Genetic Algorithms in Traveling Salesman Problem

Nevila XOXA
Albanian Academy of Science
Tirana
nevila.xoxa@akad.gov.al

Valbona DHJAKU
Credins Bank
Tirana
vdhjaku@gmail.com

Igli TAFA
Polytechnic University of Tirana
Information Technology Faculty
Computer Engineering Department
itafaj@gmail.com

Abstract

Genetic algorithms are a revolutionary technique which used operators like mutation, crossover and selection in resolving optimization problems. They have been used with success in multiple problems. The TSP (Traveling Salesman Problem) is one of these problems. It consists in finding the route with minimal length, passing by every node of a weighted graph only once. This problem is found in many real world applications therefore a good solution would be helpful. Many methods have been used in finding the best solution for the TSP but we are going to use genetic algorithms as such solution. In order to prove it we will simulate it in a test environment to watch from close the way it works and the efficiency of this algorithm in resolving our problem.

1 Presentation

Genetic algorithms are an optimization technique based on natural evolution, which includes the concepts of natural selection into a searching algorithm and offers an acceptable solution without the need of calculating all the options. Genetic algorithms are based on the concept of natural selection [Fal98]. In nature, the most adopted individuals have more survival chances and therefore even their children are more adopted and healthier than their parents.

The same idea will be applied in problems supposing as a start a group of solutions and later combining all most suitable solutions to create a new generation of solutions getting closer and closer to the required one. Genetic algorithm consists of the followings steps:

- Encoding
- Evaluation
- Selection
- Crossover
- Mutation
- Decoding

A suitable encoding is found by thinking that every solution to our problem has a unique code, in the shape of a vector or a matrix. After that a starting population is created in a completely random way. For each individual of this population a natural selection level is calculated which is used as coefficient to compare it with the other individuals and finding out which is closer to the ideal solution. Through these coefficients are used to select the individuals that will be crossed over with each other.

The crossover is the process where 2 individuals are combined to create new ones, which become part of the new generation. After that the mutation happens. Some randomly selected individuals are mutated, which means that a character of the vector is changed leading to the creation of new individuals (and therefore a new generation). This process goes on until a stopping condition is achieved. At this point the individual who is closer to the final solution is decoded and the process ends.

2. Standard Genetic Algorithm

We are given a problem with a well defined solution. The preliminary process includes finding a way of presenting such a predicted by the encoding. In this situation a genetic algorithm would work like below:

1. We start by generating a population of random solutions with n candidates of length 1 bit. (the genes)
2. We calculate function f(x) of the natural selection potential of each candidate (chromosome) of the population.
3. Repeat the steps until n descendant are created:

a. Choose a pair of chromosomes from this population, with a probability of selection as in ascending order. Each chromosome can be selected more then once as parent of the selected couple.

b. With probability pc (crossover probability) couples are crossed with each other in the random selected point in order to produce 2 descendants. If we don't have a successful crossover we can take a copy of the 2 parent and consider them as descendants.

c. Mutate each descendant in the selected gene with probability pm (mutation probability)

and put the new descendants in the new population. If n is odd one of the descendants will be randomly removed.

4. Replace the old population with the new one.
5. Restart from point2.

Each iteration of this process is called a generation.

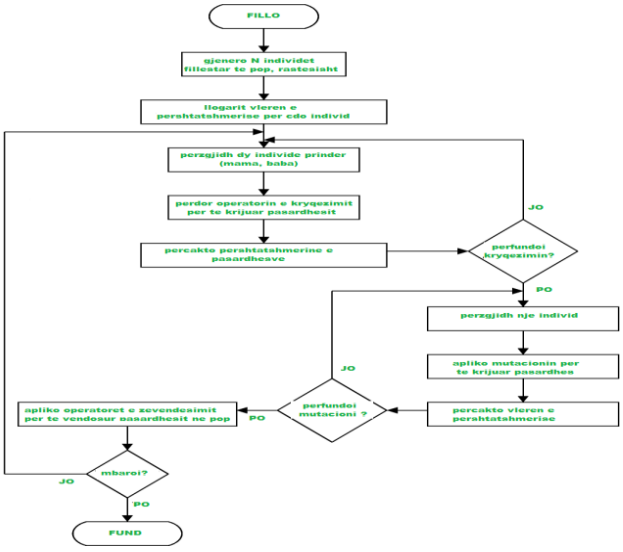


Figure 1: The evolution of a generic algorithm

3. Basic Concepts

Genetic algorithms can vary from straightforward to very difficult to understand. Before we proceed let us give a simple example of how they work. We need to maximize a function $f = -2x^2 + 4x - 5$ from a vector of whole numbers {0, 1 …15}. From straight calculation we can find that max (f) is reached for x=1.

