Information Science Letters
An International Journal

@ 2013 NSP
Natural Sciences Publishing

Routing and Capacity Assignment Problem in Computer Networks Using Genetic Algorithm

M. R. Girgis[1] , T. M. Mahmoud[1] H. F. Abd El-Hameed[2] and Z. M. El-Saghier[2]

[1]*Faculty of Engineering, inia University, El-Minia, Egypt*
[4]*Mathematics Department & Computer Science Department, Sohag University, Egypt*

Email: moheb_girgis@eun.eg, tarek_2ms@yahoo.com, hf_elsheikh@yahoo.com, zenab_elzohry@yahoo.com
Received*: 8 Feb. 2012; Revised 15 May. 2012; Accepted 23 Jul. 2012*

Abstract: In large-scale computer communication networks (e.g. the nowadays Internet), the assignment of link capacity and the selection of routes are extremely complex network optimization problems. Efficient solutions to these problems are much sought after because such solutions could lead to considerable monetary savings and better utilization of the networks. Unfortunately, as indicated by much prior theoretical research, these problems belong to the class of nonlinear combinatorial optimization problems, which are mostly (if not all) NP-hard problems. Metaheuristics, such as genetic and simulated annealing algorithms, are widely applicable heuristic optimization strategies that have shown encouraging results for a large number of difficult combinatorial optimization problems. In this paper, we propose three different approaches for solving the routing and capacity assignment (RCA) problem. The first approach uses a GA. The second approach uses the SA algorithm. The last approach combines the GA with the SA to improve the performance of the GA. The paper also presents the experiments that we have conducted to evaluate the effectiveness of the proposed approaches compared to a one of the heuristic algorithms for solving the same problem. The results show that the hybrid approach is more efficient in finding good solutions to the RCA problem compared to other techniques.

Keywords: Genetic Algorithms, Simulated Annealing, Hybrid Genetic-Simulated Annealing Algorithm, Capacity Assignment, Routing, Computer network.

INTRODUCTION

As the development of networking technologies and the increasing popularization of network based applications, computer networks are now being widely used in scientific researches, business, education, industry, national defense and other domains[1][2]. When programming or extending a computer network, designers frequently encounter the following typical problem: under the premise of a known network topology and communication requirements of all nodes, how to choose the capacity of every link and decide a route between every two nodes? This is a general network design problem, in which an optimal assignment of network link capacity and the best selection of routes are to be determined in a reasonable amount of time. This problem is also commonly referred to as the problem of capacity and flow assignment (CFA) [3]-[4]. Researchers also frequently consider a simpler version of the problem: network topology, link capacity, and communication requirements of all nodes are given, how to choose an optimal route between every two nodes? Efficient solutions to these problems are much sought after because such solutions could lead to considerable monetary savings and better utilization of the networks [5]. To tackle these problems, researchers have resorted to a wide variety of techniques. Shen et al. [3] applied a tabu search algorithm for the routing and capacity assignment problem in computer networks. Wille et al. [5] applied a Lagrangean relaxation approach for Quality of Service (QoS) networks CFA problems. Lin et al. [6] applied a genetic algorithm (GA) based approach to route selection and capacity flow assignment. Lin et al. [4] applied a ranking based selection GA for capacity and flow assignments.

GAs have gained considerable attention in recent years for solving various combinatorial optimization problems. Another metaheuristic algorithm, which has recently turned out to be one of the most powerful tools for solving hard combinatorial problems [7]-[8], is simulated annealing (SA). It is derived from thermodynamic principles.

In this paper, we propose three different approaches for solving the routing and capacity assignment (RCA) problem. The first approach uses a GA. The second approach uses the SA algorithm. The last approach combines the GA with the SA to improve the performance of the GA. The paper also presents the results of the experiments that we have conducted to evaluate the effectiveness of the proposed approaches compared to a heuristic algorithm, introduced by Bertsekas and Gallager [9], for solving the same problem.

The remainder of the paper is organized as follows. In section 2, the RCA problem is formulated. The implementation of the GA and SA algorithms for solving the RCA problem is given in sections 3 and 4, respectively. The hybrid genetic-simulated annealing algorithm for solving the RCA problem is given in section 5. The results of experiments are presented in section 6. Section 7 concludes and summarizes the main results obtained in this paper.

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. The RCA problem can be described as the following question: given the topology of a network and communication traffic between nodes, how to choose the capacity of each link and the routing between each nodes pair to ensure that the cost of network and the average message delay be minimum?. This problem is a complex nonlinear programming which has many restrained conditions and it is known to be NP-completeness [4]-[5].

