42. Genetic Algorithms and Their Applications

The first part of this chapter describes the foundation of genetic algorithms. It includes hybrid genetic algorithms, adaptive genetic algorithms and fuzzy logic controllers. After a short introduction to genetic algorithms, the second part describes combinatorial optimization problems including the knapsack problem, the minimum spanning tree problem, the set-covering problem, the bin-packing problem and the traveling-salesman problem; these are combinatorial optimization studies problems which are characterized by a finite number of feasible solutions. The third part describes network design problems. Network design and routing are important issues in the building and expansion of computer networks. In this part, the shortest-path problem, maximum-flow problem, minimum-cost-flow problem, centralized network design and multistage process-planning problem are introduced. These problems are typical network problems and have been studied for a long time. The fourth section describes scheduling problems. Many scheduling problems from manufacturing industries are quite complex in nature and very difficult to solve by conventional optimization techniques. In this part the flow-shop sequencing problem, job-shop scheduling, the resource-constrained projected scheduling problem and multiprocessor scheduling are introduced. The fifth part introduces the reliability design problem, including simple genetic algorithms for reliability optimization, reliability design with redundant units and alternatives, network reliability design and tree-based network topology design. The sixth part describes logistic problems including the linear transportation problem, the multiobjective transportation problem, the bicriteria transportation problem with fuzzy coefficients and supply-chain management network design. Finally, the last part describes location and allocation problems including the location–allocation problem, capacitated plant-location problem and the obstacle location–allocation problem.

42.1 Foundations of Genetic Algorithms

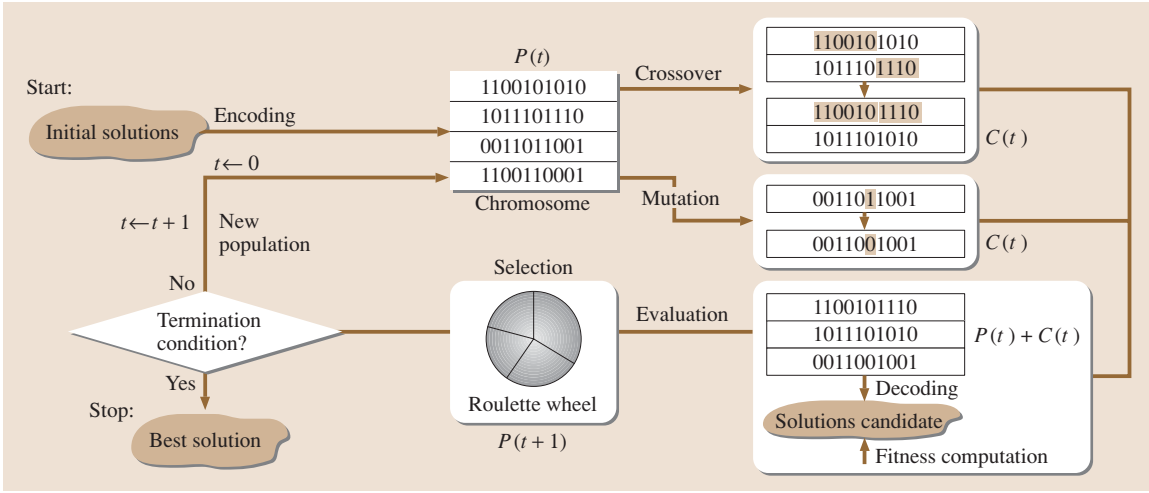

Fig. 42.1 The general structure of genetic algorithms

Recently, genetic algorithms have received considerable attention regarding their potential as an optimization technique for complex problems and have been successfully applied in the area of industrial engineering. The well-known applications include scheduling and sequencing, reliability design, vehicle routing location, transportation and many others.

42.1.1 General Structure of Genetic Algorithms

Genetic algorithms are stochastic search algorithms based on the mechanism of natural selection and natural genetics. Genetic algorithms, in contrast to conventional search techniques, start with an initial set of random solutions called the population. Each individual in the population is called a chromosome, encoding a solution to the problem at hand. A chromosome is a string of symbols, usually but not necessarily, a binary bit string. The chromosomes *evolve* through successive iterations, called generations. During each generation, the chromosomes are *evaluated*, using some measures of fitness [42.1]. To create the next generation, new chromosomes, called offspring, are formed by either merging two chromosomes from the current generation using a *crossover* operator or modifying a chromosome using a mutation operator. A new generation is formed by selecting, according to the fitness values, some of the parents and offspring, and rejecting others so as to keep the population size constant.

Fitter chromosomes have higher probabilities of being selected. After several generations, the algorithms converge to the best chromosome, which we hope represents the optimum or suboptimal solution to the problem

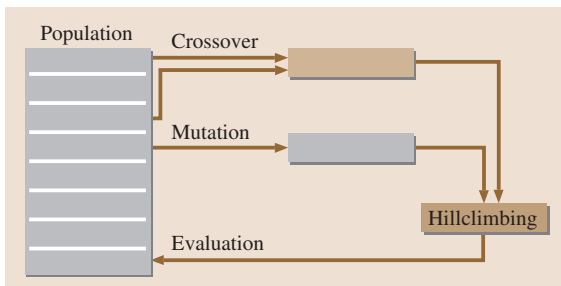

Fig. 42.2 General structure of hybrid genetic algorithms

when decoded. Let $P(t)$ and $C(t)$ be parents and offspring in the current generation t; the general structure of genetic algorithms Fig. 42.1 is described as follows:

```
procedure: genetic algorithms
begin
  t ← 0;              // t: generation number
  initialize P(t);    // P(t): population of individuals
  evaluate P(t);
  while (not termination condition) do
        crossover P(t) to yield C(t) ; // C(t): offspring
        mutation P(t) to yield C(t);
        evaluate C(t) ;
        select P(t+1) from P(t) and C(t) ;
        t ← t+1;
  end
end
```

Crossover is the main genetic operator. It operates on two chromosomes at a time and generates offspring by combining both chromosomes' features. A simple way to achieve crossover would be to choose a random cut-point and generate the offspring by combining the segment of one parent to the left of the cut-point with the segment of the other parent to the right of the cut-point.

Mutation is a background operator, which produces spontaneous random changes in various chromosomes. A simple way to achieve mutation would be to alter one or more genes.

42.1.2 Hybrid Genetic Algorithms

Genetic algorithms (GAs) have proved to be a versatile and effective approach for solving optimization problems. Nevertheless, there are many situations in which the simple GA does not perform particularly well, and various methods of have been proposed [42.2].

One of the most common forms of hybrid genetic algorithms is to incorporate local optimization as an add-on extra to the canonical GA loop of recombination and selection [42.3, 4]. With the hybrid approach, local optimization such as hill-climbing is applied to each newly generated offspring to move it to a local optimum before injecting it into the population. Genetic algorithms are used to perform global exploration among the population while heuristic methods are used to perform local exploitation around chromosomes. Because of the complementary properties of genetic algorithms and conventional heuristics, the hybrid approach often outperforms either method operating alone. Some work has been done to reveal the natural mechanism behind such a hybrid approach, among which is Lamarckian evolution. Let $P(t)$ and $C(t)$ be parents and offspring in the current generation t. The general structure of hybrid genetic algorithms is described as follows; see Fig. 42.2.

```
procedure: hybrid genetic algorithms
begin
  t ← 0 ;             // t: generation number
  initialize P(t);    // P(t): population of individuals
  evaluate P(t);
  while (not termination condition) do
        crossover P(t) to yield C(t);
                    // C(t): offspring
        mutation P(t) to yield C(t);
        locally search C(t);
        evaluate C(t);
        selection P(t+1) from P(t) and C(t);
        t  ←  t+1 ;
  end
end
```

42.1.3 Adaptive Genetic Algorithms

There are two basic approaches to applying the genetic algorithms to a given problem: 1) to adapt a problem to the genetic algorithms, 2) to adapt the genetic algorithms to a problem.

Genetic algorithms were first created as a kind of generic and weak method featuring binary encoding and binary genetic operators. This approach requires a modification of the original problem into an appropriate from suitable for the genetic algorithms, as shown in Fig. 42.3.

To overcome such problems, various nonstandard implementations of the genetic algorithm have been created for particular problems, which leave the problem unchanged and adapt the genetic algorithms by modifying a chromosome representation of a potential solution and applying appropriate genetic operators, as shown in Fig. 42.4. This approach has been successfully applied in the area of industrial engineering and is becoming the main approach in recent applications of genetic algorithms [42.5].

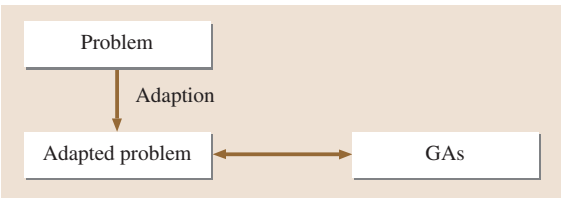

Fig. 42.3 Adapt a problem to the genetic algorithms

42.1.4 Fuzzy Logic Controller

Fuzzy logic is much closer in spirit to human thinking and natural language than the traditional logical systems. In essence, the fuzzy logic controller provides an algorithm which can convert a linguistic control strategy based on expert knowledge into an automatic control strategy. In particular, this methodology appears very useful when the processes are too complex for analysis by conventional techniques or when the available sources of information are interpreted qualitatively, inexactly, or with uncertainty [42.3].

The pioneering work to extend the fuzzy logic technique to adjust the strategy parameters of genetic algorithms dynamically was carried out by *Xu* and *Vukovich* [42.5]. The main idea is to use a fuzzy logic controller to compute new strategy parameter values that will be used by the genetic algorithms. A fuzzy logic controller is comprised of four principal components:

1. a knowledge base,
2. a fuzzification interface,
3. an inference system,
4. a defuzzification interface.

The experts' knowledge is stored in the knowledge base in the form of linguistic control rules. The inference system is the kernel of the controller, which provides an approximate reasoning based on the knowledge base. The generic structure of a fuzzy logic controller is shown in Fig. 42.5.

42.1.5 Multiobjective Optimization Problems

During the last two decades, genetic algorithms have received considerable attention regarding their potential as a novel approach to multiobjective optimization problems, known as evolutionary multiobjective optimization or genetic multiobjective optimization.

Multiobjective optimization problem with q objectives and m constraints will be formulated as follows:

$$\max \left[z_1 = f_1(\boldsymbol{x}), z_2 = f_2(\boldsymbol{x}), \ldots, z_q = f_q(\boldsymbol{x})\right] , \tag{42.1}$$

$$\text{s.t.} \quad g_i(\boldsymbol{x}) \le 0, \quad i = 1, 2, \ldots, m . \tag{42.2}$$

A. The Concept of a Pareto Solution

In most existing methods, Pareto solutions are identified at each generation and are only used to calculate fitness values or ranks for each chromosome. No mechanism is provided to guarantee that the Pareto solutions generated during the evolutionary process enter the next generation. A special pool for preserving the Pareto solutions is added onto the basic structure of genetic algorithms. At each generation, the set of Pareto solutions $E(t)$ is updated by deleting all dominated solutions and adding all newly generated Pareto solutions [42.5]. The overall structure of the approach is given as follows:

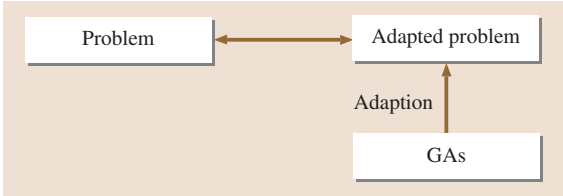

Fig. 42.4 Adapt the genetic algorithms to a problem

procedure: Pareto genetic algorithms
begin
  $t \leftarrow 0$; // t: generation number
  initialize $P(t)$; // $P(t)$: population of individuals
  objective $P(t)$;
  create Pareto $E(t)$;
  fitness $eval(P)$;
  while (not termination condition) **do**
    crossover $P(t)$ to yield $C(t)$;
      //$P(t)$: population of individuals
    mutation $P(t)$ to yield $C(t)$;
    objective $C(t)$;
    update Pareto $E(P, C)$;
    fitness $eval(P, C)$;
    selection $P(t+1)$ from $P(t)$ and $C(t)$;
    $t \leftarrow t+1$;
  end
end

B. Adaptive Weight Approach

Gen and *Cheng* proposed an adaptive weights approach which utilizes some useful information from the current population to readjust weights to obtain a search pressure towards a positive ideal point [42.6, 7].

