

Genetic Algorithm Solution of the Knapsack Problem Used in Finding Full Issues in the Holy Quran Based on the Number (19)

Fadi A. O. Najadat¹, Ghassan G. Kanaan¹, Raed K. Kanaan¹, Omar S. Aldabbas¹ & Riyad F. Al-Shalabi¹

¹University of Banking and Financial Sciences, Amman, Jordan

Correspondence: Fadi A. O. Najadat, University of Banking and Financial Sciences, Amman, Jordan. E-mail: fadi_najadat@yahoo.com

Received: December 31, 2012 Accepted: February 3, 2013 Online Published: March 4, 2013

doi:10.5539/v6n2p18

URL: <http://dx.doi.org/10.5539/v6n2p18>

Abstract

The Holy Quran is the biggest Miracle of Muslims everywhere and at every time; therefore, it is valid for every time and place. Actually, researches and studies into the Holy Quran that aim to uncover new miracles within are considered as a kind of worship for Muslims researchers since it facilitates the Islamic mission and clarifies the vague picture of Islam throughout the world. From this perspective, the researchers have selected the vague miracle of the number 19 in the Holy Quran to examine through this study.

In this study, the researchers describe a special algorithm that facilitates the retrieval process of the full-issues according to the conditions of number 19 miracle in the Holy Quran. This algorithm is derived from the genetic algorithm that is used to solve the knapsack problem, since it is so similar to the case of this study. However, in the results of this study many full-issues have appeared that insure the truth of the number 19 miracle in the Holy Quran and thus support the miracle of the whole Holy Quran.

Keywords: information retrieval (IR), genetic algorithm (GA), knapsack problem (KP), number 19 in Holy Quran, full issues in the Quran

1. Introduction

The religion of Islam motivates the principles of thinking and scientific research; since it considers these principles as a presentation of Muslims full faith in Allah almighty, angels, the Holy Quran and other divine books, the prophet Mohammad, peace be upon him, and all of the previous prophets, doomsday, and fate. Moreover, the Holy Quran is not only considered to be the holy book of the Islamic religion, but it is also considered to be the miracle of the prophet Mohammad, peace be upon him. In fact, the Holy Quran is considered an immortal miracle since it speaks to the minds at every time and place (Al-Kaheel, 2010). Therefore, reading and understanding the hidden meanings in verses of the Holy Quran is one of the most valuable acts of worship. In addition, the understanding of the Holy Quran meanings is one of the significant keys for the discovery of the hidden miracles of the Holy Quran. Moreover, discovering new miracles will lead to an increase in the faith of the Islamic religion. In addition, it provides new proof of the truth of the Holy Quran and the truth of the Islamic religion based on it (Al-Refae'i, 2006).

Al-Refae'i (2006) has studied the Holy Quran from a semantic perspective in order to get as much understanding as possible of the hidden meaning of the Holy Quran. This perspective of Al-Refae'i has arisen from the fact of the miracle in the formulation of the words and verses in the Holy Quran in addition to the miracle of the arrangement of the words and verses within this miracle book (Al-Refae'i, 2006). The Holy Quran is considered to be a miraculous book. Therefore, the Holy Quran has a huge number of miracles, including the arrangement of the words in the verses, their numbers, and the Ottoman writing of the words (Jarrar, 2001; Al-Kabbani, 2004). From this perspective, Al-Refae'i has focused on the Ottoman writing of the words, since it appears that the same word can be written in different shapes according to its place and meaning (Al-Refae'i, 2006).

2. Number 19 in the Holy Quran

The Holy Quran has a lot of miracles that are still undiscovered until this day. Surely, there is no doubt that the miracles of Holy Quran will stay mysterious until the last day of worldly life, since the Holy Quran is considered as a miracle for every place and at every time (Al-Refae'i, 2006; Jarrar, 2001). This does not mean that the miracles of the Holy Quran are non-detectable, but it means that each time and place will help researchers in the

field of the Holy Quran discovering and identifying new miracles in this holy book. Therefore, in this era of technological developments that change everything in human life, one of the newly discovered miracles is the numeric secrets in the Holy Quran, especially the secrets of the number 19. The Holy Quran contains many verses, which insures that there is a numeric miracle in this great book; consider the following verse:

