ELSEVIER

Advances in Engineering Software 32 (2001) 731–744

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

Network design techniques using adapted genetic algorithms

Mitsuo Gen[a,*], Runwei Cheng[a], Shumuel S. Oren[b]

[a]*Department of Industrial and Information Systems Engineering, Ashikaga Institute of Technology, Ashikaga 326-8558, Japan*
[b]*Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA 94720, USA*

Received 29 March 2000; revised 8 November 2000; accepted 22 November 2000

Abstract

In recent years we have evidenced an extensive effort in the development of computer communication networks, which have deeply integrated in human being's everyday life. One of important aspects of the network design process is the topological design problem involved in establishing a communication network. However, with the increase of the problem scale, the conventional techniques are facing the challenge to effectively and efficiently solve those complicated network design problems. In this article, we summarized recent research works on network design problems by using genetic algorithms (GAs), including multistage process planning (MPP) problem, fixed charge transportation problem (fc-TP), minimum spanning tree problem, centralized network design, local area network (LAN) design and shortest path problem. All these problems are illustrated from the point of genetic representation encoding skill and the genetic operators with hybrid strategies. Large quantities of numerical experiments show the effectiveness and efficiency of such kind of GA-based approach.

Keywords: Genetic algorithms; Network design; Multistage process planning; Minimum spanning tree; Fixed charge transportation problems; Centralized network design; Local area network design; and Bicriteria shortest path problem

1. Introduction

Genetic algorithms (GAs) are one of the most powerful and broadly applicable stochastic search and optimization techniques based on principles from evolution theory [19,25]. Over the past few years, the GAs community has turned much of its attention toward the optimization of network design problems [12,13]. This paper is intended as a text covering applications of GAs to some difficult-to-solve network design problems inherent in industrial engineering and the computer communication network [16].

2. Adaptation of GAs

GAs were first created as a kind of generic and weak method featuring in a binary encoding and binary genetic operators. This approach requires a modification of an original problem into an appropriate form suitable for the GAs, as shown in Fig. 1. The approach includes a mapping between potential solutions and binary representations, taking care of decoders or repair procedures, etc. For complex problems, such an approach usually fails to provide successful applications.

To overcome such problems, various non-standard implementations of the GAs have been created for particular problems. As shown in Fig. 2, this approach leaves the problem unchanged and adapts the GAs by modifying the chromosome representation of a potential solution and applying appropriate genetic operators. An encoding method can be either direct or indirect. In the direct encoding method, the whole solution for a given problem is used as a chromosome. For a complex problem, however, such a method will make almost all of conventional genetic operators unusable because large amounts of offspring will be infeasible or illegal. In general, it is not a good choice to use the whole original solution of a given problem as the chromosome representation because many real problems are too complex to have a suitable implementation with a whole solution representation. On the contrary, in the indirect encoding method, just a necessary part of a solution is used in a chromosome. A decoder is then used to produce solutions. A decoder is a problem-specific and determining procedure to generate solutions according to chromosomes produced by GAs. With this method, the GAs will focus their search solely on the interesting part of a solution space.

A third approach is to adapt both the GAs and a given problem, as shown in Fig. 3. A common feature of

* Corresponding author. Tel.: +81-284-62-065; fax: +81-284-64-1071.
E-mail addresses: gen@ashitech.ac.jp (M. Gen), runweicheng@hotmail.com (R. Cheng), oren@ieor.berkeley.edu (S.S. Oren).

0965-9978/01/$ - see front matter
PII: S0965-9978(01)00007-2

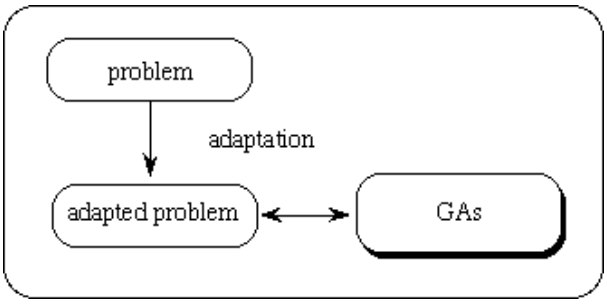

Fig. 1. Adapting a problem to the GAs.

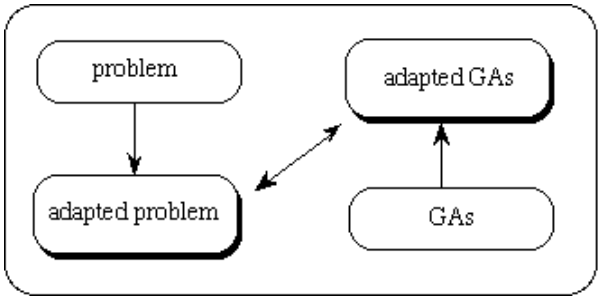

Fig. 3. Adapt both GAs and problems.

combinatorial optimization problems is to find a permutation and/or a combination of some items under some side constraints. If the permutation and/or combination can be determined, a solution then can be derived with a problem-specific procedure. With the approach, the GAs are used to evolve an appropriate permutation and/or combination of some items under consideration, and a heuristic method is subsequently used to construct a solution according to the permutation and combination [4].

This approach has been successfully applied in the area of industrial engineering and has recently become the main approach for practical use of GAs in network designs and optimizations [5].

3. Multistage process planning problems

Multistage process planning (MPP) problem is abundant in manufacturing systems. It provides a detailed description of manufacturing capabilities and requirements for transforming a raw stock of materials into a completed product through multi-stage process. The problem can be classified into two types: variant MPP and generative MPP [3]. Recently the generative MPP problem has received more attention since it can accommodate both general and automated process planning models, which is particularly important in flexible manufacturing systems [31].

The MPP system usually consists of a series of machining operations, such as turning, drilling, grinding, finishing and so on, to transform a part into its final shape or product. The whole process can be divided into several stages. At each stage, there is a set of similar manufacturing operations. The MPP problem is to find the optimal process planning among all possible alternatives given certain criteria such as minimum cost, minimum time, maximum quality or under multiple of these criteria. Fig. 4 shows a simple example of the MPP problem by means of network flows.

For an n-stage MPP problem, let s_k be a state at stage k, let x_k be the decision variable to determine a transition from a given state s_k to a state of next stage $k-1$ and $D_k(s_k)$ be the set of possible decisions under state s_k at stage k. A state corresponds to a node in the graph of flow networks and a decision corresponds to a directed arc in the graph. The MPP problem can be formulated as follows:

$$\min_{x_k \in D_k(s_k), k=1,2,\ldots,n} V(x_1, x_2, \ldots, x_n) = \sum_{k=1}^{n} v_k(s_k, x_k), \tag{1}$$

where $v_k(s_k, x_k)$ represents the criterion to determine x_k under state s_k at stage k, usually defined as a real positive number, such as cost, time or distance. The problem can be modeled as a dynamic recurrence expression and then approached by dynamic programming method [8]. Obviously, with increased problem scales, too many stages and states must be considered, which will greatly affect the efficiency of the dynamic programming method. This is usually known as dimension explosion. Awadh et al. proposed a binary string based-GA to deal with the MPP problem [1]. Zhou and Gen examined the problem with multiple objectives and developed a state permutation encoding method [46].

3.1. Representation

In the state permutation encoding, the position of a gene is used to indicate the stage and the value of the gene is used

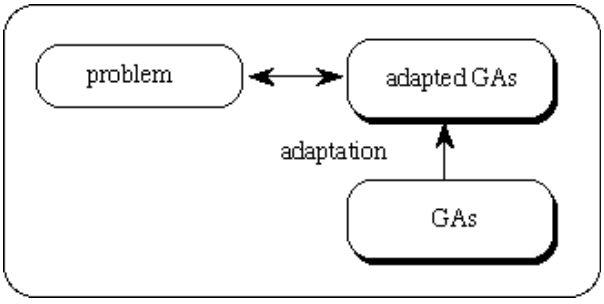

Fig. 2. Adapting the GAs to a problem.