The RCA problem can be characterized as follows [9]:

Given: *Topology and traffic requirement (flow) between the nodes*

Performance Constraints: *Delay Time*

Variables: *Link capacities and Flow assignment*

Goal*: Minimize cost*

The design objectives in the route and capacity assign problem are to:

1. Keep the average delay per packet or message below a given level.
2. Minimize a combination of capital investment and operating costs while meeting objective 1.

   A common, broadly stated formulation of the problem is as follows:

The flow on each link (i, j) is denoted F_{ij} and is expressed in the same units as capacity C_{ij} . We assume that there is a given input flow for each origin-destination (OD) pair and γ is the sum of these. The link flows F_{ij} depend on the known input flows and the scheme used for routing. When the flows F_{ij} are known, the problem is to minimize the linear cost:

$$Z = \sum_{(i,j)} d_{ij}(C_{ij}), \tag{1}$$

subject to the average delay constraint:

$$\frac{1}{\gamma}\sum_{(i,j)} \frac{F_{ij}}{C_{ij} - F_{ij}} \leq T, \tag{2}$$

where γ is the total arrival rate into the network, $d_{ij}(C_{ij})$ is the cost of leasing C_{ij} in link (i, j),.and T is a given threshold.

Implementing a GA for Solving the RCA Problem

Genetic algorithms (GAs) were first introduced and investigated by Holland [10][11]. The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representations of the problem to be solved. According to the attributes of the problem, different positions of each chromosome are encoded as bits, characters, or numbers. These positions are sometimes referred to as genes and are

changed randomly within a range during evolution. The set of chromosomes during a stage of evolution are called a population. An evaluation function is used to calculate the "goodness" of each chromosome [12].
During evaluation, two basic operators, crossover and mutation, are used to simulate the natural reproduction and mutation of species. The selection of chromosomes for survival and combination is biased towards the fittest chromosomes. .
GAs appear ideal to design communication network with the capability of handling discrete values, multi objective functions, and multi constraint problems. This section presents an integrated solution to the RCA problem using a GA approach.

3.1 The GA for the Routing Problem

The routing algorithm is that part of the network layer software responsible for deciding a route of each packet from its source node to its destination based on the network topology.
The GA for solving the routing problem selects the best path between each origin and destination by using genetic operators. The initial population should contain some valid routes. The large number of possible routes between each node pair may exceed memory constraints. To keep the routing tables small, the number of initial candidate routes is limited. This limit is defined as the population size. For a given topology, the capacity assignment and routing problems require the simultaneous optimization of both the flow and link capacities [13].

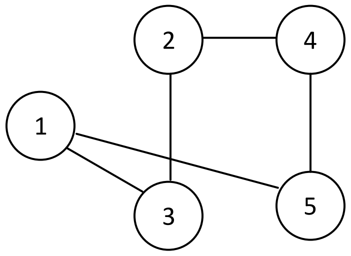

Figure 1 An example network

3.1.1 Representation

In this problem, we consider the routing scheme as the GA chromosome. A path list represents routing chromosome P = (P_1,P_2…,$P_{N(N-1)/2}$), which represents the entire network. Each path P_k is a particular route between nodes i and j [14]. For example, a path list P for the network example shown in Figure 4.4 could be as follows:
P = [(1,3,2), (1,3), (1,5,4), (1,5), (2,3), (2,4), (2,4,5), (3,2,4), (3,2,4), (4,5)].
A path (route) is encoded by listing the nodes from its source to its destination based on the topology of the network. For example, in the network of Figure 1, a path from node 1 to node 5 is represented as a list of nodes along the path as shown in Figure 2:

1	**3**	**2**	**4**	**5**

Figure 2 A Path structure
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.

3.1.2 Initial population

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

3.1.3 Evaluation

Evaluation of fitness value of routes is based on the costs of links between nodes in the route. Each solution in the population will be assigned a value calculated from its total link cost. The fitness function of a chromosome is calculated as follows:
Step 1: Evaluate the cost for the chromosome by the cost function:

$$\text{eval}(x) = \sum_{k=1}^{n_p} \sum_{P_{ij} \in x} d_{ij}(C_{ij}), \quad (3)$$

where n_p number of paths in routing table x, and P_{ij} is a path between node i and node j.