(كَيْتَبُ مَرْفُومٍ) (المطففين، Sura # 83، Verse # 9)

The number (19) is considered as a key secret code to discover new miracles from the Holy Quran since it has been agreed that the secret numeric code of the Holy Quran is built on the basis of the number (19). This miracle insures that the Holy Quran has a special arrangement that protects the whole Quran from the attempt to change any letter in this great book as the following verse says:

(إِنَّا نَحْنُ نَزَّلْنَا الذِّكْرَ وَإِنَّا لَهُ لَحَافِظُونَ) (الحجر، Sura # 15، Verse # 9)

There are many verses that indicate the existence of a miracle involving the number (19) in the Holy Quran; for example, the first verse in the Holy Quran consists of 19 letters exactly, as all can see in the following verse (Al-Kaheel, 2012):

(بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ) (الفاتحة، Sura# 1، verse# 1)

However, Al-Refae'i showed that giving numeric values for each word in the Holy Quran according to the frequency of each Arabic letter within is a very helpful step in analyzing the secret code behind the number (19) as shown in Table 1. The most repeated letter in the Holy Quran is found to be the (الالف) so it gains the weight of 1, the Arabic letter (اللام) gains the second level since it is repeated so many times that it has the weight of 2, and so on (Al-Refae'i, 2006; Hassan, 2011). Therefore, the weight of each word is calculated as the sum of the weights of the letters in that word. For instance, the word (الله) consists of (ا+ل+ل+ه), so the weight of the word (الله) is 12, which comes from replacing each letter by its weight as follows:

$$12 = (7 \leftarrow ه) + (2 \leftarrow ل) + (2 \leftarrow ل) + (1 \leftarrow ا)$$

For another instance, the word (محمد) consists of (م+ح+م+د); therefore, the weight of the word (محمد) will be 42, which comes from replacing each letter by its weight as follows:

$$42 = (16 \leftarrow د) + (4 \leftarrow م) + (18 \leftarrow ح) + (4 \leftarrow م)$$

Table 1. Weights of the Arabic letters (Al-Refae'i, 2006)

Letter	Weight	Letter	Weight
ا، ي، ء، أ	1	س	15
ل	2	د	16
ن	3	ذ	17
م	4	ح	18
و، ؤ	5	ج	19
ي، ئ، ث، يـ ^ا	6	خ	20
ه، ة	7	ش	21
ر	8	ص	22
ب	9	ض	23
ك	10	ز	24
ت	11	ث	25
ع	12	ط	26
ف	13	غ	27
ق	14	ظ	28

For a more complex example of the secret code in this miracle, the word (محمد) has the same weight as the word (رسول الله) which can be considered as a sub-miracle that has been discovered from the miracle of the number 19 in the Holy Quran. However, calculating the weight of each verse requires the calculating of the weight of each word in the verse then summing them to calculate the weight of the verse (Al-Refae'i, 2006).

According to Al-Refae'i (2006), the miracle of 19 in the Holy Quran appears when calculating the weights of

whole included verses that share the same topic in the Holy Quran. It appears that the total weight of a set of verses on the same topic has to be divisible by the number 19 to consider this set as a full issue. Surprisingly, it appears that the result of the division represents the weight of a related word or phrase. This can be very helpful, especially in finding suitable solutions in the explanation of the disputed issues in Al-Tafseer of the vague verses in the Holy Quran.

3. Some Full Issues

Al-Refae'i (2006) discovered many full issues that have appeared during his research such as the full issue of the word Mohammad (محمد) and the full issue of the word Quran (قران).

3.1 The Full Issue of the Word Mohammad (محمد)

According to Figure 1, the total weight of these verses is (798) which is divisible by the number 19 and gives the dividend of 42. It appears that these verses concern the prophet Mohammad, peace be upon him. From another point, the weight of the word Mohammad (محمد) is equal to the sum of the weights of the included letters (4+18+4+16=42). Multiplying 19 by the weight of the word Mohammad (محمد), which is 42, gives the result 798. Finally, multiplying 19 by the weight of Mohammad (محمد) is equal to the total weight of the verses in the class. Therefore, this full issue is about the prophet Mohammad although the word Mohammad (محمد) is not mentioned in the verses exactly (Al-Refae'i, 2006).