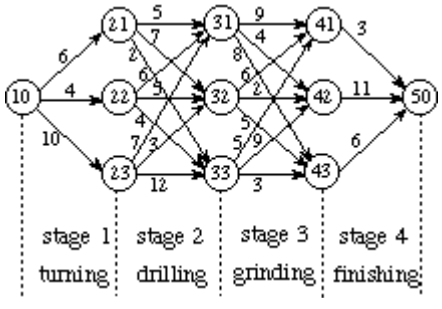

Fig. 4. Flow network for a simple MPP problem.

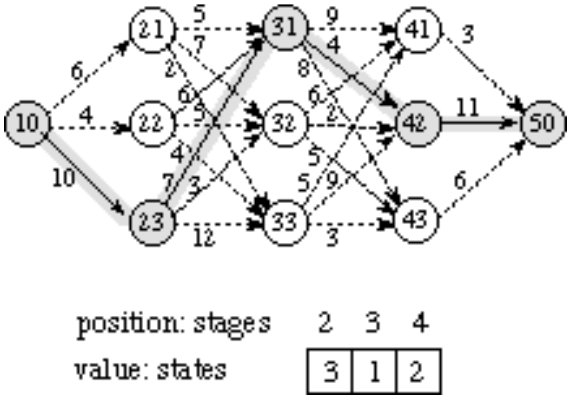

Fig. 5. State permutation encoding.

to indicate a state at that stage. Because the state for the first stage is always fixed, so we do not need to encode this state in a chromosome. It means that for a given problem with n stages, we have the length of an encoding $n - 1$. The encoding can save much more computer memory compared with a binary encoding.

For example, [3 1 2] is an encoding for a feasible solution to the instance given in Fig. 5, where the first gene '3' corresponds to node 23, the second gene '1' corresponds to node 31 and the last gene '2' corresponds to node 42. Therefore, we have a processing plan (s_{10}, s_{23}, s_{31}, s_{42}, s_{50}). This encoding method is able to generate all feasible individuals through genetic operations such as crossover or mutation. Initial population is generated randomly with the number of all possible states in corresponding stage (Fig. 5).

3.2. Genetic operators

Only a neighborhood search-based mutation was used to produce more adapted offspring. It works with the following three major steps:

1. Determine the mutated gene randomly for a given chromosome.
2. Create a set of neighbor by replacing the mutated gene with all its possible states.
3. Select the best one from the neighbors as the offspring.

Fig. 6 shows an example of the neighborhood search-based mutation method.

3.3. Evaluation

When apply GA in multiobjective context, a crucial issue is how to evaluate the credit of chromosomes and then how to represent the credit as fitness values. Zhou and Gen used a weighted-sum approach, which assigns weights to each objective function, combines the weighted objectives into a single objective function, and takes its reciprocal as the

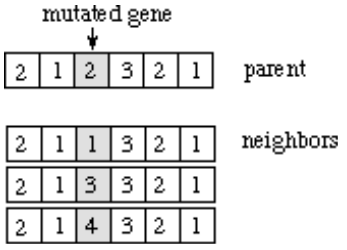

Fig. 6. Illustration of mutation with neighborhood search.

fitness value:

$$\text{eval}(v) = 1/\sum_{k=1}^{p} w_k V_k(x_1, x_2, \ldots, x_n), \tag{2}$$

where v denotes a given chromosome, p the number of objectives, w_k the weight coefficient assigned to objective V_k, which is determined by an *adaptive weights approach* [6]. In the adaptive weight approach, weights will be adaptively adjusted according to the current generation in order to obtain a search pressure towards to the positive ideal point. Because some useful information from the current population is used to readjust the weights at each generation, the approach gives a selection pressure towards to positive ideal point. Let t be index of current generation, $P(t)$ current population, $E(t)$ current Pareto solutions. The overall procedure of the proposed method is given below.

Begin
  $t = 0$
  Initialize $P(t)$
  Objectives $P(t)$
  Pareto $E(t)$
  Fitness $P(t)$
  While (not termination condition) **do**
  Begin
    Mutation $P(t)$
    Objective $P(t)$
    Update Pareto $E(t)$
    Fitness $P(t)$
    Select $P(t + 1)$ from $P(t)$
    $t = t + 1$
  End
End

3.4. Example

To demonstrate the performance of the proposed method, an instance of the MPP problem with 15 stages and 89 nodes was generated. Two attributes were defined on each arc: processing time and cost. The values for processing were randomly generated over [1, 50] and the values of cost were generated over [1, 100], respectively. The objectives were to minimize total processing time and total cost, respectively.

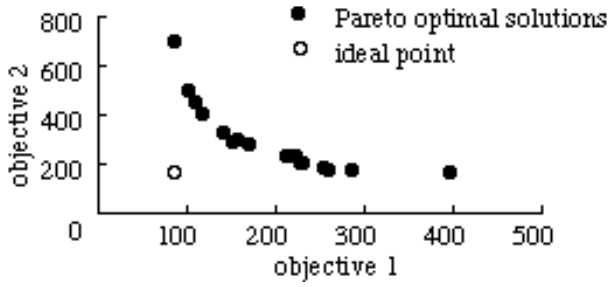

Fig. 7. Pareto solutions of the multiple objective MPP problem.

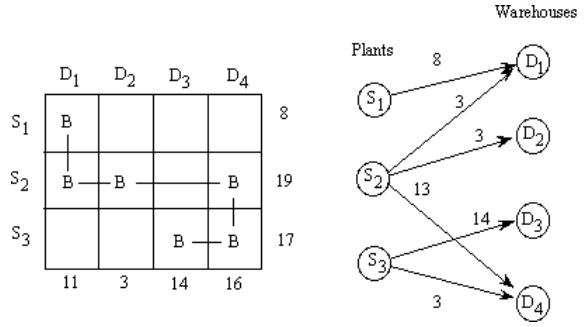

Fig. 8. A transportation alternative with a spanning tree structure.

The results obtained by the proposed GA approach are shown in Fig. 7.

Compared with the enumeration of all Pareto solutions [46], which took 3840 seconds CPU time, 85% results obtained by the proposed method are real Pareto optimal solutions, and, it took only 2.36 s CPU time on average. It is evident that the adaptive weight approach plays an important role in guiding genetic search towards the ideal point and enforces all chromosomes approach to the ideal point as close as possible in the evolutionary process.

4. Fixed charge transportation problem

The fixed charge transportation problem (fc-TP) is an extension of the transportation problem (TP) and many practical transportation and distribution problems can be formulated as the problem [27,32]. For instance, in a TP, a fixed cost may be incurred for each shipment between a given plant and a given consumer and a facility of a plant or warehouse may result in a fixed amount on investment. The fc-TP problem is much more difficult to solve due to the presence of fixed costs, which cause discontinuities in the objective function [2,8,27,46]. Given m plants and n consumers, the problem can be formulated as follows:

$$\text{fc-TP}: \qquad \min f(x) = \sum_{i=1}^{m}\sum_{j=1}^{n}(f_{ij}(x) + d_{ij}g(x_{ij})), \tag{3}$$

$$\text{s. t.} \sum_{j=1}^{n} x_{ij} \le a_i, \qquad i = 1,2,\ldots,m, \tag{4}$$

$$\sum_{i=1}^{m} x_{ij} \ge b_j, \qquad j = 1,2,\ldots,n, \tag{5}$$

$$x_{ij} \ge 0 \qquad \forall i,j, \tag{6}$$

where $x = [x_{ij}]$ is the unknown quantity to be transported from plant i to consumer j, $f_{ij}(x)$ is the objective function of shipping, and

$$g(x_{ij}) = \begin{cases} 1 & \text{if } x_{ij} > 0, \\ 0 & \text{otherwise}, \end{cases} \tag{7}$$

where d_{ij} is the fixed cost. Several methods have been proposed for the fc-TP range from exact solution algorithms to heuristic methods. Recently, Gottlieb and Paulmann [22] proposed a GA based on permutation representation for this problem. Sun et al. [43] proposed a tabu search method. Since the solution of the problem has a network structure characterized as spanning tree, Gen and Li proposed a spanning tree-based GA [17,18,32]. Fig. 8 shows a simple exam ple of transportation alternatives expressed as a spanning tree.

