# Chapter 6
# Genetic Algorithms and Evolutionary Computing

## 6.1 Introduction

Evolutionary algorithms are inspired from the Nature's ability to evolve. Evolutionary algorithms are a component of evolutionary computing in the field of Artificial Intelligence. They are inspired from the biological evolution of random population by employing various modifying operations on the basic pattern of the candidates of the population. In nature, evolution through such modification happens in such a way that the next population will consist of members that are comparatively more fit to survive in the given situation. In a case where the modification results in poorer candidates, they cannot survive, and hence they will be automatically deselected from the population. Only the fit candidates will survive in the population. That is why such an approach is called the survival of the fittest approach. A broad outline of the approach is given as follows.

- Generate initial population by selecting random individuals.
- Apply one or more evaluation functions for the candidates.
- Select fit (good quality) candidates and push them in the next generation directly, if they are up to the mark.
- Select some strong candidates and modify them to generate even stronger candidates and push them to the next generation.
- Repeat the procedure until the population evolves towards a solution.

Evolutionary Algorithms is an umbrella term used to describe a consortium of various computational models/techniques following the basic principle of evolution. All the computational techniques that fall under the area of evolutionary algorithms apply various modifications to the population and preserve better candidates, and hence evolve the population. Techniques such as genetic algorithms, genetic programming, evolutionary programming, gene expression programming, evolution



R. Akerkar, P.S. Sajja, *Intelligent Techniques for Data Science*,
DOI 10.1007/978-3-319-29206-9_6

**Table 6.1** Evolutionary algorithm techniques

| Technique | Description |
|---|---|
| Genetic algorithm | It is the most popular technique that falls under evolutionary algorithms. |
| | The candidates are represented by strings over a finite set of alphabets, numbers or symbols. Traditionally binary number system is used. |
| | It is often used to solve optimization problems. |
| Genetic programming | The candidates within the population are in form of computer code. The code that can solve a given computational problem is considered to be a fit candidate. |
| | Often the individuals are represented through tree encoding strategy. |
| Evolutionary strategy | The candidates in populations are represented as real valued vectors. The techniques generally use self-adaptive mutation rates. |
| Evolutionary programming | This technique is very much similar to genetic programming. The difference lies in the structure of the computer code used as individuals. Here, the structure of computer code is fixed; however, parameters are allowed to evolve. |
| Gene expression programming | This technique also uses various computer codes as individuals in the population. Here, computer codes of different sizes are encoded into linear chromosomes of pre-defined length using tree data structure. |
| | The individuals represented using tree structures evolve by changing their sizes, shapes, and composition. |
| Differential evolution | Differential evolutions use a similar evolutionary approach for real numbers with specific mutation and crossover operators. |
| | Here, the candidates are represented in the form of vectors and emphasis is given on vector differences. This technique is basically suited for numerical optimization. |
| Neuro-evolution | This is similar to genetic programming; however, the difference is in the structure of the candidate. Here, the individuals of a population are artificial neural networks. |
| Learning classifier system | Here, the individuals of a population are classifiers represented in the form of rules or conditions. The rules can be encoded in binary (direct approach) or using artificial neural network or symbolic expression (indirect approach). |
| Human-based genetic algorithm | The technique is similar to genetic algorithm or genetic programming. However, the human is allowed to direct the evolutionary process. Often the human serves as the objective function and evaluates potential solutions. |

strategy, differential evolution, neuro-evolution and learning classifier systems fall under the umbrella of evolutionary algorithms. Table 6.1 describes the techniques in brief.

For any evolutionary system, the fundamental components are encoding strategy using the individuals that are represented, modifying operators and evaluation functions. Modification of operators generate new candidates to form a new generation from which better candidates are selected. It is the evaluation function that preserves the good quality solutions within the populations. However, sometimes

the weak candidates may survive and become parents as some of their characteristics are really strong. This way, one may claim that evolutionary algorithms are stochastic. The common characteristics of evolutionary algorithms are listed below.

- Evolutionary algorithms preserve population, modify their candidates and evolve stronger candidates.
- Evolutionary algorithms modify candidates by combining adorable characteristics of individual candidates to generate a new candidate that is stronger than the original candidates.
- Evolutionary algorithms are stochastic in nature.

## 6.2 Genetic Algorithms

Being a candidate of the evolutionary algorithm consortium, a genetic algorithm (GA) follows the basic principle of evolution as stated earlier (refer to Section 1, broad outline of evolutionary algorithm). The genetic algorithm is generally used for searching and optimization problems in an adaptive manner. The process of evolution is very similar to biological evolution and works according to the survival of the fittest principle introduced by the famous naturalist and geologist Charles Darwin. It is also called the principle of natural selection. In nature, It is observed that species and individuals compete for resources such as food, water, shelter and mates. Those who can succeed in these exercises will survive and can produce many offspring. Losers are not able to survive, they cannot produce offspring, and hence the gene pattern of weak individuals cannot be transferred. Only the genes from adaptive and fit individuals will be passed to the next generations. The offspring in many cases have better combinations of genes from their parents and become much fitter than the parents. Generation by generation, the species will become stronger. See the popular story of giraffes in Fig. 6.1.

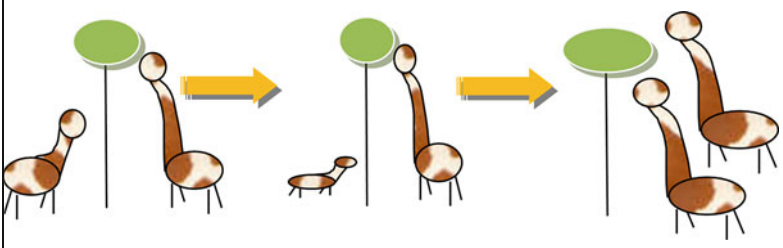
A typical Genetic algorithm is given at Fig. 6.2 and illustrated in Fig. 6.3.

The basic principles of genetic algorithms were first explored by John Holland (1975). Since then, many other scientists have been working on the technique to solve problems in various domains. Genetic algorithms are adaptive and robust in nature. They are useful while the processing voluminous data, which are complex and interrelated. Genetic algorithms are useful and efficient when:

- The search space is large, complex, or poorly understood.
- Domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space.
- No mathematical analysis is available.

Furthermore, genetic algorithms are random algorithms—the course taken by the algorithm is determined by random numbers. In particular, if you ask a random

It is believed that the early giraffes were not tall like today's giraffes. They were much shorter in comparison. The Charles Darwin survival of the fittest theory explains how all giraffes at present have long neck. He explained that, at the time of drought, during scarce situation, some of the giraffes with long necks, could managed to eat more leaves from the far branches could survive. Giraffes with short height (and neck) could not survive and died. Most of the survived giraffes had long neck and good height. The next generation produced by these tall giraffes was having many tall giraffes as the parents are tall. In such a way the whole generation went through the genetic modification, parents' genes of good height were transformed to the children. By accident, if there were any short giraffes, they could not survive; as they are not fit for the environment. By this way the species of giraffes has become tall.



**Tall** giraffes live and breed & short giraffes give up … ends up with all tall giraffes.

In the case of giraffe, the characteristic of good height is very important to survive, and giraffes who could maintain good height could survive. This shows that the Nature will support the solutions with the desired characteristics. Solutions which are fit and strong can only survive.