$$\begin{aligned} 266 &= \text{﴿ تَلَوْا عَلَيْهِمْ ءَايَاتِكَ وَيُعَلِّمُهُمُ الْكِتَابَ وَالْحِكْمَةَ وَيُزَكِّيهِمْ ﴾} \\ 269 &= \text{﴿ تَلَوْا عَلَيْكُمْ ءَايَاتِنَا وَيُزَكِّيكُمْ وَيُعَلِّمُكُمُ الْكِتَابَ وَالْحِكْمَةَ ﴾} \\ 263 &= \text{﴿ تَلَوْا عَلَيْهِمْ ءَايَاتِهِ وَيُزَكِّيهِمْ وَيُعَلِّمُهُمُ الْكِتَابَ وَالْحِكْمَةَ ﴾} \end{aligned}$$

Figure 1. The full issue of the word Mohammad (محمد) (Al-Refae'i, 2006)

3.2 The Full Issue of the Word Quran (قران)

According to Figure 2, the total weight of these verses is (494) which is divisible by the number 19 and gives the result of 26. It appears that these verses are concerned with the Holy Quran. From another perspective, the weight of the word Quran (قران) is equal to the sum of the weights of the included letters (14+8+1+3=26). By multiplying 19 by the weight of the word Quran (قران), which is 26, the result will be 494. Finally, multiplying 19 by the weight of (قران) is equal to the total weight of the verses in the class; therefore, this full issue is about the Holy Quran although the word Quran (قران) is not mentioned in the verses exactly (Al-Refae'i, 2006).

$$\begin{aligned} 159 &= [16 : \text{القيامة}] \text{﴿ لَا تُحَرِّكْ بِهِ لِسَانَكَ لِتَعْجَلَ بِهِ ﴾} \\ 108 &= [17 : \text{القيامة}] \text{﴿ إِنَّ عَلَيْنَا جَمْعَهُ وَقُرْءَانَهُ ﴾} \\ 144 &= [18 : \text{القيامة}] \text{﴿ فَإِذَا قَرَأْتَهُ فَانْبَعِثْهُ قُرْءَانَهُ ﴾} \\ 83 &= [19 : \text{القيامة}] \text{﴿ ثُمَّ إِنَّ عَلَيْنَا بَيِّنَاتِهِ ﴾} \end{aligned}$$

Figure 2. The full issue of the word Quran (قران) (Al-Refae'i, 2006)

There are many more examples of full issues throughout Al-Refae'i's book since this miracle explains many issues in the Holy Quran. Moreover, many vague issues have appeared through the research, such as the issue of Isra and Al Maraj that appeared to be an incomplete issue in regard to the miracle of (19) in the Holy Quran. Therefore, the process of search for the secrets of the Holy Quran does not stop with Al-Refae'i's research. Instead every Muslim researcher has an obligation to continue this important research (Al-Refae'i, 2006).

4. Genetic Algorithms

Genetic algorithms are considered to be an artificial intelligence application, since the science of artificial intelligence tries to simulate human behaviours using algorithms to create intelligent machines (El-Mihoub et al., 2006). A genetic algorithm is an algorithm that simulates the evolution of living organisms in a manner to contribute to the enhancement of computerized systems (Sudhakaran & Raj, 2010). Therefore, genetic algorithms are considered to be “Bio-Inspired” applications (Stein et al., 2004). Genetic algorithms were developed by a professor of computer science from the University of Michigan named John Holland. Actually, John Holland got the idea of genetic algorithms from Darwin’s theory of “origin of species”. In the year 1970, the first genetic algorithm was presented officially by John Holland in the United States of America. Since then, genetic algorithms have been used to solve many programming problems in the science of artificial intelligence in many different types of computerized systems (Thierens, 1999).

Genetic algorithms follow Darwin’s theory of “the origin of species” in order to simulate the evolution of living organisms in an intelligent computerized system. The theory of the origin of species explains the evolution of living organisms in terms of their genetic evolution; in other words, Darwin’s theory of the origin of species concentrates on comparing the genetic characteristics of the parents and grandparents with the genetic characteristics of their descendants in order to discover the evolution in their genes (Kuilekov et al., 2003). A genetic algorithm simulates the principles of Darwin’s theory in order to find an optimum solution from a huge number of solutions in a computerized system (Nemati, 1994). A genetic algorithm also can be defined as a search algorithm that operates in a heuristic manner based on ideas about the evolution and selection of living organisms (Kosorukoff, 2012). From another perspective, genetic algorithms can be defined as a branch of computerized evolution, which is considered as an area of artificial intelligence that uses the concepts of Darwin’s theory in order to find the fittest solution for a specific problem (Sandurkar & Chen, 1998).