For the examined solutions at each generation, we define two extreme points: the maximum extreme point z^+ and the minimum extreme pint z^- in criteria space as follows:

$$z^+ = \left(z_1^{\max}, z_2^{\max}, \ldots, z_q^{\max}\right) , \tag{42.3}$$

$$z^- = \left(z_1^{\min}, z_2^{\min}, \ldots, z_q^{\min}\right) , \tag{42.4}$$

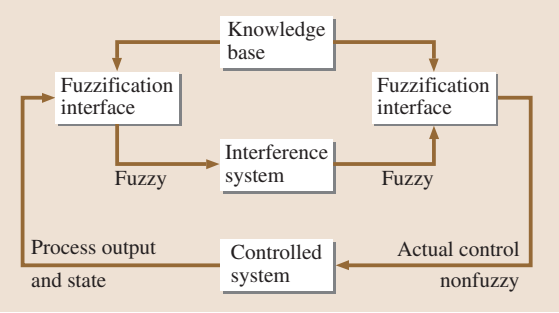

Fig. 42.5 Generic structure of a fuzzy logic controller

where $z_k^{\min}$ and $z_k^{\max}$ are the maximal value and minimal value for objective k in the current population. Let P denote the set of the current population. For a given individual $\boldsymbol{x}$, the maximal value and minimal value for each objective are defined as

$$z_k^{\max} = \max\{f_k(\boldsymbol{x}) | \boldsymbol{x} \in P\}, \quad k = 1, 2, \ldots, q\,, \tag{42.5}$$

$$z_k^{\min} = \min x\{f_k(\boldsymbol{x}) | \boldsymbol{x} \in P\}, \quad k = 1, 2, \ldots, q\,. \tag{42.6}$$

The hyperparallelogram defined by the two extreme points is a minimal hyperparallelogram containing all current solutions. The two extreme points are renewed at each generation. The maximum extreme point will gradually approximate the positive ideal point. The adaptive weight for objective k is calculated by

$$w_k = \frac{1}{z_k^{\max} - z_k^{\min}}, \quad k = 1, 2, \ldots, q\,. \tag{42.7}$$

For a given individual $\boldsymbol{x}$, the weighted-sum objective function is given by

$$z(\boldsymbol{x}) = \sum_{k=1}^{q} w_k \left(z_k - z_k^{\min}\right) \tag{42.8}$$

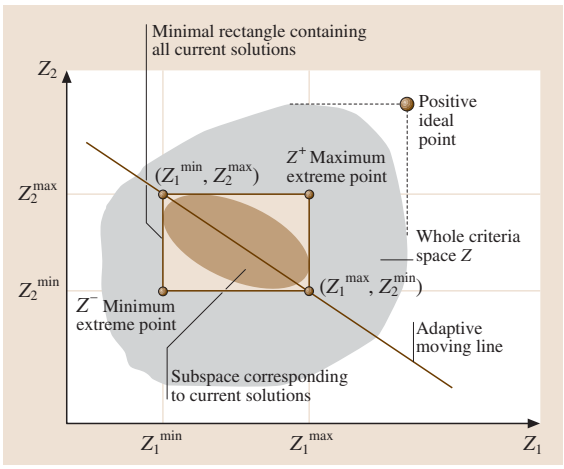

Fig. 42.6 Adaptive weights and adaptive hyperplane

$$= \sum_{k=1}^{q} \frac{z_k - z_k^{\min}}{z_k^{\max} - z_k^{\min}} \tag{42.9}$$

$$= \sum_{k=1}^{q} \frac{f_k(\boldsymbol{x}) - z_k^{\min}}{z_k^{\max} - z_k^{\min}}\,. \tag{42.10}$$

As the extreme points are renewed at each generation, the weights are renewed accordingly. Figure 42.6 is a hyperplane defined by the following extreme points in the current solutions

$$(z_1^{\max}, z_2^{\min}, \ldots, z_k^{\min}, \ldots, z_q^{\min})\,, \tag{42.11}$$

$$\ldots$$

$$(z_1^{\min}, z_2^{\min}, \ldots, z_k^{\max}, \ldots, z_q^{\min})\,, \tag{42.12}$$

$$\ldots$$

$$(z_1^{\min}, z_2^{\min}, \ldots, z_k^{\min}, \ldots, z_q^{\max})\,. \tag{42.13}$$

It is an adaptive moving line defined by the extreme points $(z_1^{\max}, z_2^{\min})$ and $(z_1^{\min}, z_2^{\max})$, as shown Fig. 42.6. The rectangle defined by the extreme points $(z_1^{\max}, z_2^{\min})$ and $(z_1^{\min}, z_2^{\max})$ is the minimal rectangle containing all current solutions.

42.2 Combinatorial Optimization Problems

Combinatorial optimization studies problems which are characterized by a finite number of feasible solutions. An important and widespread area of application concerns the efficient use of scarce resources to increase productivity. Typical problems include set covering, bin packing, knapsack, quadratic assignment, minimum spanning tree, machine scheduling, sequencing and balancing, cellular manufacturing design, vehicle routing, facility location and layout, traveling-salesman problem, and so on.

42.2.1 Knapsack Problem

Suppose that we want to fill up a knapsack by selecting some objects among various objects (generally called *items*). There are n different items available and each item j has a *weight* of w_j and a *profit* of p_j. The knapsack can hold a weight of at most W. The problem is to find an optimal subset of items so as to maximize the total profit subject to the knapsack's weight capacity. The profits, weights and capacity are positive integers [42.8].

Let x_j be binary variables given by

$$x_j = \begin{cases} 1 & \text{if item } j \text{ is selected}, \\ 0 & \text{otherwise}. \end{cases} \tag{42.14}$$

The knapsack problem can be mathematically formulated as

$$\max \sum_{j=1}^{n} p_j x_j \,, \tag{42.15}$$

$$\text{s.t.} \sum_{j=1}^{n} w_j x_j \le W \,, \tag{42.16}$$

$$x_j = 0 \quad \text{or } 1 \; j = 1, 2, \ldots, n \,. \tag{42.17}$$

Binary Representation Approach

The binary string is a natural representation for the knapsack problem, where one means the inclusion and zero the exclusion of one of the n items from the knapsack. For example, a solution for the 10-item problem can be represented as the following bit string:

$$x = (x_1 x_2 \cdots x_{10}) \\ (0\,1\,0\,1\,0\,0\,0\,0\,1\,0)\,,$$

meaning that items 2, 4 and 9 are selected for inclusion in the knapsack.

42.2.2 Minimum Spanning Tree Problem

Consider a connected undirected graph $G = (V, E)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is a finite set of *vertices* representing terminals or telecommunication stations etc., $E = \{e_{ij} | e_{ij} = (v_i, v_j), v_i, v_j \in V\}$ is a finite set of *edges* representing connections between these terminals or stations. Each edge has an associated positive real number denoted by $W = \{w_{ij} | w_{ij} = w(v_i, v_j), w_{ij} > 0, v_i, v_j \in V\}$ representing distance, cost and so on. The vertices and edges are sometimes referred to as *nodes* and *links* respectively [42.9].

Based on their different backgrounds, many researchers have proposed varieties of spanning tree problems with some constraints on them, such as the spanning tree problem with a degree constraint, the stochastic spanning tree problem, the quadratic spanning tree problem, the multi-criteria spanning tree problem and the spanning tree problem with a constraint on the number of leaves or leaf-constrained spanning tree problem [42.10, 11].

A spanning tree is a minimal set of edges from E that connects all the vertices in V and therefore at least one spanning tree can be found in graph G. The minimum spanning tree is just one of the spanning trees whose total weight of all edges is minimal. It can be formulated as

$$\min z(x) = \sum_{i=1}^{n-1} \sum_{j=2}^{n} w_{ij} x_{ij} \,, \tag{42.18}$$

$$\text{s.t.} \sum_{i=1}^{n-1} \sum_{j=2}^{n} x_{ij} = n - 1 \,; \tag{42.19}$$

$$\sum_{i \in S} \sum_{\substack{j \in S \\ j > 1}} x_{ij} \le |S| - 1, \; S \subseteq V \backslash \{1\}, \; |S| \ge 2 \,, \tag{42.20}$$

$$x_{ij} = 0 \text{ or } 1 \,, \; i = 1, 2, \ldots, n-1, \\ = 2, 3, \ldots, n \,, \tag{42.21}$$

where

$$x_{ij} = \begin{cases} 1, & \text{if edge } (i, j) \text{ is selected in a spanning tree} \\ 0, & \text{otherwise} \end{cases} \tag{42.22}$$

and T is a set of the spanning trees of graph G.

A. Tree Encodings

For the minimum spanning tree (MST) problem, the method of encoding a tree is critical for the genetic algorithm approach because the solution should be a tree.

If we associate an index k with each edge, i. e., $E = \{e_k\}$, $k = 1, 2, \ldots, K$, where K is the number of edges in a graph, a bit string can represent a candidate solution by indicating which edges are used in a spanning tree, as illustrated in Fig. 42.7.

B. Genetic Approach

Representation. The chromosome representation for a spanning tree should contain, implicitly or explicitly, the degree on each vertex. Among the several tree encodings, only the Prüfer number encoding explicitly

contains the information of vertex degree, i.e. that any vertex with degree d will appear exactly $d-1$ times in the encoding. Thus the Prüfer number encoding is adopted.

Crossover and Mutation. Prüfer number encoding can still represent a tree after any crossover or mutation operations. Simply, the one-point crossover operator is used, as illustrated in Fig. 42.8. Mutation is performed as random perturbation within the permissive integer from 1 to n (n is the number of vertices in graph). An example is given in Fig. 42.9

42.2.3 Set-Covering Problem

The problem is to cover the rows of an m-row/n-column zero–one matrix by a subset of columns at minimal cost. Considering a vector $\boldsymbol{n}$ that x_j is $0-1$ variable that takes on the 3 value 1, if item j is selected (with a cost $c_j > 0$). The set-covering the problem is then formulated as

$$\min z(x) = \sum_{j=1}^{n} c_j x_j \,, \tag{42.23}$$

$$s.t. \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad i = 1, 2, \ldots, m \,, \tag{42.24}$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n \,. \tag{42.25}$$

Genetic Approach

Representation. The fitness of an individual $f(x)$ is calculated simply by

$$f(x) = \sum_{j=1}^{n} c_j x_j \,. \tag{42.26}$$

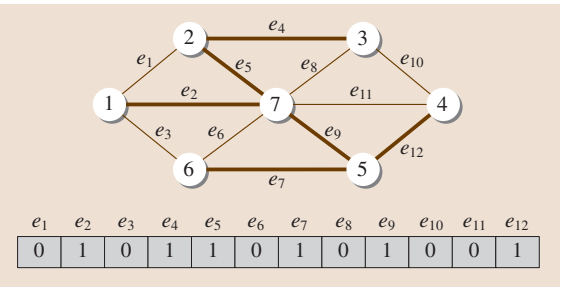

Fig. 42.7 A graph with its edge encoding for a spanning tree

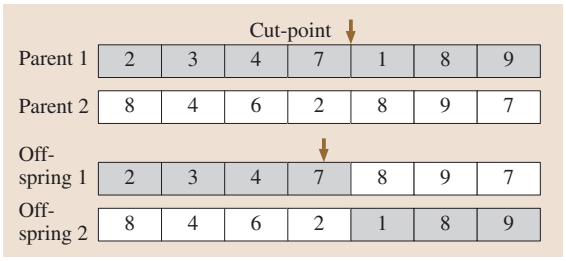

Fig. 42.8 Illustration of the crossover operation

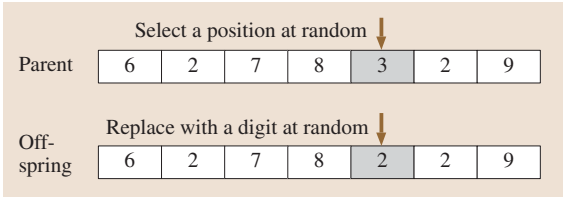

Fig. 42.9 Illustration of the mutation operation

The initial population can be generated randomly.

Genetic Operators. *Beasley* and *Chu* proposed a generalized fitness-based crossover operator called the fusion operator [42.9].