Step 2: Evaluate the delay time for each path by the delay function:

$$D = \frac{1}{\gamma} \sum_{(i,j)} \frac{F_{ij}}{C_{ij} - F_{ij}}, \quad (4)$$

then sum the delay time for all paths and store it as the delay time of this chromosome.

3.1.4 Selection

Selection is an operator to select two parent chromosomes (i.e., routing tables) for generating new chromosomes. In the selection, the fitness value of each string is utilized for selecting parent strings. The fitness value of each chromosome is determined as described in Sec. 3.1.3. A chromosome with a high fitness value (minimum cost subject to the delay constraint) has more chance to be selected as one of parents than a chromosome with a low fitness value (higher cost). In the selection process, we applied the average-fitness selection scheme proposed by Girgis et al. [15].

3.1.5 Crossover operator

The crossover operator exchanges sub-routes between two chromosomes. First, select the chromosomes according the probability of 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 [16]. 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 sub-routes. Figure 3 shows an example of applying the crossover operator on 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 sub-routes as shown in Figure 3.

Path P1	**1**	**5**	**3**	**4**
Path P2	**1**	**3**	**2**	**4**

$\downarrow$

Child P1'	**1**	**5**	**3**	**2**	**4**
Child P2'	**1**	**3**	**4**		

Figure 3 Example of crossover

When a common node in a pair of chromosomes does not exist, a crossing site cannot be selected; therefore, it is impossible to perform the crossover operator.

3.1.6 Mutation operator

The path mutation operator generates an alternative path from a chromosome. To perform a mutation, first, 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. Figure 4 shows an example of a mutation operator. First, suppose that node 5 is selected as a mutation node. Second, node 3 is randomly selected from the neighbors of the mutation node. Third, generate random path from node 1 to node 3 (r_1), and random path between node 3 to node 4 (r_2). We finish the mutation by connecting the sub-routes r_1 and r_2. An offspring V' isn't generated if any duplication of nodes exists in sub-routes r_1 and r_2. This is because we need to avoid creating any loop in a route.

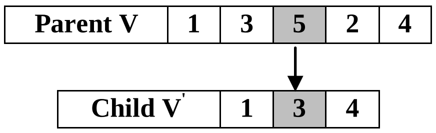

Figure 4 Example of mutation

3.1.7 Overall routing GA

Given the capacities of all links, the steps of the overall GA that solves the flow assignment (routing) problem are as follows:

- Step 1 Set the parameters. Set the population size (pop_size), crossover rate (P_c), mutation rate (P_m), delay time threshold (T) and the maximum generation (maxgens), and initialize the generation number g=0.
- Step 2 Initialization:
  - (2.1) Generate the initial population randomly.
  - (2.2) Assign flow for each chromosome (routing table).
  - (2.3) Sum the flow on the links in each chromosome in the initial population.
  - (2.4) Calculate the fitness of each chromosome in the population using Eq. (3).
  - (2.5) Check the delay time of each chromosome to see whether it is less than T. If not, generate another chromosome.
  - (2.6) If the edge flow exceeds the capacity on an edge, set the capacity of this edge to a higher link type than the assigned flow.
  - (2.7) Save the best chromosome of the current generation.
- Step 3 Select candidate chromosomes from the current population using the average fitness selection method.
- Step 4 Perform the crossover and mutation. To obtain two children candidate chromosomes, apply the crossover operator and the mutation operator to the selected chromosomes according to P_c and P_m, respectively.
- Step 5 (5.1) Assign flow for each child (routing table).
  - (5.2) Sum the flow on the links in each child.
  - (5.3) Calculate the fitness of each child using Eq. (3).
  - (5.4) Check the delay time of each child to see whether it is less than T. If not, generate another chromosome.
  - (5.5) If the edge flow exceeds the capacity on an edge, set the capacity of this edge to a higher link type than the assigned flow.
- Step 6 Check the number of children. If number of children < pop_size–1, go to step 3; otherwise, go to step 7.
- Step 7 Establish the new population. Replace the parents with children.
- Step 8 Calculate eval(x) for each chromosome in the new population using Eq. (3).
- Step 9 Get the best chromosome of the new population and compare it with the best one of the previous generation. If it is better, replace the best chromosome of the previous generation with it.
- Step 10 Perform the terminating test. If g < maxgens, set g = g+1, go to step 3 for the next generation; otherwise terminate.

The required optimal network routing table will be the one represented by the best chromosome of all generations.

