336	—	—	—
10	3	0.945002	—	—	—	0.927336	—	—	—
15	6	0.917781	—	—	—	0.881592	—	—	—
20	13	0.927262	—	—	—	0.895543	—	—	—
30	19	—	0.919642	0.862682	0.8959	—	0.847595	0.752405	0.807822
50	22	—	0.905096	0.864982	0.886874	—	0.847053	0.74346	0.796436
100	40	—	0.864804	0.811001	0.83788	—	0.775108	0.586768	0.669208
200	25	—	0.949631	0.909122	0.937051	—	0.842965	0.725463	0.775536
		GGAFAS 3				GGAFAS 4			
n	m	Exact_Max	Max_FT	Min_FT	FT_Avg	Exact_Max	Max_FT	Min_FT	FT_Avg
6	4	0.85344	—	—	—	0.85344	—	—	—
7	3	0.89544	—	—	—	0.89544	—	—	—
8	3	0.90272	—	—	—	0.90272	—	—	—
9	3	0.953907	—	—	—	0.953907	—	—	—
10	3	0.945002	—	—	—	0.945002	—	—	—
15	6	0.939392*	—	—	—	0.927822	—	—	—
20	13	0.951664*	—	—	—	0.933935	—	—	—
30	19	—	0.955008*	0.936481	0.945964	—	0.931501	0.905442	0.919903
50	22	—	0.947798*	0.929994	0.940892	—	0.923432	0.89685	0.909386
100	40	—	0.919355*	0.897167	0.90743	—	0.882147	0.850134	0.866312
200	25	—	0.976545*	0.970773	0.973711	—	0.966712	0.932327	0.952358

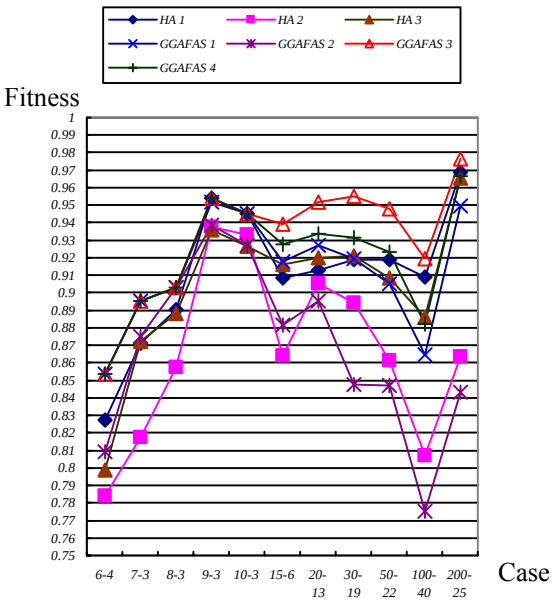

Figure 3 the experimental results of each GGAFAS compare with each EA.

Table 3 the relative errors of all cases in small-scale.

err (%)	6-4	7-3	8-3	9-3	10-3
HA 1	-3.03	-2.71	-1.39	0	0
HA 2	-8.14	-8.69	-5.01	-1.71	-1.23
HA 3	-6.43	-2.61	-1.63	-1.92	-1.97
GGAFAS1	0	0	0	-0.22	0
GGAFAS 2	-5.15	-2.13	0	-1.63	-1.87
GGAFAS3	0	0	0	0	0
GGAFAS 4	0	0	0	0	0

5. Conclusion

In this paper, we used the concept of grouping genetic algorithm to solve the file assignment in a star-topology network problem subject to constraints of nodes' capacity and files' copies. We have shown that this problem belongs to NP-hard. In the proposed algorithms, we developed the repair procedure to tune those infeasible chromosomes to feasible ones. We also have compared our proposed algorithms with exact and heuristic algorithms.

The exact algorithm usually can find the exact solution in the small-scale; however, it takes much execution time. The heuristic algorithms generally take less execution time than GGAFASs, but it is only able to find out a near-optimal in the middle-scale and large-scale. Experimental results have shown that GGAFASs led to better solutions than heuristics in almost all cases.