3.1 Encoding

The encoding process is usually the hardest one in resolving a problem with genetic algorithms [Mit91]. When applied in a specific problem it is usually hard to find a correct presentation of the solution. Taking into consideration that we have to encode many possible solutions to create a population the easiest way to represent it is through a vector of 0's and 1's. Anyway based on the kind of characters used for presentation encoding is divided in 4 groups:

– Binary Encoding
  Usually used in problems with small complicity or when the solution is expected to be a number.

Chromosome1	110101110010
Chromosome2	011010011101

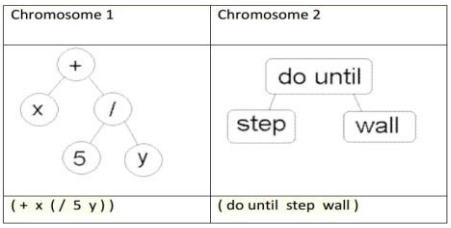
– Encoding with real numbers
  Used in problems that requires optimization of functions and classified as the most convenient [Wro96].

Chromosome1	1 5 2 3 5 2 6 4 6 9 8
Chromosome2	8 6 3 6 3 9 6 3 1 5 8

– Character Encoding [Rei94]. Best solution for combinatory problems.

Chromosome1	1.23, 2.12, 3.14, 0.34, 4.62
Chromosome2	ABDJEIFJDHDDLDFLFEGT

– Encoding with structure of data [Mic94]. Used in real world problems



Let's take into consideration the problem above. Our possible solutions are clearly numbers, therefore we use binary encoding. Therefore to represent numbers from 0 to 15 we need 4 bits which in genetic algorithms will be called genes and the vectors formed by then will be called chromosomes.

For example: 1 → 0001 . . . . . 15 → 1111

Now we in a random way we generate a population from these chromosomes.

3.2 Selection

The main principle on which genetic algorithms is essentially Darwinian natural selection. Selection provides a driving force in genetic algorithms. Very strongly, genetic research will be completed ahead of time, with little power, progress of evolution will be slower than usual. Usually, a pressure lower selection suggested early genetic research in favor of an exploration of wider space research, while a high pressure selection is recommended at the end of genetic research to narrow the space research, also achieving convergence solution. Selecting drives genetic research towards a promising area of research in space [Psg91]. During the past two decades, more selection methods are proposed, examined, compared listed below:

– Roulette wheel selection
– $(\mu + \lambda)$- selection
– Tournament selection
– Steady- state reproduction

- Ranking and scaling
- Sharing

Roulette wheel selection, proposed by Holland is the best type of selection known so far. The basic idea of this method is to determine the probability of selection or survival probability for each chromosome in proportion to the value of adaptability. So a roulette wheel model can be made to the appearance of these probabilities. Then the selection process is done by turning the wheel as often as population size, each time when a chromosome selected only for the young population. Wheels present stochastic method as a procedure sample. Baker proposed a universal stochastic method which requires only one rotation. Wheel is modeled like a roulette wheel, with an equal number of areas with the population. The basic strategy which is built on this approach is to maintain the expected number of copies of chromosomes in the new generation.

In contrast to proportional selection, selection $(\mu + \lambda)$ and (μ, λ) of the proposed Back deterministic procedures which are selecting the best chromosomes from parents and offspring. We note that the two methods of selection prevent duplicated chromosomes from the population, many researchers prefer to use this method to work with combinatorial optimization problems [Gol89]. Trunkan selection and blocking it are also regarding the procedures stochastic which classify individuals based on eligibility and select the best parents. Elite selection of commonly used as supplementary selections chromosomes proportional to preserve the best of the new generation, though not selected during the process of selection proportional.

