Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

SIMULATION OPTIMIZATION EMBEDDED PARTICLE SWARM OPTIMIZATION FOR RELIABLE SERVER ASSIGNMENT

Sadan Kulturel-Konak

Management Information Systems
Pennsylvania State University Berks
Reading, PA 19610, USA

Abdullah Konak

Information Sciences and Technology
Pennsylvania State University Berks
Reading, PA 19610, USA

ABSTRACT

A reliable server assignment (RSA) problem in networks is defined as determining a deployment of identical servers to maximize a measure of service availability. In networks, the communication between a client and a server might be interrupted since the server itself is offline or unreachable as a result of catastrophic network failures. In this paper, a novel simulation optimization approach is developed based on a Monte Carlo (MC) simulation and embedded into Particle Swarm Optimization (PSO) to solve the RSA problem. The experimental results show that the simulation optimization embedded PSO is an effective heuristic method.

1 INTRODUCTION

The availability and accessibility of servers in communications networks are crucial. For example, a computer network will not properly function if the servers providing the Domain Name System (DNS) service to the network are inaccessible. Therefore, the Reliable Server Assignment (RSA) problem is an important problem to tackle with. In this paper, the RSA problem in communications networks is defined and solved using a simulation optimization approach based on Particle Swarm Optimization (PSO), which is a recent nature inspired metaheuristic method. The term "simulation optimization" is usually used for the problems where the relationship between the input variables and the system output is very complex in order to estimate the output for a given setting of the input variables. Simulation optimization approaches are among the most frequently used tools to solve many real-world problems. Andradottir (1998) and Fu (2002) surveyed common approaches to simulation optimization. Discussion about the latest advancements in simulation optimization can also be found in April et al. (2003), Law and McComas (2000 and 2002), Ling, Liang and Da-zhong (2003), Olafsson and Kim (2002), and Swisher et al. (2000).

Consider an undirected network $G=(N, E)$, where $N=\{1,\ldots,n\}$ is the node set, $E=\{(i, j)\}$ is the edge set, and $(i, j)\equiv(j, i)$. The cost of deploying and maintaining a server at node i is given as c_i. For the successful operation of the network, each node must have access to at least one server. However, both nodes and edges of the network are subject to failure with known probabilities. Let $r_{(i, j)}$ be the reliability of edge (i, j) and r_i be the reliability of node i. When edge (i, j) fails, the communication between nodes i and j is interrupted if it cannot be rerouted through an alternative path. When a node fails, all of its incident edges become inoperative. Therefore, if a server node fails, the server located on this node becomes unreachable by all users. The RSA problem is defined as determining a deployment of servers to maximize a measure of service availability while not exceeding the available budget and is formulated as follows:

$$\max \quad z = R(\{s_1, \ldots, s_n\} \mid G)$$

$$\sum_{i=1}^{n} c_i s_i \leq C$$

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

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), $R(\ )$ is a measure of the service availability, and C is the allowable budget for the server deployment and maintenance in the network.

Section 2 gives a summary of the network reliability and availability measures defined in the literature. In this paper, a new reliability measure, which is called *critical service rate* (CSR) and proposed by Kulturel-Konak and Konak (2008), is used. CSR is defined as the probability that more than a predetermined fraction (α) of the operational nodes has access to at least one server in case of a catastrophic component failure as follows:

$$\text{CSR} = \Pr\{\frac{\sum_{i \in N} \tau_i(\mathbf{X})}{\sum_{i \in N} \delta_i(\mathbf{X})} \geq \alpha\}$$

where $\mathbf{X}$ denotes a state vector of the network; $\tau_i(\mathbf{X})$=1 if there exists at least one path between node i and a server node, and $\tau_i(\mathbf{X})$=0 otherwise; $\delta_i(\mathbf{X})$=1 if node i is operational in state $\mathbf{X}$, $\delta_i(\mathbf{X})$=0 otherwise; and α is the critical service level.

The organization of the paper is as follows: Section 2 provides the background on the RSA problem. Then, the novel simulation optimization embedded PSO is introduced in Section 3 followed by experimental results in Section 4. Finally, Section 5 concludes the paper and presents the future directions in the area of RSA.