**Fig. 6.1** Tale of giraffes evolution

algorithm to optimize the same problem twice in precisely the same manner, you will get two different answers. Sometimes you'd like to get the same answer to the same question. If so, this is against random algorithms—though it would be possible to always use the same seed for a random number generator.

The common principles of the genetic algorithm are discussed in the next section.

Begin

- Generate initial population of individuals (known or random).
- Encode the individuals by taking suitable encoding stream. Also determine string length of encoded string for an individual.
- Repeat the steps given below till the desired fitness level of the individuals is reached or improvement is not further observed or up to predefined number of iterations (say maximum 100).
    - Apply genetic operators and generate new population.
    - Evaluate fitness of the individuals.
    - Select stronger individuals and remove poor ones from the new population.

End

**Fig. 6.2** Typical genetic algorithm

## 6.3 Basic Principles of Genetic Algorithms

Before application of genetic algorithms to a problem, the individuals from the problem domain must be represented in the form of genes of predetermined length. As genetic algorithms follow the concept of evolutionary algorithms, this starts with a population encompassing randomly selected (if not known) and encoded individuals. These individuals are frequently modified and evaluated for their fitness, until a satisfactory solution is reached. For these, there needs to be an encoding strategy (for representation of characteristics of individuals), genetic operators such as mutation and crossover (to modify individuals), and fitness functions (to push good quality individuals into the next generation). This section describes these basic principles with necessary examples.

### *6.3.1 Encoding Individuals*

Individuals selected from the domain of interest need to be encoded in the form of genes. Each gene or set of genes represents a characteristic. A set of genes is also commonly known as a gene string, phenotype or genotype. However, a genotype is the set of genes in the string which is responsible for a particular characteristic, whereas a phenotype is the physical expression, or characteristics. Each individual is encoded and represented in the form of a series of genes/genotype. That is why

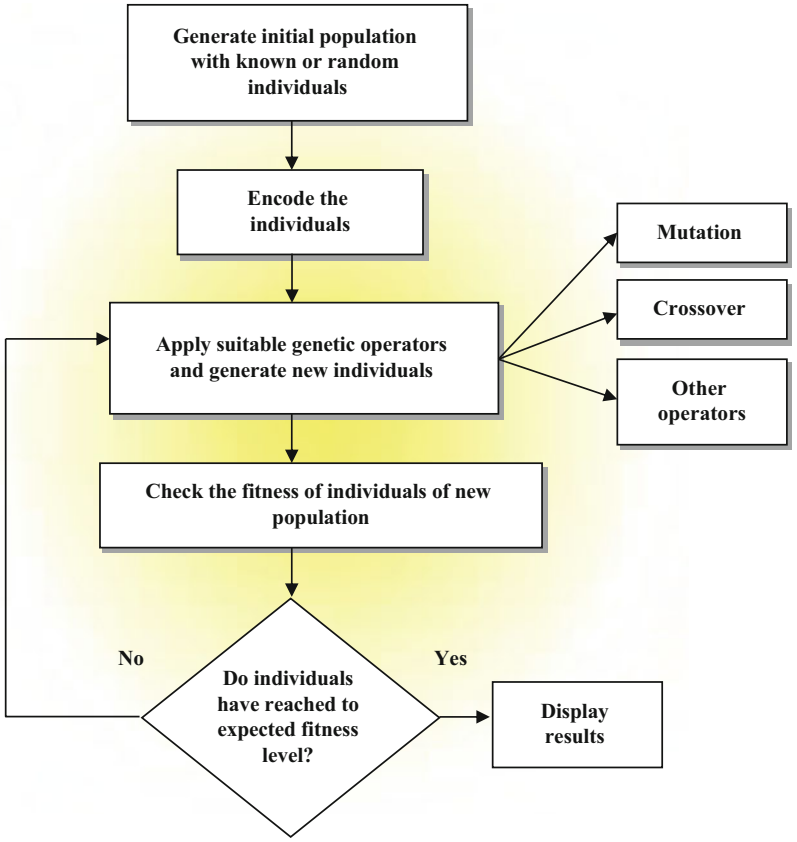



**Fig. 6.3** Genetic algorithm illustration

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Individual X | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| Individual Y | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

**Fig. 6.4** Binary encoding

an individual is identified as a chromosome. John Holland (1975) has used binary values to encode individuals. In binary encoding, each individual is represented in the form of a binary number (a number system having only two symbols, 0 and 1, with base 2). Each binary digit (bit) represents a gene.

Figure 6.4 shows an example of individuals X and Y encoded in a binary number system with a string length of 8.

After encoding, initialization of the population is done. While initializing a population, random individuals from the search space are chosen or it can be filled with known values.

### 6.3.2 Mutation

Mutation is basically a bit flipping operation. It is a method of changing a gene with another valid value, with the help of a mutation probability, say $P_{Mutation}$. It is done as follows. A number between 0 and 1 is chosen at random. If the random number is smaller than $P_{Mutation}$, then the mutation is done at the current bit, otherwise the bit is not altered. For binary encoding, a mutation means to flip one or more randomly chosen bits from 1 to 0 or from 0 to 1. Figure 6.5 shows two original individuals X and Y, as shown in Fig. 6.4, and a mutation operation on them. On the individual represented by label X, a two-site mutation is done at position 3 and position 5. On the individual represented by Y, a two-site mutation is done at position 1 and position 7.

### 6.3.3 Crossover

Using mutation alone is not sufficient; rather, it is just like a random walk through the search space. Other genetic operators must be used along with mutation. Crossover is another such operator. Crossover selects substrings of genes of the same length from parent individuals (often called mates) from the same point, replaces them, and generates a new individual. The crossover point can be selected randomly. This reproduction operator proceeds as follows.

- The reproduction operator selects at random a pair of two individual strings.
- A cross-site is selected at random within the string length.
- The position values are swapped between two strings following the cross-site.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Individual X | 1 | 0 | **0** | 1 | **1** | 0 | 0 | 1 |
| New Individual X | 1 | 0 | **1** | 1 | **0** | 0 | 0 | 1 |
| | | | | | | | | |
| Individual Y | **0** | 1 | 1 | 1 | 0 | 0 | **1** | 0 |
| New Individual Y | **1** | 1 | 1 | 1 | 0 | 0 | **0** | 0 |

**Fig. 6.5** Two-site mutation on individual X and individual Y

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Individual X | 1 | **0** | **0** | **1** | 1 | 0 | 0 | 1 |
| Individual Y | 0 | **1** | **1** | **1** | 0 | 0 | 1 | 0 |
| | | | | | | | | |
| Individual P | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| Individual Q | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |

**Fig. 6.6** Crossover on binary encoding with string length 3, position 2

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Individual X | 1 | **0** | **0** | **1** | 1 | 0 | **0** | **1** |
| Individual Y | 0 | **1** | **1** | **1** | 0 | 0 | **1** | **0** |
| | | | | | | | | |
| Individual P | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| Individual Q | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

**Fig. 6.7** Crossover on binary encoding with string length 3, position 2 and string length 2, position 7

Such operation results in two new individuals. Crossover selects some desirable characteristics from one parent and other characteristics from another parent and creates a new individual by taking the best of both (just like a father's height and a mother's skin in a child!). Figure 6.6 illustrates the process of crossover.

Simultaneous crossover can also be done at multiple sites, as shown in Fig. 6.7.