Let P_1 and P_2 be the parent strings. Let f_{P_1} and f_{P_2} be the fitness values of the parent strings P_1 and P_2, respectively. Let C be the child string. The fusion operator works as follows:

Procedure: Fusion Operator.

Step 1. $i = 1$.
Step 2. If $P_1[i] = P_2[i]$, then $C[i] \leftarrow P_1[i] = P_2[i]$.
Step 3. If $P_1[i] \neq P_2[i]$, then
(a) $C[i] \leftarrow P_1[i]$
with probability $p = f_{P_2}/(f_{P_1} + f_{P_2})$.
(b) $C[i] \leftarrow P_2[i]$ with probability $1 - p$.
Step 4. If $i = n$, stop; otherwise, set $i \leftarrow i + 1$ and go to step 1.

42.2.4 Bin-Packing Problem

The bin-packing problem consists of placing n objects into a number of bins (at most n bins). Each object has a weight ($w_i > 0$) and each bin has a limited bin capacity ($c_i > 0$). The problem is to find the best assignment of objects to bins such that the total weight of the objects in each bin does not exceed its capacity and the number of bins used is minimized.

A mathematical formulation for the bin-packing problem is given as follows [42.8]:

$$\min z(\boldsymbol{y}) = \sum_{i=1}^{n} y_i \,, \tag{42.27}$$

$$\text{s.t.} \sum_{j=1}^{n} w_j x_{ij} \le c_i y_i \,, \quad i \in N = \{1, 2, \ldots, n\} \,, \tag{42.28}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \,, \quad j \in N \,, \tag{42.29}$$

$$y_i = 0 \text{ or } 1, \quad i \in N \,, \tag{42.30}$$

$$x_{ij} = 0 \text{ or } 1, \quad i,\ j \in N \,, \tag{42.31}$$

where

$$y_i = \begin{cases} 1, & \text{if bin } i \text{ is used} \\ 0, & \text{otherwise} \,, \end{cases} \tag{42.32}$$

$$x_{ij} = \begin{cases} 1, & \text{if object } j \text{ is assigned to bin } i \\ 0, & \text{otherwise} \,. \end{cases} \tag{42.33}$$

Genetic Approach

Representation. The most straightforward approach is to encode the membership of objects in the solution. For instance, the chromosome 1 4 2 3 5 2 would encode a solution where the first object is in bin 1, the second in bin 4, the third in bin 2, the fourth in bin 3, the fifth in bin 5 and the sixth in bin 2. This representation for the bin-packing problem is illustrated in Fig. 42.10.

Genetic Operators

Procedure: Crossover [42.12].

Step 1. Select at random two crossing sites, delimiting the crossing section, in each of the two parents.
Step 2. Inject the contents of the crossing section of the first parent at the first crossing site of the second parent.
Step 3. Eliminate all objects now occurring twice from the bins they were members of in the second parent, so that the *old* membership of these objects gives way to the membership specified by the *new* injected bins. Consequently, some of the *old* groups coming from the second parent are altered.
Step 4. If necessary, adapt the resulting bins, according to the hard constraints and the cost function to optimize.
Step 5. Apply steps 2–4 to the two parents with their roles permuted to generate the second child.

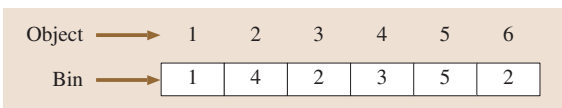

Fig. 42.10 Representation of membership of objects

42.2.5 Traveling-Salesman Problem

The traveling-salesman problem (TSP) is one of the most widely studied combinatorial optimization problems. Its statement is deceptively simple: a salesman seeks the shortest tour through n cities.

For example, a tour of a nine-city TSP

$$3-2-5-4-7-1-6-9-8$$

is simply represented as follows:

$$[3-2-5-4-7-1-6-9-8] \,.$$

This representation is also called a path representation or order representation. This representation may lead to illegal tours if the traditional one-point crossover operator is used, therefore many crossover operators have been investigated for it. Another method is the random keys representation. This representation encodes a solution with random numbers from (0,1). These values are used as sort keys to decode the solution.

For example, a chromosome for a nine-city problem may be

$$[0.23 \;\; 0.82 \;\; 0.45 \;\; 0.74 \;\; 0.87 \;\; 0.11 \;\; 0.56 \;\; 0.69 \;\; 0.78]$$

Where position i in the list represents city i. The random number in position i determines the visiting order of city i in a TSP tour. We sort the random keys in ascending order to get the following tour:

$$6-1-3-7-8-4-9-2-5$$

Genetic Approach

Representation. Permutation representation is perhaps the most natural representation of a TSP tour, where cities are listed in the order in which they are visited [42.13, 14].

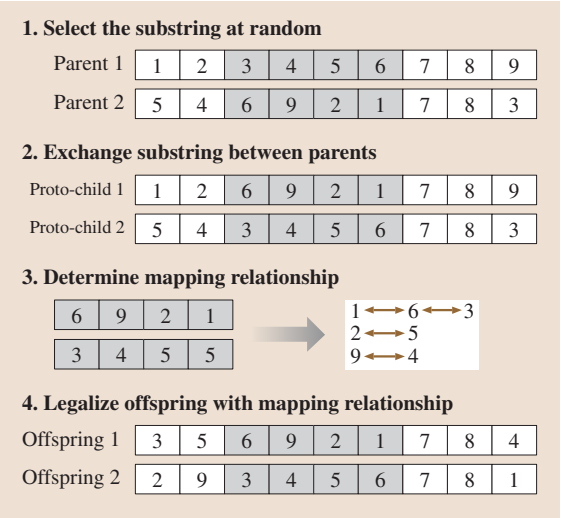

Fig. 42.11 Illustration of the PMX operator

Crossover Operators

Procedure: partial-mapped crossover (PMX) [42.14].

Step 1. Select two positions along the string uniformly at random.
Step 2. Exchange two substrings between parents to produce proto-children.
Step 3. Determine the mapping relationship between two mapping sections.

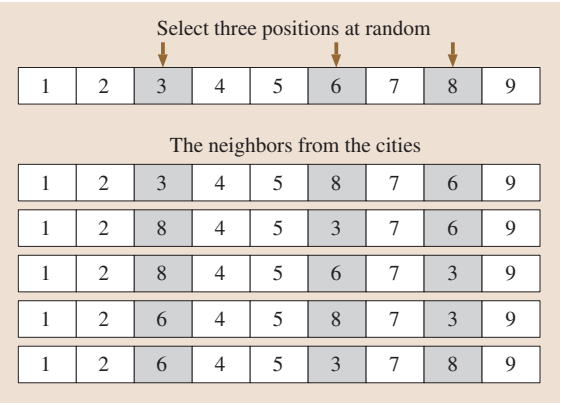

Fig. 42.12 Illustration of the heuristic mutation operator

Step 4. Legalize offspring with the *mapping relationship*.

The procedure is illustrated in Fig. 42.11.

Mutation Operators

Procedure: heuristic mutation [42.15, 16].

Step 1. Pick n genes at random.
Step 2. Generate neighbors according to all possible permutation of the selected genes.
Step 3. Evaluate all neighbors and select the best one as offspring.

The procedure is illustrated in Fig. 42.12.

42.3 Network Design Problems

Network design and routing are one of important issues in the building and expansion of computer networks. Many ideas and methods have been proposed and tested in the past two decades. Recently, there is an increasing interest in applying genetic algorithms to problems related to computer network [42.17].

42.3.1 Shortest-Path Problem

An undirected graph $G = (V, E)$ comprises a set of nodes $V = \{1, 2, \cdots, n\}$ and a set of edges $E \in V \times V$ connecting nodes in V. Corresponding to each edge, there are two nonnegative numbers c_{ij}^1 and c_{ij}^2 representing the cost and distance, or other items of interest, from node i to node j. A path from node i to node j is a sequence of edges (i, l), (l, m), $\cdots$, (k, j) from E in which no node appears more than once. A path can also be equivalently represented as a sequence of nodes $(i, l, m, \cdots, k, j)$. For the example given in Fig. 42.13, $(1, 4)$, $(4, 3)$, $(3, 5)$, $(5, 6)$ is a path from node 1 to node 6. The node representation is $(1, 4, 3, 5, 6)$.

Let 1 denote the initial node and n denote the end node of the path. Let x_{ij} be an indicator variable defined

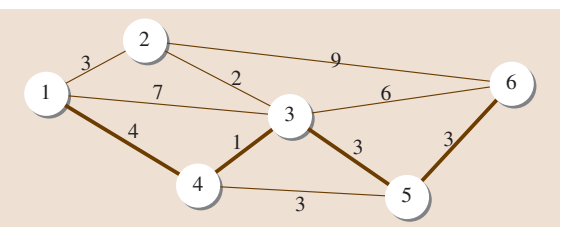

Fig. 42.13 Simple undirected graph with six nodes and 10 edges

as follows:

$$x_{ij} = \begin{cases} 1\,, & \text{if edge } (i,\, j) \text{ is included in the path} \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{42.34}$$

The bicriteria shortest-path problem can be formulated as follows:

$$\min z^1(x) = \sum_i \sum_j c_{ij}^1 x_{ij}\,, \tag{42.35}$$

$$\min z^2(x) = \sum_i \sum_j c_{ij}^2 x_{ij}\,, \tag{42.36}$$

$$\text{s.t.} \sum_j x_{ij} \le 2\,, \quad \forall i \in V\,, \tag{42.37}$$

$$\sum_{j \ne k} x_{ij} \ge x_{ik}\,, \quad \forall (i,k) \in E, \quad \forall i \in V \backslash \{1, n\}\,, \tag{42.38}$$

$$\sum_j x_{1j} = \sum_j x_{jn} = 1\,, \quad \forall i,\ j \in V\,, \tag{42.39}$$

$$x_{ij} = x_{ji}\,, \quad \forall (i,\, j) \in E\,, \tag{42.40}$$

$$0 \le x_{ij} \le 1\,, \quad \forall (i,\, j) \in E\,. \tag{42.41}$$

Genetic Approach

Priority-Based Encoding [42.18–20]. The position of a gene is used to represent a node and the value is used to represent the priority of the node for constructing a path among the candidates. The encoding method is denoted by priority-based encoding. The path corresponding to a given chromosome is generated by a sequential node-appending procedure, beginning from the specified node 1 and terminating at the specified node n.

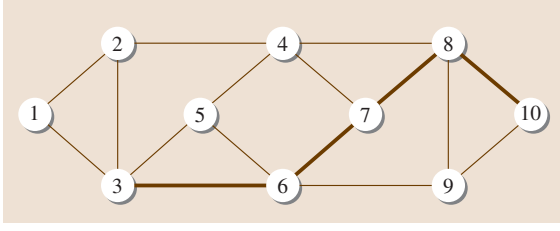

Fig. 42.14 Simple undirected graph with 10 nodes and 16 edges

Position: node ID	1	2	3	4	5	6	7	8	9	10
Value: priority	7	3	4	6	2	5	8	10	1	9

Fig. 42.15 Example of priority-based encoding

Consider the undirected graph shown in Fig. 42.14. Suppose we are going to find a path from node 1 to node 10. An encoding of the instance is given in Fig. 42.15. 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 the adjacency relation among nodes. The priorities of them are 3 and 4, respectively. 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 the next position and select the one with the highest priority among them. These steps are repeated until we obtain a complete path (1, 3, 6, 7, 8, 10).

For an n-node problem, let Ω be a set containing integers from 1 up to n, that is, $\Omega = \{1, 2, \ldots, n\}$, let p_i denote the priority for node i, which is a random integer exclusively from the set Ω. Priorities p_i of all nodes satisfy the following conditions:

$$p_i \ne p_j, \quad p_i,\ p_j \in \Omega, \quad i \ne j,\ i, j = 1, 2, \ldots, n \tag{42.42}$$

Then the priority-based encoding can be formally defined as

$$[p_1 \;\; p_2 \;\; \cdots \;\; p_n]\,.$$

Genetic Operators

Here the position-based crossover operator proposed by *Syswerda* is adopted [42.21]. It can be viewed as a kind of uniform crossover operator for integer permutation representation together with a pairing procedure, as shown in Fig. 42.16. Essentially, it takes some genes from one parent at random and fills the vacuum position with genes from the other parent using a left-to-right scan. The swap mutation operator is used here, which simply selects two positions at random and swaps their contents as shown in Fig. 42.17.