Distribution techniques, presented by Richardson for a Golberg and multimodal optimized functions, are used to maintain the diversity of the population. A distribution function is a way to determine the degradation of the eligibility of an individual from a neighboring individual in a certain distance. Degradation, the probability of reproducing individuals within a community falls while other individuals are encouraged to japing successor.

In our problem, simplicity and efficiency that will be used wheeling roulette, in which a group of individuals will choose randomly, but the appropriateness of assessing proportionality in the section above.

3.3 Crossover

Crossover may be eligible a straightforward procedure. In our example, which used the simplest case crossover, we randomly select two chromosomes to cross, I randomly select an intersection point, and then we exchange all the genes that are after that point. In our case:

v1 = 0111

v2 = 1100

We can suppose that the crossover point is randomly selected after the 2nd gene.

v1 = 01|11

v2 = 11|00

By exchanging genes we would have:

v1’ = 01|00 = 4

v2’ = 11|11= 15

Now we have 2 new chromosomes which will be moved to create the new population.

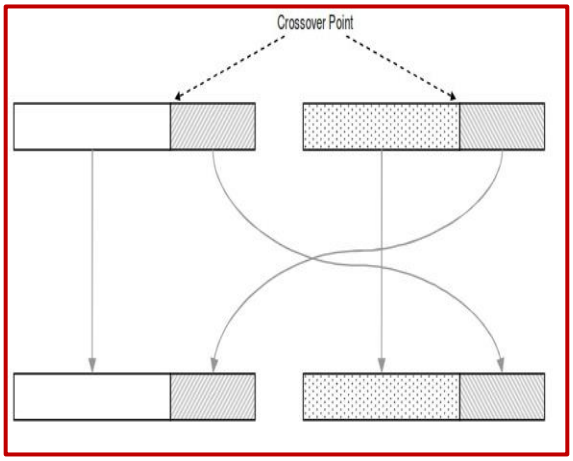


Figure 2: Crossover process with one point

Not every junction chromosome used. We reiterated that the function of assessment gives or any chromosomal 'points' that are used to determine the probability that any chromosome crosses, with the help of selection mentioned above, chromosomes are selected to be crossed by chance and the greatest opportunity is given to those with 'points' higher. We use distribution collection created in the evaluation process to select chromosomes by various selection methods. Generate random numbers between 0 and 1 and choose which chromosome corresponds to our distribution. We repeat to find the second and then the intersection of two young individuals become part of the new generation. The process continues until a new generation is filled. Not necessarily better than the first settler.

There are many types’ intersection routines, some of which would later face. Sometimes we need us move the intersection routines to ensure that the chromosomes do not conclude logically incorrect.

3.4 Mutation

Mutation is the technique which enables us to process not stalled local optimization. As a result of the process to chance, occasionally we have chromosomes from optimal local close but not that global. For this reason chromosomes from optimal local close to the junction will be solved because they will have greater adaptability, but then the chances will be smaller to achieve global from optimal. So completely random mutation as a way remains the only option for finding a possible solution. Mutation is applied after the intersection in one of the new generation of chromosomes. Then randomly selected a point (gene) on chromosome selected and change it. For demonstration, in our example we have:

v1 = 0111

If we suppose that randomly has been selected as mutation point the 3rd gene then v1 would become:

v1' = 0101

Another inversion is a form of mutation. It is usually used in special cases which will see later. Now we will demonstrate inversion in the example we have taken. Inversion consists of random selection of two points Inverted in vector and then every bit For example:

v2 = 1100

We select 2 points:

v2 = 1|10|0

Invert the 2 genes:

v2' = 1|01|0

If we had a bigger chromosome then we would have:

v3= 101101101001

v3= 101|101101|001

v3'= 101|010010|001

3.5 Decoding

Decoding is the last step to be followed in an algorithm genetic, it's just the reverse process of coding, where the final data or response algorithm to a problem, which come in the form of selected coding return a response accessible to the user or process. In this case, it made the transition from binary to decimal conversion by the basic rules of the way.

4. Traveling Salesman Problem

4.1 Entry

The problem of vendors traveling (Traveling Salesman Problem, TSP) addresses the problem of finding a way to visit a number of cities, with the proviso that each city is visited only once, the journey ends in the city of the left and this length be much smaller. The first example of TSP was given by Euler in 1759, whose problem was to displace the horse in any chess position only once. Fame took in a book written by a German seller BF Voigt in 1832 how to become a successful traveling salesman. He cited TSP, not with the same name, but suggested that an important aspect was that every city to be visited only once.