### *6.3.4 Fitness Function*

Fitness function plays an important role in determining the quality of individuals and hence the quality of the population providing the end solutions. It is the control that is set on the individuals for allowing them into the next generation: individuals that are fit as per the fitness function that only would be pushed further. It is observed that tighter and better fitness function results in high quality individuals in a generation. A fitness function must be well defined in order to allow individuals with similar fitness to be close and together. The fitness function must lead the evolution towards valid and good individuals. The fitness function also works as a penalty function for poor individuals. Some penalty functions also provide completion cost. The completion cost is factor showing expected cost of converting an invalid individual into a valid one. In some cases, instead of exact fitness or penalty function, approximate functions are also used.

### 6.3.5 *Selection*

A selection operator chooses good individuals for further operations. It generally copies good strings, but does not create new ones. Individual solutions are selected through a well-defined fitness function. Many definitions of selection operators have been in application. John Holland's (1975) *fitness-proportionate selection* is a pioneer. In the method defined by the Holland, individuals are selected with a probability proportional to their relative fitness. Such fitness-proportionate selection is also known as Roulette-wheel selection. If $f_i$ is the fitness of individual $i$ in the population, its probability of being selected is $P_i = f_i / \sum f_i$, where i takes values from 1 to $N$, where N is the number of individuals in the population. The name 'Roulette-wheel solution' comes from the analogy of a typical Roulette wheel in casino games. Each individual represents a pocket on the wheel; the size of the pockets is proportionate to the probability of the individual being selected. Instead of probability, percentages can also be considered. Selecting N individuals from the population is equivalent to playing N games on the Roulette wheel, as each candidate is drawn independently. That is, selection is done by rotating the wheel a number of times equal to the population size. Figure 6.8 illustrates the typical Roulette wheel selection.

Because of the probability function utilized here, in the Roulette wheel selection method, there is a chance that stronger individuals with a higher fitness value may be deselected. With this approach, another chance is that weaker individuals may be selected. It is beneficial sometimes; as such weaker solutions may have a few genes that are really strong. A strong part of the weaker individual may then be further considered for crossover function.

Another method that eliminates a fixed percentage of the weakest candidate is the tournament selection method. Tournament selection works as follows.

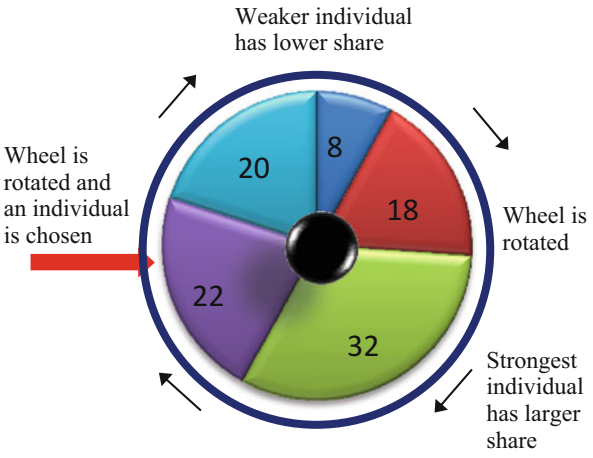



**Fig. 6.8** Roulette wheel selection

**Table 6.2** Tournament selection

| |
|---|
| Begin |
| Determine size of tournament, say K |
| Create mating pool M |
| Selected individual set A |
| Determine probability P |
| Select K individuals from the population at random |
| Repeat for K individuals |
| {Evaluate fitness and store in A} |
| Select best from A and insert it to the mating pool M with probability P |
| Select second best from A and insert it to the mating pool M with probability $P\ (1-P)$ |
| Select third best from A and insert it to the mating pool M with probability $P\,(1-P)\,(1-P)$ |
| End |

- A pair of individuals are selected randomly and evaluated for their fitness.
- The stronger individual is inserted into a mating pool.
- This process is repeated until the mating pool is filled completely.
- Modifying operators and probability based fitness functions are applied to these individuals.

Here, we are selecting an individual from a pair, hence it is called binary tournament. At a time many individuals (say n) can also be selected for comparison, this is called larger tournament with size n. If the tournament size is larger, strong individuals have better chances to be selected. To make better individuals win, one can consider a probability factor along with comparison through fitness function, where an individual will win the tournament with probability $p$ (usually above average, i.e., $>0.5$). Table 6.2 displays a broad outline of the tournament selection.

While selecting individuals, some researchers adopt a methodology for selection such that no individual of the current population will be duplicated. Some select n new individuals and delete $n$ individuals and keep the population in fixed size and steady state status. Another possibility is to deselect all and add any number of new individuals.

Using selection alone will tend to fill the population with copies of the best individual from the population. However, the selection cannot introduce new individuals into the population. For inclusion of new individuals, one has to have the help of operators such as mutation and crossover.

### *6.3.6 Other Encoding Strategies*

So far we have seen binary encoding strategy for individuals with fixed length bit strings, which is a popular encoding strategy. One can also use symbols, alphabets, and tree structures for encoding. Figure 6.9 presents mutation and crossover operations on trees X and Y, respectively.

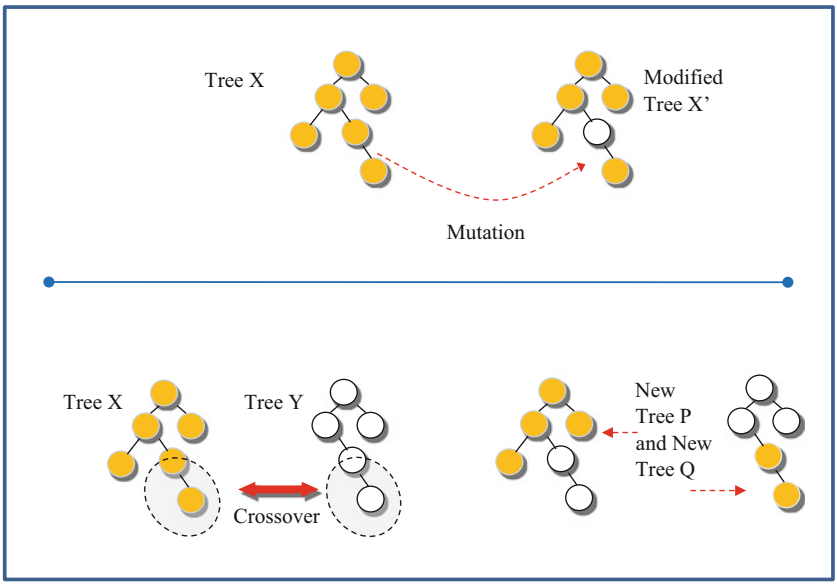



**Fig. 6.9** Tree encoding and operations

Generally, tree encoding would be used in evolving programs in genetic programming as well as gene expression programming. The syntax of the instructions can be represented in the form of trees on which operations like selection, mutation, and crossover are carried out. The resulting offspring are evaluated against a fitness function.

**Real World Case 1: Airplane Design Optimization**
A primary airplane design can be realized by means of genetic algorithms (GA). The airplane key parameters are mapped into a chromosome-like string. These include the wing, tail and fuselage geometry, thrust requirements and operating parameters. GA operators are performed on a population of such strings and natural selection is expected to occur. The design performance is obtained by using the airplane range as the fitness function. GA cannot only solve design issues, but also project them forward to analyze limitations and possible point failures in the future so these can be avoided.

## 6.4 Example of Function Optimization using Genetic Algorithm

This example demonstrates use of the genetic algorithm for function optimization. Consider a function $f(x, y) = x * y$ for maximization within the interval of [0, 30]. A genetic algorithm finds values of $x$ and $y$ (an order pair $(x,y)$) that produce maximum values of the function $f(x, y)$ from the given interval [0, 30].

