World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:3, 2007

A Genetic and Simulated Annealing Based Algorithms for Solving the Flow Assignment Problem in Computer Networks

Tarek M. Mahmoud

***Abstract*—**Selecting the routes and the assignment of link flow in a computer communication networks are extremely complex combinatorial optimization problems. Metaheuristics, such as genetic or simulated annealing algorithms, are widely applicable heuristic optimization strategies that have shown encouraging results for a large number of difficult combinatorial optimization problems. This paper considers the route selection and hence the flow assignment problem. A genetic algorithm and simulated annealing algorithm are used to solve this problem. A new hybrid algorithm combining the genetic with the simulated annealing algorithm is introduced. A modification of the genetic algorithm is also introduced. Computational experiments with sample networks are reported. The results show that the proposed modified genetic algorithm is efficient in finding good solutions of the flow assignment problem compared with other techniques.

***Keywords*—**Genetic Algorithms, Flow Assignment, Routing, Computer network, Simulated Annealing.

I. Introduction

NETWORK design is a fundamental problem with a large scope of applications that have given rise to many different models and solution approaches [6], [10]. The general network design problem involves the minimization of a cost objective function over a lot of design variables, such as link capacities, flow assignment, network topology, and node locations. This problem belongs to the class of combinatorial problems. Efficient solutions to this problem are much sought after because such solutions could lead to better utilization of the networks. The traditional Lagrange relaxation and sub-gradient optimization methods can be used for tackling this problem. The results generated by these methods are locally optimal instead of globally optimal [7]. The flow assignment (FA) problem focuses on assigning the traffic requirements on the best routes used by nodes in the network in order to ensure an acceptable performance level at a minimum cost.

For solving the FA problem, we consider different approaches. The first one uses the genetic algorithm (GA). The second approach uses the simulated annealing algorithm (SA).

Tarek M. Mahmoud is with the Computer Science Dept., Faculty of Science, Minia University, El-Minia, Egypt (e-mail: tarek_2ms@yahoo.com).

The third approach combines GA with simulated annealing (SA) to improve the performance of the GA. In the fourth approach, a new modification of the GA is introduced.

GAs have gained considerable attention in recent years for solving various combinatorial optimization problems. It is an evolutionary technique that simulates the process of natural evolution and applies genetic search operators like recombination, mutation, and selection to a sequence of alleles. The sequence of alleles is the equivalent of a chromosome in nature. GA can be used to solve variety of different problems using a survival of the fittest idea [4], [8]. SA is a metaheuristic algorithm derived from thermodynamic principles. It has recently turned out to be one of the most powerful tools for solving hard combinatorial problems [5], [9].

The remainder of the paper is organized as follows. In section 2, the FA problem is formulated. Section 3 gives a review of both the GA and SA. The implementation of the GA and SA for solving the FA problem is given in section 4 and 5. The hybrid genetic-simulated annealing algorithm to solve the FA problem is given in section 6. In section 7, a modification of the GA is introduced. The results of computational experiments are presented in section 8. Section 9 concludes and summarizes the main results obtained in this paper.

II. Problem Description and Formulation

A computer network can be modeled as an undirected graph G = (N, L), in which the sets of nodes N and links L represent computer sites and communication cables, respectively. There are communication demands between n different nodes. The demands are specified by an n x n demand matrix M = (m_{ij}), where m_{ij} is the amount of traffic required between n_i and n_j. The arrival rate on each link (i, j) is denoted by λ_{ij} and is expressed in the same units as capacity C_{ij}.