Standard problem or called as itinerant vendor's symmetrical problem can be expressed mathematically as follows:

Given a weighted graph G = (V, E) where weights CIJ crossing between nodes i and j is not a negative value, find any connection node to have a minimum total cost.

Currently the only way to guarantee the optimal solution to this problem of any magnitude is calculated every opportunity connection between them and finding it with lesser cost [Hgl92]. Any point of contact for a number of towns in particular need calculation of n! Possibilities of connections, with a growing number of cities done this method inefficient, and unable to find the cost of each link in polynomial time.

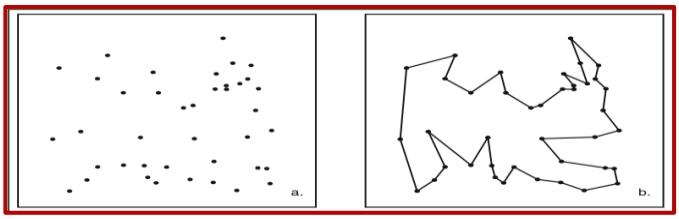


Figure 3: Example of a TSP problem

4.2 Why TSP? Applications

The problem of itinerant vendors there are plenty of different applications in the real world, which makes it a very common problem to be solved. Here we will explain some of these applications. For example, some car router problems can be addressed and resolved as TSP. Here the problem lies in finding out who the client will be served and by whom the car, and the minimum number of machines necessary to serve each customer. There are many variations and problems involving minimum time to serve each customer. We can deal with such problems in the form of a TSP-art.

Hardware problem connecting with "wireless" can be modeled as a TSP. We have a set of modules, each with a certain number of pin. We need to connect pins with several groups of conductors to each other, but each pin apogee is not to have more than two connections and the length of the conductor is minimized.

Another found an application by Plate, Lowe and Chandrasekaran control in aircraft gas turbines. Consisting of several wind sensor located in the perimeter and located on each level of the turbine to ensure a uniform flow of gas. The positioning of wind indicators in order to minimize the fuel consumption can be modeled as a symmetric TSP problem.

Turn the work of a single machine with the time given to each work and preparation for any work time is also a TSP. We can minimize the work of a total aligning things in the right order.

A robot must perform a set of operations to meet a process. In this application, compared with sequences of a machine works, we have advantages and limitations which can be viewed as an asymmetric TSP [Jtr94].

Problems related to the acceleration of the work of drilling electronic circuits, which consists Opening of holes with different diameter for various electronic elements, changing the head of drill spends more time with it to a factory cost counties. Control of the work of the drilling head can be treated as TSP.

As these and many other applications give a particular importance to this problem, as in aeronautics, robotics, industry, transport, telecommunications etc.

5. Genetic Algorithms As Solution For TSP

5.1 Adaptation

One of the methods for optimization is also tsp GA (Genetic Algorithms). In the first chapter presented in general terms how this method proceeds to the provision of an optimal solution to a problem. In this part we will focus on how helps an algorithm based on natural evolution for solving a problem so large it is TSP-ja. Step will adapt to any functional link genetic algorithms TSP-in.

5.1.1 Adapted Encoding

We saw above basic ways of encoding a basic problem using vectors to 0 and 1, who presented numbers in binary form. The question arises how to introduce so easily malleable and efficient problem of itinerant vendors? In the same way we can use vectors encoding numbers as 12345, letters ABCDE vectors, also any other form of strings of symbols, enough to make sense of the problem. Below we present the main ways of presenting the TSP's so malleable by genetic algorithms [Che00].

Matrix

Because in itself a problem as TSP-ja is itself a graph. We can create a matrix of connections, which consists of one and zero, where 1 defines the connection node of the j-in and 0 division. We can then treat this matrix in unmodified form or working with extending linearly according to the ranks.

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Vectorial

Form the first to introduce the problem of itinerant vectors symbols can be cyclic, in the form of vectors:

V = b1 b2 ... bn...