**Encoding**
Individuals are encoded using binary digits (bits) with length 5. The possible values of x and y are between 0 and 30. These numbers can be represented into five bits maximum. Taking more binary bits results in larger numbers than 30 and fall out of the specified interval for values $x$ and $y$. Hence, such numbers are invalid.

**Fitness Function**
The fitness function definition is already provided in the problem definition. The function definition ($x*y$) will be the fitness function. The individuals x and y are added to check the fitness.

The initial population is selected randomly as denoted in Table 6.3.

On the initial population, fitness of individuals are calculated, according to which a Roulette wheel is designed, which is shown in Fig. 6.10.

Selection, mutation and crossover operations are carried out on the basis of the Roulette wheel count shown in Fig. 6.10. The resulting individuals are denoted in Table 6.4.

**Table 6.3** Initial population

| Sr. no. of individual | Individuals (First 5 bits for $x$ and last 5 bits for $y$) | Decimal values of $x$ and $y$ | Fitness $f(x, y) = x*y$ (Value in Decimal) | Roulette wheel selection count |
|---|---|---|---|---|
| 1 | 0101101010 | 10,11 | 110 | 1 |
| 2 | 1100001100 | 24,12 | 288 | 2 |
| 3 | 0100100101 | 09,05 | 45 | 0 |
| 4 | 0101100111 | 11,07 | 77 | 1 |

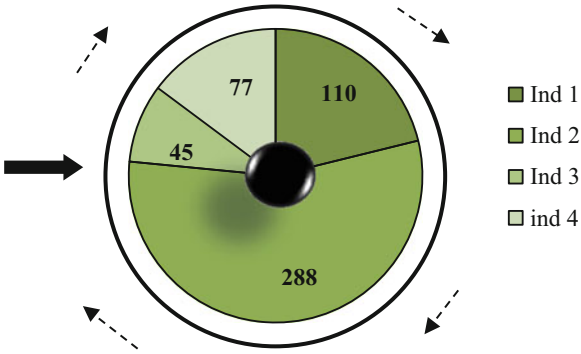



**Fig. 6.10** Roulette wheel selection for the example

**Table 6.4** Operations on the population

| Sr. no. of selected individual | Individuals (First 5 bits for $x$ and last 5 bits for $y$) | New individual after mutation or crossover | Operation site (and string length) | Fitness $f(x, y) = x*y$ (value in decimal) |
|---|---|---|---|---|
| 1 | 01**0**11 **0**1010 | 01**1**11 **1**1010 | Mutation at position 3 and 6 | 390 |
| 2 | 11000 **0**1100 | 11000 **1**1100 | Mutation at position 6 | 672 |
| 3 | 110**00** 01100 | 110**11** 01100 | Crossover with mate 4, site 4, length 2 | 324 |
| 4 | 010**11** 00111 | 010**00** 00111 | Crossover with mate 3, site 4, length 2 | 56 |

**Table 6.5** Second generation

| Sr. no. of individual (new) | Individuals (first 5 bits for $x$ and last 5 bits for $y$) | Decimal value of $x$ and $y$ | Fitness $f(x, y) = x*y$ (value in decimal) | Roulette wheel selection count |
|---|---|---|---|---|
| 1 | 01111 11010 | 15, 26 | 390 | 1 |
| 2 | 11000 11100 | 24, 28 | 672 | 2 |
| 3 | 11011 01100 | 27, 12 | 324 | 0 |
| 4 | 01000 00111 | 8, 7 | 56 | 1 |

One can see in Table 6.4 that individual 3 is removed from the population and individual 2 is duplicated. This decision is taken from the experiments of the Roulette wheel. The count of the roulette wheel is denoted in Table 6.3. It is obvious that individual 2 is the strongest (fittest) in the first generation, as shown in Table 6.3. The poorest individual is the individual 3; hence, it is removed from the next generation. After selection of stronger individuals in the next generation, new individuals are created by applying the mutation and crossover function. Table 6.5 shows the second generation.

One can observe an increase in fitness values for majority of the individuals in Table 6.5, illustrating the second generation of the population. The Roulette wheel for the second generation individuals is shown in Fig. 6.11.

Continuing the evolution, one can reach to the optimum solution. Here, we are have the advantage of knowing the solution (through application of the traditional method) and we may smartly select operations (mutation and crossover) in such a way that we can reach the solution in a few generations. In the case of real problems, where traditional solutions are not available, one can set the maximum number of iterations of evolution or the minimum improvement factor in the solution. If there is no significant improvement after evolution, and stagnation is achieved, the process may be stopped. Ideally, the genetic algorithm should evolve in such a way that all its individuals are successively showing better fitness.

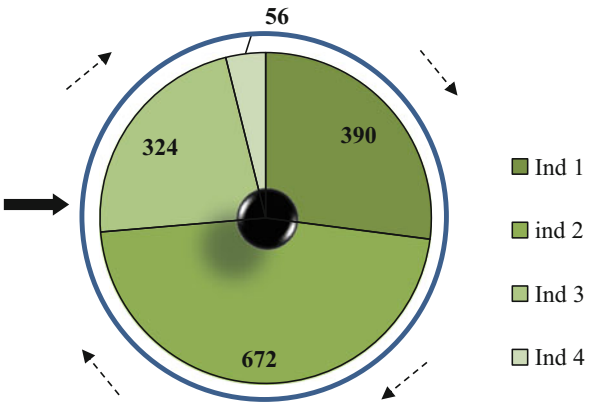



**Fig. 6.11** Roulette wheel for the second generation individuals

## 6.5 Schemata and Schema Theorem

Schema is a template that identifies a subset of strings with similarities at certain string positions. The notion of schema was introduced by John Holland (1975). A schema is a template defined over the alphabet $\{0, 1, *\}$, which describes a pattern of bit strings in the search space $\{0, 1\}^L$ (the set of bits strings of length L). For each of the L bit positions, the template either specifies the value at that position (1 or 0), or indicates by the symbol * (referred to as 'don't care' or wildcard symbol) that either value is allowed.

The schema **1 0 * * 1 0 *** is a template for the following strings:

1 0 11 1 0 1
1 0 00 1 0 0
1 0 01 1 0 1
1 0 01 1 0 0
1 0 10 1 0 1
1 0 10 1 0 0

### *6.5.1 Instance, Defined Bits, and Order of Schema*

A bit string $x$ that matches a schema's **S** pattern is said to be an instance of **S**. In a schema, 1s (ones) and 0s (zeroes) are referred to as defined bits. The total number of defined bits in the schema is considered as an order of a schema. The defining length of a schema is the distance between the leftmost and rightmost defined bit in the string, as shown in Table 6.6.