2 BACKGROUND

The RSA problem that is defined in the previous section is closely related with the *p*-median problem. The deterministic *p*-median problem is originally defined by Hakimi (1964) as locating *p* identical services at *p* distinct nodes of a network to minimize the total weighted distance between nodes and the closest servers. Several researchers (Drezner 1987; Nel and Colbourn 1990; Melachrinoudis and Helander 1996; Nakaniwa et al. 2000; Snyder and Daskin 2005; Eiselt, Gendreau and Laporte 1992) study the reliable *p*-median problem, which is to try to minimize the service unavailability due to the infrastructure disruptions or component failures. In the reliable *p*-median problem, the nodes and/or edges of the underlying network are subject to failure with known probabilities. The overall objective is to maximize service availability, which is usually expressed in the form of an expected value of service level or a probability that the service is reachable. Therefore, the evaluation of the objective function requires techniques from network reliability analysis which is concerned with computing the probability that a network maintains a desired connectivity. However, evaluating this type of a stochastic function is a daunting task requiring considering all possible failure scenarios. In fact, most network reliability problems are *NP*-hard (Ball 1980). Because of its difficulty, in the literature the reliable *p*-median problem has usually been studied either for special cases or with assumptions to make the evaluation of the objective function computationally feasible.

Nel and Coulbourn (1990) formulate the reliable *p*-median problem to identify a single node on a network such that the expected number of nodes connected to that node is maximized in the presence of edge failures. Since computing this expected value is intractable, an efficient upper bound on the two-terminal reliability is utilized to identify promising nodes. A similar problem on tree networks with unreliable edges is studied by Melachrinoudis and Helander (1996). Unlike the general networks considered in our paper, on a tree network, it is computationally feasible to compute the objective function. Berman and Drezner (2003) also study the stochastic one-median problem on general networks

where the objective is to maximize the probability of reaching all nodes from the service center within a time threshold.

As mentioned earlier, Eiselt, Gendreau and Laporte (1992) study the reliable *p*-median problem on general networks. However, they consider a special case of the problem where only a single edge failure is considered at a time. They propose an algorithm to transform a general network into a tree network while preserving failure probabilities. Then, the problem is solved on the transformed tree network. Note that, his transformation is only possible because of the assumption that only a single node failure could occur at a time. Eiselt, Gendreau and Laporte (1996) extend this approach to networks with unreliable nodes.

Berman, Drezner and Wesolowsky (2003) define a reliable *p*-median on distribution networks to minimize the expected amount of unsatisfied demand. Nakaniwa et al. (2002) study the optimal mirror Web server assignment problem considering reliability. In their problem, network edges are perfectly reliable and nodes are subject to failure. For a given server to node assignment, it is assumed that users are served by the closest Web server. Therefore, the probability that a user could access to a particular server is computed as a function of the failure rates of the nodes on the shortest path from the user to the server. An integer programming formulation is developed to maximize the overall system reliability under cost, delay and capacity constraints. Snyder and Daskin (2005) formulate a reliable *p*-median problem where nodes and edges of the network are perfectly reliable, but service facilities fail with known probabilities. For each customer, a primary facility and a set of backup facilities are determined using a Lagrangian relaxation algorithm. Snyder and Daskin (2006) present a novel robustness measure that combines the objectives of minimizing expected cost and minimizing worst-case cost or regret, which is called *p*-robust. For many instances of the problems, finding a feasible solution and even determining whether the instance is feasible are difficult. Therefore, they first discuss a mechanism to assess the infeasibility, and then suggest a heuristic method to solve the problem.

3 SIMULATION OPTIMIZATION EMBEDDED PARTICLE SWARM OPTIMIZATION

Exactly evaluating CSR for the problems studied in this paper is not practical in reasonable CPU times. Therefore, the objective functions of solutions created by the PSO algorithm are evaluated by a Monte Carlo (MC) simulation. The PSO algorithm, the MC simulation, and a hashing method are integrated in a novel way to minimize the computational effort to efficiently evaluate candidate solutions and reduce the effect of the noise in the objective function due to the evaluation process by the MC simulation.

