Journal of the Korean Society of Marine Engineering, Vol. 38, No. 10, pp. 1340~1346, 2014
J. Korean Soc. of Marine Engineering (JKOSME)
http://dx.doi.org/10.5916/jkosme.2014.38.10.1340

ISSN 2234-7925 (Print)
ISSN 2234-8352 (Online)
Original Paper

Metaheuristics for reliable server assignment problems

Kil-Woong Jang[1] · Jae-Hwan Kim†

(Received October 22, 2014 ; Revised December 24, 2014 ; Accepted December 30, 2014)

Abstract: Previous studies of reliable server assignment considered only to assign the same cost of server, that is, homogeneous servers. In this paper, we generally deal with reliable server assignment with different server costs, i.e., heterogeneous servers. We formulate this problem as a nonlinear integer programming mathematically. Our problem is defined as determining a deployment of heterogeneous servers to maximize a measure of service availability. We propose two metaheuristic algorithms (tabu search and particle swarm optimization) for solving the problem of reliable server assignment. From the computational results, we notice that our tabu search outstandingly outperforms particle swarm optimization for all test problems. In terms of solution quality and computing time, the proposed method is recommended as a promising metaheuristic for a kind of reliability optimization problems including reliable sever assignment and e-Navigation system.

Keywords: Reliable server assignment, Tabu search, Particle swarm optimization

Nomenclature

α	critical service level i.e., predetermined fraction of the operational nodes
K	number of simulation replications using Crude Monte Carlo sampling to calculate CSR
n	number of nodes
c_i	cost of deploying and maintaining a server at node i
re_i, rn_i, rs_i	reliability of edge, node, and server, respectively
s_i	binary server assignment decision variable indicating whether a server is assigned to node i ($s_i = 1$) or not ($s_i = 0$)
C	budget limit
$iter$	number of iterations of tabu search
$Stopiter$	the predetermined maximum number of iterations
x_0	initial feasible solution
x_c	current solution
x_{bf}	best feasible solution found so far
$x_{b\inf}$	best infesible solution found so far

Acronums

RSA	reliable server assignment
TS	tabu search
PSO	particle swarm optimization
CSR	critical service rate defined in Konak *et al.* **[1]**
ACO	ant colony optimization
CSA	clonal selection algorithm

1. Introduction

RSA problems in networks are defined as determining a deployment of servers to maximize a measure of service availability as follows.

$$\begin{aligned} &\text{Maximize } z = CSR \\ &s.t \quad \sum_{i=1}^{n} c_i s_i \le C \\ &s_i \in \{0,\ 1\} \end{aligned} \tag{1}$$

where s_i is the binary server assignment decision variable indicating whether a server is assigned to node i (s_i =1) or not (s_i =0). The reliability measure of CSR (critical service rate)

† Corresponding Author (ORCID: http://orcid.org/0000-0002-8248-6325): Department of Data Information, Korea Maritime and Ocean University, Dongsam-dong, Yeongdo-gu, Busan 606-791, Korea, E-mail: jhkim@kmou.ac.kr, Tel: 051-410-4374
1 Department of Data Information, E-mail: jangkw@kmou.ac.kr, Tel: 051-410-4375

is originally proposed by Konak *et. al.* [1]. They defined the CSR as the probability that more than a predetermined fraction (α) of the operational nodes have access to at least one server in case of a component failure.

The CSR is as follows:

$$CSR(\mathbf{S}) = \Pr\left(\frac{\sum_{i \in V} \delta_i (\mathbf{X} \mid \mathbf{S})}{\sum_{i \in V} v_i (\mathbf{X})} \geq \alpha \right) \qquad (2)$$

where $\mathbf{X}$ denotes a state vector of the network such that at least one node is operational, $\delta_i (\mathbf{X} \mid \mathbf{S})$ =1 if there exists at least one path between node i and a server node, and $\delta_i (\mathbf{X} \mid \mathbf{S})$ =0 otherwise, $v_i (\mathbf{X})$ =1 if node i is operational in state $\mathbf{X}$, and $v_i (\mathbf{X})$ =0 otherwise. The CSR measure defined in (1) considers reachability of servers only by operational nodes because it is assumed that , when a node fails, the users of that node cannot access network services. In practice, a network continues to operate even though several of its nodes fail or become disconnected, and CSR takes into account this operational aspect of networks. The RSA problem defined in this paper is closely related with the *p*-median problem. The deterministic p-median problem was originally proposed by Hakimi [2]. Several authors([3]-[8]) dealt with the reliable p-median problem, which is concerned with the service unavailability due to the infrastructure disruptions or component failures. The network reliability optimization problems are known as NP-hard [9].