**Table 6.6** Order, length and instances of schema

| Schema | Order | Length | Instances |
|---|---|---|---|
| 1 1 * * 1 1 0 * | 5 | 6 | 1 1 11 1 1 0 1 |
| | | | 1 1 01 1 1 0 0 |
| * * * * 1 1 0 * | 3 | 2 | 11 10 1 1 0 0 |
| | | | 1011 1 1 0 1 |
| **0*0*1 | 3 | 4 | 1101011 |
| | | | 1001001 |
| 1**0*1 | 3 | 5 | 100011 |
| | | | 111011 |

### *6.5.2 Importance of Schema*

Schema in one way provides an explanation of how a genetic algorithm works. The basic idea is to evaluate fitness of a schema (that is, group of individuals represented by the schema), instead of applying the fitness function to randomly generated individuals from the whole population. Each schema represents a set of individuals following the common pattern. Evaluating the pattern reduces the burden of testing each individual following the schema. It can be said that the fitness of the schema is the fitness of the individuals following the pattern specified by the schema. This is an important and efficient way to manage a big search space and impart generalization in the evolution process. This way, schemas will be helpful in searching good regions of the search space. A schema that provides above-average results has greater chances of providing better elements in terms of fitness. As per the Building Block Hypothesis (Hayes, 2007), the genetic algorithm initially detects biases towards higher fitness in some low-order schemas (with a small number of defined bits) and over time, detects the same in high-order schemas. John Holland (1975) has also suggested that the observed best schemas will, on average, be allocated an exponentially increasing number of samples in the next generation. Holland also suggested that an individual may satisfy more than one schema, some of which are stronger and some of which are weaker. Indirectly, the manipulation is done on these schema in such a way that stronger schema will survive. This is called an implicit parallelism.

## 6.6 Application Specific Genetic Operators

Typical genetic operators such as crossover and mutation may not be suitable for many applications. Consider the travelling sales person problem. The traveling salesperson problem is about finding an optimum path of a tour from a given set of cities so that each city is visited only once and the total distance travelled is minimized. To encode the possible routes as individuals, either city initials (alphabets) or symbols are used. One can also use numbers. If there are five different

cities to be travelled once with the minimum effort, the possible paths using numbers as encoding strategy can be given as follows:

**Route 1** : **(1 2 3 4 5)**
**Route 2** : **(2 3 4 5 1)**

The standard mutation and crossover operators here create illegal/invalid solutions/plans. See the following example, in which Route 1 undergoes a mutation operation at location 1. City 1 can be replaced with one of the remaining cities (2, 3, 4, or 5). This results in an illegal plan.

**Original Route 1** : **(1 2 3 4 5)**
**Mutated Route 1** : **(2 2 3 4 5) (invalid)**

Similarly, a crossover (at location 2 with string length 3) also results in illegal solutions, as follows:

**Route 1** : **(1 2 3 4 5)**
**Route 2** : **(2 3 4 5 1)**

**New Offspring Route 3** : **(1 3 4 5 5)** (invalid)
**New Offspring Route 4** : **(2 2 3 4 1)** (invalid)

To avoid such illegal solutions, one can do following.

- Try a different representation (encoding scheme).
- Design a special crossover operator that is application specific and generates valid offspring.
- Design a penalty function that removes illegal individuals by giving high penalty (say, negative or low fitness).

In the case of the travelling salesperson problem, fitness function is very straightforward and rigid. It is not advisable to change the fitness function. Encoding strategy is also very limited in this case. Here, one can consider the new design of genetic operators that are suitable to this application and generate valid individuals.

With the help of the edge recombination technique, this problem can be solved. This technique works as follows.

- Create a population with finite number of legal tours as the initial generation.
- Create an adjacency table for each city in all routes. The adjacency table contains all cities along with all their possible neighbours in all the routes from the population.
- Generate new individuals by recombining the genes used in the parent individuals as follows:

- Select the parent at random and assign its first gene as the first element in the new child.
- Select the second element for the child as follows: If there is an adjacency common to both parents, choose that element to be the next one in the child's permutation; if there is an unused adjacency available from a parent, choose it. If these two options fail, make a random selection.
- Select the remaining elements in order by repeating step

• Push the new individuals in the new population in order to evaluate its fitness.

### *6.6.1 Application of the Recombination Operator: Example*

Consider five different cities labelled 1, 2, 3, 4, and 5 for a typical traveling salesperson problem (TSP). The two randomly generated individuals are as follows:

**Route 1** : **(1 2 3 4 5)**
**Route 2** : **(2 3 4 5 1)**

The new adjacency list can be given as follows:

| Key | Adjacency list |
|---|---|
| 1 | 2,5 |
| 2 | 1,3,3 |
| 3 | 2,4,2,4 |
| 4 | 3,5,3,5 |
| 5 | 4,4,1 |

With the previous algorithm, a new offspring can be generated, with City 3 as random starting point. Let us append the starting city in the new child route, called Route 3.

**Route 3** : **3**,

From city 3, next popular destination is either city 2 or 4 as per the adjacency table. Let us select city 2. The city 2 is appended at the route 3.

**Route 3** : **3**, **2**,

From city 2, popular destinations are city 3 and 1. Since city 3 is already traversed, city 1 is selected and appended in to the Route 3.

**Route 3** : **3**, **2**, **1**,

From city 1, popular destinations are city 2 and 5. Since city 2 is already traversed, city 5 is selected and appended in to the Route 3.

**Route 3 : 3, 2, 1, 5,**

It is obvious that city 4 is not yet visited; hence we may append the city 4 directly. Otherwise, city 5's neighbours are city 4 and city 1. From the adjacency table, one can see that city 4 is comparatively more popular than city 1. City 1 is already visited, hence we select city 4. After appending city 4 in the new route, the new route will be as follows.

**Route 3 : 3, 2, 1, 5, 4**

Route 3 is a valid route as it lists each city once only. If the distances between cities are provided, one can also calculate the cost of the route. In our case, it is the fitness function.

**Real World Case 2: Shipment Routing**
Recent applications of a GA, such as the *Traveling Salesperson Problem*, can be used to plan the most efficient routes and scheduling for travel planners, traffic routers and even logistics companies. The main objectives are to find the shortest routes for traveling, the timing to avoid traffic delays, and to include pickup loads and deliveries along the way. See section 6.6 for further details.

## 6.7 Evolutionary Programming

Evolutionary Programming is inspired by the theory of evolution by means of natural selection. Precisely, the technique is inspired by macro-level or the species-level process of evolution (phenotype, hereditary, variation), and is not concerned with the genetic mechanisms of evolution (genome, chromosomes, genes, alleles). Evolutionary programming (EP) was originally developed by L. J. Fogel et al. (1966) for the evolution of finite state machines using a limited symbolic alphabet encoding. Fogel focused on the use of an evolutionary process for the development of control systems using finite state machine (FSM) representations. Fogel's early work elaborated the approach, focusing on the evolution of state machines for the prediction of symbols in time series data.

Evolutionary programming usually uses the mutation operator to create new candidate solutions from existing candidate solutions. The crossover operator that is used in some other evolutionary algorithms is not employed in evolutionary

programming. Evolutionary programming is concerned with the linkage between parent and child candidate solutions and is not concerned with surrogates for genetic mechanisms.

Fogel broadened EP to encode real numbers, thus offering a tool for variable optimization. Individuals in the EP comprise a string of real numbers, as in evolution strategies (ESs). EP differs from GAs and ESs in that there is no recombination operator. Evolution is exclusively dependent on the mutation operator, which uses a Gaussian probability distribution to perturb each variable. The standard deviations correspond to the square root of a linear transform of the parents' fitness score (the user is required to parametrize this transform).

An EP outline can be given as:

1. A current population of $\mu$ individuals is randomly initialized.
2. Fitness scores are assigned to each of the $\mu$ individuals.
3. The mutation operator is applied to each of the $\mu$ individuals in the current population to produce $\mu$ offspring.
4. Fitness scores are assigned to the $\mu$ offspring.
5. A new population of size $\mu$ is created from the $\mu$ parents and the $\mu$ offspring using tournament selection.
6. If the termination conditions are satisfied, exit; otherwise, go to step 3.