4.1. Representation

One of the classical theorems in graphical enumeration is Cayley's theorem that there are $n^{(n-2)}$ distinct labeled trees on a complete graph with n vertices. Prüfer provided a constructive proof of Cayley's theorem by establishing a one-to-one correspondence between such trees and the set of all strings of $n-2$ digits. This means that we can use only $n-2$ digits permutation to uniquely represent a tree where each digit is an integer between 1 and n inclusive. This permutation is usually known as Prüfer number [39]. The transportation alternatives can be encoded with Prüfer number as illustrated in Fig. 9.

Procedure: convert a tree to a Prüfer number

Step 1. Let i be the lowest-numbered leaf node in tree T. Let j be the node that is the predecessor of i. Then j becomes the rightmost digit of the Prüfer number $P(T)$, $P(T)$ is built up by appending digits to the right; thus $P(T)$ is built and read from left to right.

Step 2. Remove i and the edge (i, j) from further consideration. Thus, i is no longer considered at all. If i is the only successor of j, then j becomes a leaf node.

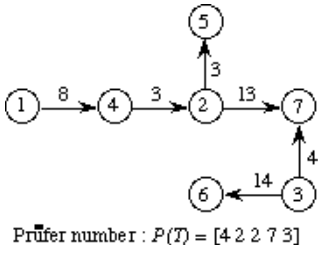

Fig. 9. A transportation alternative and its Prüfer Number encoding.

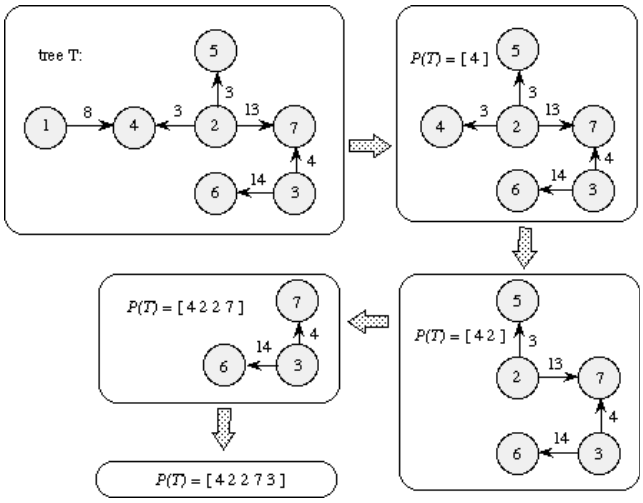

Fig. 10. Illustration of the encoding procedure.

Step 3. If only two nodes remain to be considered, $P(T)$ has been formed, and stop; otherwise, return to step 1.

Fig. 10 illustrated the converting a transportation tree to its corresponding Prüfer number.

Procedure: convert Prüfer number to transportation tree

Step 1. Let $P(T)$ be the original Prüfer number and let $\bar{P}(T)$ be the set of all nodes that are not part of $P(T)$ and designed as eligible for consideration.

Step 2. Repeat the process (2.1)–(2.5) until no digits are left in $P(T)$.

(2.1) Let i be the lowest numbered eligible node in $\bar{P}(T)$. Let j be the leftmost digit of $P(T)$.

(2.2) If i and j are not in the same set S or D, add the edge (i, j) to tree T. Otherwise, select the next digit k from $P(T)$ that is not included in the same set with i, exchange j with k, and add the edge (i, k) to the tree T.

(2.3) Remove j (or k) from $P(T)$ and i from $\bar{P}(T)$. If j (or k) does not occur anywhere in the remaining part of from $P(T)$, put it into $\bar{P}(T)$. Designate i as no longer eligible.

(2.4) Assign the available amount of units to $x_{ij} = \min\{a_i, b_j\}$ (or $x_{ik} = \min\{a_i, b_k\}$) to the edge (i, j) (or (i, k)), where $i \in S$and $j, k \in D$.

(2.5) Update availability $a_i = a_i - x_{ij}$ and $b_j = b_j - x_{ij}$ (or $b_k = b_k - x_{ik}$).

Step 3. If no digits remain $P(T)$ then there are exactly two nodes, i and j, still eligible in $\bar{P}(T)$ for consideration. Add edge (i, j) to tree T and form a tree with $m + n - 1$ edges.

Step 4. If there are no available units to assign, stop. Otherwise, supply r and demand s remain. Add the edge (r, s) to the tree and assign the available amount $x_{rs} = a_r = b_s$ to the edge. If there exists a cycle, remove the edge that is assigned zero flow. A new spanning tree is formed with $m + n - 1$ edges.

The decoding procedure from a Prüfer number to a transportation tree is illustrated in Fig. 11.

Since a transportation tree is a special type of spanning

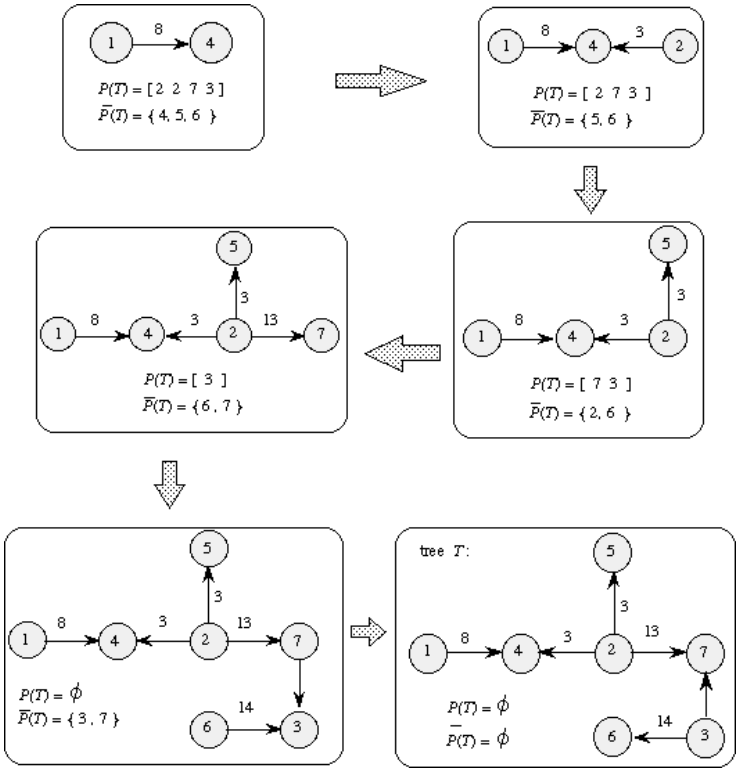

Fig. 11. Illustration of the decoding procedure.

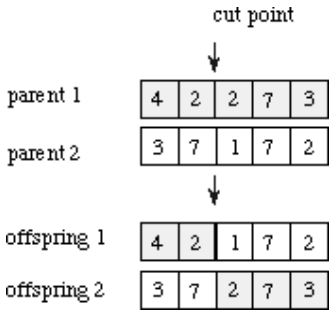

Fig. 12. One-cut point crossover.

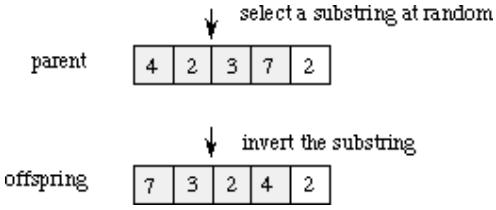

Fig. 14. Illustration of inversion mutation.

tree, the Prüfer number encoding may correspond to an infeasible solution. Gen and Li designed a criterion for checking the feasibility of chromosomes.

4.2. Genetic operators

One-cut point crossover, inversion and displacement mutation were used to explore new solutions. Fig. 12 illustrates the one-cut point crossover operations. The corresponding trees and transportation graphs are given in Fig. 13.