Melachrinoud is *et al*. [5] suggested a similar problem on tree networks with unreliable edges. Note that, unlike general networks as considered in this paper, on a tree network, it is computationally feasible to compute this objective function.

In the meanwhile, Eiselt *et al*. [10] proposed the reliable *p*-median problem on general networks. However, they dealt with a case of when only a single edge failure is considered at a time and extended this approach to networks with unreliable nodes. Berman *et al.* [11] suggested a reliable *p*-median on distribution networks to minimize the expected amount of satisfied demand. Nakaniwa *et al.* [12] considered the optimal mirror Web server assignment problem considering reliability. In this problem, edges are perfectly reliable and nodes are subject to failure. The RSA problem with the new reliability measure of CSR was solution methods by three nature-inspired metaheuristics, that is, ACO, PSO, and CSA.

However, the server cost c_i was fixed to the same value of c for all servers in Konak *et al.* [1]. That is, they dealt with only the case of homogeneous severs. In this paper, we generally tackle the above RSA problem with the different value of c_i 's for each node of server, i.e., heterogeneous servers. We also propose two metaheuristic algorithm (TS and PSO) for solving the RSA with heterogeneous servers. From computational results, we noticed that TS outstandingly outperforms PSO for all test problems.

The rest of this paper is organized as follows. Two metaheuristic algorithms for solving the RSA problem is developed in Section 2. In Section 3, we illustrate three examples of the RSA problem. In Section 4, we evaluate the performance of two proposed algorithm through the computational efforts. Finally, conclusions and future research are discussed in Section 5.

2. Solution Methods

Konak *et al.* [1] developed three metaheuristic algorithms for the RSA problem, that is, ACO, PSO, and CSA. In this paper, we propose an efficient TS for the RSA problem. Also we compare our TS with PSO which is the best for this problem in the literature.

2.1 TS Algorithm

The TS algorithm, first proposed by Glover [13], is a metaheuristic method to expand its search beyond local optimality using adaptive memory. The adaptive memory is a mechanism based on the tabu list of prohibited moves. The tabu list is one of the mechanism to prevent cycling and guide the search towards unexplored region of the solution space. The TS generally adopts the penalty function to allow to explore the search towards the attractive infeasible region. The TS has been successfully applied to many combinatorial optimization problems such as vehicle routing problems, travelling salesman problems, time tabling problems, and resource allocation problems, etc.

<u>General steps of TS</u>

The general steps of TS can be summarized as follows:

Step 0: (Initialization) Set *iter*=0, and initialize tabu list.

Step 1: Randomly generate the initial solution x_0.

Set $x_c = x_0$ and $x_{bf} = x_{b\inf} = x_c$.

Step 2: a. Set *iter*=*iter*+1.

Generate the neighborhood of the current solution x_c by the defined move.

b. Select the best neighborhood which is not in the tabu list. Store it as the new current solution x_c. Update the tabu list.

Step 3: If x_c is feasible, then go to step 4.

Else go to step 5.

Step 4: If $CSR(x_c) > CSR(x_{bf})$, then set $x_{bf} = x_c$, *iter*=0, and initialize the tabu list. Go to step 2.a.

Else go to step 6.

Step 5: If $CSR_p(x_c) > CSR_p(x_{b\inf})$ then $x_{b\inf} = x_c$, *iter*=0, and initialize the tabu list. Go to step 2.a.

Step 6: If *iter* > *Stopiter*, then go to step 7.

Else go to Step 2.a.

Step 7: (End) Stop with the best feasible solution found so far.

Penalty function

TS generally adopts the penalty function to allow to explore the search towards the promising infeasible region. Our TS adopts the following penalty function CSR_p (see **[14]**) to handle the budget constraint (C).

$$CSR_p = \begin{cases} CSR \times \left[\dfrac{C}{\sum_{i=1}^{n} c_i s_i} \right]^2 & \text{if } \sum_{i=1}^{n} c_i s_i > C \\ CSR & \text{otherwise} \end{cases} \tag{3}$$

The exact computation of CSR is intractable for the size of the problem studied in this paper. Therefore, Crude Monte Carlo simulation is used to evaluate the objective function of solutions created by the metaheuristic approaches.