3.2 Capacity Assignment Problem

The capacity assignment problem focuses on finding the best possible set of capacities for the links that satisfies the traffic requirements in a computer network while minimizing the cost and/or delay time. This is accomplished by selecting the capacity of each link from a discrete set of capacities. The capacity assignment problem can be stated mathematically as follows [14]: *Find the best link capacities in order to Minimize cost Z, given by Eq. (1), and Minimize delay D, given by Eq. (4).*

The currently acclaimed solutions to the capacity assignment problem are primarily based on heuristics that attempt to determine the best capacity of each link once the set of requirements are specified. In this section we will provide a GA for solving this capacity assignment problem that includes the routing GA, described in Sec. 3.1, in the flow assignment step.

3.2.1 Representation

Solutions to the capacity assignment problem are coded in the form of chromosomes made up of a sequence of integer numbers 0, 1, 2, 3, … that symbolize the existence or nonexistence of a link. The numbers 1, 2, 3, … refer to the available link capacities and a link with capacity 0 indicates that the link does not exist. In our implementation, the different capacity types are shown in Table 1, [14].

Table 1 The capacity types of links

Link Capacity Type	Capacity (Mbps)	Costs units/kilometer
1	6	1
2	45	4
3	150	9

The length of the chromosome is n(n-1)/2, where n is the number of nodes. For example, the link capacity of the network shown in Figure 1 can be represented as integer string as shown in Figure 5.

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

Figure 5 A capacity chromosome of the network of Figure 1.

Note that the position k of a link (i,j) in the chromosome is determined by the following formula [15]:

$$k = \frac{n(n-1)}{2} - \frac{(n-i)(n-i+1)}{2} + (j-i) \tag{5}$$

3.2.2 Initial population

We generate randomly a finite set of individuals which represents the initial population. Each individual (chromosome) is composed of L random numbers from the set {0, 1, 2, 3}, where L is the chromosome length. The size of the set is predetermined before applying the GA procedure.

3.2.3 Evaluation

To determine the ability of a chromosome to survive and produce offspring, Eq. (1) and Eq. (4) are used as objective functions. When we apply GAs to a n-objective optimization problem, we have to evaluate the values of n objective functions for each solution. A weighted-sum objective is used to evaluate the chromosome. The way to transform the values of objective functions to the fitness value of each string x in the generation is to combine the *n* objective functions $f_i(x)$ into a scalar function as follows [18]:

$$f(x) = w_1 f_1(x) + w_2 f_2(x) + \ldots + w_n f_n(x) \tag{6}$$

where f(x) is the fitness function of x, and $w_1,\ldots,w_n$ are non-negative weights for the n objectives. The weight values are determined as follows [19]:

$$w_i = r_i/(r_1 + r_2 + \ldots + r_n) \tag{7}$$

where $r_1,r_2,\ldots,r_n$ are non-negative random real numbers (or non-negative random integers). The fitness function with various weights is utilized in the selection operator.

We can define the evaluation function eval(x) of the capacity assignment problem with weighted-sum of the cost function, Eq. (1), and delay time function, Eq. (4), as follows [17]:

$$\text{eval}(x) = w_1 \sum_{(i,j)} d_{ij}(C_{ij}) + w_2 \frac{1}{\gamma} \sum_{(i,j)} \frac{F_{ij}}{C_{ij} - F_{ij}} \quad (8)$$

3.2.4 Selection

Selection is an operator to select two parent strings for generating new strings (i.e., offspring). In the selection, the fitness value of each string is utilized for selecting parent strings. A string with a high fitness value has more chance to be selected as one of parents than a string with a low fitness value. Again, we applied the average-fitness selection scheme [17].

3.2.5 Crossover and Mutation operators

Single-point crossover operation is used. The bit flip mutation is employed, which is performed on a bit-by-bit basis. If bit i has a value $t \neq 0$, then its new value is randomly chosen from the set {1, 2, 3} - t.

3.2.6 Overall capacity assignment GA

The steps of the overall GA algorithm that solves the capacity assignment problem are as follows:

Step 1 Input the problem data and set the GA parameters: Input the network topology and OD matrix. Set the population size (pop_size), crossover rate (P_c), mutation rate (P_m), delay time threshold (T) and the maximum generation (maxgens), and initialize the generation number g=0.

Step 2 Initialization:
- (2.1) Generate the initial population randomly.
- (2.2) (Assign flows): Calculate the link flows for the initial population by the routing GA given in Sec. 3.1
- (2.3) Calculate the fitness eval(x) of each candidate network x in the population using Eq. (8).
- (2.4) Save the best chromosome of the current generation .