The flow assignment problem (FA) can be described as follows: given the network topology, the traffic requirement (OD matrix), and the link capacities, minimize the average time delay of messages by selecting the route such that traffic requirements are satisfied. Once the route is decided, the flows are sent along this route from origin node (O) to

destination node (D). In this paper, as in most previous work in the literature [7], [10], and [11] we make several simplifying assumptions. We assume that nodes have practically unlimited buffers to store messages and the arrival process of messages to the network follows a Poisson distribution. The computer network can be modeled as a network of independent M/M/1 queue in which links are treated as servers with service rates proportional to the link capacities [1]. The queuing and transmission delay in link ij is given by:

$$T_{ij} = \frac{1}{\mu C_{ij} - \lambda_{ij}} \quad (1)$$

where C_{ij} is the capacity of link ij in bits per second, λ_{ij} is the arrival rate of messages to link ij, and μ^{-1} is the expected message length. Using the above notation, the expected network delay is given by:

$$T = \frac{1}{\lambda} \sum_{ij \in L} \frac{\sum_{r \in R} \lambda_{ij} \delta_{ij}^{r} x_r}{\mu C_{ij} - \sum_{r \in R} \lambda_{ij} \delta_{ij}^{r} x_r} \quad (2)$$

where

λ is the total arrival rate of messages in the network,
L is the index set of links in the network,
R is the index set of candidate routes,
δ_{ij}^{r} is an indicator function, which is 1 if link ij is used in route r and is 0 otherwise,
x_r is a decision variable which is 1 if route r is selected for message routing and 0 otherwise.

The FA problem is then to assign the traffic demand on a route r from R for each nodes pair, which can satisfy the following condition:

$$D = \min\left\{ T = \frac{1}{\lambda} \sum_{ij \in L} \frac{\sum_{r \in R} \lambda_{ij} \delta_{ij}^{r} x_r}{\mu C_{ij} - \sum_{r \in R} \lambda_{ij} \delta_{ij}^{r} x_r} \right\} \quad (3)$$

subject to:

$$C_{ij} \geq \frac{1}{\mu} \sum_{r \in R} \lambda_{ij} \delta_{ij}^{r} x_r, \quad \forall ij \in L \quad (4)$$

$$\sum_{r \in S_P} x_r = 1, \ \forall P \in \prod \quad (5)$$

$$x_r = \{0,1\}, \forall r \in R \quad (6)$$

where S_p is the index set of candidate routes for commodity p, and $\prod$ is the index set of communicating source/destination pairs in the network. The first constraint ensures that the flow on each link does not exceed its capacity, while selection of exactly one route per commodity is ensured by the second and third constraints.

III. Review of Genetic and Simulated Annealing Algorithms

In recent years, genetic algorithms (GAs), which based on the idea of natural selection and survival of the fittest, have been applied with a high degree of success to a variety of problems [3], [7]. GAs are search techniques for global optimization in a complex search space. Search space in GA is composed of possible solutions to the problem. A solution in the search space can be represented by a sequence of 0s and 1s, integers, or any other type from which a specific solution can be deduced. This solution string is referred to as the chromosome in the search space. Each chromosome has an associated objective value called the fitness value. The fitness of a chromosome corresponds to its ability to survive and reproduce offspring. A good chromosome (that has good chance to survive) is the one that has a high/low fitness value depending upon the problem (maximization/ minimization). A set of chromosomes and associated fitness values are called the population. This population at a given stage of the GA is referred to as generation. The general GA has the following elements:

- A solution encoding (representation).
- A mechanism to generate initial solutions (population) from where the iterative search will proceed.
- An evaluation function (fitness function) to rate solutions from a current population.
- Perturbation operators to create new solutions from a current population.
- Assignment to the parameters of the algorithm, and
- Stopping criteria.

