A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations

Chang Wook Ahn, *Student Member, IEEE,* and R. S. Ramakrishna, *Senior Member, IEEE*

***Abstract*—This paper presents a genetic algorithmic approach to the shortest path (SP) routing problem. Variable-length chromosomes (strings) and their genes (parameters) have been used for encoding the problem. The crossover operation exchanges partial chromosomes (partial routes) at positionally independent crossing sites and the mutation operation maintains the genetic diversity of the population. The proposed algorithm can cure all the infeasible chromosomes with a simple repair function. Crossover and mutation together provide a search capability that results in improved quality of solution and enhanced rate of convergence. This paper also develops a population-sizing equation that facilitates a solution with desired quality. It is based on the gambler's ruin model; the equation has been further enhanced and generalized, however. The equation relates the size of the population, the quality of solution, the cardinality of the alphabet, and other parameters of the proposed algorithm. Computer simulations show that the proposed algorithm exhibits a much better quality of solution (route optimality) and a much higher rate of convergence than other algorithms. The results are relatively independent of problem types (network sizes and topologies) for almost all source–destination pairs. Furthermore, simulation studies emphasize the usefulness of the population-sizing equation. The equation scales to larger networks. It is felt that it can be used for determining an adequate population size (for a desired quality of solution) in the SP routing problem.**

***Index Terms*—Gambler's ruin model, genetic algorithms, population size, shortest path routing problem.**

I. Introduction

IN MULTIHOP networks, such as the Internet and the Mobile *Ad-hoc* Networks, routing is one of the most important issues that has a significant impact on the network's performance [1], [2]. An ideal routing algorithm should strive to find an optimum path for packet transmission within a specified time so as to satisfy the Quality of Service (QoS) [2]–[4]. There are several search algorithms for the shortest path (SP) problem: the breadth-first search algorithm, the Dijkstra algorithm and the Bellman–Ford algorithm, to name a few [1]. Since these algorithms can solve SP problems in polynomial time, they will be effective in fixed infrastructure wireless or wired networks. But, they exhibit unacceptably high computational complexity for real-time communications involving rapidly changing network topologies [3], [4]. This is explained below.

We consider mobile *ad hoc* networks as target systems because they represent new generation wireless networks. Since all the nodes cooperatively maintain network connectivity without the aid of any fixed infrastructure networks, dynamic changes in network topology are possible. An optimal (shortest) path has to be computed within a very short time (i.e., a few μs) in order to support time-constrained services such as voice-, video-, and tele-conferencing [2], [4]. The indicated algorithms do not satisfy this (real-time) requirement.

In most of the current packet-switching networks, some form of SP computation is employed by routing algorithms in the network layer [2], [4]. Specifically, the network links are weighted, the weights reflecting the link transmission capacity, the congestion of networks and the estimated transmission status such as the queuing delay of head-of-line (HOL) packet or the link failure. The SP problem can be formulated as one of finding a minimal cost path that contains the designated source and destination nodes. In other words, the SP routing problem involves a classical combinatorial optimization problem arising in many design and planning contexts [2]–[7]. Since neural networks (NNs) [2]–[4] and genetic algorithms (GAs) (and other evolutionary algorithms) [5]–[16] promise solutions to such complicated problems, they have been used successfully in various practical applications. On the other hand, NNs and GAs may also not be promising candidates for supporting real-time applications in mobile *ad hoc* networks because they involve a large number of iterations in general. However, hardware implementations (e.g., field-programmable gate array (FPGA) chips) of NNs or GAs are extremely fast. Furthermore, they are not very sensitive to network size [4], [13]. The quality of the solution (i.e., computed path) returned by NNs is constrained by their inherent characteristics. GAs are flexible in this regard. The quality (of the solution) can be adjusted as a function of population. In addition, NN hardware is limited in size: it cannot accommodate networks of arbitrary size because of its physical limitation. GA hardware, on the other hand, scales well to networks that may not even fit within the memory. It is realized by employing parallel GA over several nodes. Therefore, GAs (especially hardware implementations) are clearly quite promising in this regard.

A. Review of GA-Based SP Algorithms

Investigators have applied GAs to the SP routing problem [5]–[7], multicasting routing problem [8], [9], ATM bandwidth allocation problem [10], capacity and flow assignment (CFA) problem [11], and the dynamic routing problem [12]. It is noted that all these problems can be formulated as some sort of a combinatorial optimization problem.

Munemoto's algorithm [5] is practically feasible in a wired or wireless environment. It employs variable-length chromosomes

Manuscript received October 11, 2001; revised May 26, 2002. This work was supported by the Korea Ministry of Education under the BK21 Program.

The authors are with The Department of Information and Communications, Kwang-Ju Institute of Science and Technology, Kwang-ju, Korea (e-mail: cwan@kjist.ac.kr; rsr@kjist.ac.kr).

Digital Object Identifier 10.1109/TEVC.2002.804323

for encoding the problem. Crossing sites (points) are the loci (positions of nodes in a route), where identical genes (nodes) in both the chosen chromosomes (routes) are found at the same location. Thus, it leads to a situation in which only a few crossover sites are usable for exploring feasible solutions. In other words, crossover is totally dependent on positions: indeed, identical genes should occupy the same locus for crossover. The set of candidate crossing sites is called a "potential crossing site." A locus is selected randomly to act as an actual crossing site and to partially exchange chromosomes of the parent. In the mutation phase, a gene (the mutation node) is selected randomly from the chromosome. Another gene is selected randomly from the chromosomes connected directly to the mutation node, and a mutated chromosome (alternative route) is generated by combining each partial-chromosome (partial-route) obtained by Dijkstra's algorithm. It must be noted that one partial route refers to a shortest path from the source node to the selected node and the other to a shortest path from the selected node to the destination node. But, the algorithm requires a relatively large population for an optimal solution due to the constraints on the crossover mechanism. Furthermore, it is not suitable for large networks or real-time communications, since Dijkstra's algorithm has a prohibitive computational cost.

Inagaki [6] proposed an algorithm that employs fixed (deterministic) length chromosomes. The chromosomes in the algorithm are sequences of integers and each gene represents a node ID that is selected randomly from the set of nodes connected with the node corresponding to its locus number. All the chromosomes have the same (fixed) length. In the crossover phase, one of the genes (from two parent chromosomes) is selected at the locus of the starting node ID and put in the same locus of an offspring. One of the genes is then selected randomly at the locus of the previously chosen gene's number. This process is continued until the destination node is reached. The details of mutation are not explained in the algorithm. The algorithm requires a large population to attain an optimal or high quality of solution due to its inconsistent crossover mechanism. Some offspring may generate new chromosomes that resemble the initial chromosomes in fitness, thereby retarding the process of evolution.

There are several GAs that address different kinds of routing problems, such as multiple destination or multicasting routing problems [7]–[9]. Those approaches are beyond the scope of this paper. However, the unicasting or one-destination algorithms such as the one proposed here can be extended naturally and easily to include them.

B. Review of Population-Sizing Equation

A population size that guarantees an optimal solution quickly enough is a topic of intense research [17]–[22]. Large populations usually result in better solutions, but at increased computational costs.

Holland [17] studied the (k-armed) bandit problem as a theoretical motivation for GAs. Macready and Wolpert [18] showed a mathematical flaw in Holland's analysis and provided an analytically simple bandit model that is directly applicable to optimization theory.

De Jong *et al.* [19] proposed a population-sizing equation based on the signal as well as noise characteristics of the k-armed bandit problem. Although the result explicitly exhibited the role of signal-to-noise ratio in estimating population size, the result was unverified and ignored [21], [22].

Goldberg *et al.* [20] developed the first population-sizing equation based on the variance of fitness. Goldberg *et al.* [21] enhanced the equation as a conservative bound on the quality of GAs. The population-sizing equation permits accurate statistical decision making among competing building blocks. The population-sizing relation conservatively bounds the actual accuracy of GA convergence as long as all major sources of noise (collateral noise) are considered in the sizing calculation.

Harik *et al.* [22] also developed a population-sizing equation that is inspired by the classical random walk problem: the gambler's ruin problem in particular. Using test problems that ranged from the simple to the very difficult, the accuracy of the model was verified. Their work, however, was limited to fixed-length chromosomes. Furthermore, it required stochastic information such as the variance of fitness (noise) and the expected difference value of fitness (signal) between best and second-best building blocks (BBs)—partitions or schemata. However, such information may not be available in many practical problems such as the routing or the scheduling problems.

C. Objectives and Organization

In this paper, we propose a new GA for solving the SP routing problem. Variable-length chromosomes have been employed. Their elements represent nodes included in a path between a designated pair of source and destination nodes. The crossover exchanges partial chromosomes (partial-routes) and the mutation introduces new partial chromosomes (partial-routes). Lack of positional dependency in respect of crossing sites helps maintain diversity of the population. In addition, a simple repair function has been proposed to deal with all the infeasible chromosomes.

This paper also develops a population-sizing equation that promises a solution of desired quality for the proposed GA. It is based on the gambler's ruin model that is discussed in [22]. However, the equation in this paper has been further enhanced and generalized. It makes the relationships among the size of the population, the quality of solution, the cardinality of alphabet, and other factors of the proposed GA explicit.