In order to gain a full understanding of genetic algorithms, the biological side has to be explained briefly. Each living being consists of a set of cells in accordance with the biological perspective. In addition, each cell contains chromosomes which consist of genes. The genes are strings of Deoxyribonucleic acid (DNA); and so each gene is responsible for transferring a feature from parents to descendants such as eye color and other features (Mathew, 2002). The new generation of any living being takes all its features from its parents; half of the new generation’s genes come from one of the parents and the other half of its genes come from the other parent. The transferring of genes from parents to descendants is called the crossover process (Erben & Konstanz, 2012).

Actually, some errors may happen in the process of copying the parents’ genes to the new generation; these errors are called mutations (Bhaskar & Maheswarapu, 2011). In order to judge the descendants, they will be compared with the parents by calculating the fitness factor, which measures the success of new organisms in surviving and breeding (Gimeno & Nave, 2006). Genetic algorithm simulate the biological evolution in programming concepts with binary encoded strings, by randomly choosing half of the string from each parent and presenting the resulting strings as suggested solutions (Sugimoto, 2007).

In this way, genetic algorithms are used to solve many problems with a huge set of solutions and a requirement to find a solution in a short time such as the knapsack problem (Hristakeva & Shrestha, 2003). The knapsack problem aims to find a set of objects that have high value and low weight compared to the capacity of the knapsack (Taskiran, 2012). The solutions of the knapsack problem are presented in a binary encoding where the selected objects are represented by ‘1’ and the abandoned ones are represented by ‘0’ (Bortfeldt & Winter, 2012). Genetic algorithms are designed to find a local maximum solution for the problem, since generating the best solution will require a long period of time to find. To accomplish this, genetic algorithms use three main operators in order to build up a computerized evolutionary system for problem solving. These three operators are selection, crossover, and mutation.

4.1 Selection

The selection process is defined as the process of selecting solutions for a certain problem (Harik et al., 1999). The selection process happens randomly by using functions that generate zeroes and ones randomly. Therefore, genetic algorithms are considered to be random search processes, since the first step in the algorithm selects the solution features randomly. In addition, the random selection process makes genetic algorithms nondeterministic, since the selection process happens randomly and so the selected solution from the first run of the algorithm may be different than the result from another run. However, the random selection of solutions is followed by an evaluation for each solution with a fitness value determined by summing the values of its contents and then the fitter solutions are selected for the next step (Matthews, 2001).

The selection process aims to find suitable solutions for the next step by selecting solutions randomly then

choosing the fittest solutions. For example, in the knapsack problem the random selection of different strings of solutions is followed by calculating the values of the selected strings and comparing them with the capacity of the knapsack in order to get the fitness value for each solution, then the fittest solutions are carried on to the next step (Uyar et al., 2004; Shopova & Bancheva, 2006; Sivaraj & Ravichandran, 2011).

4.2 Crossover

The process of crossover is also a simulation of the genetic crossover in living organisms (Senaratna, 2005). Actually, the fitter solutions that selected from the first step will pass through the crossover step. Crossover means selecting the bits for the strings of new solutions by taking a half from each parent. After that, the fitness values of new solutions will be calculated in order to compare them with the parents' fitness values (Schmitt, 2001). The fitness values of new solutions give an indicator for comparing the descendants with parents in order to produce an optimum solution. The crossover process is viewed as a generation process, since it produces new solutions that may be better than their parents. In addition, the crossover step carries out an intelligent random selection process; since the randomly selected solutions are offer new solutions and the new solutions become parents for other new solutions as in the evolution process of living organisms (Yang, 2002; Paplinski, 2009; Penev, 2005).