Creating new population from a given current population can be achieved by shuffling two randomly selected chromosomes. This process is called crossover. Sometimes one or more bits of a chromosome are complemented to generate a new offspring. This process of complementation is called mutation. The GA can be summarized as follows:

Step 1: Initialize population
Step 2: Evaluate population
Step 3: Repeat Steps 4, 5, and 6 while termination criterion not reached
Step 4: Select parent chromosomes for next population
Step 5: Perform crossover and mutation
Step 6: Evaluate population

Simulated annealing algorithm (SA) is a general-purpose optimization technique and has been applied to many combinatorial optimization problems [5], [9]. The main idea behind SA is an analogy with the way in which liquids freeze and crystallize. When liquids are at a high temperature their molecules can move freely in relation to each other. As the liquid's temperature is lowered, this freedom of movement is lost and the liquid begins to solidify. If the liquid is cooled slowly enough, the molecules may become arranged in a crystallize structure. The molecules making up the crystallize

structure will be in a minimum energy state. If the liquid is cooled very rapidly it does not form such a crystallize structure, but instead forms a solid whose molecules will not be in a minimum energy state. The fundamental idea of SA is therefore that the moves made by an iterative improvement algorithm are like the re-arrangement of the molecules in a liquid that occur as it is cooled and that the energy of those molecules corresponds to the cost function which is being optimized by the iterative improvement algorithm. Thus, the SA aims to achieve a global optimum by slowly convergence to a final solution, making downwards moves with occasional upwards moves and thus hopefully ending up in a global optimum. SA can be described formally as follows: start from a random solution. Given a solution S_c, select a neighboring solution S_n and compute the difference in the objective function values, $\Delta f = f(S_n) - f(S_c)$. If the objective function is improved ($\Delta f < 0$), then replace the current solution by the new one. If $\Delta f \geq 0$, then accept a move with probability $p(\Delta f) = \exp(-\Delta f/T)$, where T is the control parameter or temperature. This probabilistic acceptance is achieved by generating a random number in [0, 1] and comparing it against the threshold $\exp(-\Delta f/T)$. If $\exp(-\Delta f/T)$ is greater than the generated random number then replace the current solution by the new one. The procedure is repeated until a stopping condition is satisfied. The SA can be summarized as follows:

Step 1: Initialize temperature T at random, and set a cooling rate.
Step 2: Initialize initial solution *Sc* at random.
Step 3: Evaluate $f(S_c)$.
Step 4: Repeat steps 5, 6, 7, and 8 while termination condition not satisfied.
Step 5: Select a solution *Sn* in the neighborhood of *Sc* at random.
Step 6: Set $\Delta f = f(S_n) - f(S_c)$ {Compare the change in objective function}
Step 7: If $\Delta f \leq 0$ then
$S_c \leftarrow S_n$ {S_n replaces S_c}
Else
If $\exp(-(\Delta f)/T) >$ random (0, 1) then
$S_c \leftarrow S_n$
Step 8: Update (T) using the relation T = α * T, where α is the cooling rate {Cooling step}

IV. Implementing the GA for Solving the Flow Assignment Problem

This section presents an implementation of the GA for identifying the link flows of the computer network that satisfies the traffic requirements while satisfying the problem constraints. Firstly, we describe how the main components of the GA are implemented to solve the FA problem. Then we give the overall algorithm used to solve this problem.

A. Genetic Representation

In route selection, and hence flow assignment, each chromosome represents a routing table which includes a path list $P_1, P_2, \ldots, P_{N(N-1)/2}$ that represents the entire network. Each path P_k is a particular route between two nodes i, j. A path (route) is encoded as list of integers by listing the nodes from its source to its destination based on the network topology [4]. If a path cannot be realized on the network, it cannot be encoded into a chromosome, which means that each step in a path must pass through a physical link in the network. The first gene on the chromosome represents the source node; the last gene represents the destination node, while other genes represent intermediate nodes along that path from the source to the destination.

B. Population Initialization

The initial process is used to compose the routing tables for all chromosomes in the current generation. Each chromosome includes a random routing table for a given topology.

C. Evaluation

To evaluate the solution quality of the flow assignment problem, equation (3) is used as the objective function to determine the ability of a chromosome to survive and produce offspring. In our implementation, a route with less average time delay is frequently employed in sending packets.