Where mug to the value implied by the position vector in that position, which means the tour goes from city to city bi. For example, the vector v1 = 3421 is meant as a tour that starts from city 1 to city 3, then from 3 to City 2 city, then the city 2 to 4 and from city to city 4 City 1. It should be noted that in presentation this way not every possible combination of numbers in the vector sense, such as those that create closed loops without going into any city such as v2 = 3412.

5.1.2 Adapted Evaluation

The evaluation process of individuals to create, which in this case are the links of successive tour presented, there will be nothing but the sum of each distance, cost or weight from node to node. Given that our main task is based on achieving the problem of finding a shortest tour, the intention of solving this problem will tend to minimize this assessment. Maintaining these distances or weights between nodes can be calculated as distance Euclidian or wasted searching algorithm can be stored in matrix form by eliminating their need occasional calculation.

5.1.3 Adapted Selection

As we mentioned, the selection is one of the most important operators in genetic algorithms [Llk86]. Given that the goal is minimizing the tour, we ask individuals, which in our case represent, with probability greater to be selected to be those with greater conformity, then the distance of the tour, the smallest defined by the evaluation operator. All selection methods can be used, but as noted in the chapter of our inquiry the best selection of selector wheel solves roulette. Assessment determines those individuals were part of the wheel which then 'rotate'

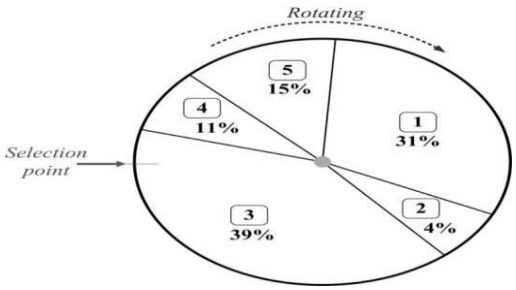


Figure 4: Roulette wheel selection

5.1.4 Adapted Crossover

We will start with the first partial junction (partially matched crossover PMX). Assume that we are using vector encoding type numbers, also reuse the intersection with two points mentioned in the first chapter.

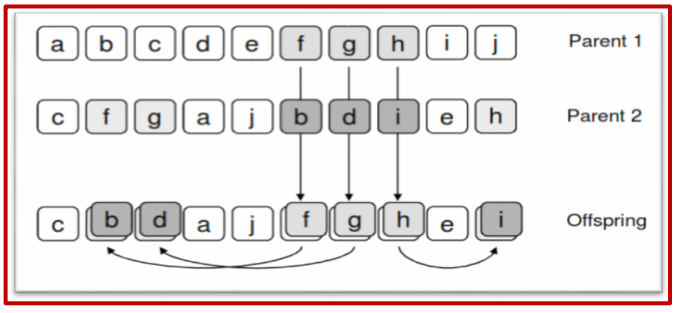


Figure 5: Partial Crossover

If we would have 2 vectors:

V1 = 1234 | 567 | 8

V2 = 8521 | 364 | 7

Crossover would be like:

V1' = 1234 | 364 | 8

V2' = 8521 | 567 | 7

Cyclic crossover CX works in a completely different way. First, this type of intersection can be used only with the presentation in the order shown, namely that where the numbers are placed in order of visits.

In this type of intersection point we do not choose at all. First choose one gene from their parents:

V1 = 12345678

V2 = 85213647

V2' = 82315647

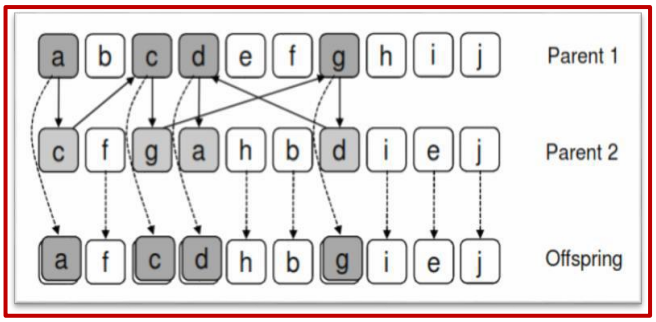


Figure 6: Cyclic Crossover

5.1.4 Adapted Mutation

So that mutations committed to provide seed even more tailored, instead of leaving the process up to chance, as noted in chapter 1, the problem of vendors strolling us come to the aid of algorithms servo controller for selecting the 'gene' that will undergo change, presented in section 2.4.2 of regulators the tour.