## 6.8 Applications of GA in Healthcare

Genetic algorithms are also considered as search and scheduling heuristics. As stated earlier, in this chapter, when a problem domain is too large and traditional solutions are too difficult to implement, genetic algorithms are most suitable. Genetic algorithms can generate a good number of possible solutions and use them to find the best solutions with the specific fitness function. Data science activities include data identification and acquisition, data curation, data analytics and data visualization. These activities deal with large amounts of data that do not follow uniform structure. Here, genetic algorithms help in identifying cluster of data, managing the data, scheduling and processing data, and helping other activities of data science in many different ways. Here are some reasons why genetic algorithms are useful for managing data science-related activities.

- Genetic algorithms are useful to manage non-linearity in the search space.
- Genetic algorithms are good in searching from large and complex search spaces using heuristics; furthermore, genetic algorithms can perform global searches efficiently on such search spaces.
- Genetic algorithms are robust and adaptive in nature.
- Genetic algorithms can be integrated with other technologies such as artificial neural network and fuzzy logic; i.e., they are extensible in nature and easy to hybridize.
- Genetic algorithms are remarkably noise tolerant and evolving in nature.

### 6.8.1 Case of Healthcare

Healthcare is defined as caring about one's heath through proper and timely diagnosis, treatment and prevention of a disease or any uncomfortable health related situation in human beings. To practice healthcare at different levels, many organized systems work together. Being a multi-disciplinary field affecting a mass of people, healthcare practice generates a large amount of data, which must be handled with care in order to improve efficiency, reduce cost and increase availability. The area is simultaneously most important and complex. It is important, because it deals with the basic health of mankind, and complex because of its volume, its unstructured nature and variety of content. This is an area that involves more complicated data in large volume than many other industries. Furthermore, data that are the base of problem solving and decision making also have relative importance and are dynamic in nature. For example, in the health care domain, some viruses become resistant to an antibiotic after a certain period. This leads to the requirement of designing new antibiotic drugs. For this, a long history of many patients treated with similar antibiotics must be studied in order to determine the pattern of virus behaviour. Knowing such patterns would be helpful in designing new, powerful and effective antibiotic drugs. However, a new antibiotic drug will soon become ineffective as viruses become resistant to this new drug also. Further analysis and pattern-finding exercises need to be done, obviously more complex than the previous exercise. Day by day, the procedure becomes more complex and requires dedicated approaches such as genetic algorithms. This shows that with a large pool of complex data, the time has come to utilize data science techniques for healthcare-related data. Healthcare practitioners, insurers, researchers, and healthcare facilitators can be benefited by the use of advanced and latest trend applications in the field of data science.

As stated, the large pool of data generated through healthcare practices can be handled with data science. There are so many activities where healthcare data needs to be managed by handling its sheer volume, complexity and diversity. Data science techniques are helpful in facilitating cost effective data handling for decision making, diagnosing, resource management and care delivery. Data science activities such as data acquisition, data pre-processing and curation, data analytics and data visualization can be helpful in not only managing the data, but also in discovery of knowledge and improving the quality of services provided to patients, who are the ultimate end user of the facility.

Typically, healthcare information includes information related to the patients, doctors and other experts, resources, drugs and insurance claims. Information about these components is not available in a straightforward distinct manner, but in the form of complex mixture. Furthermore, the information is multi-media. It can be partially digitalized and stored in the form of multi-media and partially manual documents. Health informatics can be further subdivided into various domains, such as image informatics, neuro-informatics, clinical informatics, public health informatics, and bioinformatics.

Evolutionary algorithms or genetic algorithms can be useful for managing patient databases for various applications. To demonstrate use of genetic algorithms, a case of automatic scheduling of patients is discussed here. Automatic scheduling of patients is possible for a limited domain; however, in domains with wide scope, it is a very complex problem to automate. Here, one has to take care of patient's emergency, doctor's availability, infrastructure and resources availability as well as insurance-related issues. Hard computing methods are very rigid in nature and hence may not be suitable for applications. Genetic algorithms' capability of handling large search space can be utilized here. This approach may not offer any perfect or best solution; however, it offers good and acceptable solutions, which can be practically possible.

### *6.8.2 Patients Scheduling System Using Genetic Algorithm*

Genetic algorithm, as discussed earlier in this chapter, starts with initial population, with randomly selected individual for the domains. Later on, the initial population undergoes fitness testing. A few elements may be removed, which are poor in terms of fitness, and some good candidates are preserved. Later, to generate more candidate solutions, operators such as mutation and crossover can be applied. This procedure is repeated until the satisfactorily strong candidate is evolved.

Let us consider information related to the task. As patients are at the centre of all the activities, the very first thing to come to mind is information related to patients. Information such as a patient's identification number, name, contact information, category of healthcare service required, and other personal information such as blood group and age, is basic information associated with a set of patients. Similarly, information about infrastructure such as branch or specialty (section) of the hospital, operation theatre, room allocated, ICU facility availed, nurses allocated, and medicines used are also required. Information about doctors, outside experts, supporting staff and medical insurance also plays an important role. The scheduling of surgery may depend on a patient's present conditions, availability of doctors and anaesthetists, and even on medical policy verification. Figure 6.12 illustrates the complexity of the scenario.

To authenticate the schedule once it is proposed, some common validations that can be used in these situations are as follows:

- No patient can undergo many activities at a time. Rather, typical sets of activities are well defined.
- Every resource can be used once at a time. That is, a unit of an operation theatre must be utilized at a time for a single surgery. Similarly, the disposable resources can be used only once at any time.
- Each activity can be performed once.
- Each activity can be performed in a specific order.

The candidate solutions (individuals) must be checked for the above constraints. However, satisfying many such validations, a candidate solution does not become

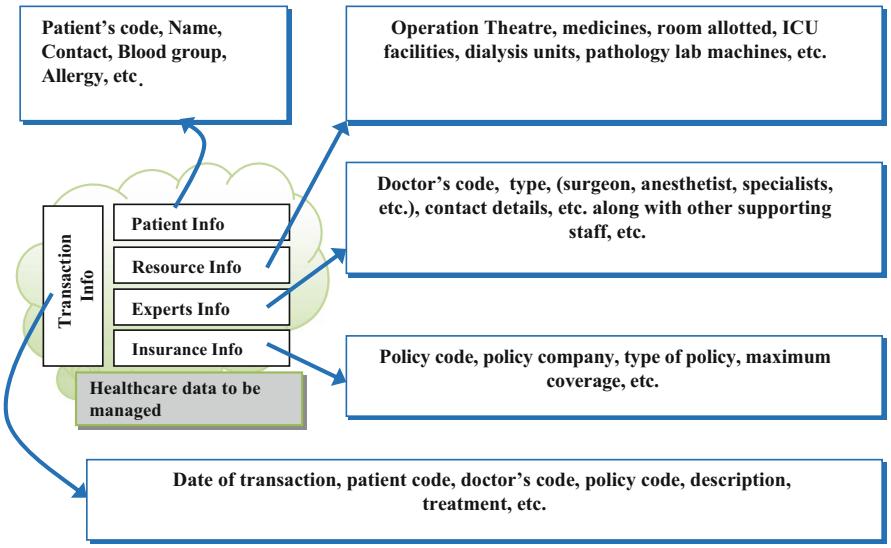



**Fig. 6.12** Possible data in the application

the optimum solution. It has to survive the fitness function designed. A better idea is to apply the constraints at the time of encoding a possible solution. Various constraints, as mentioned above, are collected in order to design a well-defined constraints set, which can be used to validate the solutions suggested by the algorithm.