Step 3 Select candidate chromosomes from the current population by the average fitness method.

Step 4 Perform the crossover and mutation. To obtain two children, apply the crossover and the mutation operators to the selected chromosomes according to Pc and Pm, respectively.

Step 5 Calculate the link flows for each child.

Step 6 Check the number of children. If number of children < pop_size–1, go to Step 3; otherwise, go to step 7.

Step 7 Establish the new population. Replace the parents with children.

Step 8 Calculate eval(x) for each network in the new population.

Step 9 Get the best chromosome of the new population and compare it with the best one of the previous generation. If it is better, replace the best chromosome of the previous generation with it.

Step 10 Perform the terminating test. If g < maxgens, set g = g+1, go to step 3 for the next generation; otherwise terminate.

The required optimal network capacity assignment will be the one represented by the best chromosome of all generations.

Implementation of Simulated Annealing Algorithm for Solving the RCA Problem

Simulated annealing (SA), developed by Metropolis, is an important stochastic adaptive method based on the physical process of annealing. In this algorithm, a solid is heated up to a liquid phase by increasing temperature and followed by slowly lowering the temperature for cooling. SA is good at hill climbing for optimum solutions. However, due to the random processes to search for the minimum energy state, the convergence speed of simulated annealing is very slow [20]. 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 (Äf < 0), then replace the current solution by the new one. If Äf ≥ 0, then accept a move with probability exp(-Äf/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(-Äf/T). If exp(-Äf/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 following subsections describe the basic elements of the SA algorithm for solving the capacity assignment problem, followed by the overall algorithm for solving the whole RCA problem.

4.1 Solution space and energy function

The most obvious choices for the solution space S and the energy function f are the set F of all solutions to the capacity assignment problem and the fitness function *f(G)*, respectively. Solutions to the capacity assignment problem are coded in the form of vectors, as the chromosomes of the capacity assignment GA. The energy function is the same as the fitness function given by Eq. (4.8).

4.2 The Basic iteration

At each step, the SA heuristic considers some neighbor S' of the current state *S* and probabilistically decides between moving the system to state S' or staying put in state *S*. The probabilistic decisions are chosen so that the system ultimately tends to move to states of lower energy. Typically, this step is repeated until the system reaches a state, which is good enough for the application, or until a given computation budget has been exhausted.

4.3 Neighborhood structure

We define the neighborhood of a network G as being composed of all graphs with different link capacities with respect to G. So, the selection of a solution in the neighborhood of the initial solution can be obtained by changing randomly one link capacity**.**

4.4 Rules of acceptance

The principle of SA requires that one accepts, occasionally and under the control of the “temperature”, an increase in the energy of the current state enables it to be pulled out of a local minimum. The rule of acceptance generally used is the Metropolis rule described above.

4.5 Cooling schedule

For the cooling schedule, we have chosen a heuristic proposed by Lundy and Mees [21], and used with good results in [22]. This schedule performs only one iteration for each temperature level ($N(t)=1, \forall t$) and uses a temperature function $T(t+1)=T(t)/(1+B\times T(t))$. This function decreases more slowly than the exponential schedule, but without guaranteeing global convergence. The actual speed of descent in temperature is proportional to the cooling constant B; Connolly [20] suggests taking $B<<T_{\circ}$.

4.6 Stopping criterion

The stopping criterion selected is a total number of iterations: the search ends when t= MAX_R (maximum number of repetitions). In our implementation, this is equivalent to setting a final temperature level, as follows: $T_f = T_{\circ} /(1+MAX\_R\times B\times T_{\circ})$.

The above choices of SA parameters are suitable for the capacity assignment problem. But there still some free parameters: the initial temperature, $T_{\circ}$; the cooling constant B; the total number of repetitions MAX_R. All these must be fixed in a problem by problem basis. The steps of the SA algorithm for solving the capacity assignment problem are shown in Figure 6.