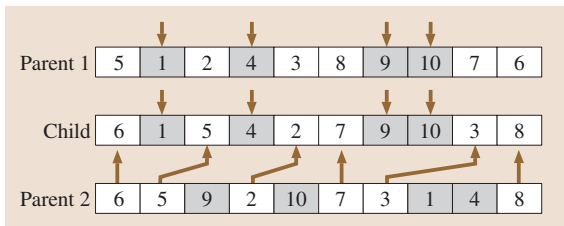

Fig. 42.16 Position-based crossover operator

42.3.2 Maximum-Flow Problem

There have been many applications of this problem in the real world. One of them is to determine the maximum flow through a pipeline network. Assume that oil should be shipped from the refinery (the source) to a storage facility (the sink) along arcs of the network. Each arc has a capacity which limits the amount of flow along that arc. Here, we want to determine the largest possible flow that can be sent from the refinery to the storage facility with the restriction that no arc (pipe) capacity can be exceeded. MXF has also been applied to some other applications such as: the problem of selecting sites for an electronic message-transmission system and dynamic flows in material-handling systems [42.22–24].

A mathematical formulation for the bin-packing problem is given by:

$$\max f \,, \tag{42.43}$$

$$\text{s.t.} \sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = \begin{cases} f, & \text{if} \quad i = 1 \\ 0, & \text{if} \quad i = 2, 3, \\ & \quad \ldots, m-1 \\ -f, & \text{if} \quad i = m \end{cases} , \tag{42.44}$$

$$0 \le x_{ij} \le u_{ij}, \quad i,\ j = 1, 2, \ldots, m \,, \tag{42.45}$$

where f is the amount of flow in the network from node 1 to node m and u_{ij} is arc capacities.

Genetic Approach

The priority-based encoding method is used to represent the chromosome. The chromosome here is represented by m-digit numbers that are generated randomly. Each number represents the priority of the node.

Crossover. As the first step in the crossover operation, we generate random numbers γ_k in the range [0, 1] ($k = 1, 2, \ldots$,*popSize*). Next, we select the chromosomes v_k to which the crossover operation will be applied. If $\gamma_k < p_C$ then the crossover operation will be applied to chromosome v_k.

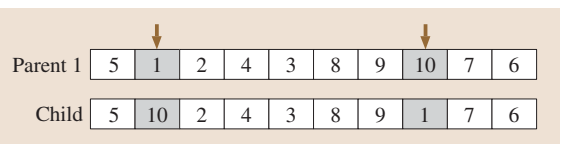

Fig. 42.17 Swap mutation operator

Mutation. Similarly, the first step in the mutation operation is to generate a random γ_r in the range [0, 1], ($r = 1, 2, \ldots$,*popSize*).

If $\gamma_r < p_M$ then the chromosome v_k $(l = (r/m +$ 1)) is chosen for the mutation operation.

42.3.3 Minimum-Cost-Flow Problem

The minimum-cost-flow problem (MCF) is known as a useful type of network optimization problem. It consists of finding the minimum-cost flows in the networks. For this problem, we are given a directed network $G = (X,\ A)$ in which each arc connecting node i and j in the network is associated with a cost c_{ij} and a capacity u_{ij}. A feasible solution to the MCF problem should satisfy two constraints. First, the flow through each arc should satisfy the capacity constraint. Second, the conservation of flow in all nodes should also be preserved. The conservation of flows here means that the flow into a node must equal the flow out of the node. The common objective is to determine the feasible network flow that minimizes the total cost.

A mathematical formulation for the bin-packing problem is given by:

$$\min z = \sum_{i=1}^{m} \sum_{j=1}^{m} c_{ij} x_{ij} \,, \tag{42.46}$$

$$\text{s.t.} \sum_{j=1}^{m} x_{ij} - \sum_{k=1}^{m} x_{ki} = b_i, i = 1, \ldots, m \tag{42.47}$$

$$x_{ij} \ge 0, i, j = 1, \ldots, m \,, \tag{42.48}$$

where x_{ij} is the flow through an arc and c_{ij} is the unit shipping cost along the arc. Equation (42.46) is called the flow-conservation or Kirchhoff equation and indicates that flow may be neither created nor destroyed in the network.

Genetic Approach

Representation. The chromosome here is represented by m-digit numbers generated randomly. Each number represents the priority of the node respectively.

Crossover. The crossover is done by selecting two chromosome randomly. We use the partially matched crossover (PMX) method for the crossover operation.

Mutation. Mutation here is done by selecting a chromosome at random. Two bit positions of the chromosome are exchanged.

42.3.4 Centralized Network Design

Consider a complete, undirected graph $G = (V,\ E)$, let $V = \{1, 2, \cdots, n\}$ be the set of nodes representing terminals. Denote the central site or *root* node as node 1, and let $E = \{(i, j)|i, j \in V\}$ be the set of edges representing all possible telecommunication wiring. For a subset of nodes $S(\subseteq V)$, define $E(S) = \{(i, j)|i, j \in S\}$ as the edges whose endpoints are both in S. Define the following binary decision variables for all edges $(i, j) \in E$:

$$x_{ij} = \begin{cases} 1\,, & \text{if edge } (i, j) \text{ is selected} \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{42.49}$$

Let c_{ij} be the fixed cost with respect to edge (i, j) in the solution, and suppose that d_i represents the demand at each node $i \in V$, where by convention the demand of the root node is $d_1 = 0$. Let $d(S)$, $S \subseteq V$ denote the sum of the demands of nodes of S. The subtree capacity is denoted with κ. The centralized network design problem can be formulated as follows [42.10]:

$$\min z = \sum_{i=1}^{n-1}\sum_{j=2}^{n} c_{ij}x_{ij}\,, \tag{42.50}$$

$$s.t. \sum_{i=1}^{n-1}\sum_{j=2}^{n} x_{ij} = 2(n-1)\,, \tag{42.51}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \le 2[|S| - \lambda(S)]\,,$$

$$S \subseteq V\backslash\{1\}\,, |S| \ge 2\,, \tag{42.52}$$

$$\sum_{i\in U}\sum_{j\in U} x_{ij} \le 2(|U| - 1)\,,\quad U \subset V\,,$$

$$|U| \ge 2\,,\quad \{1\} \in U\,, \tag{42.53}$$

$$x_{ij} = 0 \text{ or } 1\,,\quad i = 1, 2, \ldots, n-1,$$

$$j = 2, 3, \ldots, n\,. \tag{42.54}$$

Equality (42.51) is true of all spanning trees: a tree with n nodes must have $n-1$ edges. Inequality (42.53) is a standard inequality for spanning trees: if more than $|U|-1$ edges connect the nodes of a subset U, then the set U must contain a cycle. The parameter $\lambda(S)$ refers to the *bin-packing number* of set S, namely, the number of bins of size κ needed to pack the nodes of items of size d_i for all $i \in S$. These constraints are similar to those for inequality (42.53), except that they reflect the capacity constraint: if the set S does not contain the root node, then the nodes of S must be contained in at least $\lambda(S)$ different subtrees of the root.

Up to now, all heuristic algorithms for this problem are only focused on how to deal with the constraints to make the problem simpler to solve. In the cutting plane algorithms or branch-bound algorithm, the network topology of the problem are usually neglected. As a result, it leads in an exponential explosion of constraints.

In Fig. 42.18, node ID is the node number based on the depth-first search (DFS) and the degree at node ID is the number of connecting nodes.

Genetic Approach

To solve the centralized network design problem by using a genetic algorithm, a tree-based permutation encoding method is adopted to encode the candidate solutions, as illustrated in Fig. 42.18.

42.3.5 Multistage Process Planning

The multistage process planning (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 are 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 several of these criteria.

For an n-stage MPP problem, let s_k be some state at stage k, $D_k(s_k)$ be the set of possible states to be chosen at stage k, $k = 1, 2, \ldots, n$, x_k be the decision variable to determine which state to choose at stage k; obviously $x_k \in D_k(s_k)$, $k = 1, 2, \ldots, n$. Then the MPP problem can be formulated as follows:

$$\min_{\substack{x_k\in D_k(s_k)\\ k=1,2,\ldots,n}} V(x_1,\ x_2, \cdots,\ x_n) = \sum_{k=1}^{n} v_k(s_k,\ x_k)\,, \tag{42.55}$$

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 number such as cost, time, or distance.

Genetic Approach

Representation. The MPP solution can be concisely encoded in a state permutation format by concatenating all the set states of stages. This state permutation encoding has a one-to-one mapping for the MPP problem. The probability of randomly producing a process

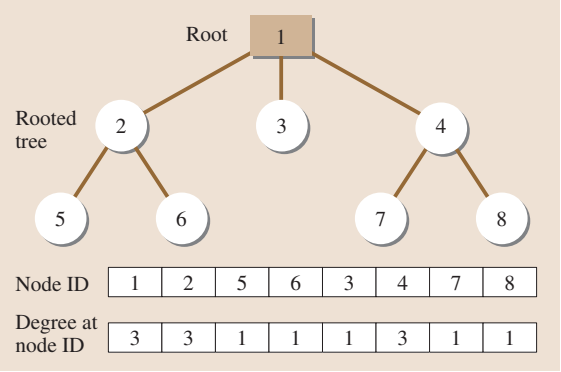

Fig. 42.18 Rooted tree and its tree-based permutation

planning is definitely 1. It is also easy to decode and evaluate. As to the initial population for an n-stage MPP problem, each individual is a permutation with $n-1$ integers whereas the integers are generated randomly with the number of all possible states in the corresponding stage.

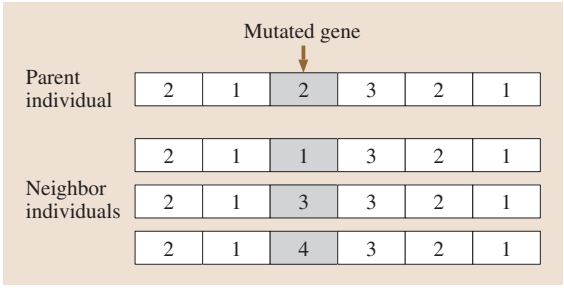

Fig. 42.19 Mutation with neighborhood search

Genetic Operation. In Zhou and Gen's method, only the mutation operation was adopted because it is easy to hybrid the neighborhood search technique to produce more adapted offspring. This hybrid mutation operation provides a great chance to evolve to the optimal solution. Figure 42.19 shows an example for this mutation operation with a neighborhood search technique supposing that the gene is at stage 3 and the number of possible states to be chosen is 4.

42.4 Scheduling Problems

Scheduling problems exist almost everywhere in real-world situations, especially in the industrial engineering world. Many scheduling problems from manufacturing industries are quite complex in nature and very difficult to solve by conventional optimization techniques.

42.4.1 Flow-Shop Sequencing Problem

The flow-shop sequencing problem is generally described as follows: there are m machines and n jobs, each job consists of m operations, and each operation requires a different machine. n jobs have to be processed in the same sequence on m machines. The processing time of job i on machine j is given by t_{ij} $(i = 1, \ldots, n;\ j = 1, \ldots, m)$. The objective is to find the sequence of jobs minimizing the maximum flow time, which is called makespan.

Heuristics for General m-Machine Problems

Genetic algorithms have been successfully applied to solve flow-shop problems. We describe Gen, Tsujimura, and Kubota's approach.

Representation. Because the flow-shop problem is essentially a permutation schedule problem [42.25–27], we can use the permutation of jobs as the representation scheme of chromosome, which is the natural representation for a sequencing problem. For example, let the k-th chromosome be

$$v_k = [3\ \ 2\ 4\ \ 1]\,,$$

meaning that the jobs sequence is $j_3,\ j_2,\ j_4,\ j_1$.

Crossover and Mutation. Here, Goldberg's PMX is used. Mutation is designed to perform random exchange; that is, it selects two genes randomly in a chromosome and exchanges their positions. An example is given in Fig. 42.20.

42.4.2 Job-Shop Scheduling

In the job-shop scheduling problem, we are given a set of jobs and a set of machines. Each machine can handle at most one job at a time. Each job consists of a chain

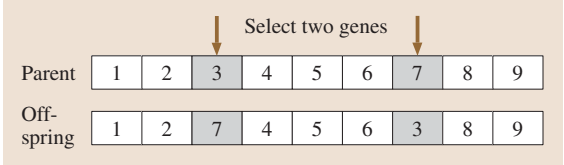

Fig. 42.20 Swap mutation

of operations, each of which needs to be processed during an uninterrupted time period of a given length on a given machine. The purpose is to find a schedule, that is, an allocation of the operations to time intervals on the machines, which has a minimum duration required to complete all jobs [42.25].