3.1 Particle Swarm Optimization

PSO was originally proposed by Eberhart and Kennedy (1995) based on the social behavior of species living in the form of swarms in nature. These species have the ability to exchange valuable information such as food locations in the habitat by means of simple interactions. As a result of such simple interactions among the members of a swarm, a global swarm behavior such as flocking may emerge. Similar to evolutionary algorithms, PSO is a population-based meta-heuristic algorithm, and population members are called particles. In PSO, the position of a particle in *n*-dimensional space represents a solution to a problem with *n* continuous decision variables. The current position of particle *j* in an *n*-dimensional solution space is represented by vector $\mathbf{S}_j=\{s_{j1},\ldots, s_{jn}\}$. In each iteration of PSO, particles move to new positions with the goal of discovering the optimal point in the search space. Each particle *j* moves to a new position in the solution space according to its velocity vector $\mathbf{V}_j=\{v_{j1},\ldots, v_{jn}\}$ as follows:

$$\mathbf{S}_j = \mathbf{S}_j + \mathbf{V}_j$$

Particles communicate with one another by broadcasting about the fitness of their best positions and learning about the best positions of others. Based on these interactions as well as their past experiences, particles adjust their velocities in each iteration. The velocity vector of each particle *j* is updated as follows:

$$\mathbf{V}_j = \omega\mathbf{V}_j + \varphi_1\mathbf{U}_1 \cdot (\mathbf{B}_j - \mathbf{S}_j) + \varphi_2\mathbf{U}_2 \cdot (\mathbf{G} - \mathbf{S}_j) \quad (1)$$

where $\mathbf{B}_j=\{b_{j1},\ldots, b_{jn}\}$ represents the position where particle j has had its best fitness so far and $\mathbf{G}=\{g_1,\ldots, g_n\}$ is the position of the global best fitness so far by any particle in the population, and ω is called the inertia coefficient used to balance exploration versus exploitation of the solution space. Values of ω close to 1 usually encourage exploration of new areas in the solution space, and for small values of ω such as $\omega < 0.4$, the search shifts to the exploitation mode. Parameters φ_1 and φ_2 are called cognition and social coefficients, respectively, and they are very important for facilitating convergence in PSO. Based on an empirical study (Ozcan and Mohan 1999), as well as theoretical results (Clerc and Kennedy 2002), φ_1 and φ_2 are recommended to be set such that $\varphi_1+\varphi_2 <4$. $\mathbf{U}_1$ and $\mathbf{U}_2$ are two vectors of n uniform random numbers between 0 and 1. Note that operator $(\cdot)$ in (1) indicates element-by-element vector multiplication.

The pseudo code of the generic PSO is presented in the following procedure for a maximization objective function $f(\ )$. In the pseudo code, $U(a, b)$ represents a random sample from the uniform distribution on $[a, b]$, and the initial positions of particles are randomly determined between the minimum and maximum values of each decision variable ($s_i^{\min}$ and $s_i^{\max}$, respectively). Particle velocities are also randomly initialized between the minimum and maximum velocity limits ($v_i^{\min}$ and $v_i^{\max}$, respectively). Velocity limits usually depend on the ranges of the decision variables.

Procedure Generic PSO (){
  For j=1...μ do {
    Set $s_{ji}=U(s_i^{\min}, s_i^{\max})$ for i=1,…, n
    Set $v_{ji}= U(v_i^{\min}, v_i^{\max})$ for i=1,…, n
    Set $\mathbf{B}_j=\mathbf{S}_j$ and update **G** if necessary
  }
  While (stopping criterion is not satisfied) {
    For j=1...μ do {
      Set $v_{ji}= v_{ji}+U(0, \varphi_1)\times(b_{ji}-s_{ji})+U(0, \varphi_2)\times(g_i-s_{ji})$ for i=1,…, n
      Set $s_{ji}= s_{ji}+ v_{ji}$ for i=1,…, n
      If ($f(\mathbf{S}_j)> \mathbf{B}_j$) set $\mathbf{B}_j=\mathbf{S}_j$
      If ($f(\mathbf{S}_j)> \mathbf{G}$) set $\mathbf{G}=\mathbf{S}_j$
    }
  }
Return **G**
}