4.7 The overall SA-based algorithm for solving RCA problem

In this subsection we present the overall algorithm for solving the whole RCA problem, which uses the SA algorithm for solving capacity assignment part of the problem. To achieve a fair comparison later with the GA for solving the same problem, the overall algorithm was designed to generate sets of POPSIZE capacity solutions in each iteration. Each capacity solution is coded as described in Sec. 4.1. In each generation, the SA algorithm, given in Figure 6, is applied to the best individual (solution) of this generation to get a better solution.

```
Select an initial state $v_c \epsilon S$;
Evaluate $v_c$;
Select an initial temperature T > 0;
Select the cooling constant B and maximum number of repetitions MAX_R;
$T_f$ = T/(1 + MAX_R * B * T);
N(t) = 1;
Set temperature change counter t = 0;
Repeat
    Set repetition counter n = 0;
    Repeat
```

Figure 6 The SA Algorithm for solving the capacity assignment problem

The steps of the proposed overall algorithm are as follows:

Step 1 Input the problem data and set the parameters: Input the network topology and OD matrix. Set the population size (pop-size), maximum generation (max-gen), and initialize generation number gen = 0.

Step 2 Initialize.
- (2.1) Randomly generate the initial population.
- (2.2) (Assign flow): calculate the link flows for each individual in the initial population by the routing algorithm given in Sec. 4.8.
- (2.3) Calculate the fitness eval(x) of each candidate network x in the population using Eq (8).
- (2.4) Take the best individual of the current generation and send it as an initial solution to the SA algorithm, given in Figure 6.
- (2.5) Save the solution obtained from the SA algorithm as the best individual of the current generation.

Step 3 Select a new population from the current population by the average fitness selection method.

Step 4 (4.1) Replace current population with the new population.
- (4.2) (Assign flow): calculate the link flows for each individual in the new population by the routing algorithm given in Sec. 4.8.
- (4.3) Calculate the fitness eval(x) of each individual x in the population using Eq (8).
- (4.4) Take the best individual of the current generation and send it as an initial solution to the SA algorithm, given in Figure 6.
- (4.5) Compare the solution obtained from the SA algorithm with the best one of the previous generation. If it is better, replace the best individual of the previous generation with it.

Step 5 Perform the terminating test. If gen < max-gen, set gen = gen +1 and return to Step 3 for the next generation. If gen = max-gen, terminate.

The required optimal network capacity assignment will be the one represented by the best individual of all generations.

4.8 The routing algorithm used in the overall SA-based algorithm

Given the capacities of all links of the given network topology in the form of a capacity solution, the steps of the algorithm that solves the flow assignment (routing) problem are as follows:

Step 1 Generate a random routing table for the given network topology, i.e. find a path between each source and destination.

Step 2 Assign flow for the links of the routing table.

Step 3 Evaluate the cost for the routing table using Eq. (3).

Step 4 Evaluate the delay time for the routing table by using the constraint (2). If the delay time does not satisfy this constraint, generate another routing table.

Step 5 If the edge flow exceeds the capacity on an edge, set the capacity of this edge to a higher link type than the assigned flow.

The Proposed Hybrid Simulated Annealing – Genetic Algorithm Approach

An essential feature of SA is that it can climb out from a local minimum, since it can accept worse neighbors at the next step [23]. So, to avoid the premature convergence and improve the hill climbing ability of the GA, we present a proposed approach in which we merge the GA with the SA algorithm (HGSA). This is done by replacing the standard crossover process of the capacity assignment GA by the SA algorithm shown in Figure 6. This combined algorithm can keep the advantages and avoid the disadvantages of both search algorithms.

The steps of the proposed HGSA are as follows:

Step 1 Input the problem data and set the parameters: Input the network topology and OD matrix. Set the population size (pop-size), maximum generation (max-gen), crossover rate (P_c), mutation rate (P_m), delay time threshold (T) and initialize generation number gen = 0.

Step 2 Initialize.

(2.1) Generate the initial population randomly.

(2.2) (Assign flows): Calculate the link flows for each chromosome in the initial population by the routing GA given in Sec. 3.1.7.

(2.3) Calculate the fitness eval(x) of each candidate network x in the population using Eq. (8).

(2.4) Save the best chromosome of the current generation.

Step 3 Select candidate networks from the current population by the average fitness selection method

Step 4 Select a number of chromosomes randomly according to P_c and send each one as an initial solution to the SA algorithm given in Figure 6.

Step 5 Replace each selected chromosome by the corresponding best solution obtained from the SA algorithm.

Step 6 Perform the mutation operator to the selected chromosomes according to P_m.

Step 7 Replace current population with the new population.

Step 8 Calculate the link flows for each individual in the new population by the routing GA.

Step 9 Calculate eval(x) of each network in the new population.

Step 10 Get the best chromosome of the new population and compare it with the best one of the pervious generation. If it is better, replace the best chromosome of the previous generation with it.

Step 11 Perform the terminating test. If $g <$ maxgens, set $g = g+1$, go to Step 3 for the next generation; otherwise terminate.