D. Selection

The selection (reproduction) operator is intended to improve the average quality of the population by giving the high fitness chromosomes (less average time delay) a better chance to get multiple copies into the next generation, whereas chromosomes with low fitness (high average time delay) have fewer copies or even none at all. Many types of selection scheme can be used [3]. One of the selection methods is based on spinning the roulette wheel pop_size times; each time a single chromosome is selected as a new offspring, where pop_size denotes the population size. The steps of this selection method are as follows [8]:
Step 1: Calculate the fitness value $f(v_i)$ for each chromosome v_i (i=1, …, pop_size).
Step 2: Find the total fitness of the population

$$F = \sum_{i=1}^{pop\_size} f(v_i)$$

Step 3: Calculate the probability of the selection p_i for each chromosome v_i where

$$p_i = f(v_i)/F, \; i = 1, 2, \ldots, pop\_size$$

Step 4: Calculate a cumulative probability q_i for each chromosome v_i where

$$q_i = \sum_{j=1}^{i} p_j, \; i = 1, 2, \ldots, pop\_size$$

Step 5: Generate a random number r from the range (0, 1).
Step 6: If $r<q_1$ then select the first chromosome (v_1); otherwise select the i-th chromosome $v_i (2 \leq i \leq pop\_size)$ such that $q_{i-1} < r \leq q_i$.

E. Crossover Operator

A crossover operator recombines the gene-codes of two parents, which are randomly selected, and produces offsprings such that the children inherit a set of building blocks from each parent. The application of crossover is governed by a crossover probability, denoted by P_c.

In each generation of the GA, the crossover operator is tried pop-size times (pop-size is the population size) and thus, the expected number of applications of crossover is P_c* pop-size.

In this paper, the crossover operator exchanges subroutes between two chromosomes. First, select the chromosomes according the probability of the crossover operator. Second, apply crossover to the selected chromosomes using the path crossover operator by selecting randomly two paths that should have the same source and destination nodes. Crossing sites for the path crossover operator are limited to nodes contained in both paths. A node is randomly selected as a crossing site from the potential crossing sites and exchanges subroutes. When applying the crossover operator to a pair of paths P_1 and P_2, the operation proceeds according to the following algorithm [4]:

Step 1: List up a set of nodes N_c included in both P_1 and P_2 (excluding source and destination nodes) as potential crossing sites.
Step 2: Select a node i as a crossing site from the N_c.
Step 3: Crossover the paths by exchanging all nodes after the crossing site i.

Fig. 1 shows an overview of the crossover operator applying for a pair of paths P_1 and P_2 from node 1 to node 4. Their potential crossing sites is node 3. When we select node 3 as a crossing site, the new offspring are generated by exchanging the subroutes as shown in the figure below.

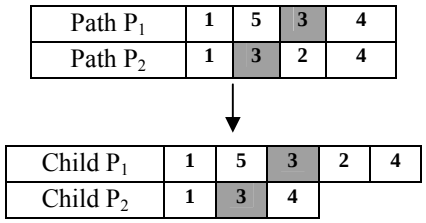

Fig. 1 Example of crossover

F. Mutation Operator

The path mutation is another genetic operator, which is applied to a randomly selected single solution (chromosome) from the population with a certain probability. It makes small random changes in the solution. These random changes will gradually add some characteristics to the population, which could not be supplied by the crossover operator. Similar to the crossover operator, the application of the mutation operator is governed by a mutation probability P_m. In each generation, the mutation operator is also tried pop-size times and thus, the expected number of mutated chromosomes is P_m* pop-size.

To perform a mutation in the FA problem, a node is randomly selected from the path, which is called a mutation node. Then another node is randomly selected from the nodes directly connected to the mutation node. If any duplication of nodes exists in the offspring path, then this path can be discarded. The path mutation operator can be described as follows [4]:

Step 1: Select mutation node i randomly from all nodes in parent V.
Step 2: Select a node j from the neighbors of the mutation node i.
Step 3: Generate a random path from the source to node j, and another random path from node j to the destination node.
Step 4: If any duplication of nodes exists in the offspring path, discard the routes and do not perform mutation. Otherwise the routes are connected to make up a mutated path.