As briefly described above, PSO is originally designed for problems with real-valued decision variables. In the binary PSO, which is introduced by Kennedy and Eberhart (1997), the values of decision variables are randomly determined using particle velocities and a logistic function as follows:

$$s_{ji} = \begin{cases} 1 & \text{if } U(0,1) < (1+\exp(-v_{ji}))^{-1} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

In (2), the value of a binary decision is defined using a logistic function. Other than this modification, the rest of the binary-PSO algorithm is the same with real-value PSO algorithm.

Compared to other meta-heuristic approaches such as Simulated Annealing (SA), Genetic Algorithms (GA), and Tabu Search (TS), the main advantage of PSO is its ease of implementation. PSO is a very simple algorithm to implement, but it has been proven to be effective to solve wide variety of problems. PSO does not rely on special operators (e.g., crossover in GA or move operator in SA and TS). PSO has a limited number of parameters to tune and the convergence properties of PSO with respect to various parameter settings are well studied in the literature (Ozcan and Mohan 1999). Equation (1) can be used in wide variety of problems where a solution can be represented as a vector.

3.2 Solution Construction

The binary PSO introduced in (Kennedy and Eberhart 1997) is used to develop the search algorithm in this paper. However, the RSA problem is a constrained problem and using (2) to determine the values of decision variables may result in many infeasible solutions. To address this problem, procedure Create_Solution() given in the following is used in this paper to construct feasible particle positions with the following probabilities:

$$p_j(i) = \frac{(1+\exp(-v_{ji}))^{-1}}{\sum_{k\in A}(1+\exp(-v_{jk}))^{-1}} \tag{3}$$

where A is the set of admissible nodes for server assignment and $p_j(i)$ represents the probability of assigning a server to node i while determining the new position of particle j based on $\mathbf{V}_j$ (i.e., $p_j(i)=\Pr\{s_{ji}=1\}$). A node is called admissible if a server has not been assigned to that node and there is an adequate residual budget (C') to assign a server to the node. Notice that in (3), the relative weights of the probabilities in (2) are preserved, and procedure Create_Solution() randomly selects nodes for server assignment based on their selection probabilities.

Procedure Create_Solution () {
    Set A=N, $C'=C$, $s_{ji}=0$ for each node $i\in N$
    While (A≠{ }) {
        Calculate $p_j(i)$ for each node $i\in$ A
        Randomly select node i from A with probability $p_j(i)$
        Set $s_{ji}=1$ and $C'=C'-c_i$
        Update set A
    }
    Return solution $\mathbf{S}_j$
}

3.3 Hashing to Check Solutions

In PSO, as the search converges, same solutions will appear in the population with increasing frequencies. Therefore, computationally expensive simulation might be used to evaluate same solutions again and again in the RSA problem. To prevent this, hashing is used to rapidly detect whether a solution has been previously investigated or not. Hashing has been previously proposed in (Woodruff and Zemel 1993; Battiti and Tecchiolli 1994) to record the solutions encountered during recent iterations in tabu search. All solutions investigated during a search are stored in a list, called solution list (SL). A hash table (HT) is used as a pointer to quickly access the solutions stored in SL. A second list, called collision list (CL) is used to store solutions with a hash collision. After a solution **S** is created, the hash value of the solution is calculated as follows:

$$d(\mathbf{S}) = \mathrm{mod}(\prod_{i=1}^{n}(e_i)^{s_i}, H)$$

where H is the hash size and e_i is a prime number corresponding to decision variable s_i. Hash table HT is an integer array such that HT[d(**S**)]=0 if a solution with a hash value of d(**S**) has not been searched yet, or HT[d(**S**)]=t if the first solution with a hash value of d(**S**) is the t^{th} solution in solution list SL. In other words, SL[t] stores the t^{th} evaluated solution without any hash collision. After calculating d(**S**) for a solution **S**, there are three cases possible.