The required optimal network capacity assignment will be the one represented by the best chromosome of all generations.

EXPERIMENTAL RESULTS

In this section, we present a comparison between the performance of a Heuristic Algorithm (HA), introduced by Bertsekas and Gallager [9], and the three proposed approaches: Genetic Algorithm (GA), Simulated Annealing (SA), and Hybrid Genetic-Simulated Algorithm (HGSA) for solving the RCA problem in computer networks. All of the experiments were carried out using Pentium ® 2.20 GHz. All the four algorithms were implemented in C++.
In the experiments, the four algorithms are applied to the example network shown in Figure 7. This network consists of node 10 and 18 full duplex links. For this network, we assumed a total arrival rate (γ) of 668 packet/second, and an average delay constraint (T) of 0.2 sec/packet. The traffic requirement for this network is shown in Table 2, and the used link capacity types are as shown in Table 1. The parameters of the GA are: Pm=0.1, Pc=0.25, and pop-size=10.

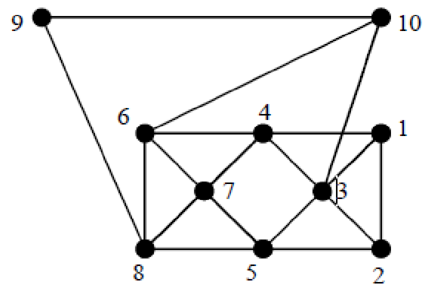

Figure 7 An example network with 10 nodes and 36 links

Table 2 Predicted traffic requirement (flow) between 10 nodes (in Mbps)

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

Table 3 The results of applying the HA to the example network

Delay	Cost	Iterations
0.06	299.079	10
0.05	261.211	20
0.05	261.211	60
0.06	259.327	100
0.06	259.327	150
0.06	259.327	200
0.04	242.079	300
0.04	241.683	500

Tables 3-6 show the results of applying the HA, GA, SA, and HGSA, to the example network. From these results we can see that, by increasing the number of iterations/generations, the cost decreases, but the delay fluctuates for all methods except the SA where it decreases.

We applied the four algorithms HA, GA, SA, and HGSA to the example network with different pop-sizes: 10, 20, and 60, and compared the cost and delay in each case. Table 7 shows the comparison between the results of the four algorithms. From this table, we can see that:

- In all cases, the HGSA reached a solution with the best cost, but the delay is a little bit higher.
- A solution with the second best cost is reached by the SA in the cases of pop-size = 10 and 20, and by the GA in the case of pop-size = 60.
- In all cases, the heuristic algorithm reached a solution with the smallest delay.

From these results, we conclude that the use of the SA with the GA enhanced the performance of the GA.

Table 4 The results of applying the GA to the example network

Delay	Cost	Maxgens for Routing GA	Maxgens for Capacity GA	Pop size
0.09	260.664	10	10	10
0.13	250.280	20	20	10
0.11	228.060	60	60	10
0.15	221.240	80	80	10
0.16	216.664	150	150	10
0.15	210.284	200	200	10
0.12	192.464	300	300	10
0.16	184.191	500	500	10

Table 5 The results of applying the SA to the example network

Delay	Cost	Maxgens	Pop size
0.15	251.013	10	10
0.11	247.933	20	10
0.11	234.440	60	10

Table 6 The results of applying the HGSA to the example network

Delay	Cost	Maxgens for Routing GA	Maxgens for Capacity GA	Pop size
0.14	246.686	10	10	10
0.18	238.620	20	20	10
0.14	225.669	60	60	10

Table 7 A comparison between the four methods HA, GA, SA, and HGSA

Best		Method	Pop-size
Delay	Cost		
0.06	299.079	Heuristic	10
0.09	260.664	GA	
0.15	251.013	SA	
0.14	246.686	GA+SA	
0.05	261.211	Heuristic	20
0.13	250.280	GA	
0.11	247.933	SA	
0.18	238.620	GA+SA	
0.05	261.211	Heuristic	60
0.11	228.060	GA	
0.11	234.440	SA	
0.14	225.669	GA+SA	

REFERENCES

[1] L. Baccouche and H. Eleuch, "*Rt-Dbp: a multi-criteria priority assignment scheme for real-time tasks scheduling*", *Appl. Math. Inf. Sci.*, Vol. **6**, No. 11, pp. 382–388 (2012).