The overall design of the solution is illustrated as shown in Fig. 6.13.

The following section describes the activities shown in Fig. 6.13 with necessary details.

### *6.8.3 Encoding of Candidates*

The encoding of an individual is done in such a way that it must represent necessary and important characteristics of an individual and simultaneously offer the ability to carry out genetic operations such as mutation and crossover. After applying crossover and mutation, again the derived solution must be checked against the constraint set to the validity of the solution. The derived solution must be feasible. Here, the individual solution is represented in multiple arrays. Each array represents the time characteristic, resource characteristic, and expert characteristic. Instead of taking multiple arrays, a single multi-dimensional array will also serve the purpose. Work done by Vili Podgorelec and Peter Kokol (1997) uses such a multi-dimensional array. An example of an encoding scheme is given at Fig. 6.14.

It is very clear from Fig. 6.14 that in a given time slot, a resource and expert can be used once only; hence leaving no chance of conflicts. Here in the $T_1$ time slot,

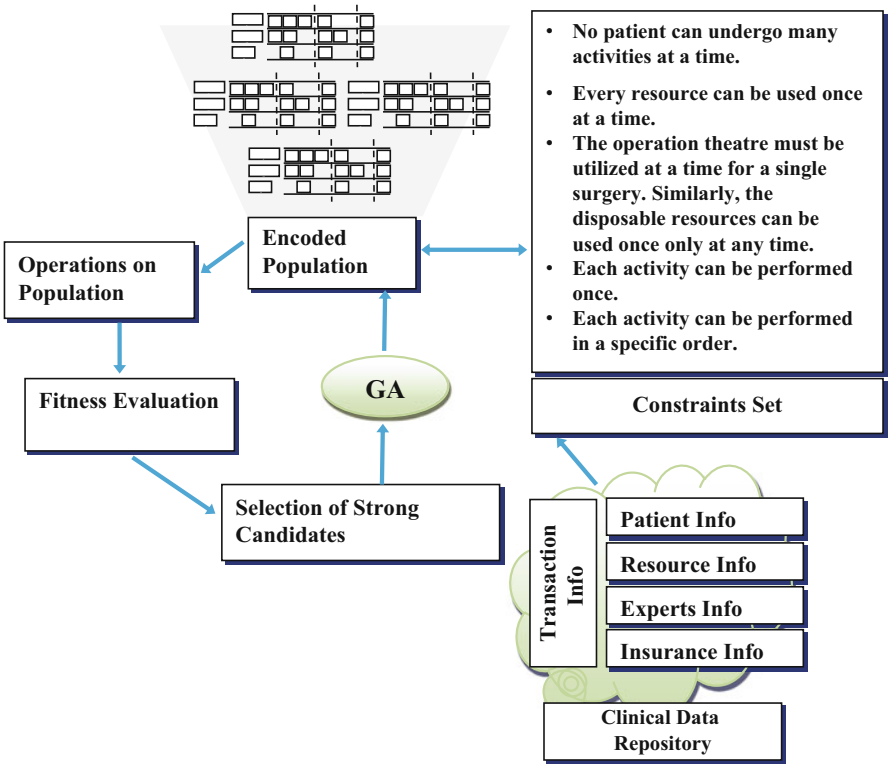



**Fig. 6.13** Scheduling of patients' activities using genetic algorithm

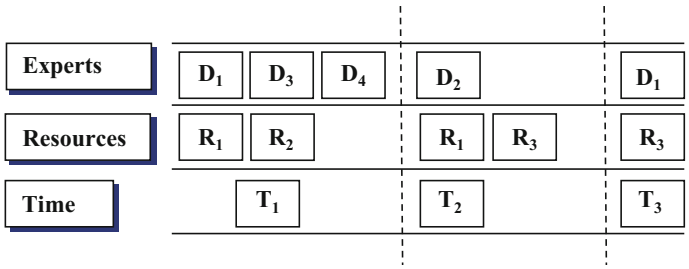



**Fig. 6.14** Example encoding of individuals

resources $R_1$ and $R_2$ are used by expert doctors $D_1$, $D_3$ and $D_4$. It is not possible to intersect the resources $R_1$ and $R_2$ or experts in a given time period $T_1$. This encoding can be further improved by adding one more layer at an appropriate place, which manages added attributes in the encoding of individuals. For example, medical insurances status or any special requirement can be added, along with resources and experts.

The initial population contains many such randomly selected individuals. Prior to addition of an individual in the initial population, it must be checked for its validity. For that, the constraint set is applied. Any invalid candidates from the solution are removed from the population. Once valid candidates of the initial population are finalized, selection, mutation and crossover operations are performed to generate more candidates. Applying genetic operators such as mutation and crossover may result in an increased (or decreased) number of candidates in a population. The modified population is now called the new generation.

### *6.8.4 Operations on Population*

This section describes the possible operations, such as selection, mutation and crossover, on the encoded individuals.

#### 6.8.4.1 Selection

Selection can be done by ranking the individual schedules. To select an appropriate schedule, some of the following criteria can be considered.

- Time taken to complete overall activities.
- Idle time of one or more resources.
- Waiting time for patients.
- Consecutive schedules for a given expert.

For each candidate of a population, fitness score is evaluated using a function of the above parameters. A schedule must have as much as possible low waiting time; i.e., fewer chances of making a resource or expert idle while simultaneously leaving enough space for them to act comfortable. Using Roulette wheel or tournament selection, the candidates are removed or preserved in the population.

#### 6.8.4.2 Mutation

Mutation requires only one candidate at a time. One candidate from the population is considered and some resources and/or experts are changed. This generates a new individual representing a novel schedule. This must be tested against the constraint set for its validity. Some resources and experts are restricted for mutation operations. Figure 6.15 illustrates the mutation operation on schedules.

In the example mentioned in Fig. 6.15, the mutation operation is done in such a way that resource 2 (labelled as R3) will not be in use consecutively.

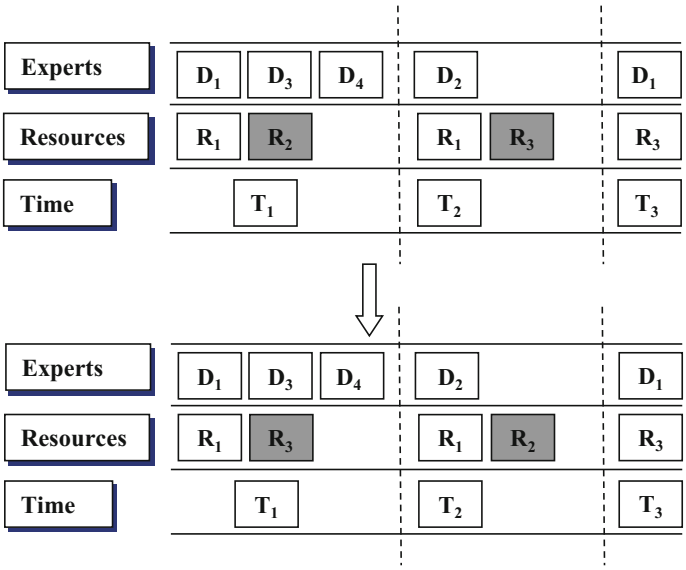



**Fig. 6.15** Mutation operation on encoded individuals

### 6.8.4.3 Crossover

In crossover of resources, experts can also be possible in order to generate new individuals. The crossover function takes two individuals and interchanges the same number of resources and experts to generate resources. It is advisable to check the validity of the resulting individuals after applying the crossover function.