Adapted Genetic Operators

During the past two decade, various crossover operators have been proposed for literal permutation encodings, such as partial-mapped crossover (PMX), order crossover (OX), cycle crossover (CX), etc.

Partial-Mapped Crossover (PMX). PMX is explained in the previous section.

Order Crossover (OX). Order crossover was proposed by Davis. OX has the following major steps [42.14]:

Step 1. Select a substring from one parent at random.
Step 2. Produce a proto-child by copying the substring into the corresponding positions as they are in the parent.
Step 3. Delete all the symbols from the second parent that are already in the substring. The resulted sequence contains the symbols the proto-child needs.
Step 4. Place the symbols into the unfixed positions of the proto-child from left to right according to the order of the sequence to produce an offspring.

Cycle Crossover (CX). Cycle crossover was proposed by *Oliver* et al.. CX works as follows [42.25]:

Step 1. Find the cycle which is defined by the corresponding positions of symbols between parents.
Step 2. Copy the symbols in the cycle to a child with the corresponding positions of one parent.
Step 3. Determine the remaining symbols for the child by deleting those symbols which are already in the cycle from the other parent.
Step 4. Fill the child with the remaining symbols.

Mutation. It is relatively easy to make some mutation operators for the permutation representation. During the last decade, several mutation operators have been proposed for permutation representation, such as inversion, insertion, displacement, reciprocal exchange mutation, and shift mutation [42.9]. *Inversion mutation* selects two positions within a chromosome at random and then inverts the substring between these two positions. *Insertion mutation* selects a gene at random and inserts it in a random position.

42.4.3 Resource-Constrained Projected Scheduling Problem

The problem of scheduling activities under resource and precedence restrictions with the objective of minimizing the project duration is referred to as the resource-constrained project scheduling problem in the literature [42.25, 28].

The problem can be stated mathematically as follows:

$$\min\ t_n\ , \tag{42.56}$$

$$s.t.,\ t_j - t_i \geq d_i\ , \quad \forall j \in S_i\ , \tag{42.57}$$

$$\sum_{t_i \in A_{t_i}} r_{ik} \leq b_k, \quad k = 1, 2, \ldots, m\ , \tag{42.58}$$

$$t_i \geq 0, \quad i = 1, 2, \ldots, n\ , \tag{42.59}$$

where t_i is the starting time of activity i, d_i the duration (processing time) of activity i, S_i the set of successors of activity i, r_{ik} the amount of resource k required by activity i, b_k the total availability of resource k, A_{ti} the set of activities in process at time t_i, and m the number of different resource types. Activities 1 and n are dummy activities which mark the beginning and end of the project. The objective is to minimize the total project duration.

A. Priority-Based Encoding

For this problem, priority-based encoding is used; it is explained in the previous section.

B. Genetic Operators

Position-Based Crossover. The position-based crossover operator is used. This crossover is explained in the previous section.

Swap Mutation. The swap mutation operator was used here, which simply selects two positions at random and swaps their contents, as shown in Fig. 42.21.

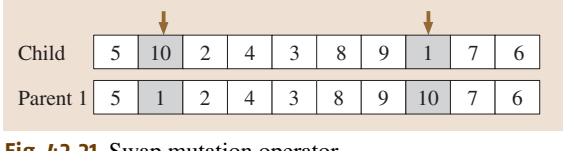

Fig. 42.21 Swap mutation operator

42.4.4 Multiprocessor Scheduling

The multiprocessor scheduling is to assign n tasks to m processors in such a way that precedence constraints are maintained, and to determine the start and finish times of each task with the objective of minimizing the completion time. There is a paper which deals with real-time tasks [42.29]. However, here we introduce an algorithm concerned with general tasks. The mathematical formulation of the problem is given as

$$\min[\max_j(x_j y_{ij})]\,, \tag{42.60}$$

$$\text{s. t.} x_k - x_j \geq p_k\,, \quad T_j \prec T_k\,, \tag{42.61}$$

$$\sum_{j=1}^{n} p_j y_{ij} \leq t_{\max}\,, \quad i = 1, \ldots, n\,, \tag{42.62}$$

$$\sum_{i=1}^{m} y_{ij} = 1\,, \quad j = 1, \ldots, m\,, \tag{42.63}$$

$$y_{ij} = 0 \;\; or \;\; 1\,, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n\,, \tag{42.64}$$

where

$$y_{ij} = \begin{cases} 1\,, & \text{if task } T_j \text{ is assigned to processor } P_i \\ 0\,, & \text{otherwise}\,, \end{cases} \tag{42.65}$$

and where $t_{\max} = \max_i(t_i)$, x_j is the completion time of task T_j, p_j is the processing time of task T_j, t_i is the time required to process all tasks assigned to process P_i, and $\prec$ represents a precedence relation; a precedence relation between tasks, $T_j \prec T_k$, means that T_k precedes T_j.

Genetic Algorithm for MSP

For the chromosome representation scheme and genetic operations, we adopt the concept of the height function [42.10], which considers precedence relations among the tasks in the implementation of a genetic algorithm.

Height Function. To facilitate the generation of the schedule and the construction of the genetic operators, we define the *height* of each task in the task graph as

$$\text{height}(T_i) = \begin{cases} 0 & \text{if pre}(T_i) = \phi \\ 1 + \max\limits_{T_j \in \text{pre}(T_i)} [\text{height}(T_j)]\,, & \text{otherwise} \end{cases} \tag{42.66}$$

$$\begin{aligned} \text{height}'(T_j) = {} & \text{rand} \in \{\max[\text{height}(T_i)] + 1, \\ & \min[\text{height}'(T_k)] - 1\} \text{ over all } T_i \in \text{pre } T_j \\ & \text{and } T_k \in \text{suc}\,(T_j)\,, \end{aligned} \tag{42.67}$$

where $\text{pre}(T_j)$ is the set of predecessors of T_j and $\text{suc}(T_j)$ is the set of successors of T_j.

Representation. The chromosome representation used here is based on the schedule of the tasks in each processor. The representation of the schedule for genetic algorithms must accommodate the precedence relations between the computational tasks.

Genetic Operators. The function of the genetic operators is to create new search nodes based on the current population of search nodes. New search nodes are typically constructed by combining or rearranging parts of the old search nodes.

Operation 1. Operation 1 is performed in the following steps

Step 1. Generate a random number c from the range $[1, \max(\text{height}')]$.
Step 2. Place the cut-point at each processor in such a way that the tasks' height$'$ before the cut-point is less than c and more than or equal to c after the cut-point.
Step 3. Exchange the second partial schedules.

Operation 2. Operation 2 is performed in the following steps

Step 1. Generate a random number c from the range $[1, \max(\text{height}')]$.
Step 2. At each processor, pick all tasks whose height$'$ is c.
Step 3. Replace the position of all tasks randomly.

42.5 Reliability Design Problem

Reliability optimization appeared in the late 1940s and was first applied to communication and transportation systems. Much of the early work was confined to the analysis of certain performance aspects of systems. One

goal of the reliability engineer is to find the best way to increase system reliability. The reliability of a system can be defined as the probability that the system has operated successfully over a specified interval of time under stated conditions.

42.5.1 Simple Genetic Algorithm for Reliability Optimization

The problem is to maximize the system reliability subject to three nonlinear constraints with parallel redundant units in subsystems that are subject to A failures, which occur when the entire subsystem is subjected to the failure condition. It can be mathematically stated as follows:

$$\max R(m) = \prod_{i=1}^{3} \left\{ 1 - [1 - (1 - q_{i1})^{m_i+1}] - \sum_{u=2}^{4} (q_{iu})^{m_i+1} \right\} , \tag{42.68}$$

$$\text{s.t.} \quad G_1(m) = (m_1+3)^2 + (m_2)^2 + (m_3)^2 \le 51 , \tag{42.69}$$

$$G_2(m) = 20 \sum_{i=1}^{3} \left[m_i + \exp(-m_i) \right] \ge 120 , \tag{42.70}$$

$$G_3(m) = 20 \sum_{i=1}^{3} \left[m_i \exp(-m_i/4) \right] \ge 65 , \tag{42.71}$$

$$1 \le m_1 \le 4, \quad 1 \le m_2, \quad m_3 \le 7 , \tag{42.72}$$

$$m_i \ge 0 : \text{integer}, \quad i = 1, 2, 3 , \tag{42.73}$$

Table 42.1 Failure modes and probabilities in each subsystem

Subsystem i	Failure modes $s_i = 4,\ h_i = 1$	Failure probability q_{iu}
1	O	0.01
	A	0.05
	A	0.10
	A	0.18
2	O	0.08
	A	0.02
	A	0.15
	A	0.12
3	O	0.04
	A	0.05
	A	0.20
	A	0.10

where $m = (m_1\, m_2\, m_3)$. The subsystems are subject to four failure modes ($s_i = 4$) with one O failure ($h_i = 1$) and three A failures, for $i = 1, 2, 3$. For each subsystem the failure probability is shown in Table 42.1.

Genetic Approach

Representation. The integer value of each variable m_i is represented as a binary string. The length of the string depends on the upper bound u_i of the redundant units. For instance, when the upper bound u_i equals 4, we need three binary bits to represent m_i. In this example, the upper bounds of the redundant units in each subsystem are $u_1 = 4$, $u_2 = 7$, $u_3 = 7$, so each decision variable m_i needs three binary bits. This means that a total of nine bits are required. If $m_1 = 2$, $m_2 = 3$, and $m_3 = 3$, we have the following chromosome:

$$\begin{aligned} \mathrm{v} &= [x_{33}\ \ x_{32}\ \ x_{31}\ \ x_{23}\ \ x_{22}\ \ x_{21}\ \ x_{13}\ \ x_{12}\ \ x_{11}] \\ &= [0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ ] \end{aligned}$$

where x_{ij} is the symbol for the j-th binary bit of variable m_i.

Crossover. One-cut-point crossover is used here.

Mutation. Mutation is performed on a bit-by-bit basis.

42.5.2 Reliability Design with Redundant Unit and Alternatives

Gen, *Yokota*, *Ida* and *Taguchi* further extended their work to the reliability optimization problem by considering both redundant units and alternative design [42.19, 30, 31].

The example used here was firstly given by *Fyffe* et al.as follows:

$$\max R(\boldsymbol{m}, \boldsymbol{\alpha}) = \prod_{i=1}^{14} \left\{ 1 - \left[1 - R_i(\alpha_i) \right]^{m_i} \right\} , \tag{42.74}$$

$$\text{s.t.} \quad G_1(\boldsymbol{m}, \boldsymbol{\alpha}) = \sum_{i=1}^{14} c_i(\alpha_i) m_i \le 130 , \tag{42.75}$$

$$G_2(\boldsymbol{m}, \boldsymbol{\alpha}) = \sum_{i=1}^{14} w_i(\alpha_i) m_i \le 170 , \tag{42.76}$$

$$1 \le m_i \le u_i , \quad \forall i , \tag{42.77}$$

$$1 \le \alpha_i \le \beta_i , \quad \forall i , \tag{42.78}$$

$$m_i , \ \alpha_i \ge 0 : \ \text{integer } \forall i , \tag{42.79}$$

where α_i represents the design alternative available for the i-th subsystem, m_i represents the identical units used

in redundancy for the ith subsystem, u_i is the upper bound of the redundant units for the i-th subsystem, and β_i is the upper bound of alternative design for the i-th subsystem.

Genetic Approach

Representation. The representation can be written as follows:

$$\boldsymbol{v}_k = [\ (\alpha_{k1}\ ,\ m_{k1})\ (\alpha_{k2}\ ,\ m_{k2})\ \cdots\ (\alpha_{k14}\ ,\ m_{k14})\ ]\ ,$$

where α_{ki} is a design alternative, m_{ki} is a redundant unit, the subscript k is the index of chromosome.

Crossover. The uniform crossover operator given by Syswerda is used here, which has been shown to be superior to traditional crossover strategies for combinatorial problem. Uniform crossover firstly generates a random crossover mask and then exchanges relative genes between parents according to the mask. A crossover mask is simply a binary string with the same size of chromosome.