The rest of the paper is organized as follows. In Section II, the proposed GA for the shortest path routing problem is described. A population-sizing equation is derived in Section III. In Section IV, the proposed algorithm and several extant algorithms are applied to numerous networks exhibiting arbitrary link cost, network size, and topology. After that, a comparative study of the results follows. Furthermore, the section verifies the accuracy of the population-sizing equation from the simulation studies. The paper concludes with a summary of the results in Section V.

II. Proposed GA for SP Routing Problem

The underlying topology of multihop networks can be specified by the directed graph $\mathbf{G} = (\mathbf{N}, \mathbf{A})$, where $\mathbf{N}$ is a set of

N nodes (vertices), and $\mathbf{A}$ is a set of its links (arcs or edges) [1]–[4]. There is a cost C_{ij} associated with each link (i, j). The costs are specified by the cost matrix $\mathbf{C} = [C_{ij}]$, where C_{ij} denotes a cost of transmitting a packet on link (i, j). Source and destination nodes are denoted by S and D, respectively. Each link has the link connection indicator denoted by I_{ij}, which plays the role of a chromosome map (masking) providing information on whether the link from node i to node j is included in a routing path or not.

It can be defined as follows:

$$I_{ij} = \begin{cases} 1, & \text{if the link from node } i \text{ to node } j \text{ exists in the routing path} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

It is obvious that all the diagonal elements of I_{ij} must be zero. Using the above definitions, the SP routing problem can be formulated as a combinatorial optimization problem minimizing the objective function (2a) as follows:

minimize

$$\sum_{i=S}^{D} \sum_{\substack{j=S \\ j \neq i}}^{D} C_{ij} \cdot I_{ij} \tag{2a}$$

subject to

$$\sum_{\substack{j=S \\ j \neq i}}^{D} I_{ij} - \sum_{\substack{j=S \\ j \neq i}}^{D} I_{ji} = \begin{cases} 1, & \text{if } i = S \\ -1, & \text{if } i = D \\ 0, & \text{otherwise} \end{cases}$$

and

$$\sum_{\substack{j=S \\ j \neq i}}^{D} I_{ij} \begin{cases} \leq 1, & \text{if } i \neq D \\ = 0, & \text{if } i = D \end{cases}$$

$$I_{ij} \in \{0, 1\}, \text{ for all } i. \tag{2b}$$

The constraint (2b) ensures that the computed result is indeed a path (without loops) between a source and a designated destination.

A. Genetic Representation

A chromosome of the proposed GA consists of sequences of positive integers that represent the IDs of nodes through which a routing path passes. Each locus of the chromosome represents an order of a node (indicated by the gene of the locus) in a routing path. The gene of first locus is always reserved for the source node. The length of the chromosome is variable, but it should not exceed the maximum length N, where N is the total number of nodes in the network, since it never needs more than N number of nodes to form a routing path. A chromosome (routing path) encodes the problem by listing up node IDs from its source node to its destination node based on topological information database (routing table) of the network. The information can be easily obtained and managed in real-time by routing protocols such as RIP [23], OSFP [24], DSDV [25], DSR [26], and VCRP [27] in wired or wireless environments, but the detailed mechanisms or other controversial issues are beyond the scope of this paper. It is noted that the topological information database of the network can be constructed easily and rapidly by such routing protocols.

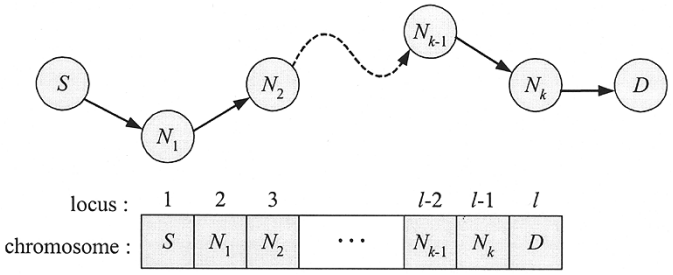


Fig. 1. Example of routing path and its encoding scheme.

An example of chromosome (routing path) encoding from node S to node D is shown in Fig. 1. The chromosome is essentially a list of nodes along the constructed path, ($S \rightarrow N_1 \rightarrow N_2 \rightarrow \cdots \rightarrow N_{k-1} \rightarrow N_k \rightarrow D$). In Fig. 1, l represents the total number of nodes forming a path.

The gene of the first locus encodes the source node, and the gene of second locus is randomly or heuristically selected from the nodes connected with the source node (S) that is represented by the front gene's allele. A chosen node is removed from the topological information database to prevent the node from being selected twice, thereby avoiding loops in the path. This process continues until the destination node is reached. It is noted that an encoding is possible only if each step of a path passes through a physical link in the network.

B. Population Initialization

In general, there are two issues to be considered for population initialization of GAs: the initial population size and the procedure to initialize the population [14], [15].

It was felt that the population size needed to increase exponentially with the complexity of the problem (i.e., the length of the chromosome) in order to generate good solutions. Recent studies have shown, however, that satisfactory results can be obtained with a much smaller population size. To summarize, a large population is quite useful, but it demands excessive costs in terms of both memory and time [14]–[22]. As would be expected, deciding adequate population size is crucial for efficiency.

Secondly, there are two ways to generate the initial population: heuristic initialization and random initialization. Although the mean fitness of the heuristic initialization is already high so that it may help the GAs to find solutions faster, it may just explore a small part of the solution space and never find global optimal solutions because of the lack of diversity in the population [15]. Therefore, random initialization is effected in this paper so that the initial population is generated with the encoding method already explained in Section II-A. Physically, the random initialization chooses genes (nodes) from the topological information database in a random manner during the encoding process. It is possible that the algorithm encounters a node for which all of whose neighboring nodes have already been visited. In this case, the defective chromosome is refreshed and reinitialized. This may induce a subtle bias in which some partial paths are more likely to be generated. However, the meager bias does not significantly affect the performance of the algorithm. It is doubly

so because it (the bias) vanishes after evolving just a few generations.

C. Fitness Function

The fitness function interprets the chromosome in terms of physical representation and evaluates its fitness based on traits of being desired in the solution [7], [15]. But, the fitness function must accurately measure the quality of the chromosomes in the population. The definition of the fitness function, therefore, is very critical [15]. The fitness function in the SP routing problem is obvious because the SP computation amounts to finding the minimal cost path. Therefore, the fitness function that involves computational efficiency and accuracy (of the fitness measurement) is defined as follows:

$$f_i = \frac{1}{\sum_{j=1}^{l_i-1} C_{g_i(j),\, g_i(j+1)}} \tag{3}$$

where f_i represents the fitness value of the ith chromosome, l_i is the length of the ith chromosome, $g_i(j)$ represents the gene (node) of the jth locus in the ith chromosome, and C is the link cost between nodes.

The fitness function of GAs is generally the objective function that requires to be optimized [7], [14], [15]. In a sense, the fitness function [(3)] can be thought of as fully reflecting the objective function [(2a)]. The fitness function has a higher value when the fitness characteristic of the chromosome is better than others. In addition, the fitness function introduces a criterion for selection of chromosomes.

D. Selection

The selection (reproduction) operator is intended to improve the average quality of the population by giving the high-quality chromosomes a better chance to get copied into the next generation [14], [15]. The selection thereby focuses the exploration on promising regions in the solution space. Selection pressure characterizes the selection schemes. It is defined as the ratio of the probability of selection of the best chromosome in the population to that of an average chromosome. Hence, a high selection pressure results in the population's reaching equilibrium very quickly, but it inevitably sacrifices genetic diversity (i.e., convergence to a suboptimal solution).

There are two basic types of selection scheme used commonly in current practice: proportionate and ordinal-based selection [14], [15]. Both selection schemes suffer when the selection pressure is inadequate (i.e., low or high).

Proportionate selection picks out chromosomes based on their fitness values relative to the fitness of the other chromosomes in the population. It is generally more sensitive to selection pressure. Hence, a scaling function is employed for redistributing the fitness range of the population in order to adapt to the selection pressure. Examples of such a selection type include roulette wheel selection, stochastic remainder selection, and stochastic universal selection.

Ordinal-based selection schemes select chromosomes based not upon their fitness, but upon their rank within the population. The chromosomes are ranked according to their fitness values. It is noted that the selection pressure (intensity) is independent of the fitness distribution of the population, and is based solely on the relative ranking of the population. Since the selection pressure is the degree to which the better chromosomes are favored, it drives the GAs toward improved population fitness over succeeding generations. However, it can become malicious when the selection pressure does not impose an adequate level. In other words, it may also suffer from high selection pressure. Tournament selection, (μ, λ) selection, truncation selection, and linear ranking selection schemes are included in the ordinal-based selection type.

On the other hand, tournament selection without replacement is perceived as an effort to keep the selection noise as low as possible [21]. Recall that tournament selection without replacement works by means of choosing nonoverlapping random sets of s chromosomes (s tournament size) from the population and then selecting the best chromosome from each set to serve as a parent for the next generation. Typically, the tournament size s is two (pairwise tournament), and it would adjust the selection pressure: the selection pressure increases as the tournament size s becomes larger [15], [22]. Recall that the selection pressure is the expected average fitness of the population after selection. As selection pressure increases, the probability of making the wrong decision increases exponentially although the convergence of the GAs may be fast [22].

Therefore, the pairwise tournament selection without replacement is employed for the proposed GA: two chromosomes are picked and the one that is fitter is selected. However, the same chromosome should not be picked twice as a parent.