4.3 Mutation

Mutation can be defined as a change of a bit form '0' to '1' or from '1' to '0' which leads to the creation of a new and different solution (Jung, 2009). In other words mutation creates new solutions that are different from their parents. In the biological perspective, mutation is often helpful, since it gives living organisms a chance to enhance their features. In genetic algorithms, the mutation process can contribute to new enhanced solutions; the mutation step requires measuring the fitness of new solutions (Bottaci, 2012). Also, mutation achieves the goal of evolutionary algorithms, since the mutation step can enhance the fitness value of new solutions. Therefore, mutation is a very significant step in the search process carried out by genetic algorithms (Korejo, 2009). There are many types of mutation such as one bit mutation, boundary mutation, non-uniform mutation, uniform mutation, and Gaussian mutation.

Those three operators of genetic algorithms are the most major steps for solving problems (Bryant & Benjamin, 2000).

5. Knapsack Problem

One of the problems that can be solved by a genetic algorithm is the knapsack problem; which is considered as one of the most complex problems in computer science (Lai, 2006). The knapsack problem can be defined as the problem of selecting a number of items to be put into a knapsack that has a certain capacity. The selected objects must have a high value as well as a suitable weight since the total weight cannot exceed the capacity of the knapsack (Dizdar et al., 2009). Therefore, the knapsack problem has a huge space of suggested solutions; and so, finding the best solution possible for the problem requires a long search time. Therefore, the knapsack problem needs a solution that gives an optimal solution in a short search time. However, the capacity of the knapsack is limited and the number of items is huge; therefore, the number of available solutions will be enormous.

Every item in the knapsack problem has a constant weight and a constant value. Typically, the suggested solutions of the knapsack problem are represented by strings of 0's and 1's. The size of each string in bits is equal to the number of items where each bit will represent one item. If the item is abandoned; the corresponding bit is represented by an s 0, otherwise, it is represented by a 1 (Sriram, 2012). The strings consist of only 0's and 1's, so a string represents each item with one bit whether it is selected or abandoned. That is,

$$S_i = \begin{cases} 0; & \text{if abandoned} \\ 1; & \text{if accepted} \end{cases} \quad (\text{Sriram, 2012}) \quad (1)$$

where:

S_i : Value of the i^{th} item value in the solution string

In order to select an item, the total weight of the selected items in a solution has to be equal to or less than the capacity W of the knapsack as Equation 2 demonstrates:

$$\begin{aligned} & \text{maximizes} \quad \sum_{i=1}^n v_i \\ & \text{Subject to} \quad \sum_{i=1}^n w_i \leq W \end{aligned} \quad (\text{Puchinger et al., 2007}) \quad (2)$$

where:

i : Item number

v_i : Value of item

w_i : Weight of item

W : Capacity of knapsack

It appears from (2) that the total weight of the items in the solution must be equal to or less than the capacity of the knapsack. Also, the total value of the items has to be the maximum obtainable one in order to select the solution. The crux of the problem is the huge number of available solutions, which require a long time to find the best solution (Puchinger et al., 2007). Actually, the knapsack problem presents a time problem, unless it is redefined to require only a suitable solution, not necessarily the best solution ever (Chekuri, 2009). It is worth mentioning that items cannot be divided into smaller pieces in the knapsack problem and so their values and weights also cannot be divided.

Each solution string for the knapsack problem has a total value and it also has a total weight (Chekuri, 2009). In order to reject the values that exceed the total capacity of knapsack, there is a value that is called the fitness value, which is calculated for each solution to give it a numeric score. The numeric scores will classify the solutions according to their suitability for the problem where the most suitable solution will be the solution that has the highest value with a total weight less than the capacity of the knapsack. Equation (3) demonstrates the process of calculating fitness values.

$$f_s = \begin{cases} \sum_{i=1}^n v_i; & \text{where } \sum_{i=1}^n w_i \leq W \\ W - \sum_{i=1}^n w_i; & \text{otherwise} \end{cases} \quad (\text{Garg et al., 2005}) \quad (3)$$

where:

f_s : Fitness value of solution

i : Item number

v_i : Value of item

w_i : Weight of item

W : Capacity of knapsack

According to (3), the fitness value of the solution will have a positive value equal to the total value of its items, if it satisfies the knapsack capacity condition, otherwise the fitness value will be stored as a negative value, so it can be rejected immediately (Tavares et al., 2005). Many researchers have tried to solve the problem of knapsack with genetic algorithms (Hristakeva & Shrestha, 2003). These genetic algorithms are designed to find a search method that can solve the knapsack problem in a short time.