To avoid generating infeasible chromosome, a checking procedure was embedded within the crossover operation. With the checking procedure, offspring always corresponds to a transportation tree.

The *inversion mutation* first selects two positions within a chromosome at random and then inverts the substring between these two positions as illustrated in Fig. 14.

The *displacement mutation* first selects a substring at random and then inserts it at a random position as illustrated in Fig. 15. These two mutation operators always generate feasible chromosomes if the parents are feasible.

4.3. Evaluation and selection

The evaluation procedure consists of two steps: (1) convert a chromosome into a tree and (2) calculate objective function. A mixed way of $(\mu + \lambda) -$ selection and *roulette wheel* selection was used in the selection procedure. This method first selects the μ best chromosomes from μ parents and λ offspring. If there are not μ different chromosomes available, the vacant pool of the population is filled up with roulette wheel selection.

4.4. Examples

Gen and Li carried out numerical experiments and compared with the matrix-based GA [44]. For the small-scale problem, there was no obvious difference on results. For the larger scale problems, the spanning tree-based GA can obtain the optimal solutions with much less CPU time than the matrix-based GA [33,34].

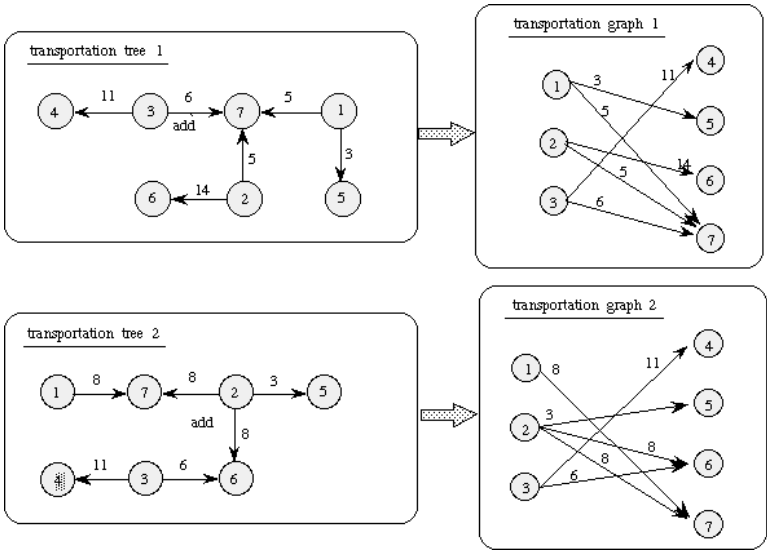

Fig. 13. Spanning trees and transportation graphs corresponding to offspring.

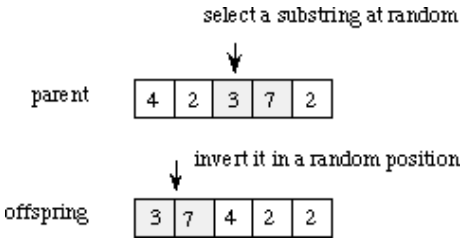

Fig. 15. Illustration of displacement mutation.

5. Minimum spanning tree problem

A spanning tree structure is one of well-used topology for telecommunication network designs, which usually consists of finding the best way to link n nodes at different locations [45]. They may be the host, concentrators, multiplexes, and terminals. In a real-life network optimization situation, a spanning tree is often required to satisfy some additional constraints, such as capacities on edges or nodes. A tree structure network with a constraint on connection among edges is known as *degree-constrained minimum spanning tree problem* (dc-MST) [38].

Considering an undirected graph $G = (V, E)$, let $V = \{1, 2, \ldots, n\}$ be the set of nodes and $E = \{(i,j) \mid i,j \in V\}$ be the set of edges. For a subset of nodes $S(\subseteq V)$, define $E(S) = \{(i,j) \mid i,j \in S\}$ be the edges whose end points are in S. Define the following binary decision variables for all edges $(i,j) \in E$.

$$x_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \text{ is selected in a spanning tree,} \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Let w_{ij} be the fixed cost related to edge (i,j), the problem can be formulated as follows:

$$\text{dc-MST}: \qquad \min z(x) = \sum_{i=1}^{n-1} \sum_{j=2}^{n} w_{ij} x_{ij}, \tag{9}$$

$$\text{s. t.} \sum_{i=1}^{n-1} \sum_{j=2}^{n} x_{ij} = n - 1, \tag{10}$$

$$\sum_{i \in S} \sum_{j \in S, j>1} x_{ij} \le |S| - 1, \; S \subseteq V \backslash \{1\}, \qquad |S| \ge 2, \tag{11}$$

$$\sum_{j=1}^{n} x_{ij} \le b_i, \qquad i = 1, 2, \ldots, n, \tag{12}$$

$$x_{ij} = 0 \text{ or } 1, \qquad i = 1, 2, \ldots, n-1, \; j = 2, 3, \ldots, n, \tag{13}$$

where b_i is the constrained degree value for node i. Inequality (12) is the constrained degree on each node. Equality (10) is true of all spanning trees. The dc-MST problem is NP-hard and there are no effective algorithms to deal with it. Zhou and Gen proposed a GAs approach to solve this problem [47,49].

5.1. Representation

For the dc-MST, there are two factors that should be taken into consideration: (1) the connectivity among vertices; (2) the degree constraint of each vertex. Zhou and Gen proposed a two-dimension structure to encode a spanning tree with degree constraints: one dimension for node permutations and the other for degree constraints [49]. The genes in the vertex dimension take exclusively the integers from 1 to n; the genes in the degree dimension take inclusively the integers from 1 to b the common upper bound value for all vertices. For an undirected tree, we can take any node as the root node and all other nodes are regarded as being connected to it hierarchically. The encoding procedure from a tree to a degree-based permutation is given below.

Procedure: degree-based permutation encoding

Step 1. Select a vertex as the root vertex in a labeled tree T, put it as the left-most one in the vertex dimension of the permutation and its degree value as the left-most digit in the degree dimension, and let that vertex be the current vertex.
Step 2. Check the successor vertex of the current vertex from left branch to right branch. If there is a successor vertex, put it into the vertex dimension and its degree value into the degree dimension next to current vertex, and then go to step 3. If there is no such vertex, let the predecessor vertex be the current vertex, and return to step 2.
Step 3. If the successor vertex is not a leaf vertex, let the successor vertex be the current vertex, then go to step 2. If the successor vertex is a leaf vertex, delete it and go to step 4.
Step 4. If all vertices have been checked, stop; otherwise, go to step 2.

Fig. 16 illustrates an example of the degree-based permutation.

It is easy to obtain a tree from the degree-based encoding. Let $P_1(k)$ denote the kth gene in the vertex dimension and $P_2(k)$ denote the kth gene in the degree dimension for a given individual. The decoding procedure works as follows:

Procedure: degree-based permutation decoding

Step 1. Set $k \leftarrow 1$ and $j \leftarrow 2$.
Step 2. Select the vertex $r = P_1(k)$ and $s = P_1(j)$. Add the first edge (r, s) into a tree.
Step 3. Let $P_2(j) = P_2(j) - 1$ and $P_2(k) = P_2(k) - 1$.
Step 4. If $P_2(j) \ge 1$, let $k \leftarrow j$ and $j \leftarrow j + 1$.
Step 5. If $k < 1$, stop.
Step 6. If $P_2(k) \ge 1$, select the vertex $r = P_1(k)$ and let $P_2(k) = P_2(k) - 1$; otherwise, let $k \leftarrow k - 1$ and go to step 5.