Defined move to generate the neighborhood

There are the following types of defined move in our TS.

i) Adding a server node to the current solution
ii) Subtracting a server node from the current solution
iii) Replacing a server node with other nodes

2.2 PSO Algorithm

PSO is a computational intelligence metaheuristic, originally developed by Kennedy and Eberthart **[15][16]**, which was inspired by social behavior of fish schooling or bird flocking. Like evolutionary and genetic algorithms, PSO is a population-based search algorithm, i.e., it moves from a set of solutions (particle's positions) to another set of solutions. The particles move through a D-dimensional space and each particle has a velocity that acts as an operator to obtain a new set of solutions. The particles adjust their movements depending on both their own experience and the population's experience. At each iteration a particle moves in a direction computed from its best visited position and the best visited position of all the particles in its neighborhood. Among the several variants of PSO, the global variant considers the neighborhood as the whole population, called the swarm, which enables the global sharing of information.

The basic elements of the PSO techniques are particle, population, velocity, inertia weight, individual best, global, learning coefficients and stopping criteria. We refer interested readers to Konak *et al*. **[1]** for the detailed steps of PSO for the RSA problem.

3. Examples

3.1 Example 1

For example 1, consider the following network in **Figure 1**.

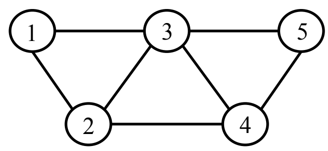

Figure 1: Network structure

The input data are given by

Node	1	2	3	4	5	C
c_i	2	3	2	3	2	6
rs_i	0.75	0.8	0.75	0.8	0.75	

where re_i =0.8 and rn_i =1.0.

Our problem is as follows:

$$\text{Maximize } z = CSR$$
$$s.t \quad 2s_1 + 3s_2 + 2s_3 + 3s_4 + 2s_5 \le 6$$
$$s_i \in \{0, 1\}$$

The reliability re_i of each edge is 0.8. In this example, we noticed that the global optimal solution is {1, 3, 5} in **Figure 2**. The value of CSR is 0.95194. The value of K, α, and *Stopiter* are 8000, 1.0, and 5, respectively. Our TS is executed on compatible with a Pentium IV 3.0 GHz. The computing time is 0.987 seconds.

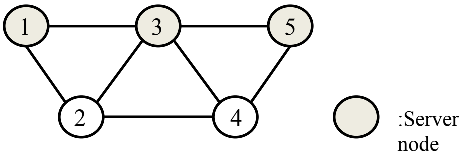

Figure 2: The optimal solution of TS

3.2 Example 2

The input data are given by

Node	1	2	3	4	5	6	7	8	9	10	11	C
c_i	3	4	5	3	4	5	3	4	5	3	4	9
rs_i	0.75	0.8	0.85	0.75	0.8	0.85	0.75	0.8	0.85	0.75	0.8	

The example 2 is same to Konak *et al.* **[1]** (**Figure 3**).

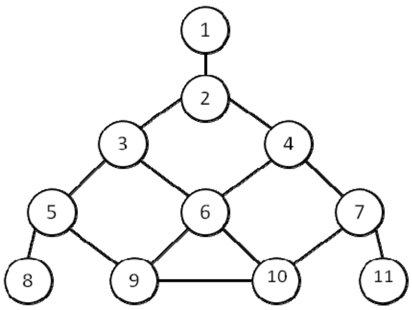

Figure 3: The network structure of Konak *et al.* **[1]**

Our problem is as follows:

Maximize $z = CSR$

s.t

$$3s_1+4s_2+5s_3+3s_4+4s_5+5s_6+3s_7+4s_8+5s_9+3s_{10}+4s_{11}\le 7$$

$$s_i \in \{0,\ 1\}$$

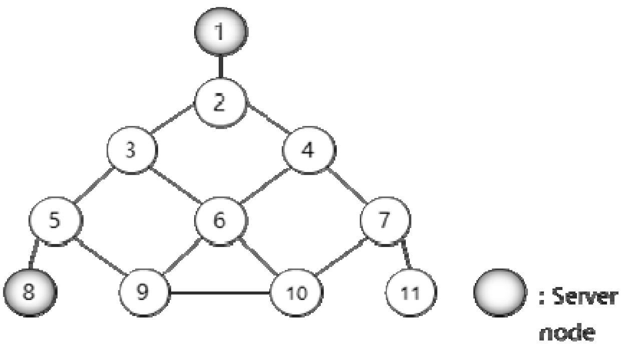

Figure 4: The optimal solution of TS

The optimal server node is {1, 8} shown in **Figure 4**, and the value of CSR is 0.67475. The value of K, α, and *Stopiter* are 8000, 1.0, and 5, respectively. The computing time is 3.735 seconds.