42.5.3 Network Reliability Design

A communication network can be represented by an undirected graph $G = (V, E)$, in which the nodes V and edges E represent computer sites and communication cables, respectively. A graph G is connected if there is at least one path between ever pair of nodes i and j, which minimally requires a spanning tree with $(n-1)$ edges. The following notations are defined to describe the optimal design problem of all-terminal reliable networks: n is the number of nodes, $x_{ij} \in (0, 1)$ is the decision variable representing the edge between node i and node j, $\boldsymbol{x}(= \{x_{12}, x_{13}, \cdots, x_{n-1,n}\})$ is a topology architecture for the network design, x^* is the best solution found so far, p is the edge reliability for all edges, q is the edge unreliability for all edges (i. e., $p+q=1$), $R(x)$ is the all-terminal reliability of the network design x, $R_{\min}$ is the network reliability requirement, $R_U(x)$ is the upper bound on the reliability of the candidate network, c_{ij} is the cost of the edge between node i and node j, $c_{\max}$ is the maximum value of c_{ij}, δ has the value of 1 if $R(x) < R_{\min}$ and is 0 otherwise, E' is a set of operational edges ($E' \subseteq E$), Ω is all operational states (E'). The optimal design of network can be represented as follows [42.10, 32]:

$$\min Z(\boldsymbol{x}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_{ij}\ , \tag{42.80}$$

$$\text{s. t.} \quad R(\boldsymbol{x}) \geq R_{\min}\ . \tag{42.81}$$

Genetic Approach

Representation. A genetic algorithm lends itself to this problem because each network design $\boldsymbol{x}$ is easily formed into a binary string which can be used as a chromosome for genetic algorithms. Each element of the chromosome represents a possible edge in the network design problem, so there are $n \times (n-1)/2$ string components in each candidate architecture Z.

Crossover. The one-cut-point crossover operation is used.

Mutation. The bit-flip mutation operation is employed, performed on a bit-by-bit basis.

42.5.4 Tree-Based Network Topology Design

Consider a local-area network (LAN) that connects m users (stations). Also, we assume the $n \times n$ service center topology matrix X_1, which represents the connection between service centers. An element $\boldsymbol{x}_{1ij}$ is represented as

$$\boldsymbol{x}_{1ij} = \begin{cases} 1\ , & \text{if the centers } i \text{ and } j \text{ are connected} \\ 0\ , & \text{otherwise}\ . \end{cases} \tag{42.82}$$

Assume that the LAN is partitioned into n segments (service centers or clusters). The users are distributed over those n service centers. The $n \times m$ clustering matrix $\boldsymbol{X}_2$ specifies which user belongs to which center. Thus

$$\boldsymbol{x}_{2ij} = \begin{cases} 1\ , & \text{if user } j \text{ belongs to center } i \\ 0\ , & \text{otherwise}\ . \end{cases} \tag{42.83}$$

A user can only belong to one center; thus, $\forall j = 1, 2, \ldots, m$, $\sum_{i=1}^{n} x_{2ij} = 1$. We define an $n \times (n+m)$ matrix $\boldsymbol{X}$ called the spanning tree matrix ($[\boldsymbol{X}_1\ \ \boldsymbol{X}_2]$). The bicriteria LAN topology design problem can be formulated as the following nonlinear 0–1 programming model [42.10, 33, 34]:

$$\max R(\boldsymbol{X})\ , \tag{42.84}$$

$$\min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{1ij} x_{1ij} + \sum_{i=1}^{n} \sum_{j=1}^{m} w_{2ij} x_{2ij}\ , \tag{42.85}$$

$$\text{s. t.} \sum_{j=1}^{m} x_{1ij} \leq g_i\ ,\quad i = 1, 2, \ldots, n\ , \tag{42.86}$$

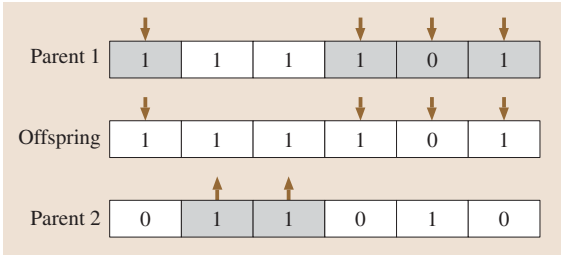

Fig. 42.22 Uniform crossover operator

$$\sum_{i=1}^{n} x_{2ij} = 1\,, \quad j = 1, 2, \ldots, m\,, \tag{42.87}$$

where $R(\boldsymbol{X})$ is the network reliability, w_{1ij} is the weight of the link between the centers i and j, w_{2ij} is the weight of the link between the center i and the user j, g_i is the maximum number that can connect to the center i.

Genetic Approach

Representation. We can easily construct an encoding as follows:

procedure: Encoding of Prüfer number

Step 1. Let node i be the smallest labeled leaf node in a labeled tree T.
Step 2. Let j be the first digit in the encoding as the node j incident to node i is uniquely determined. The encoding is built by appending digits from left to right.
Step 3. Remove node i and the link from i to j; thus we have a tree with $k-1$ nodes.
Step 4. Repeat the above steps until one link is left. We produce a Prüfer number or an encoding with $k-2$ digits between 1 and k inclusive.

Crossover. Uniform crossover is used. This type of crossover is accomplished by selecting two parent solutions and randomly taking a component from one parent to form the corresponding component of the offspring Fig. 42.22.

Mutation. Swap mutation is used, as explained in the previous section.

42.6 Logistic Network Problems

The transportation problem is a basic model in the logistic networks. Many scholars have since refined and extended the basic transportation model to include not only the determination of optimum transportation patterns but also the analysis of production scheduling problems, transshipment problems, and assignment problems.

42.6.1 Linear Transportation Problem

The linear transportation problem (LTP) involves the shipment of some homogeneous commodity from various origins or sources of supply to a set of destinations, each demanding specified levels of the commodity. The usual objective function is to minimize the total transportation cost or total weighted distance or to maximize the total profit contribution from the allocation [42.35].

Given m origins and n destinations, the transportation problem can be formulated as a linear programming model:

$$\min z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}\,, \tag{42.88}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} \le a_i, \quad i = 1, 2, \ldots, m\,, \tag{42.89}$$

$$\sum_{i=1}^{m} x_{ij} \ge b_j, \quad j = 1, 2, \ldots, n\,, \tag{42.90}$$

$$x_{ij} \ge 0, \quad \text{for all } i \text{ and } j\,, \tag{42.91}$$

where x_{ij} is the amount of units shipped from origin i to destination j; c_{ij} is the cost of shipping one unit from source i to destination j; a_i is the number of units available at origin i; and b_j is the number of units demanded at destination j.

Genetic Approach

Representation. Perhaps the matrix is the most natural representation of a solution for the transportation problem. The allocation matrix for the transportation problem can be written as follows:

$$\boldsymbol{X}_p = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \tag{42.92}$$

where $\boldsymbol{X}_p$ denotes the p-th chromosome and x_{ij} is the corresponding decision variable.

Crossover. Assume that two matrices $\boldsymbol{X}_1 = (x_{ij}^1)$ and $\boldsymbol{X}_2 = (x_{ij}^2)$ are selected as parents for the crossover operation. The crossover is performed in the following three main steps:

Step 1. Create two temporary matrices $\boldsymbol{D} = (d_{ij})$ and $\boldsymbol{R} = (r_{ij})$ as follows: $d_{ij} = \left[\left(x_{ij}^1 + x_{ij}^2\right)/2\right]$ and $r_{ij} = \left(x_{ij}^1 + x_{ij}^2\right) \mod 2$.

Step 2. Divide matrix $\boldsymbol{R}$ into two matrices $\boldsymbol{R}^1 = \left(r_{ij}^1\right)$ and $\boldsymbol{R}^2 = \left(r_{ij}^2\right)$ such that: $\boldsymbol{R} = \boldsymbol{R}^1 + \boldsymbol{R}^2$

Step 3. Then we produce two offspring of $\boldsymbol{X}_1'$ and $\boldsymbol{X}_2'$ as follows: $\boldsymbol{X}_1' = \boldsymbol{D} + \boldsymbol{R}^1$ and $\boldsymbol{X}_2' = \boldsymbol{D} + \boldsymbol{R}^2$

Mutation. The mutation is performed in following three main steps:

Step 1. Make a submatrix from the parent matrix. Randomly select $\{i_1, \cdots, i_p\}$ rows and $\{j_1, \cdots, j_q\}$ columns to create a $(p * q)$ submatrix $\boldsymbol{Y} = (y_{ij})$, where $\{i_1, \cdots, i_p\}$ is a proper subset of $\{1, 2, \ldots, m\}$ and $2 \le p \le m$, $\{j_1, \cdots, j_q\}$ is a proper subset of $\{1, 2, \ldots, n\}$ and $2 \le q \le n$, and y_{ij} takes the value of the element in the crossing position of selected row i and column j in the parent matrix.

Step 2. Reallocate commodity for the submatrix. The available amount of commodity a_i^y and the demands b_j^y for the submatrix are determined as follows:

$$a_i^y = \sum_{j \in \{j_1, \cdots, j_q\}} y_{ij}\,, \quad i = i_1, i_2, \cdots, i_p\,, \tag{42.93}$$

$$b_j^y = \sum_{i \in \{i_1, \cdots, i_p\}}^{y_{ij}}\,, \quad j = j_1, j_2, \cdots, j_q\,. \tag{42.94}$$

Step 3. Replace appropriate elements of the parent matrix by new elements from the reallocated submatrix $\boldsymbol{Y}$.

Spanning Tree-Based Approach. Transportation problems (TP) as a special type of network problem have a special data structure characterized as a transportation graph in their solutions. The spanning tree-based GA incorporating this data structure of TP was proposed by Gen and Li. This GA utilized the Prüfer number encoding based on a spanning tree, which is adopted because it is capable of representing all possible trees. Using the Prüfer number representation the memory only requires $m+n-2$ entries for a chromosome in the TP. Transportation problems have separable sets of nodes for plants and warehouses. From this point, *Gen* and *Cheng* designed the criterion for feasibility of the chromosome. The proposed spanning tree-based GA can find the optimal or near-optimal solution for transportation problems in the solution space [42.10].

42.6.2 Multiobjective Transportation Problem

In the transportation problem, multiple objectives are required in practical situations, such as minimizing transportation cost, minimizing the average shipping time to priority customers, maximizing production using a given process, minimizing fuel consumption, and so on. The traditional multiobjective transportation problem (mTP) with m plants and n warehouses can be formulated as

$$\min z_q = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^q x_{ij} \quad q = 1, 2, \ldots., Q\,, \tag{42.95}$$

$$\text{s. t.} \sum_{j=1}^{n} x_{ij} \le a_i\,, \quad i = 1, 2, \ldots, m\,, \tag{42.96}$$

$$\sum_{i=1}^{m} x_{ij} \ge b_j, \quad j = 1, 2, \ldots, n\,, \tag{42.97}$$

$$x_{ij} \ge 0\,, \quad \forall i,\ j\,, \tag{42.98}$$

where q means the q-th objective function.

Spanning Tree-based GA for Multi-objective TP

The Pareto optimal solutions are usually characterized as the solutions of the multiobjective programming problem [42.36, 37].

42.6.3 Bicriteria Transportation Problem with Fuzzy Coefficients

Consider the following two objectives: minimizing total transportation cost and minimizing total delivery time. Let $\tilde{c}_{ij}^1$ be the fuzzy data representing the transportation cost of shipping one unit from plant i to warehouse j, let $\tilde{c}_{ij}^2$ be the fuzzy data representing the delivery time of shipping one unit of the product from plant i to warehouse j, a_i be the number of units available at plant i,

and b_j be the number of units demanded at warehouse j. This problem with m plants and n warehouses can be formulated as [42.10]:

$$\min \tilde{z}_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} \tilde{c}_{ij}^1 x_{ij} \,, \tag{42.99}$$

$$\min \tilde{z}_2 = \sum_{i=1}^{m} \sum_{j=1}^{n} \tilde{c}_{ij}^2 x_{ij} \,, \tag{42.100}$$

$$\text{s. t.} \sum_{j=1}^{n} x_{ij} \le a_i \,, \quad i = 1, 2, \ldots, m \,, \tag{42.101}$$

$$\sum_{i=1}^{m} x_{ij} \ge b_j \,, \quad j = 1, 2, \ldots, n \,, \tag{42.102}$$

$$x_{ij} \ge 0 \,, \quad \forall i \,, \; j \,, \tag{42.103}$$

where x_{ij} is the unknown quantity to be transported from plant i to warehouse j.