E. Crossover

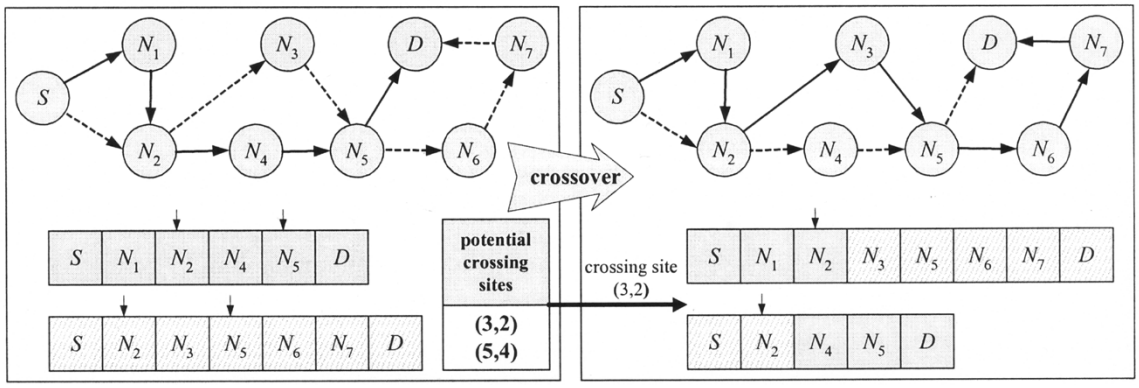
Crossover examines the current solutions in order to find better ones [14], [15]. Physically, crossover in the SP routing problem plays the role of exchanging each partial route of two chosen chromosomes in such a manner that the offspring produced by the crossover represents only one route. This dictates selection of one-point crossover as a good candidate scheme for the proposed GA. One partial route connects the source node to an intermediate node, and the other partial route connects the intermediate node to the destination node. The crossover between two dominant parents chosen by the selection gives higher probability of producing offspring having dominant traits.

But the mechanism of the crossover is not the same as that of the conventional one-point crossover. In the proposed scheme, two chromosomes chosen for crossover should have at least one common gene (node) except for source and destination nodes, but there is no requirement that they be located at the same locus. That is, the crossover does not dependent on the position of nodes in routing paths. Fig. 2(a) and (b) shows the pseudocode and an example of the crossover procedure, respectively.

As shown in Fig. 2(b), a set of pairs of nodes which are commonly included in the two (chosen) chromosomes but without positional consistency is formed first [i.e., (3,2) and (5,4)]. Such pairs are also called "potential crossing sites." Then, one pair [i.e., (3,2)] is randomly chosen and the locus of each node becomes a crossing site of each chromosome. The crossing points of two chromosomes may be different from each other. This

```
/*  C1,  C2 : Input chromosomes,  C1*,  C2* : Output chromosomes*/
/*  l1 : Length of chromosome  C1,  l2 : Length of chromosomes  C2  */
for (all  i ,  j ) { /* Find the potential crossing sites */
    if ( C1[i] == C2[ j] ) {/* If a node is commonly included in both chromosomes */
        sp[k] = (i, j) ; } /* Construct a set of potential crossing sites */
}
sc = choose_rand(sp) ; /* Randomly choose a crossing site */
C1* = C1[1 : sc(1)] // C2[sc(2) + 1 : l2] ; /* 1st exchange */
C2* = C2[1 : sc(2)] // C1[sc(1) + 1 : l1] ; /* 2nd exchange */
```

(a)



(b)

Fig. 2. Overall procedure of the crossover. (a) Pseudocode of the crossover. (b) Example of the crossover procedure.

is in contrast to the scheme adopted in Munemoto's algorithm [5]. Each partial route is exchanged and assembled and thus, two new routes are produced eventually. It is possible that loops are formed during crossover. A simple countermeasure must be taken in this regard: the rate of convergence and the quality of solution. Of course, such chromosomes (routes with loops) will gradually be weeded out in the course of a few generations because the traits of those chromosomes drive fitness values from bad to worse. Repair and penalty functions are the usual countermeasures. This is described in Section II-G.

F. Mutation

The population undergoes mutation by an actual change or flipping of one of the genes of the candidate chromosomes, thereby keeping away from local optima [14], [15]. Physically, it generates an alternative partial-route from the mutation node to the destination node in the proposed GA. A topological information database is utilized for the purpose. Of course, mutation may induce a subtle bias for reasons indicated earlier (Section II-B). However, this bias can be ignored. This is explained below.

First, mutation leads to an infinitesimal increase in the probability of inducing the bias. Second, selection and crossover heavily influence the way this bias operates. Indeed, its (bias') harmful effects vanish almost completely. Furthermore, the bias, whenever it helps in searching an optimal solution, may not induce any harmful effect at all.

Fig. 3 shows the overall procedure of the mutation operation. As can be seen from Fig. 3(b), in order to perform a mutation, a gene (i.e., node N_2) is randomly selected first from the chosen chromosome ("mutation point"). One of the nodes, connected directly to the mutation point, is chosen randomly as the first node of the alternative partial-route. The remaining procedure follows in that of Section II-A and II-B.

However, nodes already included in an upper partial route should be deleted from the database so as not to include the same node twice in the new routing path. The upper partial route represents the surviving portion of the previous route after mutation; it is the partial chromosome stretching from the first gene to the intermediate gene at the mutation point.

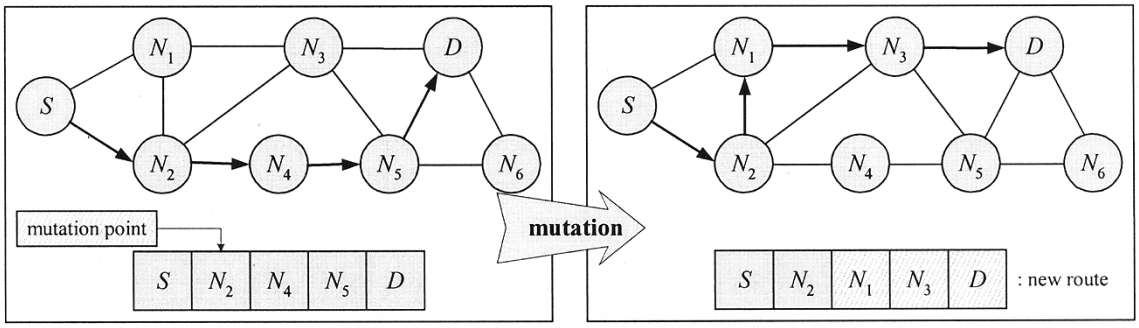
G. Repair Function

As mentioned earlier, crossover may generate infeasible chromosomes that violate the constraints of (2b), generating loops in the routing paths. It must be noted that none of the chromosomes of the initial population or after the mutation is infeasible because when once a node is chosen, it is excluded from the candidate nodes forming the rest of the path.

There are two strategies to deal with infeasible chromosomes: one is to repair them and the other is to impose a penalty [15]. Although the repair method is applied extensively, it is not always simple to cure infeasible chromosomes. A classical method employs penalty functions. It must be noted that the penalty function is critical to ensure quick convergence and high quality (of solution). But it is not easy to come up with an appropriate penalty function. Moreover, this technique may sacrifice some feasible chromosomes as well because the infeasible chromosomes might continue to be reproduced in the proposed GA. Fortunately, the mechanism that eliminates the lethal genes (forming a loop) can cure all the infeasible chro-

```
/*  C : Input chromosome,  C* : Output chromosome, T: Topological information database */
s_m = choose_rand(C) ; /* Randomly choose a node as a mutation point */
delete (T, C, s_m) ; /* Delete the nodes of upper partial-route from T */
C* = C[1 : s_m] ; /* Put the upper partial-route to the mutation chromosome */
while (1) {
    C*[s_m + 1] = choose_rand_delete (T, C*[s_m]) ; /* Randomly choose a node from T &
                                                       delete the node from T */
    if ( C*[s_m + 1] == destination ) {
        break; }
    s_m ++ ; /* Increment the  s_m  */
}
```

(a)



(b)

Fig. 3. Overall procedure of the mutation. (a) Pseudocode of the mutation. (b) Example of the mutation procedure.

mosomes in the proposed GA. The repair function finds and eliminates loops in a routing path without unduly increasing computational costs.

The proposed repair function is described in Fig. 4(a) and an example is shown in Fig. 4(b). In Fig. 4(b), one of the offspring produced after crossover becomes infeasible because the new route contains the loop $N_2 \rightarrow N_3 \rightarrow N_1 \rightarrow N_2$. The repair function detects the loop by a simple search described in Fig. 4(a). The function is linear in l, the chromosome length. After that, the lethal genes (forming the loop) that violate the constraint condition are deleted; those nodes are $\{N_3, N_1, N_2\}$ (but one of two N_2) in this example.

III. Population-Sizing Equation

In this section, the population size (number of chromosomes) that ensures a specified quality of solution is investigated by employing the gambler's ruin problem that was considered first by Harik [22]. The question as to how to choose an adequate population size for a particular domain is difficult and has puzzled practitioners for a long time [17]–[22]. If the population size is too small, it is not likely that the GAs will find solutions of high quality. However, if the population size is too large, the GAs will unnecessarily waste processing time leading to unacceptably slow convergence. Harik [22] exploited the similarity between the gambler's ruin problem and the selection mechanism of GAs for determining an adequate population size that guarantees a solution of the desired (target) quality. The (linear) ranking selection was tacitly assumed because the decision model [21] (it leads directly to results in [22]) is quite appropriate under this selection scheme. It was also assumed implicitly that mutation is not a dominant operator (i.e., crossover-intensive) because it always disrupts BBs. In order to use his results, however, several domain-dependent variables must be known such as the signal that is defined by the fitness difference between the best and second best BBs, the noise that is defined by the root mean square (rms) fitness variance of the BB that is being considered, and the number of BBs in a string. His population-sizing equation, therefore, is not suitable for applying to the SP routing problem because the GA has variable-length chromosomes. Furthermore, signal-to-noise ratio (SNR), the most important piece of information in his work is not usually known in such problems.