Case 1: If HT[d(**S**)]=0, then **S** has not been investigated before.

Case 2: If HT[d(**S**)]=t and **S**=SL[t], then **S** has been investigated before.

Case 3: If HT[d(**S**)]=t and **S**≠SL[t], then a hash collision occurs (i.e., two different solutions have the same hash value). In this case, **S** is compared with all solutions in collision list CL. If **S** is not in CL, then it has not been searched before and added to CL.

As it is demonstrated in the computational experiments, Case 3 occurs very rarely. In most cases, therefore, it is possible to check whether a solution has been previously searched or not in $O(1)$ comparisons after calculating $d(\mathbf{S})$. The procedure of checking solution using hashing is given as follows:

```
Procedure Check_Solution (S){
    Calculate d(S)
    If (HT[d(S)]=0) {
        t=t+1
        HT[d(S)]=t and SL[t]=S
        Return TRUE
    }
    If (HT[d(S)]≠0 and SL[ HT[d(S)] ]= S) Then return FALSE
    If HT[d(S)]≠0 and SL[ HT[d(S)] ] ≠S {
        If S∉ CL {
            cl=cl+1
            CL[cl]=S
            Return TRUE
        }
        Else Return FALSE
    }
}
```

3.4 Monte Carlo Simulation and Solution Evaluation

As mentioned earlier, the objective function of the problem, CSR, is estimated using MC simulation. Evaluating solutions using simulation within an optimization algorithm has two important drawbacks. First, simulation output includes a statistical error which might affect the performance of the optimization algorithm. Second, simulation is computationally expensive, especially if a small margin of estimation error is required. To remedy these problems, a hierarchical approach to solution evaluation is used in the PSO as follows. At first, a solution is evaluated using a low number of simulation replications (K_1), and if the solution seems to be promising after this first evaluation, then it is rigorously evaluated using a higher number of replications (K_2). The best b solutions found so far during the search are maintained in a sorted list called *elitist list* (EL) such that CSR(EL[1])> CSR(EL[2])>…> CSR(EL[b]). After the first evaluation, a solution is identified as promising if the solution has a better objective function value than the worst elitist solution. After rigorously evaluating a promising solution, the solution is compared with each elitist solution, and the elitist list is updated if necessary. Finally, after the whole search terminated, all elitist solutions are evaluated again using a very high number of simulation replications (K_3).

```
Procedure Evaluate_Solution (S){
If Check_Solution(S)=TRUE {
    Evaluate S using K1 simulation replication
    If (CSR(S)> the worst CSR in EL) {
        Evaluate S using K2 replications
        Update EL by including S if necessary}
    }
}
```

3.5 A PSO Algorithm to the RSA problem

The PSO algorithm proposed in this paper is based on the binary-PSO algorithm (Kennedy and Eberhart 1997) described in Section 3.1. The differences are: (*i*) solutions are randomly constructed as opposed to being randomly generated in the binary-PSO; (*ii*) hashing is used for detecting whether a solution has been previously evaluated; and (*iii*) the objective function is evaluated using simulation. In the PSO

algorithm, the best elitist solution (EL[1]) is used as the global best position so far (i.e., **G**) to update particle velocities. The overall procedure of the PSO is given below:

Procedure PSO {
  Set SL=∅, CL=∅, EL=∅
  Set t=0, cl=0
  For j=1...μ do {
    Set v_{ji}=0 for i=1,…, n
    $\boldsymbol{S}_j$=Create_Solution()
    Evaluate_Solution($\mathbf{S}_j$)
    Set $\mathbf{B}_j$=$\mathbf{S}_j$
  }
  Set **G**=EL[1]
  While (stopping criterion is not satisfied) {
    For j=1...μ do {
      Set v_{ji}= v_{ji}+$U(0, \varphi_1)\times(b_{ji}-s_{ji})$+$U(0, \varphi_2)\times(g_i-s_{ji})$ for i=1,…, n
      $\mathbf{S}_j$=Create_Solution()
      Evaluate_Solution($\mathbf{S}_j$)
      If (CSR($\mathbf{S}_j$)> $\mathbf{B}_j$) set $\mathbf{B}_j$=$\mathbf{S}_j$
      Update EL if necessary
    }
    Set **G**=EL[1]
  }
  Evaluate each elitist solution using K_3 replications
  Return the best elitist solution
}