To evaluate the performance of our TS for the general RSA problem with different costs of deploying each server, we employed the same cost of each server for Konak *et al.* **[1]**. For example, the input data for the case of α=0.9 are given by

Node	1	2	3	4	5	6	7	8	9	10	11	C
c_i	2	2	2	2	2	2	2	2	2	2	2	4

The global optimal solution of Konak *et al.* **[1]** is either {5, 11} or {7, 8} with the exact CSR=0.975395. We noticed that our TS finds one of the global optimum successfully.

3.3 Example 3

The example 3 was randomly generated as **Figure 5**. The size of node(n) is 20, and the input data are given in **Table 1**.

The problem is as follows :

Maximize $z = CSR$

$$s.t \quad \sum_{i=1}^{20} c_i s_i \le 12$$

$$s_i \in \{0,\ 1\}$$

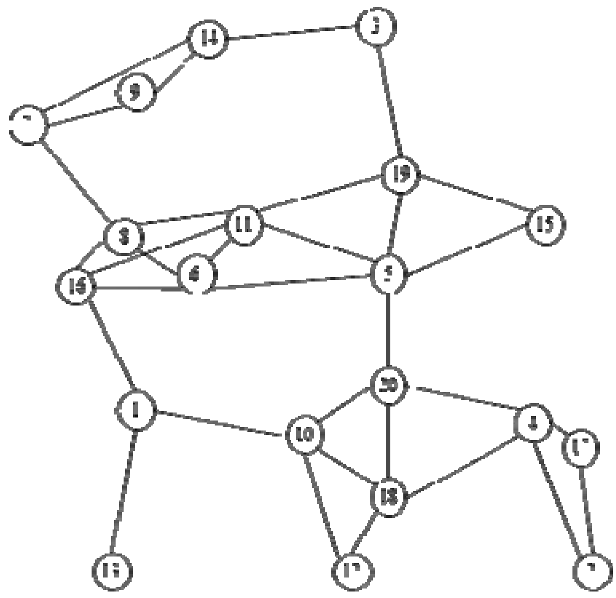

Figure 5: Randomly generated network (n=20)

Table 1: The input data of example 3

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	C
c_i	3	4	5	6	3	4	5	6	3	4	5	6	3	4	5	6	3	4	5	6	12
rs_i	.75	.8	.85	.9	.75	.8	.85	.9	.75	.8	.85	.9	.75	.8	.85	.9	.75	.8	.85	.9	

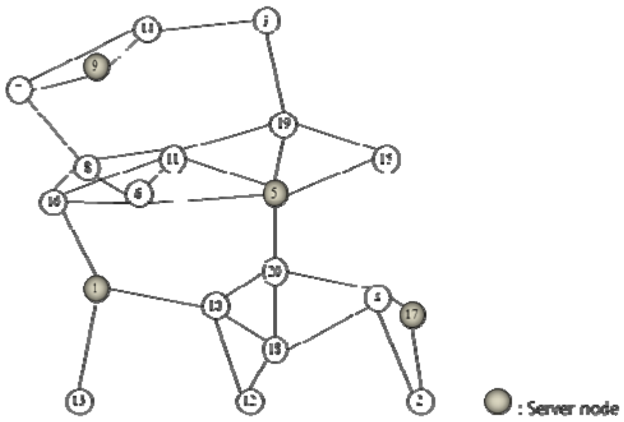

Figure 6: The optimal solution of TS

The optimal server node is {1, 5, 9, 17} shown in **Figure 6**, and the value of CSR, the measure of reliability, is 0.95104. The value of K, α, and *Stopiter* are 8000, 1.0, and 5, respectively. The computing time is 313.75 seconds.

4. Computational Results

To evaluate the performance of TS and PSO, we conducted the computational experiments for three examples of the Section 3. Each example was composed of 80 test problems, totally 240 test problems. Two algorithms (PSO and TS) were coded in C/C++ programming language, and experiments were performed on a Pentium IV 3.0 GHz PC. Performances of PSO and TS are assessed in terms of the following average relative error (A), maximum relative error (M), optimality rate (O) and average execution time (T).

$$A = \frac{1}{10}\sum_{j=1}^{10}\frac{\left(R_j^* - R_j\right)}{R_j^*}$$

$$M = \max\left\{\frac{\left(R_j^* - R_j\right)}{R_j^*}\right\}, \text{ for } j=1, 2 \ldots, 10$$

O = the number of times (out of 10 problems) that each method yields the best solution.