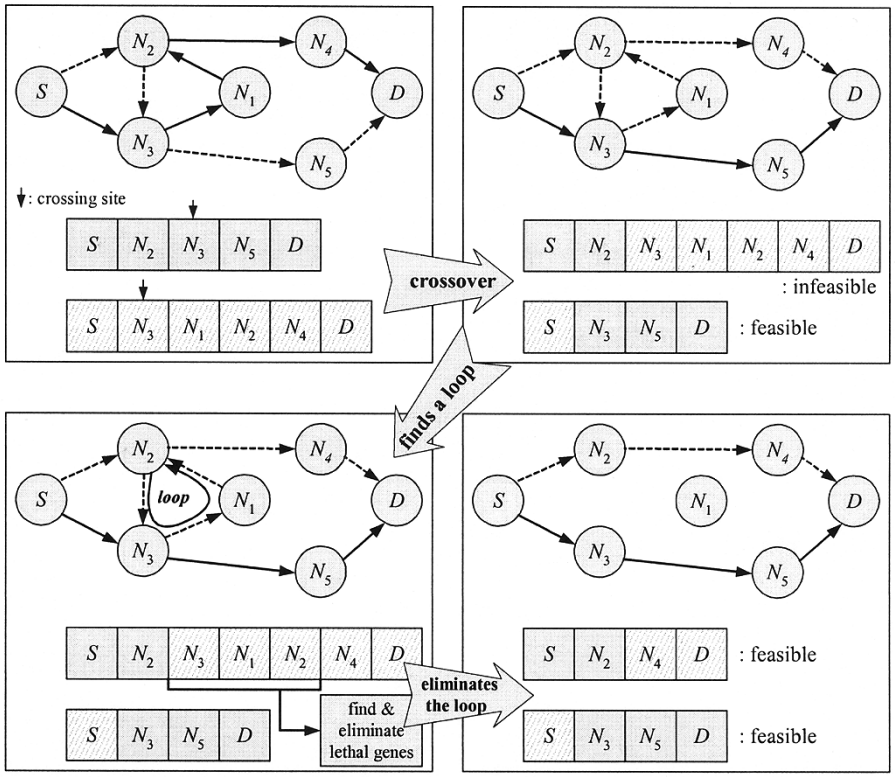
A. Decision Model of Harik

The following results follow from Harik's model of selection [22]. Consider a competition between a chromosome i_1 that contains the optimal BB, H_1 (with mean fitness $\overline{f_{H_1}}$ and fitness variance $\sigma^2_{H_1}$), and a chromosome i_2 with the second best BB, H_2 (with mean fitness $\overline{f_{H_2}}$ and fitness variance $\sigma^2_{H_2}$). The probability of deciding correctly between these two chromosomes is the same as the probability that the fitness of $i_1(f_1)$ is greater than the fitness of i_2 (f_2): the probability that $(f_1 - f_2) > 0$.

The distance between the mean fitness of chromosome with H_1 ($\overline{f_{H_1}}$) and the mean fitness of chromosome with H_2 ($\overline{f_{H_2}}$) is denoted by d (signal). Assuming that the fitness is an additive function of the fitness contributions of all the BBs, f_1 and f_2 are normally distributed (by the central limit theorem). Since

```
/* C : Input chromosome, C* : Output chromosome, l : Length of chromosome C */
for (all i , j & i<(l – j)) {
    if ( C[i] == C[l – j]) { /* Find a loop */
        C* = C[1 : i]// C[l – j +1 : l] } /* Eliminate the loop */
}
```

(a)



(b)

Fig. 4. Overall procedure of the repair function. (a) Pseudocode of the repair function. (b) Example of the repair function.

the fitness distribution of f_1 and f_2 are normal, the distribution of $(f_1 - f_2)$ is also normal.

The distribution of $(f_1 - f_2)$ is [21], [22]

$$(f_1 - f_2) \sim N\left(\overline{f_{H_1}} - \overline{f_{H_2}},\, \sigma_{H_1}^2 + \sigma_{H_2}^2\right). \tag{4}$$

Substituting d for $(\overline{f_{H_1}} - \overline{f_{H_2}})$ in the above equation, and normalizing, the probability p of making the correct decision on a single trial for the domains where BBs m (that are not competing directly) are independent and equally scaled (i.i.d) is given by [22]

$$p = \Phi\left(\frac{d}{\sqrt{\sigma_{H_1}^2 + \sigma_{H_2}^2}}\right) = \Phi\left(\frac{d}{\sqrt{2m'}\,\sigma_{\mathrm{bb}}}\right) \tag{5}$$

where Φ is the cumulative distribution of a normal distribution with zero mean and unit variance, σ_{bb} is the average rms BB standard deviation. Also, $m' = m - 1$ is the total number of collateral noise sources that are not competing directly, so the total collateral noise coming from m' is $m'\sigma_{\mathrm{bb}}^2$.

B. Generalizing the Decision Model

In general, standard deviation can be thought of as the probabilistic "width" or "spread" of distribution of a random variable. Hence, σ_{bb} (i.e., the standard deviation of BBs) indicates the "statistical length" or "spread" of fitness values of BBs from their average fitness value; indeed, the factor $2\sigma_{\mathrm{bb}}$ represents the total average range of fitness changes of all the BBs.

Let $\aleph$ be the average number of competitive BBs. Since the signal d is defined as the fitness difference between the best and second best BBs, from a statistical point of view, the best BB has the fitness value of adding average and standard deviation of BBs' fitness, and the second best BB has the value of negating a $\{2\sigma_{\mathrm{bb}}/(\aleph - 1)\}$ portion from the best fitness. Because it may be assumed that all the competing BBs can be ordered and they are distributed uniformly from the best to the worst fitness values, the interval value between inter-rank BBs is $\{2\sigma_{\mathrm{bb}}/(\aleph - 1)\}$. The first assumption is valid when ordinal-based selection (e.g., the pairwise tournament selection without replacement) is used in algorithms, and the second assumption is also valid from the

statistical considerations. Therefore, the signal d can be represented as

$$\begin{aligned} d &= \left(\left\{\overline{f_{\rm bb}} + \sigma_{\rm bb}\right\} - \left\{\overline{f_{\rm bb}} + \sigma_{\rm bb} - \frac{2\sigma_{\rm bb}}{\aleph - 1}\right\}\right) \\ &= \left(\left\{\overline{f_{\rm bb}} + \sigma_{\rm bb}\right\} - \left\{\overline{f_{\rm bb}} + \left(1 - \frac{2}{\chi^k - 1}\right)\sigma_{\rm bb}\right\}\right) \\ &= \frac{2}{\chi^k - 1}\,\sigma_{\rm bb} \end{aligned} \tag{6}$$

where $\overline{f_{\rm bb}}$ is the mean fitness of BBs, χ is the average cardinality of the alphabet, and k is the average order of BBs. Therefore, (5) can be rewritten as

$$p = \Phi\left(\frac{2}{\sqrt{2m'}\,(\chi^k - 1)}\right). \tag{7}$$

In the SP routing problem, only statistical point of view is meaningful since all the domain-dependent variables are time varying due to the changes in network topology consequent on factors such as link failure, congestion, and mobility (power on or off in wired network).

As a special case, assume that the cardinality of the alphabet is 2 (i.e., $\chi = 2$) in a uniformly scaled linear problem [i.e., one-max problem ($k = 1$)]. Similar problems were considered by Harik. He finds the signal d to be 1.0 and the BB variance $\sigma_{\rm bb}^2$ to be 0.25, so that the probability p of making the correct decision on a single trial is given by [22]

$$p = \Phi\left(\frac{d}{\sqrt{2m'}\,\sigma_{\rm bb}}\right) = \Phi\left(\frac{2}{\sqrt{2m'}}\right). \tag{8a}$$

On the other hand, (7) for the same problem can be rewritten as

$$p = \Phi\left(\frac{2}{\sqrt{2m'}\,(\chi^k - 1)}\right) = \Phi\left(\frac{2}{\sqrt{2m'}}\right). \tag{8b}$$

This is, of course, the same as (8a). It is surprising that the probability of making the correct decision can be obtained by only knowing the average number of BBs of length $m = m' + 1$. This is the same length of chromosomes in the one-max problem. No knowledge of signal and noise is required.

C. Generalizing the Population-Sizing Equation

The GA succeeds when all the n members of the population in the BBs of interest are correct. From a well-known result in the gambler's ruin (one-dimensional random walk) literature, it follows that the probability $P_{\rm bb}(x_0)$ that the GA eventually succeeds when there are x_0 initial correct BBs is [22]

$$P_{\rm bb}(x_0) = \frac{1 - \left(\frac{q}{p}\right)^{x_0}}{1 - \left(\frac{q}{p}\right)^{n}} \tag{9}$$

where $q = 1 - p$ is the probability of losing a copy of the BB in a particular competition.