G. Overall Algorithm

The GA algorithm used to solve the flow assignment problem can be described as follows:

Step 1: Input all kinds of data of the problem (network topology, OD matrix, and link capacities) and the controlling parameters of the algorithm (crossover and mutation probability, population size, and generation number).
Step 2: Randomly generate initial population, where each chromosome in the population represents a routing table for the given network.
Step 3: For each OD pair, assign the flow between the origin O and the destination D on the route connecting them, then compute the fitness of every chromosome in the current population using equation (3). Save the best chromosome.
Step 4: Select the best chromosomes (routing tables) using the roulette wheel method.
Step 5: Perform the crossover and mutation operations to get a new population.
Step 6: Repeat steps 3 – 5 until the termination condition is met.

Note that, the termination condition is met either after a specified number of generations or no improvement occurs on the best solution for successive generations. In our implementation, a specified number of generations are used as a termination condition.

V. Implementing the SA for Solving the Flow Assignment Problem

Using SA for solving the FA problem requires the determination of an initial feasible solution. In our implementation, we used the population initialization step of the GA, discussed in section III, to get an initial feasible solution. Thus, the initial solution includes a random routing table for each source-destination pair of a given network. The selecting of a solution in the neighborhood of the initial solution can be obtained by changing randomly one or more links of the route from the source to the destination. As in the

implementation of GA for solving the FA problem, equation 3 is used to compute the value of each candidate solution for the problem.

VI. Hybrid Genetic-Simulated Annealing Algorithm Approach (HGSA)

The HGSA algorithm combines both the GA and SA algorithms to solve the FA problem. This combination occurs in the selection process of the GA algorithm. A SA-selection [2] is used to choose a single candidate solution between the best of the two parents, offspring, and best solution of the generations. The selection function SA-selection(offspring[i], parent[i], best solution, T[i]) is defined, which applies the traditional SA function SA(a, b, T) multiple times to identify the single surviving candidate, where i in the SA-selection function indicates the index of the new individual. The function SA(a, b, T) calculates the acceptance probability $P = \exp(-(f(a) - f(b))/T)$, where f(x) is the cost function of a candidate x. As mentioned in section 3, if $f(a) \leq f(b)$, then candidate a will be selected. If $f(a)>f(b)$ and $P >$ random number in [0, 1), then candidate a will be selected too. Except these cases, candidate b will be selected. All possible cases of this selection process are represented in Table 1. In this SA-selection function, offspring[i] and parent[i] are compared with each other, then with best solution. To illustrate the selection cases given in Table I, consider case 1, where offspring[i] is accepted over both parent[i] and best solution, but parent[i] is accepted over best solution. Hence, offspring[i] is accepted and returned.

The HGSA algorithm for solving the FA problem has the following steps:

Step 1: Input all kinds of data of the problem (network topology, OD matrix, and link capacities) and the controlling parameters of the algorithm (crossover and mutation probability, population size, and generation number, and cooling rate).
Step 2: Randomly generate initial population, where each chromosome in the population represents a routing table for the given network.
Step 3: For each individual i randomly generate an initial temperature T[i].
Step 4: For each OD pair, assign the flow between the origin O and the destination D on the route connecting them, then compute the fitness of every chromosome in the current population.
Step 5: Save the current population as the parent population.
Step 6: Perform the crossover and mutation operations to get the offsprings.
Step 7: Find the best routing table among the parents, offsprings, and current best solution, and then update the best solution.
Step 8: For each individual of the population do
Find the i-th chromosome between parent, offspring, and best solution according to the processes given in Table I.
i-th chromosome = SA-selection(offspring[i], parent[i], best solution, T[i])
Update the fitness function of the i-th chromosome.
Set T[i] = T[i] x cooling rate
End for
Step 9: Repeat steps 4 – 8 until termination condition met.