Some of the algorithmic approaches that have been used in attempts to solve the knapsack problem are dynamic programming, brute force, branch and bound, greedy algorithms, memory functions, and genetic algorithms (Matousek, 2002). The most successful algorithm in finding a solution of the knapsack problem is the genetic algorithm since genetic algorithms optimize the total number of solutions, which leads to find a suitable solution of the problem in a quick time. Moreover, genetic algorithms are considered to be the fastest algorithm that solves knapsack problem with optimum solutions (Hristakeva & Shrestha, 2004). In addition, genetic algorithms are also characterized by using random search which implies that we consider genetic algorithms as nondeterministic. Therefore, the final result of the first run of the algorithm can be different than the final result for another run. However, genetic algorithms are also characterized by using the binary encoding method for finding solutions of knapsack problem. This also makes genetic algorithms the most suitable algorithmic approach to the knapsack problem (Gilbert & Eppstein, 2002). Finally, genetic algorithms are characterized by speed, optimizing problems, and many other features that make this approach suitable for the knapsack problem.

6. Genetic Algorithm Solution for the Knapsack Problem

According to Hristakeva and Shrestha (2003), genetic algorithm depends on random searching to solve problems that have huge amount of solutions. Therefore, genetic algorithm solves knapsack problem because this problem requires a suitable solution in a short manner of time. In addition, knapsack problem needs an algorithm that solves the problem with a suitable and optimum solution. Nevertheless, genetic algorithm presents the solutions of knapsack problem in a binary encoding where the selected item takes the '1' value and the abandoned item

takes the '0' value; and so, the solutions will be presented as binary strings of zeros and ones as Sipper (2009) pointed out. However, solutions will randomly selected regarding to the nature of genetic algorithm that followed a randomly search methods and then it tries to enhance the selected solutions in order to reach the highest possible fitness value as Molina et al. (2011) stated.

The process of selecting the solutions depends on generating random numbers between '0' and '1' values. After that, the solutions will be selected according to the generated random number as Chakaborty (2010) argued. Each solution has a fitness value that can be negative or positive; however, the initial population of solutions will delete the solutions of negative fitness values. Therefore, the initial population of solutions will only contain solutions that have positive fitness values which are considered as accepted ones. After that, the solutions that have the highest values of fitness will be selected in order to send them to the step of crossover as Carlos et al. (2010) stated. According to Djannaty and Doostdar (2008), the step of crossover is a reproduction step where the selected solutions are considered as parents' solution for the offspring solutions because offspring solutions will take the bits from the parents' solutions. After that, the fitness values of offspring solutions will be calculated in order to compare offspring solutions with the parents' solution and so the fitter solution will have a higher level than other solutions.

Enhancing the offspring solutions can be achieved through the step of mutation that changes a bit in the string of offspring solution in order to increase the fitness value. The step of mutation changes one bit from offspring solutions such as changing a '0' to '1'; therefore, the new solution will have a new fitness value. Fitness value of the new solution is an enhanced value comparing with the offspring solutions' fitness value. However, the mutation will enhance solutions and so it will decrease the total time that is needed to find the optimum solutions for the knapsack problem as Julstrom (2005) said. According to Martinjak and Golub (2006), the offspring solution will become parents in order to reproduce new solutions until genetic algorithm finds the optimum solution for knapsack problem.

According to Labeled et al. (2011), genetic algorithm firstly produces a specific number of individuals as an initial population which will be later considered as candidate solutions to the knapsack problem. The selection process produces the candidate solutions by a randomly searching process in order to find a specific number of solutions' strings. However, the randomly selection optimizes the space of available solutions that are suggested to solve certain problem. Therefore, the selected solutions have to be evaluated in order to find a measurement that can arrange the selected solutions by numeric values. The fitness value is the most suitable measurement tool that can be used to evaluate and arrange the selected solutions. It is worth to mention that, the fitness value is calculated in regards to the value and the weight of the selected items in a certain solutions. Moreover, the fitness value gives a negative values for the unacceptable solutions and positive values for the accepted one. The fitness values of accepted solutions represent the total value of the solutions; so that, the highest numeric value of fitness between solutions indicates that this solution is the fitter solution as Swetanisha and Mishra (2011) argued.