After a sufficient number of crossover and mutation operations, the population might have reached its saturation level. Any further operation might result in poorer candidates or show no improvement. Here, one can stop the process of evolution.

**Real World Case 3: Environmental Monitoring**
The objective is to perform measurements for pollution in city environments. In particular, outdoor environmental measurements for temperature, humidity and air pollution are included. Furthermore, some sensors are used to obtain such information. Mobile sensors on vehicles will be also utilized. A genetic algorithm can be adapted to the actual conditions of environmental problems, and then it can be used in environmental monitoring and environmental quality assessment.

### 6.8.5 *Other Applications*

Besides the healthcare domain, there are other domains where the genetic algorithms are used effectively. Some of the application areas are listed below:

- On the platform of Internet, there are lots of data available; much of these data are not directly useful. Consider the email correspondence between senior employees of an organization. Such a conversation contains high quality knowledge about the business. However, it may have some irrelevant information about sentiments and current trends. Further, the emails all differ in structure as far as content is considered. Other fields such as sender, receiver and subject of the mail are structured, but the mail content is in free form. Besides email, there are social network platforms, blogs and other websites also available on the same platform. Emails may refer to one of these in its content. Genetic algorithms can be helpful in managing these data by designing appropriate clustering algorithms. Collections of such data sets are available at Stanford University (Stanford Large Network Dataset Collection)[1] and at CMU,[2] which is an Enron Email Dataset prepared by the CALO (A Cognitive Assistant that Learns and Organizes) Project.
- Similar (to the above-mentioned one) applications of a genetic algorithm can be citation network analysis, collaborative platform network (such as Wikipedia) analysis, the Amazon networks, and online communities. Clustering, scheduling, analysis and visualization can be done using genetic algorithms on these domains.
- Just as clustering of knowledge or clustering of users as mentioned in previous example can occur, clustering can also be done on the Internet platform for various applications such as twitter, mail, chat, blog and other domains, and e-commerce users can also be clustered into various groups. E-commerce shopping agents/guide programmes may use such knowledge in proactively suggesting new products and assign in overall shopping. Furthermore, product feedback, promotion, product price optimization, and suggestions can also be manipulated. Even if their data are not on the Web, large and data-centric companies may also utilize genetic algorithms for analytics.
- Data mining and web mining are generic applications where the genetic algorithms can be utilized to mine complex hidden patterns on the data warehouses and the Web. For example, from large warehouses (which may be distributed in nature) or from voluminous repository of data, genetic algorithms can mine useful patterns or classification rules. Similarly, genetic algorithms can also be useful in learning concepts for large repositories. Such work is done by Jing Gao, et al. (2008).
- Finance and investment-related data are also complicated, as many parameters affect them simultaneously and continuously. Genetic algorithms may help in forecasting via the learning of complex patterns within the domain.

[1] http://snap.stanford.edu/data/

[2] http://www.cs.cmu.edu/~enron/

- In the area of automotive and engineering design, genetic algorithms can be used to find the optimum combination of best materials and best engineering design to provide faster, fuel efficient, safe and lightweight vehicles. Similar evolving hardware design concepts can also be used for the engineering data domain.
- Telecommunication data are also complex enough to handle. Genetic algorithms can be used to search for the best path and efficient routing for telephone calls. Genetic algorithms can also be utilized to optimize placement and routing of cell towers for best coverage and ease of switching. Similar applications can be designed in the fields of tourism and traffic manipulation.
- For computer gaming, even if categorized as formal tasks category because of well-defined rules (e.g., chess) future actions of the game can be suggested by the application of genetic algorithms.

Besides the above-mentioned applications in various domains, genetic algorithms may be used in scheduling, clustering, prediction, forecasting, and searching in general, by applying a human-like heuristic approach to solve challenging problems in the most efficient way.

**Real World Case 4: Gene Expression Profiling**
The development of microarray technology for taking snapshots of the genes being expressed in a cell or group of cells has been an advantage to medical research. These profiles can, for instance, distinguish between cells that are actively dividing, or show how the cells react to a particular treatment. Generic algorithms are being developed to make analysis of gene expression profiles much faster and simpler. This helps to classify what genes play a part in many diseases, and can assist in identifying genetic causes for the development of diseases. This is a key step in personalized medicine.

## 6.9 Exercises

1. Give a brief overview of a general evolutionary algorithm. Also, what are the similarities and differences between genetic algorithms and evolution strategies?
2. Explain the role of parameters of genetic algorithm that control mutation and crossover.
3. Given the following parents, $P_1$ and $P_2$, and the template T

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{P_1}$ | A | B | C | D | E | F | G | H | I | J |
| $\mathbf{P_2}$ | E | F | J | H | B | C | I | A | D | G |
| **T** | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

Show how the following crossover operators work:

- uniform crossover
- order-based crossover

with regards to genetic algorithms.

Please note that uniform crossover uses a fixed mixing ratio between two parents to enable the parent chromosomes to contribute at the gene level rather than the segment level. For example, the mixing ratio is fixed to 0.5; then the offspring will have exactly half of the genes from one parent and the remaining half from the second parent. One may choose the crossover point randomly.

4. State two aspects of a genetic algorithm's design that would cause a population to fast converge. Refer to the two fundamental stages of genetic algorithms (selection and reproduction) in your answer.
5. Explain how you would solve a travelling salesperson problem using evolutionary algorithms. Illustrate your answer with suitable examples and diagrams.
6. Describe your own idea about linking an evolutionary algorithm to any machine learning method.
7. (*Project*) When we are trying to create an efficient portfolio of stocks, we must consider some important factors. The problem is that the evaluation contains several qualitative factors, which causes most approximations to go off track. Explore a genetic algorithm approach to portfolio evaluation. By using a set of fitness heuristics over a population of stock portfolios, the objective is to find a portfolio that has a high expected return over investment.
8. (*Project*) Investigate different ways to evolve neural networks. Start with comparing techniques of creating neural networks using genetic programming (GP). For instance, the function set in the GP can be utilized to create a directed graph containing the inputs, the output and any number of intermediate nodes.
9. (*Project*) Online social networks have generated great expectations in the context of their business value. Promotional campaigns implemented in web-based social networks are growing in popularity due to a rising number of users in online communities. In order to optimize a marketing campaign, explore hybrid predictive models based on clustering algorithm, genetic algorithms and decision trees.

## References

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*. New York: Wiley.

Gao, J., Ding, B., Fan, W., Han, J., & Yu, P. (2008). Classifying data streams with skewed class distributions and concept drifts. *IEEE Internet Computing, Special Issue on Data Stream Management*, pp. 37–49.

Hayes, G. (2007, October 9). *Genetic algorithm and genetic programming.* Retrieved October 27, 2015, from http://www.inf.ed.ac.uk/teaching/courses/gagp/slides07/gagplect6.pdf

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Cambridge: The MIT Press.

Podgorelec, V., & Kokol, P. (1997). Genetic algorithm based system for patient scheduling in highly constrained situations. *Journal of Medical Systems, 21*, 417–447.