VII. A Modified Genetic Algorithm Approach (MGAA)

As mentioned earlier, the selection operator is intended to improve the average quality of the population and many types of selection scheme can be used. Calling the selection process for each generation increases the algorithm complexity. We can ignore this process and perform the reproduction process directly on the current population. In the MGAA, we generate the initial population randomly. In each generation, the new population consists of offsprings produced from mating individuals from the current population and possibly some individuals from the current population. The parents are selected for crossover and mutation according to the crossover and mutation probability. If the fitness value of the offspring has a better value than one or both of its parents then this offspring is accepted in the new population. Otherwise, mate the best of the two parents with another parent. In the later case, we can change the crossover point instead of replacing one of the parents. This step guarantees that the new population contains always the best chromosomes. Applying the traditional crossover process to generate new offspring is not always guaranteed to produce good chromosomes.

The MGAA can be summarized as follows:

Step 1: Initialize population
Step 2: Evaluate population and save the best chromosome.
Step 3: Repeat Steps 4 and 5 while termination criterion not reached
Step 4: Perform crossover and mutation
Step 5: Evaluate population and update the best chromosome.
It is clear that; the selection step in the genetic algorithm given in section 4 is eliminated. This modification speeds up the GA and guarantees that the reproduction step always produces good offsprings.

VIII. Experimental Results and Discussion

In this section, we present a comparison between the performance of the four approaches GA, SA, HGSA, and MGAA. These algorithms were implemented in C++. The results presented in this section are obtained from simulations on 2 sample networks. The topologies of these networks are as shown in Fig. 2 and 3.

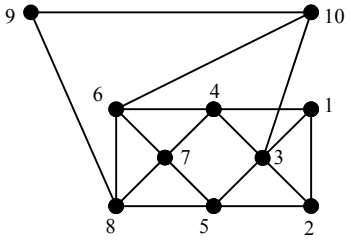

Fig. 2 A network example with 10 nodes and 36 links

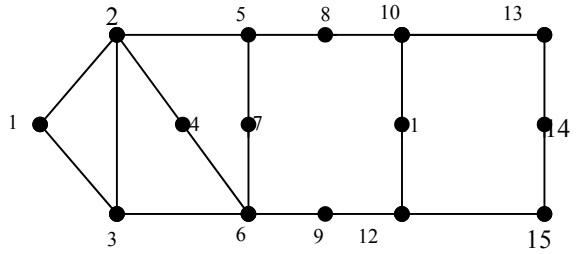

Fig. 3 A network example with 15 nodes and 38 links

In the network topology given in Fig. 2, node 10 is a geosynchronous satellite. All links are full duplex. The capacity of each terrestrial link is 38.4 kb/s and that of each satellite link is 50 kb/s. Each of the six satellite links is assumed to introduce a propagation delay of 125 ms. All link capacities in the second network topology are 50 kb/s. To handle the situation of overflow, the terms of the objective function given in equation 3 are replaced with the following terms [12]:

$$T_{ij} = \begin{cases} T_{ij}(\lambda_{ij}) & \text{if } 0 \le \lambda_{ij} \le 0.99\lambda_{ij} \\ T_{ij}(0.99\lambda_{ij}) + (\lambda_{ij} - 0.99\lambda_{ij}).T'_{ij}(0.99\lambda_{ij}) & \\ + \frac{1}{2}(\lambda_{ij} - 0.99\lambda_{ij})^2.T''_{ij}(0.99\lambda_{ij}) & \text{otherwise} \end{cases}$$

where T'_{ij} and T''_{ij} are the first and second derivatives of the objective function respectively.

The traffic matrix for the nodes of the first network example with 10 nodes and the second network example with 15 nodes is given in Table II and III.

The SA algorithm was applied using the following parameters:

- The initial temperature of the process selected randomly in the range (0..1)
- The cooling coefficient was 0.99

The genetic parameters are chosen as follows:

- The population size is 10.
- The crossover probability is 0.35.
- The mutation probability is 0.01.