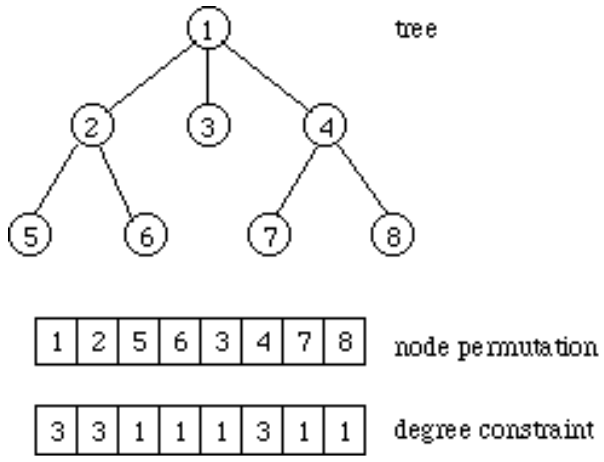

Fig. 16. A rooted tree and its degree-based permutation encoding.

Step 7. If $P_2(j) \geq 1$, select the vertex $s = P_1(j)$, and let $P_2(j) = P_2(j) - 1$; otherwise, let $j \leftarrow j + 1$ and go to step 7.

Step 8. Add the edge (r, s) into the tree and go to step 4.

5.2. Feasibility condition

In order to keep the degree constraint and connectivity among nodes, the genes in the degree dimension need to satisfy the following conditions. For an n-node tree, the total degree value for all nodes is $2(n-1)$. Suppose that d_{used} is the total degree value of the nodes whose degree value in degree dimension have been assigned and d_{rest} is the total lower bound of the degree values for all those nodes whose degree value in degree dimension have not been assigned. Then the degree value for the current node in degree dimension should hold: not less than 1 and not greater than upper bound. The degree value for the rest nodes should hold: not less than d_{rest} and not greater than $2(n-1) - d_{\text{used}}$.

5.3. Genetic operators

Since this encoding is essentially a permutation type, order crossover, swap mutation and insertion mutations were adopted. In order to avoid the illegal connection among vertices, the crossover only operates on vertex dimension and the degree dimension keeps unchanged. It is illustrated in Fig. 17. The crossover will dramatically change the tree structure among generations. This is useful for exploration on whole solution space.

Swap mutation selects two genes (vertices) at random and then swaps them. This is illustrated in Fig. 18.

Insertion mutation removes a string of genes (branch of a tree) at random and inserts it in a random position. Because removing and then inserting a string of genes causes to change the degree value of vertex, it is necessary to modify the gene (degree) in the corresponding vertex. The operation is illustrated in Fig. 19. This operator can keep good heritages from generations to generations. As a result, it makes the exploitation in the evolutionary process.

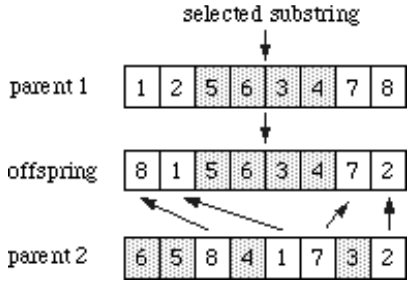

Fig. 17. Illustration of Order crossover on vertices.

6. Centralized network design problem

A centralized network is a network where all communication is to and from a single site. In such networks, terminals are connected directly to the central site. Multipoint lines are used, where groups of terminals share a tree to the center and each multipoint line is linked to the center site by one edge only. This problem is also denoted as the capacitated minimum spanning tree (c-MST) problem [10].

Considering a complete, undirected graph $G = (V, E)$, $V = \{1, 2, \ldots, n\}$ be the set of nodes and $E = \{(i,j) \mid i,j \in V\}$ be the set of edges. For a subset of nodes $S(\subseteq V)$, define $E(S) = \{(i,j) \mid i,j \in S\}$ be the edges whose endpoints are in S. Let c_{ij} be the fixed cost related to edge (i, j), and d_i be the demand at each node $i \in V$, where by convention the demand of the root node $d_1 = 0$. $d(S)$ is the sum of the demands of the nodes of S. The subtree capacity is denoted κ. The centralized network design (CND) problem can be formulated as follows:

$$\text{CND}: \qquad \min z(x) = \sum_{i=1}^{n-1} \sum_{j=2}^{n} c_{ij} x_{ij}, \tag{14}$$

$$\text{s. t.} \sum_{i=1}^{n-1} \sum_{j=2}^{n} x_{ij} = n - 1, \tag{15}$$

$$\sum_{i \in S} \sum_{j \in S, j>1} x_{ij} \leq |S| - \lambda(S), \qquad S \subseteq V \backslash \{1\}, \ |S| \geq 2, \tag{16}$$

$$\sum_{i \in U} \sum_{j \in U, j>1} x_{ij} \leq |U| - 1, \qquad U \subset V, \ |U| \geq 2, \{1\} \in U, \tag{17}$$

$$x_{ij} = 0 \text{ or } 1, \qquad i = 1, 2, \ldots, n-1, \ j = 2, 3, \ldots, n. \tag{18}$$

The parameter $\lambda(S)$ in Eq. (16) refers to the bin packing number of the set S, namely, the number of bins of size k needed to pack the nodes of items of size d_i for all $i \in S$. Up to now, all heuristic algorithms for these problems are only focused on how to deal with the constraints to make the

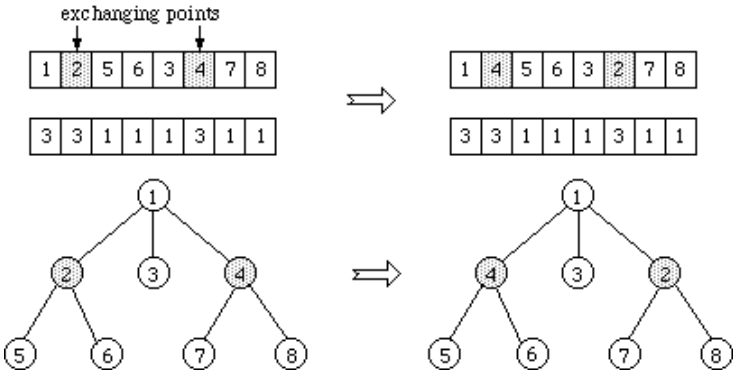

Fig. 18. Illustration of swap mutation.

problem simpler to solve. Some approaches, including cutting plane algorithms [23,24] and branch-bound algorithm [35], usually neglects the network topology of the problem. Recently, Zhou and Gen proposed a GA approach, using a degree-based permutation, explained in last section, to encode the candidate solution for the problem [48]. Zhou and Gen tested the problem given by Gavish [11]. The example consists of 16 nodes, a unit traffic between each node and node 1 and with a capacity restriction $\kappa = 5$. The best solution is 8526. After simulating 500 generations they got the optimal solution and its corresponding topology of a centralized tree. Fig. 20 illustrates the result.

7. Local area network design problem

In computer networking, local area networks (LANs) are commonly used as the communication infrastructure that meets the demands of the users in local environment. The computer networks typically consist of several LAN segments connected together via bridges. The use of transparent bridges requires loop-free paths between LANs segments. Therefore, the spanning tree topologies can be used as active LANs configurations when designing a computer network system.

Considering the LANs with n service centers and m users. Fig. 21 shows an example of the LANs with 5 centers and 23 terminals. Define the $n \times n$ service center topology matrix $\mathbf{X}_1$, whose element x_{1ij} represents whether the centers i and j are connected. Assume that LANs are partitioned into n segments (service centers). The users are distributed over those n service centers. The $n \times m$ clustering matrix $\mathbf{X}_2$ specifies which user belongs to which center, whose element x_{2ij} means whether user j belongs to center i. Define the $n(n+m)$ matrix $\mathbf{X}$ called the spanning tree matrix ($[\mathbf{X}_1\ \mathbf{X}_2]$). The $M/M/1$ model is used to describe a single cluster (LAN segment) behavior. Then we can formulate the bicriteria LAN topology design (bc-LAN) problem as the following nonlinear 0-1 programming model

$$\text{bc-LAN}: \quad \min \frac{1}{\Gamma}\left[\sum_{i=1}^{n} \frac{c_i(\mathbf{X})}{C_i - c_i(\mathbf{X})} + \sum_{i=1}^{n}\sum_{j=1}^{m} \beta_{ij}\cdot d_{ij}(\mathbf{X})\right], \tag{19}$$

$$\min \sum_{i=1}^{n-1}\sum_{j=i+1}^{m} w_{1ij}\cdot x_{1ij} + \sum_{i=1}^{n}\sum_{j=1}^{m} w_{2ij}\cdot x_{2ij}, \tag{20}$$