Genetic Approach

The proposed genetic algorithm approach is based on spanning tree. In multicriteria optimization, we are interested in finding Pareto solutions. When the coefficients of objectives are represented with fuzzy numbers, the objective values become fuzzy numbers. Since a fuzzy number represents many possible real numbers, it is not easy to compare solutions to determine which is the Pareto solution. Fuzzy ranking techniques can help us to compare fuzzy numbers. In this approach, Pareto solutions are determined based on the ranked values of fuzzy objective functions, and genetic algorithms are used to search for Pareto solutions.

Representation. The spanning-tree encoding, the Prüfer number, is used to represent the candidate solution. The criterion for the solution's feasibility designed in the proposed spanning-tree-based GA is also employed.

Crossover. For simplicity one-point crossover is used.

Mutation. Inversion mutation and displacement mutation are used.

42.6.4 Supply-Chain Management (SCM) Network Design

Supply-chain management (SCM) aims to choose the subset of plants and distribution centers to be opened and to design the distribution network strategy that can satisfy all capacities and demand requirements imposed by customers with minimum cost. We formulate the problem by using the following mixed integer linear programming model (MILP) [42.28, 38–41]:

$$\min \sum_i \sum_j s_{ij} x_{ij} + \sum_j \sum_k t_{jk} y_{jk} + \sum_k \sum_l u_{kl} z_{kl} + \sum_j f_j w_j + \sum_k g_k z_k \tag{42.104}$$

$$\text{s. t.} \sum_j x_{ij} \le a_i, \; \forall i \,, \tag{42.105}$$

$$\sum_k y_{jk} \le b_j w_j \,, \; \forall j \,, \tag{42.106}$$

$$\sum_j w_j \le P \,, \tag{42.107}$$

$$\sum_l z_{kl} \le c_k z_k, \; \forall k \,, \tag{42.108}$$

$$\sum_k z_k \le W \,, \tag{42.109}$$

$$\sum_k z_{kl} \ge d_l, \; \forall l \,, \tag{42.110}$$

$$w_j, z_k = (0, 1) \,, \; \forall j, \; k \,, \tag{42.111}$$

$$x_{ij}, y_{jk}, z_{kl} \ge 0, \; \forall i, \; j, \; k, \; l \,, \tag{42.112}$$

where i is the number of suppliers, j is the number of plants, K is the number of distribution centers, L is the number of customers, a_i is the capacity of supplier i, b_i is the capacity of plant j, c_k is the capacity of distribution center k, d_l is the demand of customer l, s_{ij} is the unit cost of production in plant j using material from supplier i, t_{jk} is the unit cost of transportation from plant j to the distribution center k, u_{kl} is the unit cost of transportation from distribution k to customer l, f_j is the fixed cost for operating plant j, g_k is the fixed cost for operating distribution center k, W is an upper limit on the total number of distribution centers that can be opened and P is an upper limit on the total number of plants that can be opened.

Here, x_{ij} is the quantity produced at plant j using raw material from supplier i, y_{jk} is the amount shipped from plant j to distribution center k and z_{kl} is the amount shipped from distribution center k to customer l. w_j and z_k are defined as

$$w_j = \begin{cases} 1, & \text{if production takes place at plant } j \\ 0, & \text{otherwise} \,, \end{cases} \tag{42.113}$$

$$z_k = \begin{cases} 1, & \text{if distribution center } k \text{ is opened} \\ 0, & \text{otherwise} . \end{cases} \tag{42.114}$$

Genetic Approach

Crossover. The crossover is done by exchanging the information of two parents to provide a powerful exploration capability. We employ a one-cut-point crossover operation, which randomly selects one cut-point and exchanges the right parts of the two parents to generate offspring.

Mutation. Modifying one or more of the gene values of an existing individual, mutation creates a new individual to increase the variability of the population. We use inversion and displacement mutation operations.

42.7 Location and Allocation Problems

Location–allocation problems arise in many practical settings. The classical single location–allocation problem is to find the single location which minimizes the summed distance from some number of fixed points, representing customers with known locations.

42.7.1 Location–Allocation Problem

There are m facilities to be located, and n customers with known locations are to be allocated to the variable facilities. Each customer has the requirement q_j, $j = 1, 2, \ldots, n$, and each facility has the capacity b_i, $i = 1, 2, \ldots, m$. We need to find the locations of facilities and allocations of customers to facilities so that the total summed distance among the customers and their serving facilities is minimized Fig. 42.23. This problem is formulated mathematically as [42.9]:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} \sqrt{(x_i - u_j)^2 + (y_i - v_j)^2}\ z_{ij} \tag{42.115}$$

$$\text{s.t.} \sum_{j=1}^{n} q_j \cdot z_{ij} \le b_i , \quad i = 1, 2, \ldots, m , \tag{42.116}$$

$$\sum_{i=1}^{m} z_{ij} = 1 , \quad j = 1, 2, \ldots, n , \tag{42.117}$$

$$z_{ij} = 0 \text{ or } 1 , \ i = 1, 2, \ldots, m, \ j = 1, 2, \ldots, n , \tag{42.118}$$

where

$$(u_j, v_j) = \text{location of customer } j, \ j = 1, 2, \ldots, n , \tag{42.119}$$

$$(x_j, y_j) = \text{location of facility } i, \tag{42.120}$$

$$\text{decision variables } i = 1, 2, \ldots, m , \tag{42.121}$$

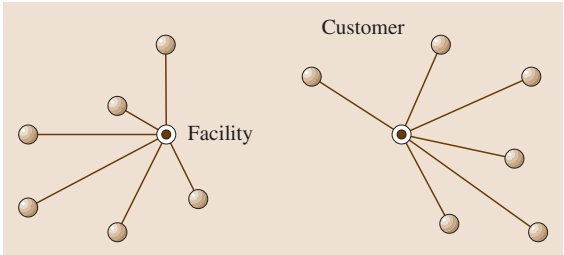

Fig. 42.23 Location–allocation problem

$z_{ij} = 0 - 1$ decision variable ,

$$z_{ij} = \begin{cases} 1, & \text{customer } j \text{ is served by facility } i \\ 0, & \text{otherwise} \end{cases} . \tag{42.122}$$

A. Genetic Approach

Representation. Since location variables are continuous, the float-value chromosome representation is used. A chromosome is given as follows:

$$\boldsymbol{c}^k = \left[\left(x_1^k, y_1^k\right) \left(x_2^k, y_2^k\right) \cdots \left(x_m^k, y_m^k\right) \right]$$

where $\left(x_i^k, y_i^k\right)$ is the location of the i-th facility in the k-th chromosome, $i = 1, 2, \ldots, m$.

Crossover. Two mating strategies are used: one is free mating, which selects two parents at random; another is dominating mating, which uses the fittest individual as a fixed parent and randomly selects another parent from the population pool. These two strategies are used alternatively in the evolutionary process.

Suppose two parents with the following chromosomes are selected to produce a child

$$\boldsymbol{c}^{k_1} = \left[\left(x_1^{k_1}, y_1^{k_1}\right) \left(x_2^{k_1}, y_2^{k_1}\right) \cdots \left(x_m^{k_1}, y_m^{k_1}\right) \right] ,$$

$$\boldsymbol{c}^{k_2} = \left[\left(x_1^{k_2}, y_1^{k_2}\right)\left(x_2^{k_2}, y_2^{k_2}\right) \cdots \left(x_m^{k_2}, y_m^{k_2}\right) \right] .$$

Only one child is allowed to be produced:

$$\begin{aligned} \boldsymbol{c} &= [(x_1, y_1)(x_2, y_2) \cdots (x_m, y_m)] , \\ x_i &= r_i \cdot x_i^{k_1} + (1 - r_i) \cdot x_i^{k_2} , \\ y_i &= r_i \cdot y_i^{k_1} + (1 - r_i) \cdot y_i^{k_2} . \end{aligned} \tag{42.123}$$

Mutation. Suppose the candidate chromosome to be mutated is as follows:

$$\boldsymbol{c}^k = \left[\left(x_1^k, y_1^k\right)\left(x_2^k, y_2^k\right) \cdots \left(x_m^k, y_m^k\right) \right]$$

then the chromosome of the child produced by subtle mutation $\boldsymbol{c} = [x_1, y_1, x_2, y_2, \cdots, x_m, y_m]$ is as follows:

$$\begin{aligned} x_i &= x_i^k + \text{random value in } [-\varepsilon, \varepsilon] , \\ y_i &= y_i^k + \text{random value in } [-\varepsilon, \varepsilon] . \end{aligned} \tag{42.124}$$

Table 42.2 Coordinates of Cooper and Rosing's example

Order number	X	Y	Order number	X	Y
1	5	9	16	53	8
2	5	24	17	1	34
3	5	48	18	33	8
4	13	4	19	3	26
5	12	19	20	17	9
6	13	39	21	53	20
7	28	37	22	24	17
8	21	45	23	40	22
9	25	50	24	22	41
10	31	9	25	7	13
11	39	2	26	5	17
12	39	16	27	39	3
13	45	22	28	50	50
14	41	30	29	16	40
15	49	31	30	22	45

B. Numerical Example

Cooper and *Rosing*'s examples are used to test the effectiveness of this method [42.42]. Cooper carefully constructed the front half data which contains three natural groups and Rosing increased the number of customers with random points. These examples provide a good benchmark to test the effectiveness of the proposed method because their global optimal solutions have already been found.

These examples include 30 customers whose location coordinates are shown in Table 42.2. Theirs is a common location–allocation problem where the requirements of the customers are treated as equal and the capacities of the facilities are assumed to be unlimited.

Both the alternative location-allocation (ALA) method and the hybrid evolutionary method (HEM) were applied to solve these examples. When using the ALA method, it was run to solve the same problem 40 times from randomly generated initial locations. The computed results are given in Table 42.3 [42.10]. In the table, the percent error was calculated by (actual value−optimal value)/optimal value ×100%.

42.7.2 Capacitated Plant Location Problem

The capacitated plant location problem (cPLP) is referred to as a fixed-charge problem to determine the locations of plants with minimal total cost, including production, shipping costs, and fixed costs where

Table 42.3 Comparison results of Cooper and Rosing's example

Problem n/m	Rosing's method optimal objective	ALA Best	ALA Percent error	HEM Best	HEM Percent error
15/2	214.281	219.2595	2.32	214.2843	0.0015
15/3	143.197	144.8724	1.17	143.2058	0.0061
15/4	113.568	115.4588	1.69	113.5887	0.0182
15/5	97.289	99.4237	2.19	97.5656	0.2843
15/6	81.264	84.0772	3.46	83.0065	2.14
30/2	447.728	450.3931	0.5952	447.73	0.0004
30/3	307.372	310.3160	0.9578	307.3743	0.0007
30/4	254.148	258.4713	1.7010	254.2246	0.0301
30/5	220.057	226.8971	3.1083	220.4335	0.1711
30/6	–	208.4301	3.4940	201.4031	0.0

the plants are located. In this case, m sources (or facility locations) produce a single commodity for n customers, each with demand of b_j ($j = 1, \ldots, n$) units. If a particular source i is opened (or facility is built), it has a fixed cost $d_i \geq 0$ and a production capacity $a_i \geq 0$ associated with it. There is also a positive cost c_{ij} for shipping a unit from source i to customer j. The problem is to determine the locations of the plants so that capacities are not exceeded and demands are met, all at a minimal total cost. The cPLP is a mixed integer program, as shown in the following [42.10]

$$\min z(x) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} d_i y_i \qquad (42.125)$$

$$\text{s.t.} \sum_{i=1}^{m} x_{ij} = b_j, \quad j = 1, 2, \ldots, n, \qquad (42.126)$$

$$\sum_{j=1}^{n} x_{ij} \leq a_i y_i, \quad i = 1, 2, \ldots, m, \qquad (42.127)$$

$$x_{ij} \geq 0, \quad \forall i, j, \qquad (42.128)$$

$$y_i = 0 \text{ or } 1, \quad i = 1, 2, \ldots, m. \qquad (42.129)$$

The variables are x_{ij} and y_i, which represent the amount shipped from plant i to warehouse j and whether a plant is open (or located) ($y_i = 1$) or closed ($y_i = 0$), respectively.