6. Reference

[1] R. Ramakumar, *Engineering Reliability: Fundamentals and Applications*, Prentice-Hall, New Jersey, 1993.

[2] D. J. Chen, K. L. Ku, M. S. Chang, and M. S. Lin, "The Distributed Program Reliability on a Star Topology: Efficient Algorithm and Approximate Solution", *Computers & Operations Research*, Vol. 27, 2000,pp. 129-142.

[3] J. S. Provan and M. O. Ball, "Disjoint products and efficient computation of reliability", *Operations Research*, Vol.26, No. 5, 1988, pp.703-715.

[4] G. J. Hwang and S. S. Tseng, "A Heuristic Task Assignment Algorithm to Maximize Reliability of a Distributed System", *IEEE Trans. Reliability*, Vol. 42. No. 3, 1993 Sep, pp.408-415.

[5] C. S. Raghavendra, S. Hariri and V. K. P. Kumar, "Reliability Analysis in Distributed Systems", *IEEE Trans. Reliability*, Vol. 37. No3, 1988 Mar, pp.352-258.

[6] J. R. Kim and M. Gen, "GA-based Reliability Design: State-of-the-Art Survey", *Computer and Industrial Engineering*, 37, 1999, pp.151-155.

[7] J. Campbell and L. Painton, "Genetic Algorithm in Optimization of System Reliability", *IEEE Trans. Reliability*, Vol. 44. No. 2, 1995 Jun, pp.172-178.

[8] S. T. Cheng, "Topological Optimization of a Reliable Communication Network", *IEEE Trans. Reliability*, Vol. 47. No. 3, 1998 Sep, pp.225-233.

[9] A. Kumar, R. M. Pathak, and Y. P.Gupta, "Genetic-Algorithm-Based Reliability Optimization for Computer Network Expansion", *IEEE Trans. Reliability*, Vol. 44. No1, 1995 Mar, pp.63-72.

[10] A. E. Smith and D. W. Coit, "Reliability Optimization of Series-Parallel Systems Using a Genetic Algorithm", *IEEE Trans. Reliability*, Vol. 45. No. 2, 1996 Jun, pp.254-260.

[11] C. C. Chiu, R. S. Chen, and Y. S. Yeh, "A Genetic Algorithm for K-node Set Reliability Optimization with Capacity Constraint of a Distributed System", *Proc. Natl. Sci. Counc. ROC(A)*, Vol. 25. No. 1, 2001, pp.27-34.

[12] G. Legault and S. Pierre, "A Genetic Algorithm for Designing Distributed Computer Network Topologies", *IEEE Trans. System, Man, and Cybernetics*, Vol. 28. No.2, 1998 Apr, pp.249-258.

[13] A. Kumar, R. M. Pathak, and Y. P. Gupta, "Genetic Algorithm Based Approach for File Allocation on Distributed Systems", *Computer Ops. Res*, Vol. 22. No 1, 1995, pp.41-54.

[14] D. J. Chen, K. M. Kavi, and P. Y. Chang, "Multimedia File Allocation on VC Networks Using Multipath Routing", *IEEE Trans. Computers,* Vol. 49. No. 9, 2000 Sep, pp.971-977.

[15] D.S. Johnson and M. R. Garey, *Computer and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, 1979.

[16] M. Gen, and R. Cheng, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York, 1997

[17] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd eds., Springer-Verlag, New York, 1999.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[19] C. L. Hwang, F. A. Tillman, V. R. Prasad, and W. Kuo, *Optimal Reliability Design Fundamentals and Applications*, Cambridge, 2001.

[20] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing", *Journal of Heuristics*, 2(1), Boston, 1996, pp5-30.

[21] C. J. Colburn, *The Combinatorics of Network Reliability*, Oxford, 1987, pp9-11.

[22] D. R. Shier, Network Reliability and Algebraic Structures, Oxford, 1991, pp8-10.