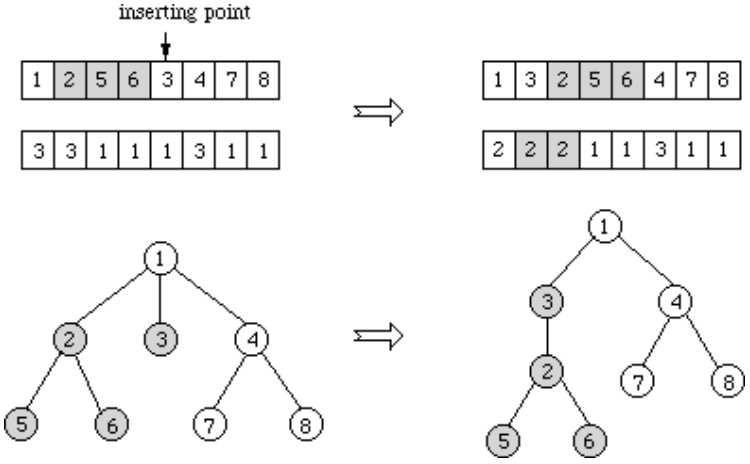

Fig. 19. Illustration of insertion mutation.

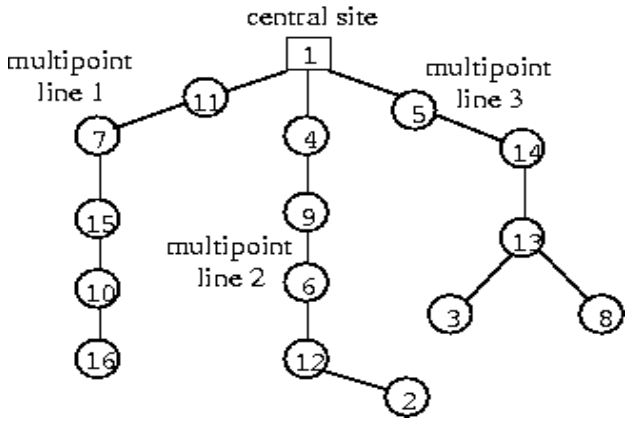

Fig. 20. Optimal solution for the centralized network.

$$\text{s. t.} \sum_{j=1}^{m} x_{2ij} \le g_i, \qquad i = 1, 2, \ldots, n, \tag{21}$$

$$\sum_{i=1}^{n} x_{2ij} = 1, \qquad j = 1, 2, \ldots, m, \tag{22}$$

$$c_i(\mathbf{X}) < C_i, \qquad i = 1, 2, \ldots, n,$$

where Γ is the total offered traffic, $c_i(\mathbf{X})$ the total traffic at center i, $d_{ij}(\mathbf{X})$ the total traffic through link (k, l), C_i the traffic capacity of center i, β_{ij} the delay per bit due to the link between centers i and j, g_i the maximum number which is capable of connecting to center i, w_{1ij} the weight of the link between centers i and j, and w_{2ij} is the weight of the link between center i and user j.

A solution to the problem can be regarded as a spanning tree, where all users are terminals or leaf nodes and all centers are internal nodes. Gen, Ida, and Kim proposed a GA to solve the bicriteria LANs topological design problem, minimizing the connecting cost and the average networks message delay [15,16,29,30]. Prüfer number was used to encode all internal nodes and all leaf nodes are not included in the encoding. GAs are used to search for Pareto optimal solutions and the TOPSIS method was used to calculate the compromise solution among Pareto solutions [26].

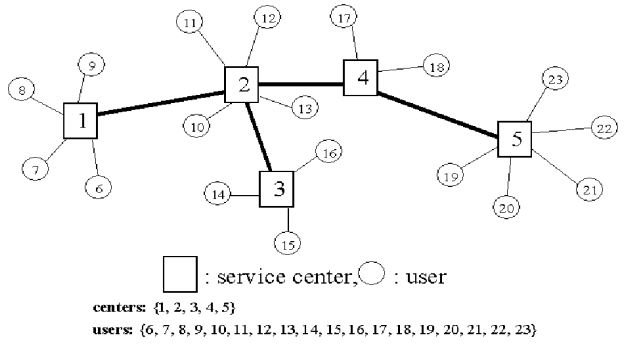

Fig. 21. The sample LANs architecture.

8. Shortest path problem

One of the most common problems encountered in analysis of networks is shortest path problem. The problem is to find a path between two designated nodes having minimum total length or cost. It is a fundamental problem that appears in many applications involving transportation, routing and communication [41,42]. In many applications, however, there are several criteria associated with traversing each edge of a network. For example, cost and time measures are both important in transportation networks, economic and ecological factors for highway construction. As a result, there has been recent interest in solving bicriteria shortest path problem. It is to find paths that are efficient with respect to both criteria. There is usually no single path that gives the shortest path with respect to both criteria. Instead, a set of Pareto optimal paths is preferred. Cheng and Gen proposed a compromise approach-based GA to solve the bicriteria shortest path problem [14]. The compromise approach, contrary to generating approach, identifies solutions, which are closest to the ideal solution as determined by some measure of distance.

8.1. Representation

How to encode a path for a graph is a critical step when apply GA. Special difficulties are (1) a path contains variable number of nodes, and (2) a random sequence of edges usually does not correspond to a path. To overcome such difficulties, Cheng and Gen proposed a *priority-based encoding* method. In this method [5], the position of a gene was used to represent a node and the value of the gene was used to represent the priority of the node for constructing a path among candidates, which is illustrated in Fig. 22.

Essentially, the mapping between encoding and path is many-to-one, which means that different chromosomes may produce to an identical shortest path. But it is easy to prove that the probability of occurrence of many-to-one mapping is very low. Therefore, at most cases, there are no trivial genetic operations associated with the encoding. It is easy to verify that any permutation of the encoding corresponds to a path so that most existing genetic operators can be easily applied to the encoding. Also any path has a corresponding encoding, therefore, any point in solution space is accessible for genetic search.

8.2. Path growth procedure

The path corresponding to a given chromosome is generated by sequential node appending procedure with beginning from the specified node 1 and terminating at the

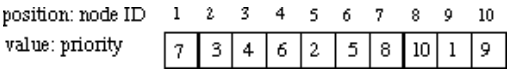

Fig. 22. An example of priority-based encoding.

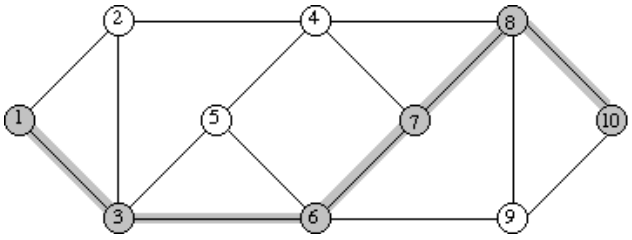

Fig. 23. A simple undirected graph with 10 nodes and 16 edges.

specified node n. At each step, there are usually several nodes available for consideration; only the node with the highest priority is added into path. Consider the undirected graph shown in Fig. 23. Suppose we want to find a path from node 1 to node 10. At the beginning, we try to find a node for the position next to node 1. Nodes 2 and 3 are eligible for the position, which can be easily fixed according to adjacent relation among nodes. The priorities of them are 3 and 4, respectively. The node 3 has the highest priority and is put into the path. The possible nodes next to node 3 are nodes 2, 5 and 6. Because node 6 has the largest priority value, it is put into the path. Then we form the set of nodes available for next position and select the one with the highest priority among them. Repeat these steps until we obtain a complete path (1, 3, 6, 7, 8, 10). A detailed description of the path growth procedure is given in Ref. [13] (Fig. 23).