Since (9) is a conditional probability given that the GA starts with x_0 correct BBs, the probability that the GA succeeds can be formed by

$$\begin{aligned} P_{\rm bb} &= \sum_{i=0}^{n} P_{\rm bb}(i) \cdot P(i) \\ &= \sum_{i=0}^{n} \left[\frac{1 - \left(\frac{q}{p}\right)^{i}}{1 - \left(\frac{q}{p}\right)^{n}}\right] \cdot \binom{n}{i} \left(\frac{1}{\chi^k}\right)^{i} \left(1 - \frac{1}{\chi^k}\right)^{n-i} \end{aligned} \tag{10}$$

where $P(i)$ represents the probability that the GA starts with i correct BBs. Using binomial expansion, it can be arranged as follows:

$$P_{\rm bb} = \frac{1 - \left(1 - \frac{1 - \left(\frac{q}{p}\right)}{\chi^k}\right)^{n}}{1 - \left(\frac{q}{p}\right)^{n}} = \frac{1 - \left(1 - \frac{2p-1}{\chi^k \cdot p}\right)^{n}}{1 - \left(\frac{1-p}{p}\right)^{n}}. \tag{11}$$

The remaining derivation refers to [22]. The $P_{\rm bb}$ may be approximated as

$$P_{\rm bb} \approx 1 - \left(1 - \frac{2p-1}{\chi^k \cdot p}\right)^{n}. \tag{12}$$

Hence

$$n = \frac{\ln(\alpha)}{\ln\left(1 - \frac{2p-1}{\chi^k \cdot p}\right)} \tag{13}$$

where $\alpha = 1 - P_{\rm bb}$ is the probability of GA failure. Physically, the probability of GA failure α represents the fact that the GA converges to one of the local optimal solutions.

Since $(2p-1)/(\chi^k \cdot p)$ tends to be a small number, $\ln(1 - (2p-1)/(\chi^k \cdot p))$ can be approximated as $-((2p-1)/(\chi^k \cdot p))$. Thus, (13) can be rewritten as follows:

$$n = -\chi^k \cdot \ln(\alpha) \cdot \frac{p}{2p-1}. \tag{14}$$

An approximate value of p, using the first two terms of the power series expansion of the normal distribution is given by [22]

$$p = \frac{1}{2} + \frac{1}{\sqrt{2\pi}}\,z. \tag{15}$$

From (7), z is found to be $2/(\sqrt{2m'}\,(\chi^k - 1))$. As a result, a fairly general population-sizing equation can be written as follows:

$$\begin{aligned} n &= -\frac{\chi^k}{2} \cdot \ln(\alpha) \cdot \left(z^{-1}\sqrt{\frac{\pi}{2}} + 1\right) \\ &= -\frac{\chi^k}{2} \cdot \ln(\alpha) \cdot \left(\frac{\chi^k - 1}{2} \cdot \sqrt{\pi m'} + 1\right). \end{aligned} \tag{16}$$

Since the length of BBs is almost one in our problem (i.e., one-max problem) as explained later (Section IV-C), the crossover can hardly disrupt the BBs. When the number of genes in the BBs (average order) becomes large, the probability of disrupting the BBs is increased; thus, the population size may be increased to reach a particular quality of solution. This is the reason why a higher probability of disrupting the BBs drives the probability of making the correct decision on a single

trial p toward smaller values so that the population size n must be increased for achieving the same GA failure probability α. This can be inferred from (14).

However, we can observe an interesting consequence from the experiments of Harik [22]: the population size necessary for obtaining a desired quality of solution is not strongly affected by the average order, even if it is considerably large ($k \geq 4$). Thus, it is as if the population size is not strongly affected by the (one-point) crossover.

Although the mutation operation may disrupt the BBs and retard convergence of BBs, it eventually ensures a better quality of solution by introducing new chromosomes (maintaining the diversity of the population) that help the GA avoid local convergence. Thus, the population will not be increased by the mutation. In other words, the ultimate population size (for a solution of desired quality) may not be increased by these operations because the minor harmful effects of the crossover are offset by the beneficial effects of mutation.

Equation (16) is applicable only to a crossover-intensive GA with little or no mutation. There are some approaches in evolutionary computation, and some problems (notably neural networks), that employ mutation as the dominant operator. In these approaches, (16) is not really useful for determining a population size that guarantees a specified quality of solution. The evolutionary checkers player coevolved with a (fully connected feed forward) neural network with an input layer, two hidden layers, and an output node [28] provides a good example in this regard. It needs a tiny population of only 30 to evolve thousands of weights of the neural network. However, the proposed SP algorithm employs ordinal-based selection (i.e., pairwise tournament selection without replacement) and works with crossover as the dominant operator. Therefore, the population-sizing equation is used in the proposed GA for the SP routing problem in the presence of changes in network topology.

It must be noted that (16) does not require any knowledge of signal and noise which may not be available in advance in most practical problems. Instead, the equation approximates such stochastic information for all the selection mechanisms. Of course, the approximation may induce some discrepancy that depends on the selection mechanism. It can, however, be concluded [21] that the equation provides an upper bound (i.e., overestimation of population size to obtain a target quality) for ordinal-based selection. Furthermore, the equation may be quite accurate for the (linear) ranking selection because the selection scheme was implicitly assumed in [21] as explained in an earlier section.

D. Deciding the Parameters in the SP Routing Problem

The problem of determining the parameters in (16) remains. As χ is the average cardinality of the alphabet, it can be found by the average link connectivity in the network. In other words, the parameter χ physically represents the average number of nodes that each gene can take. As mentioned before, the parameter $\alpha(=1 - P_{\rm bb})$ is the GA failure (i.e., route failure) probability; GA failure implies that the computed route is not optimal. The average order k of BBs can be modeled as a linear combination of one-max function and deceptive function in the SP routing problem. That is

$$k = \sum_{x=1}^{N} (c_x \cdot x) \tag{17}$$

where N is the number of nodes, and c_x, the weighted average coefficient (WAC) is a domain-dependent parameter. The sum of all the WACs is 1.

In (17), we get the one-max problem when x is equal to 1, and the deceptive problem when x is greater than 1. In addition, the total number of collateral noise sources m' is $(m-1)$, where m is the average number of BBs. In the routing problem, m is calculated as L/k, where L is the average length of chromosomes that is defined by the number of nodes whose average cost is not greater than that of the overall network. Of course, the BBs may be inherently interdependent. However, the average number of BBs (L/k) will be a reasonable approximation from a statistical point of view, if an accurate average order k is used. This is explained below.

The possibility of finding a shortest path will be quite high when each node chooses a lowest/best-cost node among its own neighbors as a forward node in a route (as happens in a greedy algorithm). If the shortest path is always found in this manner, the problem can be modeled as the one-max problem. But, a globally optimal path is possible even when locally nonoptimal selections are made. It is well known that locally optimal selections may be misleading. These reflect interdependence among BBs. The idea is to make an attempt to spread this potential (to mislead) over the average length of chromosomes, and thereby weaken it. The chromosomes can then be modeled as independent BBs. This is discussed Section IV-C.

IV. Experiments and Discussion

In this section, the proposed GA is compared with Munetomo's [5] and Inagaki's [6] algorithms through computer simulations. All the simulations were performed with MATLAB 5.3 on Pentium III processor (850-MHz clock). As described in Section II-D, the proposed GA employs pairwise tournament selection (i.e., tournament size $s = 2$) without replacement. In all the experiments, the mutation probability is set to 0.05 (a typical mutation option), and each experiment is terminated when all the chromosomes have converged to the same solution. A convergence test for termination is performed before applying mutation as otherwise uninvited evolution may follow due to the placement of the mutation operator in the loop. However, this strategy does not affect the results.

Each solution is compared with Dijkstra's SP [1] solution. In other words, Dijkstra's algorithm provides a reference point. Furthermore, the accuracy and the scalability of the population-sizing equation are also verified through simulation studies.

A. Simulation Results for a Fixed Network With 20 Nodes

The simulation studies involve the deterministic, weighted network topology (with 20 nodes) depicted in Fig. 5(d). The