The computer simulation results generated by the GA, SA, HGSA, and MGAA for the two network topologies given in Figs. 2, 3 are shown in Figs. 4, 5. The figures show the relation between the number of generations and the fitness (average time delay) values for the FA problem. As can be seen from Fig. 4 and 5, MGAA is better than GA, HGSA, and SA algorithms.

According to the simulation results, the hybridization between SA and GA improves the performance of the GA in solving the FA problem. It is clear also that SA is better than GA in solving the FA problem. As mentioned above, the MGAA has another advantage than the traditional GA. The time used in the selection process either using the roulette wheel discussed in section IV.D or any other selection method is saved. Our implementation of the MGAA applies in the reproduction step one-cut point crossover.

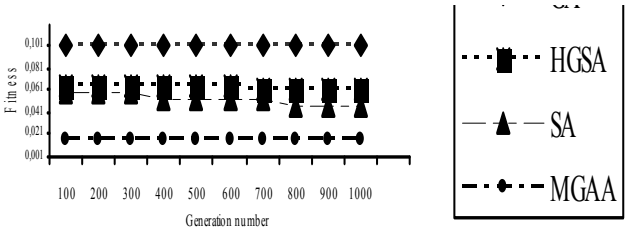

Fig. 4 Comparison chart for fitness values using GA, SA, HGSA and MGAA for the first network example

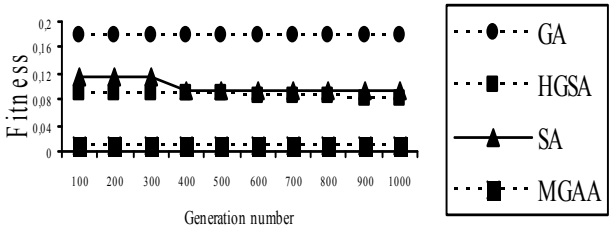

Fig. 5 Comparison chart for fitness values using GA, SA, HGSA and MGAA for the second network example

IX. Conclusion

In this paper, we studied the route selection and flow assignment in computer networks, and the use of a genetic algorithm and simulated annealing algorithm for solving the problem. To improve the performance of the genetic algorithm, hybridization between the genetic and the simulated annealing algorithm is introduced. A modification of the genetic algorithm is also introduced.

We have compared the performance of the four algorithms in solving the route selection and flow assignment problem. Our experimental results showed that the proposed modified genetic algorithm provided better solutions than the traditional genetic, simulated annealing, and hybrid genetic-simulated annealing algorithms.

For the same problem, the simulated annealing and its hybridization with the genetic algorithm have better results than the traditional genetic algorithm. The modified genetic algorithm can be used in any problems to which the genetic algorithm approach is applicable. Further modifications of the initial population step can improve the performance of the genetic algorithm. Instead of generating the initial population randomly, we can use any method to generate only better chromosomes.

References

[1] D. Berttsekas, and R. Gallager. "Data Networks", Second Edition, Prentice Hall, Englwood Cliffs, New Jersey (1992).
[2] H. Chang, and P. Jung, "SA-selection-based Genetic Algorithm for the Design of Fuzzy Controller", International Journal of Control, Automation, and Systems, Vol. 3, no. 2, pp. 236-243, June (2005).
[3] W. Chang and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations", IEEE Transaction on Evolutionary Computation, Vol.6, No. 6, December (2002).
[4] M. Gen and R. Cheng, "Genetic algorithms and engineering optimization", John Wiley&Sons, Inc., (2000).