R_j = the system reliability (CSR) obtained by each method for each test problem j.

R_j^* = the best system reliability obtained by both of PSO and TS.

In our experiments, the stopping criterion of TS was defined as 5 iterations without finding an improvement in the best feasible solution, and PSO generated 300 populations for the initial solution. Each method was applied 10 times with different starting initial solution for each test problem.

The computational results for example 1, example 2, and example 3 are summarized in **Table 2**, **Table 3**, and **Table 4**, respectively. From the computational results for example 1, PSO was same to TS in terms of the quality of the solution, even though the computing time of PSO was almost 6 times as much as that of TS. However, for example 2 and 3, the performance of PSO was very poor than that of TS in terms of the quality of the solution and computing time. From the computational results, we noticed that our TS outstandingly outperforms PSO for all test problems.

5. Conclusions

Konak *et al.* **[1]** originally proposed the RSA problem using the new reliability measure of CSR. They also suggested the solution methods by three nature-inspired metaheuristics, that is, ACO, PSO, and CSA. However, the server cost c_i was fixed to c for all servers in Konak *et al.* **[1]**. That is, they considered only the case of homogeneous severs of the RSA problem. In this paper, we generally tackled the above RSA problem with different c_i's for each node of server, i.e., heterogeneous servers, and proposed an efficient TS for the RSA problem. Also we compared our TS with PSO, which is the best for this problem in the literature, in terms of the quality of the solution and the computing time. From the computational results, we noticed that our TS outstandingly out performs PSO for all test problems.

In terms of solution quality and the computing time, our TS is recommended as a promising metaheuristic for a kind of reliability optimization problems including the RSA problem.

Table 2: Computational results for example 1

No.	(*n*, *rn*, α, *re*)	PSO				TS			
		A	M	O	T	A	M	O	T
1	(5, 1.0, 0.9, 0.8)	0.0	0.0	10/10	6.74	0.0	0.0	10/10	1.75
2	(5, 1.0, 0.9, 0.9)	0.0	0.0	10/10	6.35	0.0	0.0	10/10	1.67
3	(5, 1.0, 1.0, 0.8)	0.0	0.0	10/10	6.69	0.0	0.0	10/10	1.72
4	(5, 1.0, 1.0, 0.9)	0.0	0.0	10/10	6.44	0.0	0.0	10/10	1.66
5	(5, .95, 0.9, 0.8)	0.0	0.0	10/10	6.66	0.0	0.0	10/10	1.75
6	(5, .95, 0.9, 0.9)	0.0	0.0	10/10	6.39	0.0	0.0	10/10	1.65
7	(5, .95, 1.0, 0.8)	0.0	0.0	10/10	6.67	0.0	0.0	10/10	1.72
8	(5, .95, 1.0, 0.9)	0.0	0.0	10/10	6.37	0.0	0.0	10/10	1.66

Table 3: Computational results for example 2

No.	(*n*, *rn*, α, *re*)	PSO				TS			
		A	M	O	T	A	M	O	T
1	(11, 1.0, 0.9, 0.8)	0.0016	0.0074	3/10	19.16	0.0	0.0	10/10	10.63
2	(11, 1.0, 0.9, 0.9)	0.001	0.0029	2/10	18.47	0.0	0.0	10/10	10.90
3	(11, 1.0, 1.0, 0.8)	0.0747	0.1262	4/10	19.13	0.0	0.0	10/10	10.80
4	(11, 1.0, 1.0, 0.9)	0.069	0.0805	1/10	18.32	0.0	0.0	10/10	10.27
5	(11, .95, 0.9, 0.8)	0.0039	0.0316	3/10	18.46	0.0	0.0	10/10	10.60
6	(11, .95, 0.9, 0.9)	0.0041	0.0072	4/10	17.83	0.0	0.0	10/10	9.83
7	(11, .95, 1.0, 0.8)	0.0585	0.1197	5/10	18.19	0.0	0.0	10/10	10.10
8	(11, .95, 1.0, 0.9)	0.027	0.0934	7/10	17.69	0.0	0.0	10/10	9.77