[2] A.Jemai, A. Mastouri and H. Eleuch, *"A tabu search algorithm for the routing and capacity assignment problem in computer networks", Appl. Math. Inf. Sci.* Vol. **5**, No. 3, pp. 655–667 (2011).

[3] Jian Shen, Fuyong Xu, and Peng Zheng, *"Study of key pre-distribution schemes in wireless sensor networks: case of BROSK (use of WSNet)"*, Computers & Operations Research, Vol. **32**, No. 11, pp. 2785–2800 (2005).

[4] Guangming Lin, Chengbo Huang, Shaobin Zhan, Xin Lu and Yunting Lu, *"Ranking Based Selection Genetic Algorithm for Capacity Flow Assignments"*, Communications in Computer and Information Science, Vol. **107**, pp. 97-107 (2010).

[5] Emilio C.G. Wille, Marco Mellia, Emilio Leonardi, Marco Ajmone Marsan, "A Lagrangean relaxation approach for QoS networks CFA problems", *International Journal of Electronics and Communications (AEÜ),* Vol. **63**, pp. 743–753 (2009).

[6] Xiao-Hui Lin, Yu-Kwong Kwok, Vincent K.N. Lau, *"A genetic algorithm based approach to route selection and capacity flow assignment", Computer Communications,* Vol. **26**, pp. 961–974 (2003).

[7] Tianze Xu , Heng Wei , and Zhuan-De Wang, *"Study on continuous network design problem using simulated annealing and genetic algorithm", Expert Systems with Applications,* Vol. **36**, No. 2, pp. 2735-2741 (2009).

[8] Nasser A. El-Sherbeny, "Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods", *Journal of King Saud University- Science*, Vol. **22**, No. 3, pp. 123–131, 2010.

[9] D. Bertsekas and R. Gallager, *Data network*, 2nd Edition, Prentice Hall of India, New Delhi, (1997).

[10] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, Ann Arbor, IL (1975).

[11] J. H. Holland, "*Genetic algorithms", Scientific American*, Vol. **267**, No. 1, pp. 66–72, July (1992).

[12] Celso C. Ribeiro , Simone L. Martins , and Isabel Rosseti, *"Metaheuristics for optimization problems in computer communications", Computer Communications,* Vol. **30**, No. 4, pp. 656–669, February (2007).

[13] M. Munetomo, Y. Takai, and Y. Sato, "*An Adaptive network routing algorithm employing path genetic operators",* In Proceedings of the 7th International Conference on Genetic Algorithms", Morgan Kaufmann Publishers, San Francisco, pp. 643-649 (1997).

[14] K. Ko, K. Tang, C. Chan, K. Man and S. Kwong, *"Using genetic algorithms to design mesh networks", IEEE Computer,* Vol. **30**, No. 8, pp. 56–61, Aug (1997).

[15] M. Girgis, T. Mahmoud, and A. Rabie, "A genetic algorithm for solving the delay-constrained minimum-cost network design problem", *Proceedings of first international conference on intelligent computing information systems, ICICIS 2002,* Cairo, Egypt, pp. 110-116 (2002).

[16] L. Eshelman, M. Mathias, and J. Schaffer, "Crossover operator biases: exploiting in the population distribution", In T. Bäck, editor, *In Proceedings of the 7th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers*, San Francisco, pp. 354-361, USA (1997).

[17] M. Gen and R. Cheng, *Genetic Algorithms and engineering optimization,* John Wiley & Sons, Inc.,(2000).

[18] T. Murata, "*Genetic algorithms for multiobjective optimization",* Ph.D. dissertation, University of Osaka Prefecture, Japan (1996).

[19] R. Cheng and M. Gen, "An adaptive superplane approach for multiple objective", Technical Report, Ashikaga Institute of Technology (1998).

[120] Shun-Fa Hwang, and Rong-Song He, *"Improving real-parameter genetic algorithm with simulated annealing for engineering problems",* Advances in Engineering Software, Vol. **37**, No. 6, pp 406-418 (2006).

[21] M. Lundy, and A. Mees, "Convergence of an annealing algorithm*", Mathematical programming*, Vol. **34**, No. 1, pp. 111–124 (1986).

[22] D. Connolly, "An improved annealing scheme for the QAP", *European Journal of Operational Research*, Vol. 46, No. 1, pp. 93-100 (1990).

[23] R. Tavakkoli-Moghaddam, N. Safaei, Y. Gholipour, "A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length", *Applied Mathematics and Computation*, Vol. **176**, pp. 445–454 (2006).