4 EXPERIMENTAL RESULTS

The PSO algorithm is tested using random problems ranging from 30 to 100 nodes as given in Table 1. The performance of the PSO algorithm is compared with the Ant Colony Optimization (ACO) algorithm given in (Kulturel-Konak and Konak 2008). For each test problem group, ten random instances are created by randomly assigning node and edge reliabilities from [0.90, 0.95] and node costs from [1, 2]. Each random problem instance is solved by both PSO and ACO algorithms for ten replications for a budget constraint of C=8. The solutions found by the PSO and ACO algorithms are compared using a Multivariate Analysis of Variance (MANOVA) model to test whether there are significant differences among their performances. For each problem group, a MANOVA model is built to compare the best and the worst elitist solutions found by the PSO and ACO algorithms. The MANOVA models include the ten problem instances and the heuristics as fixed factors and ten random replications as a random factor. In all runs, the simulation related parameters are $K_1=10^3$, $K_2=8\text{x}10^3$, $K_3=10^5$ (i.e., the number of simulation replications), H=99001 (i.e., the Hash size), and |EL|=20 (i.e., the elitist list). PSO parameters φ_1 and φ_2 are set to 2 as generally recommended in the PSO literature. The population size is 50, and the number of solutions searched so far (maximum of 8000 solutions) is used as the stopping criterion for both algorithms. The averaged CSRs of the best elitist and the worst elitist solutions (i.e., EL[1] and EL[20], respectively) and p-value of the MANOVA models are given in Table 1. Based on the MANOVA results, the PSO algorithm outperforms the ACO algorithm in terms of the best and worst elitist solutions in the majority of the problem groups. Only for the problems with 30 nodes, the worst elitist solutions found by the ACO are statistically better than the ones found by the PSO. The difference between two algorithms is statistically significant with p-values of almost zero in many cases. ACO algorithms are known to quickly converge that may become a drawback in the large-sized problems studied in this paper

because the search may stagnate around a few good solutions discovered early in the search. In this paper, the main difference between the ACO and the PSO is in the calculation of the probabilities of assigning servers to nodes in the solution construction procedure. In the ACO, all ants use the same pheromone trails to calculate those probabilities, while in the PSO, each particle maintains its individual velocity vector which may aim a different direction in the search space. This causes a slow convergence of the PSO, but also reduces the likelihood of being trapped in local optima and encourages exploration of different regions in the search space.

In Table 1, (Elitist Range)/σ is the ratio of the CSR range of the elitist list (i.e., the difference between the best and the worst elitist solutions) to the estimation of the standard deviation in simulation. (Elitist Range)/σ is an indicator for the quality of the best solution with respect to the solutions in the elitist list. In each case, the gap between the best and worst elitist solutions is much larger than the estimated standard deviation. Therefore, it can be assured with a high confidence that the final elitist list will include the true best solutions found during the search. Collision% is the ratio of the |CL| to |CL|+|SL| at the termination (i.e., the percent of hash table collisions). A small value of Collision% indicates how efficiently hashing can detect whether a solution has been previously investigated or not. This ratio is very low for all cases. For example, 2.5% collision rate means that the solution check can be performed in $O(1)$ time (i.e., checking the hash table after calculating the hash value) in 97.5% of solutions searched. Finally, average CPU seconds (with the 3.0 GHz Intel Xeon E5450 Quad-Core Processors and 32 GB memory) are given for the PSO algorithm.

Table 1: Comparisons of the PSO and the ACO algorithms and computational results.