How early will revisit the 2-opt operators. Randomly select two connections (a, b) and (b, c) from our tour and check if we can find another way to link these four joints in order to achieve a lower cost. To do this check if:

Cab + CCD> Cac + CDB

If the inequality completed Casualty replace connections (a, b) and (c, d) with the new links (a, c) and (d, b). We note that it was assumed that a, b, c, d displayed in the order of the tour though b to c are not connected. We also have 3-opt operator who controls randomly selected 3 links instead of two. If we have connections (a, b), (c, d) and (e, f), check whether:

Cab + CCD + Cef > Cac + CBE + CDF

If completed on like is replaced connections for these 6 nods.

6. Simulations with Different Algorithms

	Nr of nodes	Optimal Result	Best approximate result		Average result from GA	
Bay29	29	9074	9074	0.00 %	9075	0.01 %
Eli51	51	426	434	1.87 %	441	3.52 %
Berlin 52	52	7542	7544	0.02 %	7774	3.07 %
KroA 100	100	21282	21600	1.49 %	22445	5.46 %
KroA 200	200	29368	31864	8.49 %	32399	10.32 %

Conclusions

Genetic algorithms, their development, are playing a very important role in resolving various problem-based approach and anticipation. They have led an evolutionary step forward calculation, and as term comprehensive computerization. Genetic algorithms, through its junction operators, mutation and selection enhanced algorithm so create a more flexible and suitable for any application.

The problem of itinerant vendors, a conceptual basis of a problem that arises in many real-life applications. Center of attention of many mathematicians studies because of its importance, has expanded so much in terms of different types of treatment, as well as the techniques and methods needed to solve it [Aff09].

In our subject matter treated as the use of genetic algorithms to find an approximation to the problem of itinerant vendors and based on simulations performed, genetic algorithms constitute a relatively good proxy tool to the problem of itinerant vendors. He responds ideally to smaller nodes number 30 also gives a satisfactory approximation and faster even more problems with joints. As we see in the summary table relative dependence calculated approximation does not exist only in relation to the number of nodes, but also their position and weight. A look at the results given by the algorithm to Berlin 52 Eli51 and draw the conclusion that genetic algorithm shows more efficiency in large instance. Problems where accuracy is the primary element, a greater number generation need to get the best of him.

As in many other applications, genetic algorithms can be used effectively for obtaining a satisfactory approximation to the problem of itinerant vendors [Hau04]. As a method which is based on more research in the future, an even higher performance can be achieved by him.

References

[Fal98] Emanuel Falkenauer. Genetic Algorithms and Grouping Problems. John Wiley and Sons, 1998.

[Gol89] David E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning.Addison-Wesley, 1989.

[Hgl92] Abdollah Homaifar, Shanguchuan Guan, and Gunar E. Liepins. Schema analysis of the traveling salesman problem using genetic algorithms. Complex Systems, 1992.

[Psg91] Prasanna Jog, Jung Y. Suh, and Dirk Van Gucht. Parallel genetic algorithms applied to the traveling salesman problem. SIAM Journal of Optimization, 1991.

[Jtr94] Michael Junger, Stefan Thienel, and Gerard Reinelt. Provably good solutions for the traveling salesman problem. Mathematical Methods of Operations Research, 1994.

[Llk86] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. The Traveling Salesman. John Wiley and Sons, 1986.

[Mic94] Zbigniew Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 2nd edition, 1994.

[Rei94] Gerard Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications. Springer- Verlag, 1994.

[Wro96] Jakub Wroblewski. Theoretical foundations of order-based genetic algorithms. Fundamenta Informaticae, 1996.

[Che00] Genetic.Algorithms.and.EngineeringOptimization M.Gen R.Cheng (Wiley_2000)

[Mit91] An.Introduction.to.Genetic.Algorithms 5ed M.Mitchell, 1991

[Aff09] Genetic.Algorithms.and.Genetic.Programming , M.Affenzeller, 2009

[Hau04] Practical.Genetic.Algorithms 2ed, R.L.Haupt dhe S.E.Haupt (Wiley_2004)