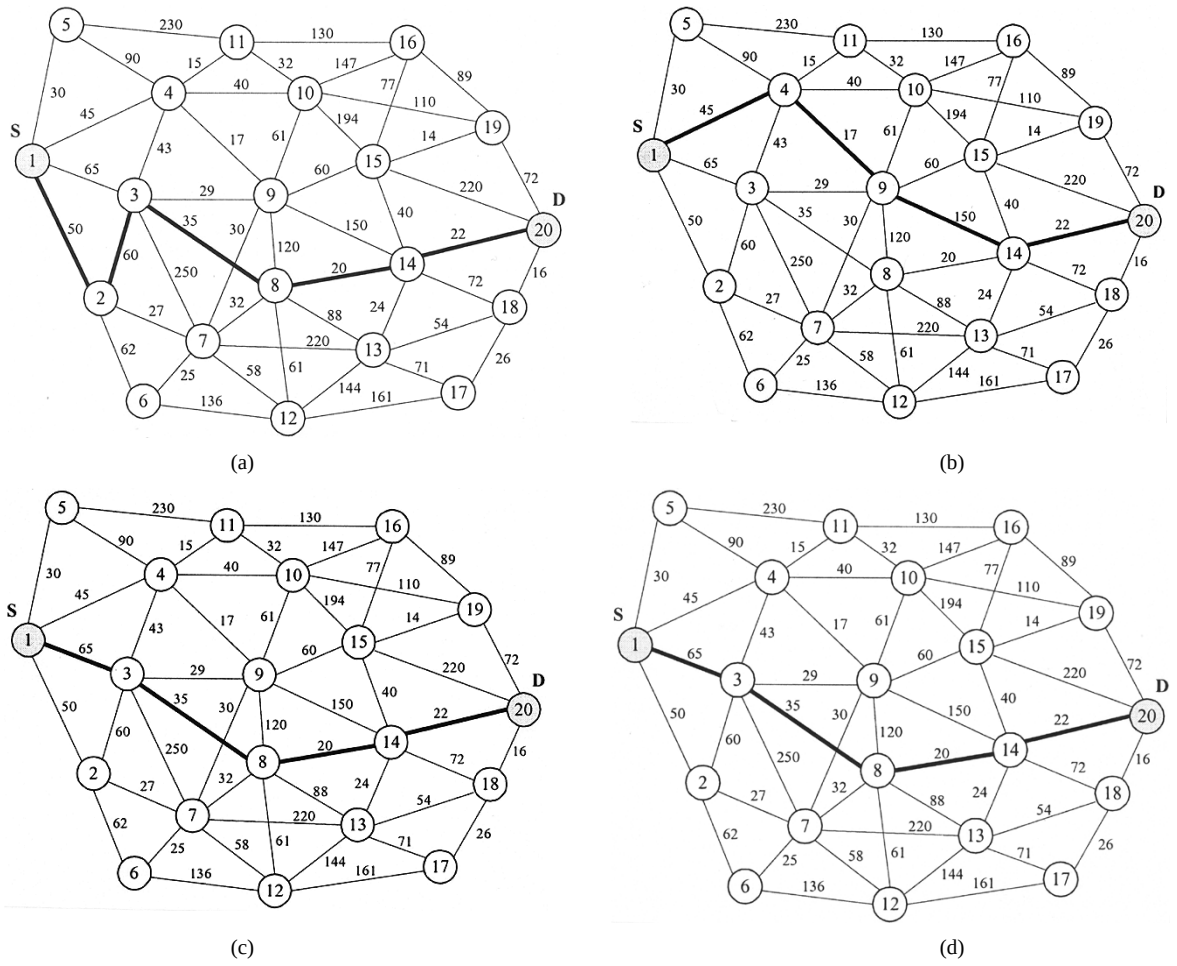


Fig. 5. Comparison results for the paths found by each algorithm. (a) Result of the Munemoto's algorithm (total path costs: 187). (b) Result of the Inagaki's algorithm (total path costs: 234). (c) Result of the proposed algorithm (total path costs: 142). (d) Example network with optimal path in bold line (optimal path costs: 142).

bold line shows an optimal path. With a view to focus exclusively on fair comparison of algorithms on the basis of performance, the population size is taken to be the same as the number of nodes in the network. Population size and its influence on quality of solution are investigated later (Section IV-C).

Fig. 5(a)–(d) show the shortest paths (bold lines) found by the algorithms for the indicated source–destination pair. It is seen that the path computed by the proposed algorithm coincides with that found by Dijkstra's algorithm. The latter is known to always return the shortest path. Munetomo's [5] and Inagaki's [6] algorithms, on the other hand, settle for a suboptimal path.

Fig. 6 compares objective-function values returned by the algorithms. It is seen that the proposed GA exhibits the fastest rate of convergence because the number of generations up to convergence is the smallest. The algorithm converging through smaller generations has better convergence performance because all the algorithms have the same population size in the experiment. In general, however, convergence performance must be compared with the average number of fitness function evaluations until the GAs reach equal quality of solutions [29]. A detailed explanation will be given later (Section IV-B). In addition, it indeed converges to a value that is exactly the same as the Dijkstra-value (the optimal route), notwithstanding the somewhat inherent initial disadvantage.

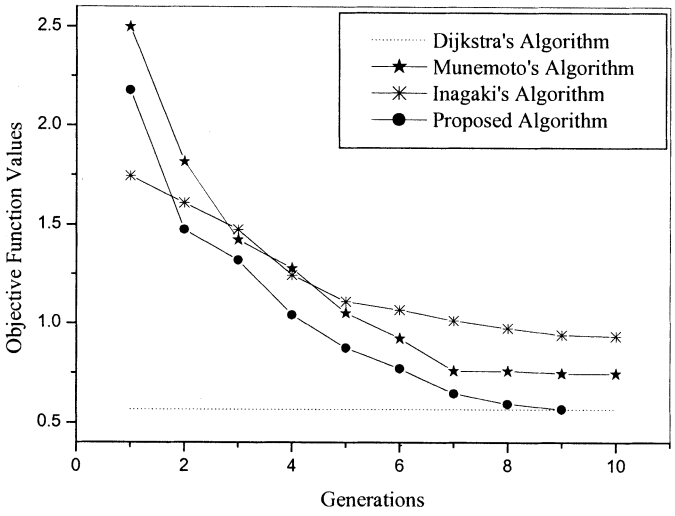


Fig. 6. Convergence property of each algorithm.

B. Simulation Results for Random Network Topologies

In this section, we verify that the previous results (the quality of solution and the convergence speed) hold for all kinds of problems (networks types and scales). Networks with 15–50

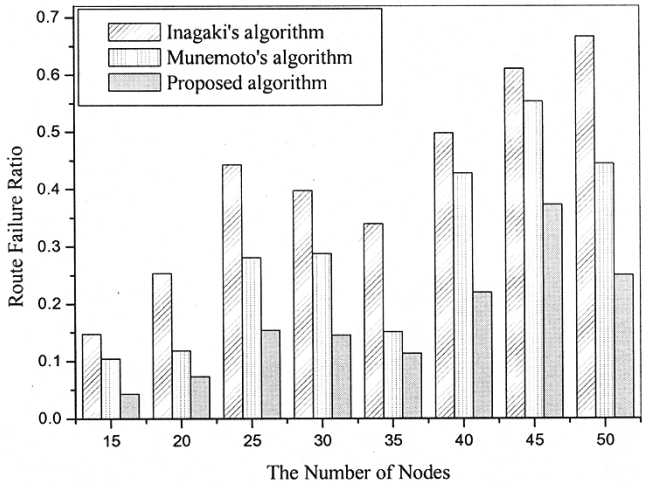


Fig. 7. Comparison results of the quality of solution for each algorithm.

nodes, and randomly assigned (normalized) link costs were investigated. As described earlier (Section I), mobile *ad hoc* networks provide acceptable targets. Applications include military battlefield (e.g., moving platoon or company), rescue missions, conference room and so on [25]–[27]. They involve networks with sizes that range from small to medium (e.g., tens of nodes). Simulations reflect this practical reality. A possible implication is that the proposed algorithm scales well to larger networks.

First, the quality of solution (route optimality) for each GA is investigated. The route optimality is defined as the percentage of time that the GA finds the global optimum (i.e., the shortest path). The route failure ratio is the inverse of route optimality. It is asymptotically the probability that the computed route is not optimal, because it is the relative frequency of route failure. The population size of each GA is also taken to be the same as the number of nodes in the networks. A total of 1000 random network topologies were considered in each case.

The quality of solutions of the algorithms is compared in Fig. 7. From the figure, we can see that the quality of the solution of the proposed GA is much higher than that of the other algorithms. In case of 30 nodes, for example, the proposed GA outperforms Inagaki's GA and Munemoto's GA with $prob. < 0.26$ and $prob. < 0.15$, respectively. The results are collected in Table I. The proposed GA attains a 0.1712 route failure ratio (82.88% route optimality) with a population size equal to the number of nodes in the networks. The proposed GA is better than Inagaki's GA and Munemoto's GA with $prob. < 0.25$ and $prob. < 0.13$, respectively. Meanwhile, STD of route failure ratio (probability) for the proposed GA amounts to 0.1067 comparing favorably with 0.1745 for Inagaki's GA and 0.167 for Munemoto's GA. It means that the proposed algorithm retains its robustness amidst changing network topologies with regard to the quality of solution (route optimality). The benefits accrue from the compound effects of effective search capability of the crossover and the diversity maintenance by mutation.

Second, the convergence speed of every GA is investigated. Convergence performance is investigated in terms of the average number of fitness function evaluations. Cantŭ-Paz [29] suggests the total execution time (i.e., the number of fitness function evaluations) required to find a solution of the same average quality as a fair comparison criterion. In other words, the number of fitness function evaluations directly measures the rank (excellence) of convergence performance only if all the GAs converge to solutions with identical quality. Unfortunately, finding the exact population size for a particular quality of solution for each algorithm is very difficult. However, determining the population size with certain constraints is relatively easy. We can determine the population size for each algorithm by exhaustive search for each algorithm so as to achieve almost the same quality of solution (route optimality). The proposed algorithm seems to have the most satisfactory performance and Inagakis' the least. The reason is quite apparent: the proposed algorithm involves the smallest number of fitness function evaluations. That means faster convergence. Networks with 15–50 nodes and randomly assigned link costs were also studied.

The results in respect of number of fitness function evaluations are shown in Fig. 8. From the figure, we can see that the convergence rate of the proposed algorithm is much higher than that of the other algorithms in every case, because the average number of fitness function evaluations needed to reach similar quality of solution (i.e., maximum difference is about 4%) is smaller than any other algorithm. In the case of 30 nodes, for instance, the proposed GA is faster than Inagaki's GA and Munemoto's GA with $prob. < 0.87$ and $prob. < 0.48$ (i.e., 7.27 times and 1.92 times), respectively. The results are collected in Table II.