8.3. Genetic operators

The nature of the proposed encoding is a kind of permutation representation. A number of recombination operators have been investigated for permutation representation. Here the position-based crossover was adopted. Essentially, it takes some genes from one parent at random and fills the vacuum position with genes from the other parent by a left-to-right scan (Fig. 24).

The swap mutation operator was used, which simply select two positions at random and swap their contents as shown in Fig. 25.

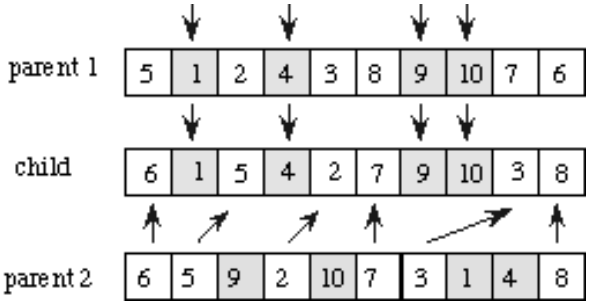

Fig. 24. The position-based crossover operator.

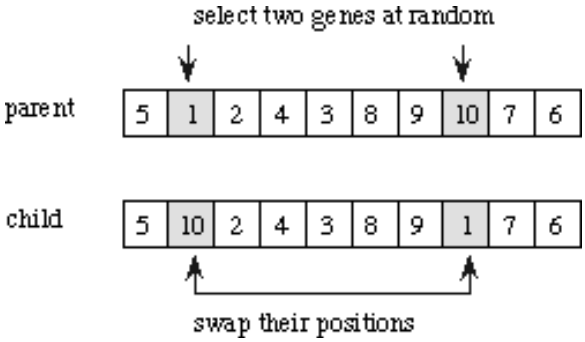

Fig. 25. The swap mutation operator.

8.4. Compromise approach

This approach can be regarded as a kind of mathematical formulation of goal seeking behavior in terms of a distance function. It identifies solutions that are closet to the ideal solution as determined by following weighted L_p-norm:

$$r(z;p,w) = \left[\sum_{k=1}^{2} w_k^p |z_k - z_k^*|^p\right]^{1/p}. \tag{23}$$

The weight w_k is assigned to each objective to signal the different degrees of importance. The parameter p is used to reflect the emphasis of decision-makers. When $p = 1$, the sum of regrets of all objectives is most emphasized. When p turns to infinity, the individual regret of objectives is most emphasized. In general, the choice of p reflects the concerns with respect to the maximal regret among objectives. The larger the value of p is, the greater the concern.

$z^* = (z_1^*, z_1^*)$ is the ideal solution to the problem. For the bicriteria shortest path problem, the ideal point can be easily obtained by solving two single criterion problems. For many complex problems, to obtain an ideal point is also a difficult task. To overcome the difficulty, a concept of *proxy ideal point* was suggested to replace the ideal point. The proxy ideal point is the ideal point corresponding to current generation, not to a given problem. In the other words, it is calculated in the

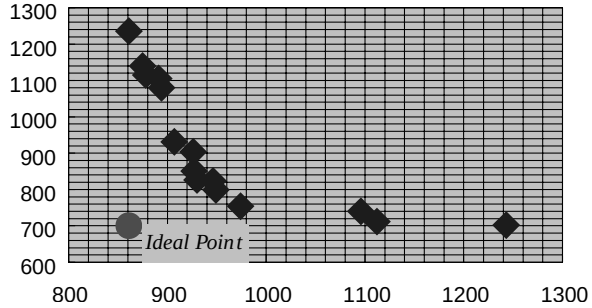

Fig. 26. Pareto solutions for the test problem.

Table 1
Results on the random test problems

	Ideal solution	$w_1 = 0.5,\ w_2 = 0.5$	$w_1 = 0.2,\ w_2 = 0.8$	$w_1 = 0.8,\ w_2 = 0.2$
L_1-norm	(861,702)	(974,754)	(1112,712)	(930,826)
L_2-norm	(861,702)	(974,754)	(974,754)	(930,826)
L_∞-norm	(861,702)	(1243,702)	(1243,702)	(861,1236)

explored partial solution space but not the whole solution space. The proxy ideal point is easy to obtain at each generation, as given in Eq. (24). Along with evolutionary process, the proxy ideal point will gradually approximate to the real ideal point

$$z^1_{\min} = \min\{z^1(x) \mid x \in P\}, \qquad z^2_{\min} = \min\{z^2(x) \mid x \in P\}. \tag{24}$$

8.5. Evaluation and selection

Since the smaller the regret value, the better the individual, we have to convert the regret value into the fitness value in order to ensure that a fitter individual has a larger fitness value. Let $r(x)$ denote the regret value of individual, $r_{\max}$ the biggest regret value, and $r_{\min}$ the smallest regret value in current generation. The transformation is given as follows:

$$\text{eval}(x) = \frac{r_{\max} - r(x) + \gamma}{r_{\min} - r_{\min} + \gamma}, \tag{25}$$

where γ is a positive real number which is usually restricted within the open interval (0,1). The purpose of using it is twofold: (1) to prevent Eq. (25) from zero division and (2) to make it possible to adjust the selection behavior from fitness-proportional selection to pure random selection.

The roulette wheel approach was adopted as the selection procedure, which is one of the fitness-proportional selections. The *elitist* way was combined with this approach in order to preserve the best chromosome in the next generation and overcome the stochastic errors of sampling. With the elitist selection, if the best individual in the current generation is not reproduced into the new generation, remove one individual randomly from the new population and add the best one to the new population.

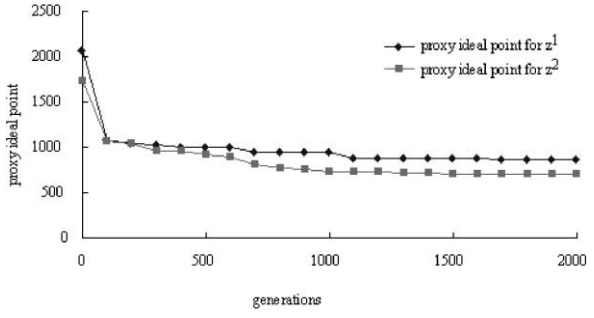

Fig. 27. Evolutionary process of proxy ideal solution.

8.6. Overall procedure of GA

The overall procedure is summarized as follows:

Step 1. Set parameters and read in the data of a given instance.
Step 2. Generate initial population randomly.
Step 3. Decode chromosomes into paths.
Step 4. Fitness calculation: (1) calculate objectives for each decoded individual, (2) calculate proxy ideal point in current population, (3) calculate regret value for each individual, and (4) convert regret value to fitness value.
Step 5. Select next generation with roulette wheel method.
Step 6. If the maximal generation is reached, stop.
Step 7. Produce offspring with crossover and mutation.
Step 8. Go back to step 3.

8.7. Examples

The randomly generated test problem is a non-planar and undirected graph with 100 nodes and 473 edges. Each edge is associated with two numbers: time and cost. The evolutionary environment is set as follows: population size 40, maximum generation 1000, crossover ratio 0.4 and mutation ration 0.2. The obtained Pareto solutions are depicted in Figs. 26–27.

The ideal solution and compromise solutions for different weights are given in Table 1, where the ideal solution is obtained by running Floyd–Warshall algorithm twice [36], each time on one objective.

The compromise approach-based GAs can be readily applied to other multiple objective optimization problems and the priority based encoding method can be easily used in other highly constrained combinatorial optimization problems.

9. Conclusions

With the development of modern society, the data communication has become very important part in human being's life [2]. Actually, this trend will continue to the next century or even further future. Simultaneously, it also brings about many problems related with varieties of network designs to us. In this paper, we just gave out a brief review about our recent research works in this field [7,9,20,21,28,37,40]. Different from many other conven-

tional techniques, we developed the GAs to deal with all these network design problems. Our limited computational experience showed that the GAs approach is competitive to solve such kinds of networks design problems like MPP problem, minimum spanning tree problem, centralized network design, local network design, and fc-TP transportation problem. Our recent experiences also showed that GA is a very effective means to solve location-allocation problems [7,9,20,21,28,37]. Especially, with the increase of problem scale and some complicated constraints, the GAs showed their even great potential power to cope with all these network design problems. From this point of view, the GAs for them are not only means of algorithms or techniques, but also a kind of art in the sense that the problems were solved in coding space instead of the solutions space themselves. The paper therefore focused on such kind of state-of-the-art in the GAs approach on these network design problems.