$(n, m)^*$	Best Elitist			Worst Elitist					
	ACO	PSO	*p*-value	ACO	PSO	*p*-value	(Elitist Range)/σ	Collision %	CPU Sec.
(30,36)	0.88711	0.88744	0.613	0.86984	0.86897	0.035	24.0	0.4	65
(40,53)	0.88728	0.88799	0.000	0.87091	0.87314	0.000	36.8	0.7	111
(50,98)	0.98518	0.98504	0.190	0.97948	0.98027	0.000	19.8	1.6	261
(60,118)	0.99340	0.99376	0.000	0.98956	0.99047	0.000	17.3	1.9	429
(70,138)	0.98891	0.98934	0.000	0.98355	0.98512	0.000	17.8	2.5	589
(80,158)	0.97732	0.97799	0.000	0.96927	0.97207	0.000	11.6	3.0	881
(100,115)	0.40935	0.41664	0.000	0.37216	0.38985	0.000	13.2	3.6	1446

*: n is the number of nodes and m is the number of links.

5 CONCLUSIONS

The reliable server assignment problem (RSA) in networks is studied in this paper. A novel simulation optimization approach is developed based on a MC simulation and embedded into PSO to solve the RSA problem. During the search, hashing method is used to rapidly detect whether a solution has been previously investigated or not in order to prevent costly evaluation process of same solutions again and again. The experimental study shows that the PSO algorithm is promising.

REFERENCES

Andradottir, S. 1998. Simulation Optimization. J. Banks, J. S. Carson, B. L. Nelson and D. M. Nicol, *eds. Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice.* John Wiley, New York.

April, J., F. Glover, J. P. Kelly, and M. Laguna. 2003. Practical introduction to simulation optimization. In *Proceedings of 2003 Winter Simulation Conference*, eds. S. E. Chick, P. J. Sanchez, D. M. Ferrin, D. J. Morrice. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ball, M. 1980. Complexity of Network Reliability Calculation. *Networks* 10: 153-165.

Battiti, R., and G. Tecchiolli. 1994. The reactive Tabu search. *ORSA Journal on Computing* 6: 126-140.

Berman, O., and Z. Drezner. 2003. A probabilistic one-centre location problem on a network. *Journal of the Operational Research Society* 54: 871-877.

Berman, O., Z. Drezner, and G. O. Wesolowsky. 2003. Locating service facilities whose reliability is distance dependent. *Computers & Operations Research* 30: 1683-1695.

Clerc, M., and J. Kennedy. 2002. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation* 6: 58-73.

Drezner, Z. 1987. Heuristic solution methods for two location problems with unreliable facilities. *Journal of the Operational Research Society* 38: 509-514.

Eberhart, R. C., and J. Kennedy. 1995. A new optimizer using particle swarm theory. *Sixth International Symposium on Micro Machine and Human Science*, Nagoya.

Eiselt, H. A., M. Gendreau, and G. Laporte. 1992. Location of facilities on a network subject to a single-edge failure. *Networks* 22: 231-246.

Eiselt, H. A., M. Gendreau, and G. Laporte. 1996. Optimal location of facilities on a network with an unreliable node or link. *Information Processing Letters* 58: 71-74.

Fu, M. C. 2002. Optimization for simulation: theory vs. practice. *INFORMS Journal on Computing* 14: 192-215.

Hakimi, S. L. 1964. Optimum locations of switching centers and absolute centers and medians of graph. *Operations Research* 12: 450-459.

Kennedy, J., and R. C. Eberhart. 1997. Discrete binary version of the particle swarm algorithm. *the IEEE International Conference on Systems, Man and Cybernetics*.

Kulturel-Konak, S., and A. Konak. 2008. Reliable Network Server Assignment using an Ant Colony Approach. *ESREL- Annual Conference of European Safety and Reliability Association (ESRA) and Society for Risk Analysis Europe (SRA-E)*, Valencia, Spain, Taylor & Francis Group, London.

Law, A. M., and M. G. McComas. 2000. Simulation-based optimization. In *Proceedings of the 2000 Winter Simulation Conference*, eds. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Law, A. M., and M. G. McComas. 2002. Simulation-based optimization. In *Proceedings of the 2002 Winter Simulation Conference* ed. E. Yücesan, C. H. Chen, J. L. Snowdon, J. M. Charnes. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ling, W., Z. Liang, and Z. Da-zhong. 2003. Advances in simulation optimization. *Control and Decision* 18: 257-271.