[5] Y.,Habib, M. Sait and H. Adiche, "Evolutionary algorithms, simulated annealing and tabu search: a comparative study", Engineering Applications of Artificial Intelligence 14, pp. 167-181, (2001).
[6] K. Ko, K. Tang, C. Chan, K. Man and S. Kwong, "Using genetic algorithms to design mesh networks", IEEE Computer, pp 56-61, (1997).
[7] X. Lin, Y. Kwok and V. Lau, "A genetic algorithm based approach to route selection and capacity flow assignment", Computer Communications 26, pp 961-974, (2003).
[8] Z. Michalewicz, "Genetic algorithms + data structure = evolution programs", 3rd edition, Springer Verlag, New York, USA, (1996).
[9] P. Mills, E. Tsang, Q. Zhang and J. Ford, "A survey of AI-based meta-heuristics for dealing with local optima in local search", Technical Report Series, Report No. CSM-416, September (2004).
[10] J. Shen, F. Xu and P. Zheng,"A tabu search algorithm for routing and capacity assignment problem in computer networks", Computers & Operations Research 32, pp 2785– 2800, (2005).
[11] K. Walkowiak, "Ant algorithm for flow assignment in connection-oriented networks", International Journal of Applied Mathematics and Computer Science, 15, pp 205-220, (2005).
[12] Z. Wang, D. Browning, "An Optimal Distributed Routing Algorithm", IEEE Transaction on Communication, vol.39, no. 9, (1991).

TABLE I
SA-SELECTION FUNCTION FOR CHOOSING THE SURVIVING SOLUTION AMONG OFFSPRING, BEST SOLUTION, AND PARENT

Case	SA-selection(offspring[i], parent[i], best solution, T[i])			
	SA(offspring[i], parent[i], T[i])	SA(offspring[i], best solution, T[i])	SA(parent[i], best solution, T[i])	return
1	offspring[i]	offspring[i]	parent[i]	offspring[i]
2	offspring[i]	offspring[i]	best solution	offspring[i]
3	offspring[i]	best solution	parent[i]	offspring[i]
4	offspring[i]	best solution	best solution	best solution
5	parent[i]	offspring[i]	parent[i]	parent[i]
6	parent[i]	offspring[i]	best solution	offspring[i]
7	parent[i]	best solution	parent[i]	parent[i]
8	parent[i]	best solution	best solution	best solution

TABLE II
TRAFFIC REQUIREMENT OF THE FIRST NETWORK EXAMPLE WITH 10 NODES

	1	2	3	4	5	6	7	8	9	10
1	0	5	5	0	0	5	0	10	0	0
2	5	0	10	5	5	3	0	10	0	0
3	5	10	0	5	0	3	5	5	20	0
4	0	5	5	0	5	0	5	5	5	0
5	0	5	0	5	0	5	0	0	5	0
6	5	3	3	0	5	0	0	0	5	0
7	0	0	5	5	0	0	0	5	0	0
8	10	10	5	5	0	0	5	0	5	0
9	0	0	20	5	5	5	0	5	0	0
10	0	0	0	0	0	0	0	0	0	0

TABLE III
TRAFFIC REQUIREMENT OF THE SECOND NETWORK EXAMPLE WITH 15 NODES

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	10	3	0	0	5	0	8	0	3	0	6	0	4	0
2	10	0	8	5	0	4	10	0	0	5	0	7	0	0	9
3	3	8	0	0	6	8	0	2	0	5	0	8	0	5	0
4	0	5	0	0	10	9	0	3	0	6	0	4	0	0	2
5	0	0	6	10	0	0	5	6	0	3	0	6	8	10	0
6	5	4	8	9	0	0	5	0	7	7	4	3	0	0	5
7	0	10	0	0	5	5	0	5	2	4	0	7	0	8	0
8	8	0	2	3	6	0	5	0	11	12	0	5	0	8	0
9	0	0	0	0	0	7	2	11	0	10	0	10	0	4	0
10	3	5	5	6	3	7	4	12	10	0	5	0	5	5	0
11	0	0	0	0	0	4	0	0	0	0	0	4	0	7	0
12	6	7	8	4	6	3	7	5	10	0	2	0	5	2	5
13	0	0	0	0	8	0	0	0	0	5	0	5	0	5	5
14	4	0	5	0	10	0	8	8	4	5	7	2	5	0	8
15	0	9	0	2	0	5	0	0	0	0	0	5	5	8	0