Acknowledgements

This research work was supported by the International Scientific Research Program, the Grant-in-Aid for Scientific Research (No. 10044173: 1998.4-2001.3) by the Ministry of Education, Science and Culture, the Japanese Government.

References

[1] Awadh B, Sepehri N, Hawaleshka O. A computer-aided process planning model based on genetic algorithms. Comput Oper Res 1995;22:841–56.
[2] Bertsekas D, Gallager R. Data networks. 2nd ed. New Jersey: Prentice-Hall, 1992.
[3] Chang TC, Wysk RA. An introduction to automated process planning systems. Englewood Cliffs: Prentice-Hall, 1985.
[4] Cheng R. Study on Genetic Algorithm-based Optimal Scheduling Techniques, PhD dissertation, Tokyo Institute of Technology, Japan, 997.
[5] Cheng R, Gen M. An evolution program for the resource-constrained project scheduling problem. Comput Integrated Manufacturing 1998;11(3):274–87.
[6] Cheng R, Gen M, Oren S. An adaptive hyperplane approach for multiple objective optimisation problems with complex constraints. Proc GECOO 2000:299–306.
[7] Cooper L. Location-allocation problems. Oper Res 1963;11(3):331–44.
[8] Cormen TH, Leiserson CE, Rivest RL. Introduction to algorithms. Cambridge, MA: MIT Press, 1990.
[9] Domschke K, Drex A. An international bibliography on location and layout planning. Heidelberg: Springer, 1984.
[10] Gavish B. Topological design of centralized computer networks: formulation and algorithms. Networks 1982;12:355–77.
[11] Gavish B. Augmented Lagrangian-based algorithms for centralized network design. IEEE Trans Commun 1985;COM-33:1247–57.
[12] Gen M, Cheng R. Genetic algorithms and engineering design. New York: Wiley, 1997.
[13] Gen M, Cheng R. Genetic algorithms and engineering optimisation. New York: Wiley, 2000.
[14] Gen M, Cheng W, Wang D. Genetic algorithms for solving shortest path problems. In: Proceedings of IEEE International Conference on Evolutionary Computation, Indianapolis, Indiana. 1997;401–6.
[15] Gen M, Ida K, Kim JR. A spanning tree-based genetic algorithm for bicriteria topological network design. In: Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, Alaska. 1998;15–20.
[16] Gen M, Kim JR. In: Dagli CH, editor. GA-based optimal network design: a state-of-the-art survey. Intelligent engineering systems, through artificial neural networks, vol. 8. New York: ASME Press, 1998. p. 247–52.
[17] Gen M, Li Y. Solving multiobjective transportation problem by spanning tree-based genetic algorithm. In: Parmee I, editor. Adaptive computing in design and manufacture. Berlin: Springer, 1998. p. 98–108.
[18] Gen M, Li, YZ. Hybrid genetic algorithms for transportation problem in logistics. J Logistics; Res Appl, 1999 (forthcoming).
[19] Goldberg DE. Genetic algorithms in search, optimisation and machine learning. Reading: Addison-Wesley, 1989.
[20] Gong D, Gen M, Yamazaki G, Xu W. Hybrid evolutionary method for obstacle location–allocation problem. Comput Ind Engng 1995;29(1–4):525–30.
[21] Gong D, Gen M, Yamazaki G, Xu W. Hybrid evolutionary method for capacitated location–allocation. Engng Des Automation 1997;3(2):166–73.
[22] Gottlieb J, Paulmann L. Genetic algorithms for the fixed charge transportation problem. In: Proceedings of IEEE International Conference on Evolutionary Computation, Anchorage, Alaska. 1998;330–5.
[23] Gouveia L. A 2n constraint formulation for the capacitated spanning tree problem. Oper Res 1995;43(1):130–41.
[24] Hall L. Experience with a cutting plane algorithm for the capacitated minimal spanning tree problem. INFORMS J Comput 1996;8(3):219–34.
[25] Holland JH. Adaptation in natural and artificial systems. Cambridge, MA: MIT Press, 1975.
[26] Hwang C, Yoon K. Multiple attribute decision-makings: methods and applications. Berlin: Springer, 1981.
[27] Jensen AP, Barnes JW. Network flow programming. New York: Wiley, 1980.
[28] Katz I, Cooper L. Facility location in the presence of forbidden regions; I. formulation and the case of the euclidean distance with one forbidden circle. Eur J Oper Res 1981;6:166–73.
[29] Kim JR, Gen M, Ida K. Bicriteria network design using spanning tree-based genetic algorithm. Artificial Life Robotics 1999;3:65–72.
[30] Kim JR, Gen M, Yamashiro M. A bi-level hierarchical GA for reliable network topology design. In: Proceedings of the Seventh European Congress on Intelligent Techniques and Soft Computing, Session CD-7, Aachen, 1999.
[31] Kusiak A, Finke G. Selection of process plans in automated manufacturing systems. IEEE J Robotics Automation 1988;4(4):397–402.
[32] Li YZ, Gen M. Spanning tree-based genetic algorithm for bicriteria transportation problem with fuzzy coefficients. Aust J Intelligent Inform Process Syst 1997;4(3/4):220–9.
[33] Li YZ, Gen M, Ida Y. Fixed charge transportation problem by spanning tree-based genetic algorithm. Beijing Math 1998;4(2):239–49.
[34] Li YZ. Study on hybridized genetic algorithm for production distribution planning problems, PhD dissertation, Ashikaga Institute of Technology, Japan, 1999.
[35] Malik K, Yu G. A branch and bound algorithm for the capacitated minimum spanning tree problem. Networks 1993;23:525–32.
[36] McHugh JA. Algorithmic graph theory. New Jersey: Prentice-Hall, 1990.
[37] Multagh B, Niwattisyawong S. An efficient method for the muti-depot location-allocation problem. J Oper Res Soc 1984;33:629–34.
[38] Narula SC, Ho CA. Degree-constrained minimum spanning tree. Comput Oper Res 1980;7:239–49.
[39] Prüfer H. Neuer beweis eines satzes über permutation. Arch Math Phys 1918;27:742–4.

[40] Rosing K. An optimal method for solving (generalized) multi-weber problem. Eur Oper Res 1992;58:479–86.
[41] Sancho NG. A multi-objective routing problem. Engng Optimisation 1986;10:71–6.
[42] Sniedovich M. A multi-objective routing problem revisited. Engng Optimisation 1988;13:99–108.
[43] Sun M, Aronson JE, Mckeown PG, Drinka D. A tabu search heuristic procedure for the fixed charge transportation problem. Eur J Oper Res 1998;106:441–56.
[44] Vignaux GA, Michalewicz Z. A genetic algorithm for the linear transportation problem. IEEE Trans Syst Man Cybern 1991;21:445–52.
[45] Walters GA, Smith DK. Evolutionary design algorithm for optimal layout of tree networks. Engng Optimisation 1995;24:261–81.
[46] Zhou G, Gen M. In: Porto B, editor. Evolutionary computation on multicriteria production process planning problem. Proceedings of the IEEE International Conference on Evolutionary Computation, 1997. p. 419–24.
[47] Zhou G, Gen M. A note on genetic algorithm approach to the degree-constrained spanning tree problems. Networks 1997;30:105–9.
[48] Zhou G, Gen M. Approach to degree-constrained minimum spanning tree problem using genetic algorithm. Engng Des Automation 1997;3(2):157–65.
[49] Zhou G. Study on Constrained Spanning Tree Problems with Genetic Algorithms, PhD dissertation, Ashikaga Institute of Technology, Japan, 1999.