Spanning Tree-Based GA for Plant Location Problems

The spanning tree-based GA for the capacitated plant location problem is the same as that of the fixed-charge transportation problem except there is a different evaluation function in the evolutionary process.

42.7.3 Obstacle Location–Allocation Problem

There are n customers with known locations and m facilities to be built to supply some kind of services to all customers, for example, supplying materials or energy. There are also p obstacles representing some forbidden areas. The formulation of the mathematical model is based on the following assumptions:

- customer j has service demand q_j, $j = 1, 2, \ldots, n$,
- facility i has service capacity b_i, $i = 1, 2, \ldots, m$,
- each customer should be served by only one facility,
- new facilities should not be built within any obstacle,
- connecting paths between facilities and customers should not be allowed to pass through any of the obstacles.

The problem is to choose the best locations for facilities so that the sum of distances between customers and their serving facilities is minimal, as illustrated in Fig. 42.24. The obstacle location–allocation problem can be formulated as follows [42.9]:

$$\min f(\mathrm{D}, z) = \sum_{i=1}^{m}\sum_{j=1}^{n} t(\mathrm{D}_i, \mathrm{C}_j)\cdot z_{ij} \qquad (42.130)$$

$$\text{s.t.} \sum_{j=1}^{n} d_j z_{ij} \leq q_i, \quad i = 1, 2, \cdots, m, \qquad (42.131)$$

$$\sum_{i=1}^{m} z_{ij} = 1, \quad j = 1, 2, \cdots, n, \qquad (42.132)$$

$$\mathrm{D}_i = (x_i, y_i) \notin \mathrm{Q}_k, \quad i = 1, 2, \cdots, m, k = 1, 2, \cdots, q, \qquad (42.133)$$

$$(x_i, y_i) \in \mathrm{R}_\mathrm{T}, \quad i = 1, 2, \cdots, m, \qquad (42.134)$$

$$x_i, y_j \in R \quad i = 1, 2, \cdots, m \qquad (42.135)$$

$$z_{ij} = 1 \text{ or } 0, \quad i = 1, 2, \cdots, m, j = 1, 2, \cdots, n, \qquad (42.136)$$

where $\mathrm{C}_j = (u_j, v_j)$ is the location of the j-th customer, $\mathrm{D}_i = (x_i, y_i)$ is the decision variable, the location of the j-th distribution center DC_i should not fall within any of the obstacles, $t(\mathrm{D}_i,\mathrm{C}_j)$ is the shortest connecting path from the set of possible paths between the distribution center DC_i and the customer C_j which avoids all obstacles, R_T is the total area considered for the location and allocation problem and z_{ij} is a 0–1 decision variable; $z_{ij} = 1$ indicates that the j-th customer is served by DC_i, $z_{ij} = 0$ otherwise.

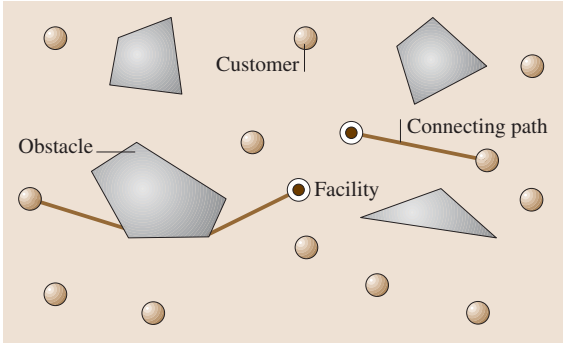

Fig. 42.24 Obstacle location–allocation problem

Hybrid Evolutionary Method

Since there are obstacles, the locations of the chromosome produced by initialization, crossover and mutation procedure may become infeasible. Generally, there are three kinds of methods to treat infeasible chromosomes. The first is to discard it, but according to the experience of other researchers this method may lead to very low efficiency. The second is to add a penalty to infeasible chromosomes. The third is to repair the infeasible chromosome according to the characteristics of the specified problem.

References

42.1 D. Fogel, A. Ghozeil: Using Fitness Distributions to Design More Efficient Evolutionary Computations, Proc. of the Third IEEE conference on Evolutionary Computation, Nagoya 1996, ed. by D. Fogel (IEEE Press, Nagoya 1996) 11–19

42.2 M. Gen, R. Cheng: Evolutionary network design: Hybrid genetic algorithms approach, Inter. J. Comp. Intell. & Appl. **3**, 357–380 (2003)

42.3 Y. Yun, M. Gen (Eds.): Adaptive hybrid genetic algorithm with fuzzy logic controller. In: *Fuzzy Sets Based Heuristics for Optimization*, ed. by J. L. Verdegay (Springer, New York 2003) pp. 251–263

42.4 C. Y. Lee, Y. S. Yun, M. Gen: Reliability optimization design for complex systems by hybrid GA with fuzzy logic controller and local search, IEICE Trans. Electr. **E85-A**, 880–891 (2002)

42.5 H. Xu, G. Vukovich: Fuzzy Evolutionary Algorithms, Automatic Robot Trajectory Generation, Proc. of the First IEEE Conference on Evolutionary Computation, Orlando 1998, ed. by D. Fogel (IEEE Press, Piscataway 1998) 595–600

42.6 H. Ishii, H. Shiode, T. Murata: A multiobjective genetic local search algorithm and its application to flowshop scheduling, IEEE Trans. Syst. Man Cyber. **28**, 392–403 (1998)

42.7 M. Gen, J. R. Kim: GA-based Optimization of Reliability Design. In: *Evolutionary Design by Computers*, ed. by P. Bentley (Morgan Kaufman, San Francisco 1999) pp. 191–218

42.8 S. Martello, P. Toth: *Knapsack Problems: Algorithms and Computer Implementations* (Wiley, Chichester 1990)

42.9 M. Gen, R. Cheng: *Genetic Algorithms & Engineering Design* (Wiley, New York 1997)

42.10 M. Gen, R. Cheng: *Genetic Algorithms & Engineering Optimization* (Wiley, New York 2000)

42.11 L. Lin, M. Gen: Node-based genetic algorithm for communication spanning tree problem, IEICE Trans. on Comm. **E89-B**(4), 1091–1098 (2006)

42.12 E. Falkenauer: Tapping the full power of genetic algorithms through suitable representation, local optimization: Application to bin packing. In: *Evolutionary Algorithms in Management Applications*, ed. by J. Biethahn, V. Nissen (Springer, Berlin Heidelberg New York 1995) pp. 167–182

42.13 L. Davis (Ed.): *Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York 1991)

42.14 Z. Michalewicz: *Genetic Algorithm + Data Structure = Evolution Programs*, 2nd edn. (Springer, New York 1994)

42.15 D. Goldberg, R. Lingle: Loci and The Traveling Salesman Problem, Proc. of the First International Conference on Genetic Algorithms, New Jersey 1985, ed. by J. Grefenstette (Lawrence Erlbaum Associates, Hillsdale 1985) 154–192

42.16 R. Cheng, M. Gen: Evolution Program for Resource Constrained Project Scheduling Problem, Proc. of the First IEEE Conference on Evolutionary Computation, Orlando 1994, ed. by D. Fogel (IEEE Press, Orlando 1994) 736–741

42.17 M. Gen, A. Kumar, J. R. Kim: Recent network design techniques using evolutionary algorithms, Int. J. Prod. Econ. **98**(2), 251.261 (2005)

42.18 R. Cheng, M. Gen: Resource constrained project scheduling problem using genetic algorithm, Inter. J. Intell. Autom. Soft Comp. **3**, 273–286 (1997)

42.19 T. Yokota, M. Gen, K. Ida, T. Taguchi: Optimal design of system reliability by an approved genetic algorithm, Trans. Inst. Electron. Inf. Commun. Eng. **J78A**, 702–209 (1995)

42.20 M. Gen, L. Lin: A new approach for shortest path routing problem by random key-based GA, Genetic and Evol. Comp. Conf., Seattle (2006)

42.21 G. Syswerda: Scheduling Optimization Using Genetic Algorithms. In: *Handbook of Genetic Algorithms*, ed. by L. Davis (Van Nostrand Reinhold, NewYork 1991) pp. 332–349

42.22 M. Gen, L. Lin, R. Cheng: Bicriteria network optimization problem using priority-based genetic algorithm, IEEJ Trans. Elect. Inf. Sys. **124**, 1972–1978 (2004)

42.23 L. Lin, M. Gen: Bicriteria network design problem using interactive adaptive-weight GA and priority-based encoding method, IEEE Trans. Evol. Comput (in reviewing)

42.24 M. Gen, L. Lin: Multi-objective hybrid genetic algorithm for bicriteria network design problem, Compl. Int. **11**, 73–83 (2005)

42.25 C. Cheng, V. Vempati, N. Aljaber: An application of genetic algorithms for flow shop problems, Eur. J. Oper. Res. **80**, 389–396 (1995)

42.26 H. IIshibuchi, N. Yamamoto, T. Murata, H. Tanaka: Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems, Fuzzy Sets and Systems **67**, 81–100 (1994)

42.27 K. Baker: *Introduction to Sequencing and Scheduling* (Wiley, New York 1974)

42.28 M. Gen, K. W. Kim, G. Yamazaki: Project scheduling using hybrid genetic algorithm with fuzzy logic controller in scm environment, J. Tsinghua Sci. Technol. **8**, 1, 19–29 (2003)

42.29 M. Yoo, M. Gen: Scheduling algorithm for real-time tasks using multiobjective hybrid genetic algorithm in heterogeneous multiprocessors system, Comp. Oper. Res. in press

42.30 G. Syswerda: Uniform Crossover in Genetic Algorithm, Proc. of the 3rd International Conference on Genetic Algorithms, San Francisco 1989, ed. by J. Schaffer (Morgan Kaufmann, San Francisco 1989) 2–9

42.31 M. Gen, K. Ida, T. Taguchi: System Reliability Optimization with Several Failure Modes by Genetic Algorithm, Proceedings of International Conference on Computers & Industrial Engineering, Japan, Ashikaga 1994, ed. by M. Gen, Ashikaga 1994) 349–351

42.32 L. Lin, M. Gen: A self control genetic algorithm for reliable communication network design, Proc. of IEEE Congress on Evol. Comp. Vancouver, IEEE Press (2006)

42.33 G. Zhou, M. Gen: A genetic algorithm approach on tree-like telecommunication network design problem, J. Oper. Res. Soc. **54**, 248–254 (2003)

42.34 A. Syarif, M. Gen: Solving exclusionary side constrained transportation problem by using a aybrid spanning tree-based genetic algorithm, J. Intell. Manuf. **14**, 389–399 (2003)

42.35 F. Budnick, D. McLeavey, R. Mojena: *Principles of Research for Management*, 2nd edn. (Irwin, Homewood 1998)

42.36 M. Gen, Y. Li: Solving Multi-Objective Transportation Problem by Spanning Tree-Based Genetic Algorithm. In: *Adaptive Computing in Design and Manufacture*, ed. by I. Parmee (Springer, New York 1998) pp. 95–108

42.37 M. Gen, A. Syalif: Hybrid genetic algorithm for multi-time period production/distribution planning, Comp. Ind. Eng. **48**(4), 799–809 (2005)

42.38 A. Syarif, Y. S. Yun, M. Gen: Study on multi-stage logistics chain network: A spanning tree-based genetic algorithm approach, Comp. Ind. Eng. **43**, 299–314 (2002)

42.39 M. Gen, A. Syarif: Multi-stage Supply Chain Network by Hybrid Genetic Algorithms with Fuzzy Logic Controller. In: *Fuzzy Sets based Heuristics for Optimization*, ed. by J. L. Verdegay (Springer, New York 2003) pp. 181–196

42.40 G. Zhou, H. Min, M. Gen: A genetic algorithm approach to the bi-criteria allocation of customers to warehouses, Int. J. Prod. Econ. **86**, 35–45 (2003)

42.41 M. Gen, F. Altiparmak, L. Lin: A genetic algorithm for two-stage transportation problem using priority-based encoding, OR Spectrum (2006)

42.42 K. Rosing: An optimal method for solving (generalized) multi-weber problem, Eur. J. Oper. Res. **58**, 414–426 (1992)