From Table II, it can be seen that the proposed GA converges to a solution with about 0.2 route failure ratio (80% route optimality) in about 122 fitness function evaluations. The convergence speed of the proposed GA is superior to that of Inagaki's GA and Munemoto's GA with $prob. < 0.85$ and $prob. < 0.52$ (i.e., 6.63 and 2.06 times), respectively. It is noted that such improvement serves as a lower bound of convergence gain because the proposed GA still attains better quality of solution than other algorithms. Furthermore, the STD of fitness function evaluations for the proposed GA is about 32, while it is about 182 for Inagaki's GA and about 59 for Munemoto's GA. It also implies that the proposed GA is indeed insensitive to network topologies, as far as convergence is concerned.

In order to further compare the convergence performance of the proposed GA with that of Dijkstra's algorithm, direct (real) computation time obtained from previous experiments (Fig. 7) is presented in Fig. 9. The average computation time of Dijkstra is 0.14 s and that of the proposed GA is 0.067 s. The proposed GA is faster than Dijkstra's algorithm with $prob. < 0.522$, although it may confront the route failure situation with $prob. < 0.1712$. The computation time of the proposed GA does not increase significantly with the network size while it does in case of Dijkstra's algorithm. It is noted that both the algorithms are, per se, inadequate for real-time communications in mobile ad-hoc networks. However, the proposed GA, with its extremely fast hardware counterpart, passes the test (Section I).

C. Experimental Verification of the Population-Sizing Equation

In this section, we verify that the generalized and enhanced version of the population-sizing equation employing the gam-

TABLE I
PERFORMANCE COMPARISON OF THE QUALITY OF SOLUTION

Performance Measure \ Algorithms		Inagaki's algorithm	Munemoto's algorithm	Proposed algorithm
Route Failure Ratio	Average	0.4195	0.2959	0.1712
	STD	0.1745	0.1670	0.1067

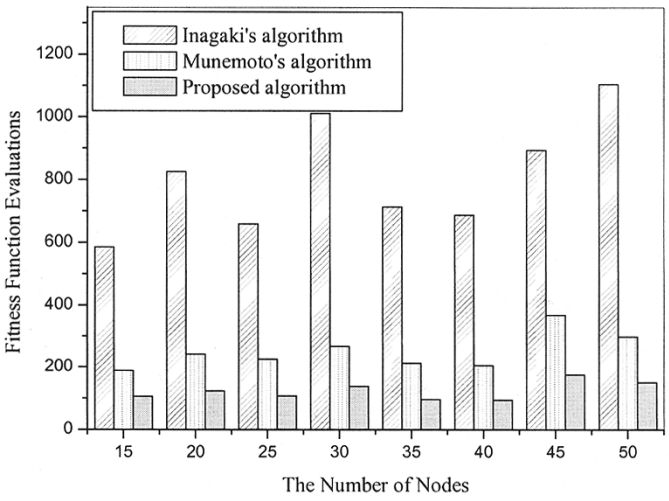


Fig. 8. Comparison results of the rate of convergence for each algorithm.

bler's ruin problem fairly accurately predicts the quality of solutions computed by the proposed GA.

For an accurate assessment, all the results must be averaged over 10 000 random network topologies, because the best quality of solution in the experiment guarantees a 10^{-2} route failure ratio (99% route optimality). In other words, we must experiment until the number of suboptimal solutions is at least 100; thus, 10 000 network topologies are needed for supporting up to 10^{-2} route failure ratio. So, the experiments are run ten times with a total of 1000 random network topologies with 15–50 nodes, and randomly assigned link costs (normalized).

In order to apply the population-sizing equation [(16)], the domain-dependent variables must be properly assigned. The average order k is a unique domain-dependent variable not yet specified. Assuming that it is likely that an average order of more than two is very rare, the parameter k can be approximated by a two-term weighted average as follows:

$$k \approx \sum_{x=1}^{2} (c_x \cdot x). \tag{18}$$

The reason for this assumption is explained below. When k is 2, half the nodes in a route choose lower/worse nodes among their own neighbors as forward nodes in order to build an optimal path (i.e., quite misleading). Thus, $k > 2$ can be ignored since this situation ($k = 2$) itself is relatively rare in practice. Determining the coefficients is a very difficult problem. They are also sensitive to network size and topology. We observed that plotting the average deceptive size against the number of nodes on a log–log scale results in an almost straight line, which means that the coefficient of the average order 2 (c_2) can be approximated with a general power-law equation [29] as follows:

$$c_2 = A \cdot N^B. \tag{19}$$

Here, N is the number of nodes in the networks, and A and B are domain-dependent constants. The value of A and B can be computed by transcendental cognition as follows:

$$A = 10^{-2} \cdot (1 - \alpha)^2 \quad \text{and} \quad B = 1.0 \tag{20}$$

where $\alpha = 1 - P_{\text{bb}}$ is also the probability of GA failure (route failure probability).

Therefore, the average order may be calculated as follows:

$$k = 1 \cdot c_1 + 2 \cdot c_2 = 1 + c_2 = 1 + 10^{-2} \cdot (1 - \alpha)^2 \cdot N. \tag{21}$$

From (21), we can see that the average order k is around 1 if the network does not have a large number of nodes. In that case, the probability of disruption of the BBs by crossover is very small. It is noted that if the average order k becomes large, the probability becomes large too and the population size might be affected. As mentioned earlier (Section III-C), however, the increasing average order does not strongly induce any increment in the population size [22].

Experiments concerning the population-sizing equation are summarized in Fig. 10. In Fig. 10, the dotted line is the target route failure probability and the thin line with a symbol is the experimental result with the population size obtained by (16). From the figure, we can see that the equation accurately predicts a population size that is adequate for reaching the desired (target) solution on the order of route failure probability of 0.1 (i.e., better than 90% optimality), and satisfactorily estimates the population size as an upper bound in cases of worse route failure probabilities. Thus, the equation can be used for determining the population size for a desired quality of solution.

On the other hand, the scalability of the population-sizing equation with regard to he number of nodes (network size) is investigated only in two cases, *viz.*, 0.01 (99%) and 0.1 (90%) route failure probabilities (route optimality). The results of the experiments are shown in Fig. 11. It can be seen that the equation is quite scalable at 0.01 route failure ratio. Although it is not quite scalable at 0.1 route failure ratio, the quality of solution is not so much influenced by the number of nodes since the variation of the quality of solutions is about 5%.

As a result, it may be inferred that the population-sizing equation is scalable, a characteristic that is quite important in practical problems.

TABLE II
PERFORMANCE COMPARISON OF THE RATE OF CONVERGENCE

Performance Measure \ Algorithms		Inagaki's algorithm	Munemoto's algorithm	Proposed algorithm
Achieved Route Failure Ratio		0.2438	0.2209	0.2014
Number of Fitness Function Evaluations	Average	810.5440	251.9105	122.3158
	STD	181.9882	58.9393	31.6627

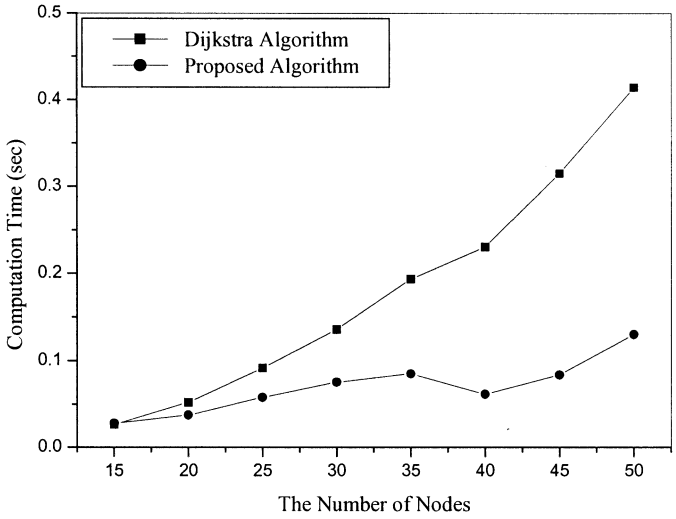


Fig. 9. Convergence performance between the Dijkstra's algorithm and the proposed GA.

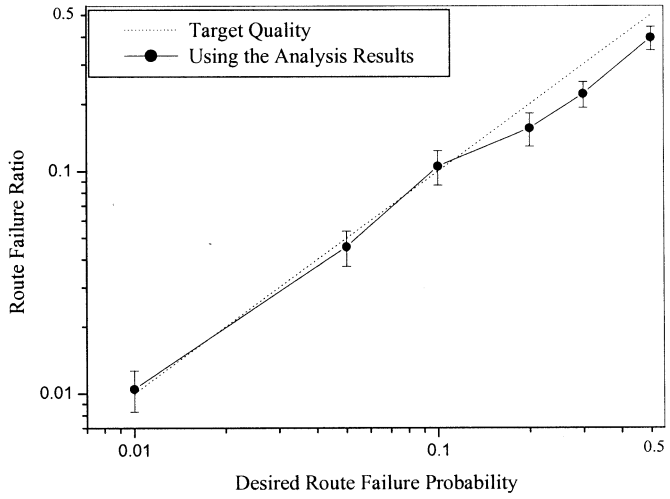


Fig. 10. Desired route failure probability versus route failure ratio.