Melachrinoudis, E., and M. E. Helander. 1996. A single facility location problem on a tree with unreliable edges. *Networks* 27: 219-237.

Nakaniwa, A., J. Takahashi, H. Ebara, and H. Okada. 2000. Reliability-based optimal allocation of mirror servers for Internet, San Francisco, CA, USA, Institute of Electrical and Electronics Engineers Inc., Piscataway, NJ, USA.

Nakaniwa, A., J. Takahashi, H. Ebara, and H. Okada. 2002. Reliability-based mirroring of servers in distributed networks. *IEICE Transactions on Communications* E85-B: 540-549.

Nel, L. D., and C. J. Colbourn. 1990. Locating a broadcast facility in an unreliable network. *INFOR* 28: 363-379.

Olafsson, S., and J. Kim. 2002. Simulation optimization. . In *Proceedings of the 2002 Winter Simulation Conference* ed. E. Yücesan, C. H. Chen, J. L. Snowdon, J. M. Charnes. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Ozcan, E., and C. K. Mohan. 1999. Particle swarm optimization: surfing the waves. *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, Piscataway, NJ, USA, IEEE.

Snyder, L. V., and M. S. Daskin. 2005. Reliability models for facility location: the expected failure cost case. *Transportation Science* 39: 400-416.

Snyder, L. V., and M. S. Daskin. 2006. Stochastic p-robust location problems. *IIE Transactions (Institute of Industrial Engineers)* 38: 971-985.

Swisher, J. R., P. D. Hyden, S. H. Jacobson, and L. W. Schruben. 2000. A survey of simulation optimization techniques and procedures. In *Proceedings of the 2000 Winter Simulation Conference*,

eds. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Woodruff, D. L., and E. Zemel. 1993. Hashing vectors for tabu search. *Annals of Operations Research* 41: 123-137.

AUTHOR BIOGRAPHIES

SADAN KULTUREL-KONAK is an Associate Professor of Management Information Systems at the Pennsylvania State University Berks. She received her degrees in Industrial Engineering; B.S. from Gazi University, Turkey, M.S.s from Middle East Technical University, Turkey and from the University of Pittsburgh, and Ph.D. from Auburn University. Her research interests are in modeling and optimization of complex systems and robustness under uncertainty. Dr. Kulturel uses exact solution methods such as mixed integer programming as well as metaheuristic techniques such as Tabu Search, Genetic Algorithms, and Ant Colony Approaches with applications to facility layout, reliability, scheduling, telecommunications, etc. Her other research interests consist of women in Information Systems and Engineering, and Teaching and Learning with Virtual Teams. She has published her research in *IIE Transactions*, *Operations Research Letters*, *INFORMS Journal on Computing*, *INFORMS Transactions on Education*, *International Journal of Production Research*, *European Journal of Operational Research*, and *Journal of Intelligent Manufacturing*. Dr. Kulturel teaches courses on management information systems, project management, business statistics and entrepreneurial leadership. She is a member of *INFORMS*, *IIE*, *Alpha Phi Mu*, and *Phi Kappa Phi*. Her email address is <sadan@psu.edu>.

ABDULLAH KONAK is an Associate Professor of Information Sciences and Technology at the Pennsylvania State University Berks. He received his degrees in Industrial Engineering; B.S. from Yildiz Technical University, Turkey, M.S. from Bradley University, and Ph.D. from University of Pittsburgh. His current research interest is in the application of Operations Research techniques to complex problems, including such topics as telecommunication network design, network reliability analysis/optimization, facilities design, and data mining. He has published papers in *IIE Transactions*, *Operations Research Letters*, *OMEGA- The International Journal of Management Science*, *IEEE Transactions on Reliability*, *International Journal of Modeling and Simulation*, *International Journal of Production Research*, *Engineering Optimization*, and *Journal of Intelligent Manufacturing*. He is a member of IIE and INFORMS. His email address is <konak@psu.edu>.