Table 4: Computational results for example 3

No.	(*n*, *rn*, α, *re*)	PSO				TS			
		A	M	O	T	A	M	O	T
1	(20, 1.0, 0.9, 0.8)	0.0353	0.0388	0/10	147.32	0.0194	0.0324	4/10	57.5
2	(20, 1.0, 0.9, 0.9)	0.0117	0.0128	0/10	139.4	0.0038	0.0128	7/10	75.39
3	(20, 1.0, 1.0, 0.8)	0.0523	0.0964	1/10	147.98	0.0	0.0	10/10	55.03
4	(20, 1.0, 1.0, 0.9)	0.0173	0.0373	2/10	140.1	0.0	0.0	10/10	56.85
5	(20, .95, 0.9, 0.8)	0.0606	0.0763	0/10	134.6	0.0	0.0	10/10	62.61
6	(20, .95, 0.9, 0.9)	0.0291	0.0452	2/10	134.32	0.0	0.0	10/10	51.48
7	(20, .95, 1.0, 0.8)	0.0617	0.0968	1/10	145.31	0.0	0.0	10/10	56.38
8	(20, .95, 1.0, 0.9)	0.0845	0.1129	0/10	131.83	0.0	0.0	10/10	56.77

To achieve a better solution quality, the development of more powerful metaheuristics and hybrid metaheuristics for the RSA problem would be performed in the future research.

Acknowledgements

This paper is extended and updated from the short version that appeared in the Proceedings of the International symposium on Marine Engineering and Technology (ISMT 2014), held at Paradise Hotel, Busan, Korea on September 17-19, 2014.

References

[1] A. Konak and S. Kulturel-Konak, "Reliable sever assignment in networks using nature inspired metaheuristics," IEEE Transactions On Reliability, vol. 60, no. 2, pp. 381-393, 2011.

[2] S. L. Hakimi, "Optimum locations of switching centers and absolute centers and median of graph," Operations Research, vol. 12, no. 3, pp .450-459, 1964.

[3] Z. Drener, "Heuristic solution methods for two location problems with unreliable facilities," Journal of the Operational Research Society, vol. 38, no. 6, pp. 509-514, 1987.

[4] L. D. Nel and C. J. Colbourn, “Locating a broadcast facility in an unreliable network,” INFOR, vol. 28, no. 4, pp. 363-379, 1990.

[5] E. Melachrinoudis and M. E. Helander, “A single facility location problem on a tree with unreliable edges,” Networks, vol. 27, no. 3, pp. 219-237, 1996.

[6] A. Nakaniwa, J. Takahashi, H. ebara, and H. Okada, “Reliability-based optimal allocation of mirror servers for internet,” IEEE Global Telecommunications Conference, pp. 1571-1577, 2000.

[7] L. V. Snyder and M. S. Daskin, “Reliability models for facility location: The expected failure cost case,” Transportation Science, vol. 39, no. 3, pp. 400-416, 2005.

[8] H. A. Eiselt, M. Gendreau, and G. Laporte, “Location of facilities on a network subject to a single-edge failure,” Networks, vol. 22, no. 3, pp. 231-246. 1992.

[9] M. Ball, “Complexity of network reliability computations,” Networks, vol. 10, no. 2, pp. 153-165, 1980.

[10] H. A. Eiselt, M. Gendreau, and G. Laporte, “Optimal location of facilities on a network with an unreliable node or link,” Information Processing Letters, vol. 58, no. 2, pp. 71-74, 1996.

[11] O. Berman, Z. Drezner, and G. O. Wesolowsky, “Locating service facilities whose reliability is distance dependent,” Computers and Operations Research, vol. 30, no. 11, pp. 1683-1695, 2003.

[12] A. Nakaniwa, J. Takahashi, H. Ebara, and H. Okada, “Reliability-based mirroring of servers in distributed networks,” IEICE Transactions on Communications, vol. E85-B, no. 2, pp. 540-549, 2002.

[13] F. Glover, “Future paths for integer programming and links to artificial intelligence,” Computers and Operations Research, vol. 13, no. 5, pp. 533-549, 1986

[14] W. C. Yeh, “A two-stage discrete PSO for the problem of multiple multi-level redundancy allocation in series systems,” Expert Systems with Applications, vol. 36, no. 5, pp. 9192-9200, 2009.

[15] R. C. Eberhart and J. Kennedy, “A new optimizer using particle swarwm theory,” Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39-43, 1995.

[16] J. Kennedy and R. C. Eberhart, “Discrete binary version of the particle swarm algorithm,” Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 4104-4108, 1997.