V. CONCLUDING REMARKS

This paper presented a genetic algorithm for solving the SP routing problem. The crossover and the mutation operations work on variable-length chromosomes. The crossover is simple and independent of the location of crossing site. Consequently, the algorithm can search the solution space in a very effective manner. The mutation introduces, in part, a new alternative route. In essence, it maintains the diversity of population thereby avoiding local traps. A treatment for infeasible solutions

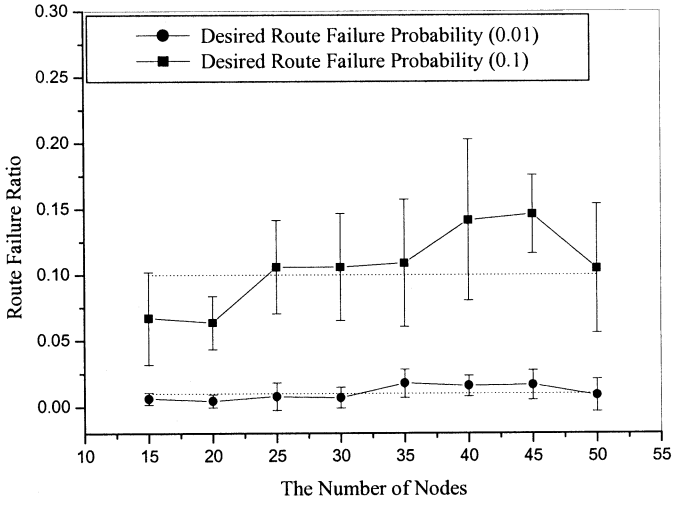


Fig. 11. Number of nodes versus route failure ratio.

(chromosomes) has also been investigated without unduly compromising on computational requirements. Simulation studies show that the algorithm is indeed insensitive to variations in network topologies in respect of both route optimality (quality of solution) and convergence speed. The quality of solution has empirically been found to be better than that of other algorithms; indeed, the 0.1712 route failure ratio (82.88% route optimality) is attained with the population size that is equal to the number of nodes in the networks. The route failure (optimality) performance of the proposed GA was better than those of Inagaki's GA and Munemoto's GA with $prob. < 0.25$ and $prob. < 0.13$, respectively. Furthermore, convergence was seen to occur in about 122 fitness function evaluations with up to about 0.2 quality of solution (80% route optimality). The convergence performance of the proposed GA was better than those of Inagaki's GA and Munemoto's GA with $prob. < 0.85$ and $prob. < 0.52$, respectively. In addition, the real computation time of the proposed GA was shorter than that of Dijkstra's algorithm. However, hardware implementation may be necessary for applications involving real-time services in a dynamic network topology.

An equation that helps in determining an adequate population size for obtaining a desired (quality) solution has also been investigated in this paper. It is based on the gambler's ruin model. It is a fairly generalized form of it, though. It is noted that the equation does not require information such as the signal or variance of competing BBs. It is difficult or impossible to obtain such information in multihop network environments in any case. The accuracy of the equation was verified with experiments by

specifying different solution qualities ranging from 0.01 to 0.5 route failure probability (50∼99% route optimality). The results showed that the predictions of the equation are acceptable in the (practically) operational region [which is better than 0.1 route failure probability (90% route optimality)]. It can be used as an upper bound elsewhere. Furthermore, the equation scales with problem difficulty (network size).

The proposed algorithm can search the solution space effectively and speedily compared with other extant algorithms. The population-sizing equation appears to be a conservative tool to determine a population size in the routing problem.

ACKNOWLEDGMENT

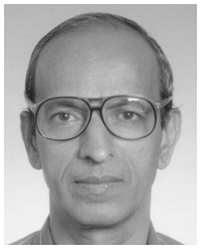
The authors would like to thank D. B. Fogel and the reviewers for thoughtful, constructive comments, and suggestions.

REFERENCES

[1] W. Stalling, *High-Speed Networks: TCP/IP and ATM Design Principles*. Englewood Cliffs, NJ: Prentice-Hall, 1998.
[2] M. K. Ali and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," *IEEE Trans. Neural Networks*, vol. 4, pp. 941–954, Nov. 1993.
[3] D. C. Park and S. E. Choi, "A neural network based multi-destination routing algorithm for communication network," in *Proc. Joint Conf. Neural Networks*, 1998, pp. 1673–1678.
[4] C. W. Ahn, R. S. Ramakrishna, C. G. Kang, and I. C. Choi, "Shortest path routing algorithm using hopfield neural network," *Electron. Lett.*, vol. 37, no. 19, pp. 1176–1178, Sept. 2001.
[5] M. Munemoto, Y. Takai, and Y. Sato, "A migration scheme for the genetic adaptive routing algorithm," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, 1998, pp. 2774–2779.
[6] J. Inagaki, M. Haseyama, and H. Kitajima, "A genetic algorithm for determining multiple routes and its applications," in *Proc. IEEE Int. Symp. Circuits and Systems*, 1999, pp. 137–140.
[7] Y. Leung, G. Li, and Z. B. Xu, "A genetic algorithm for the multiple destination routing problems," *IEEE Trans. Evol. Comput.*, vol. 2, pp. 150–161, Nov. 1998.
[8] Z. Xiawei, C. Changjia, and Z. Gang, "A genetic algorithm for multicasting routing problem," in *Proc. Int. Conf. Communication Technology (WCC-ICCT 2000)*, 2000, pp. 1248–1253.
[9] Q. Zhang and Y. W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 53–62, Apr. 1999.
[10] H. Pan and I. Y. Wang, "The bandwidth allocation of ATM through genetic algorithm," in *Proc. IEEE GLOBECOM'91*, 1991, pp. 125–129.
[11] M. E. Mostafa and S. M. A. Eid, "A genetic algorithm for joint optimization of capacity and flow assignment in packet switched networks," in *Proc. 17th National Radio Science Conf.*, 2000, pp. C5-1–C5-6.
[12] N. Shimamoto, A. Hiramatsu, and K. Yamasaki, "A dynamic routing control based on a genetic algorithm," in *Proc. IEEE Int. Conf. Neural Networks*, 1993, pp. 1123–1128.
[13] G. Tufte and P. C. Haddow, "Prototyping a GA pipeline for complete hardware evolution," in *Proc. 1st NASA/DoD Workshop on Evolvable Hardware*, 1999, pp. 76–84.
[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.
[15] X. Hue, "Genetic algorithms for optimization: Background and applications," Edinburgh Parallel Computing Centre, Univ. Edinburgh, Edinburgh, Scotland, Ver 1.0, Feb. 1997.
[16] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1989, pp. 2–9.
[17] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.
[18] W. G. Macready and D. H. Wolpert, "Bandit problems and the exploration/exploitation tradeoff," *IEEE Trans. Evol. Comput.*, vol. 2, pp. 2–22, Apr. 1998.
[19] K. A. DeJong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. Michigan, Ann Arbor, MI, 1975.
[20] D. E. Goldberg and M. Rundnick, "Genetic algorithms and the variance of fitness," *Complex Syst.*, vol. 5, no. 3, pp. 265–278, 1991.
[21] D. E. Goldberg, K. Deb, and J. H. Clark, "Genetic algorithms, noise, and the sizing of populations," *Complex Syst.*, vol. 6, pp. 333–362, 1992.
[22] G. Harik, E. Cantŭ-Paz, D. E. Goldberg, and B. L. Miller, "The Gambler's ruin problem, genetic algorithms, and the sizing of populations," *Evol. Comput.*, vol. 7, no. 3, pp. 231–253, 1999.
[23] C. Hedrick, "Routing information protocol," *RFC 1058*, June 1988.
[24] J. Moy, "Open shortest path first protocol," *RFC 1583*, Mar. 1994.
[25] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *Comput. Commun. Rev.*, pp. 234–244, Oct. 1994.
[26] S. Murthy and J. J. Garcia-Luna-Aceves, "An efficient routing protocol for wireless networks," *ACM Mobile Networks Applic. J.*, pp. 183–197, Oct. 1996.
[27] C. W. Ahn, R. S. Ramakrishna, and C. G. Kang, "Efficient clustering-based routing protocol in mobile *ad-hoc* networks," in *Proc. IEEE Vehicular Technology Conf. (VTC'02)*, Sept. 2002, pp. 1647–1651.
[28] K. Chellapilla and D. B. Fogel, "Evolving an expert checkers playing program without using human expertise," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 422–428, Aug. 2001.
[29] E. Cantŭ-Paz, *Efficient and Accurate Parallel Genetic Algorithms*. Norwell, MA: Kluwer, 2000.

Chang Wook Ahn (S'02) was born in Muan, Chonnam, Korea, in 1977. He received the B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, in 1998 and 2000, respectively. He is currently working toward the Ph.D. degree at the Department of Information and Communications, Kwang-Ju Institute of Science and Technology, Kwang-ju, Korea.

His research interests include genetic algorithms, multiobjective optimization, neural networks, and the applications of evolutionary techniques to wireless networks.



R. S. Ramakrishna (M'93–SM'98) received the Ph.D. degree from the Indian Institute of Technology, Kanpur, India in 1979.

From 1980 to 1996, he was an Associate Professor and Professor in the Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, India. He is currently a Professor in the Department of Information and Communications, Kwang-Ju Institute of Science and Technology (K-JIST), Kwang-ju, Korea.

His research interests include evolutionary algorithms, parallel and distributed computing, computer graphics, and quantum computing.