The fitter solutions will be selected to implement the crossover step on them in order to produce new solutions. The offspring solutions will also have fitness values in regards to their values and weights as Liu and Liu (2012) stated. The fitness values of offspring solutions may be higher than the fitness values of the parents' solutions. On the other hand, the fitness values of the offspring may be less than the fitness values of the parents' solutions. However, the fitter solutions will be selected to mutate them with one of the mutation types as the single bit mutation. After mutation applying to the offspring solutions, the fitness value will be recalculated again in order to know the acceptance degree of the new solutions. However, the new offspring solution will be the parents' solutions in order to regenerate new solutions as Sunanda and Garg (2009) pointed out.

It appears that genetic algorithm is a so suitable solution for knapsack problem since this algorithm optimizes the huge number of solutions that are available for solving knapsack problem. In addition to that, genetic algorithm solves knapsack problem by an evolutionary algorithm so that the selected solutions will be enhanced until the searching process reaches the deadline of genetic algorithm; the final solution will be the best solution from the other solutions as Gallardo et al. (2001) stated. However, there are other features that make genetic algorithm suits the knapsack problem as the quickness in finding solutions. Genetic algorithm distinguishes with the quickness feature which appears obviously when comparing it with the long time that taken by traditional solutions to solving knapsack problem as Alves and Almeida (2007) argued.

According to Taishan (2010), it is worth to mention how the genetic algorithm stops the searching process in order to gain a full understanding for the nature of knapsack problem's final solution. The deadline appears when the termination condition achieved. The termination condition is the stability of fitness value since the stability

of the fitness value means that the genetic algorithm is generating the same solution or similar solutions and so this value will be the best generated value. However, the solution of knapsack problem that is generated by genetic algorithm is considered as a suitable solution and also it is considered as the best of all tested solutions but not all solutions since it selects solutions randomly. According to Zorman et al. (2002), the termination process can also happen when a predetermined condition of an accepted solutions' range fitness value is achieved. Moreover, the genetic algorithm also terminates the flow of the algorithm when the maximum number of tries is exceeded in order to avoid entering infinite loops. However, all of termination conditions have to be implemented in the genetic algorithm that will be used for solving knapsack problem since each one of those conditions completes the other one as Han and Kim (2001) pointed out.

Genetic algorithm is considered to be a nondeterministic algorithm since it depends on random numbers; therefore, the final result of the first run of the program might be different than another run for the program as Mishra et al. (2005) stated. Using of genetic algorithm cannot assure that the final result will be the best solution for the knapsack problem, but it can assure that the final result will be an optimum solution for the knapsack problem since there is not any indication to know the best solution for the problem as Nabil et al. (2012) pointed out. However, it appears that genetic algorithm solves knapsack problem effectively in regards to the saving both of time and effort that are needed to solve the problem with the traditional searching methods and algorithm. Moreover, the ability of genetic algorithm to deal with binary encoding method also contributes in making the genetic algorithm suitable for solving knapsack problem as Hristakeva and Shrestha (2004) argued.

7. The Research Problem

The miracle of the number 19 in the Holy Quran is one of the most important miracles that contribute effectively to discovering new miracles in the Holy Quran, which can thus be considered as sub-miracles (Al-Refae'i, 2006). These sub-miracles contribute by enhancing the understanding of the hidden meanings in the verses of the Holy Quran. Actually, finding the full issues based on the conditions of number 19 miracle of the Holy Quran is a complex process, especially when the searching and calculating processes are executed manually. However, the miracle of the number 19 appeared in an era of technological developments in various fields, which indicates that this miracle needs a technological approach to find the full issues by using it throughout the Holy Quran. Complicated issues cannot be discovered using only manual techniques; therefore, the researchers designed a suitable program for applying an efficient algorithm that can search and find the full issues easily in order to discover new full issues. Therefore, the researchers constructed an algorithm based on the genetic algorithm that has been used to solve the knapsack problem with the goal of enhancing the search process according to special conditions related to the number 19 miracle.

8. Experiments and Results

First of all, the words of the Quran verses have been normalized in order to process the Ottoman script writing as the normal spelling to simplify the programming. First, the diacritical marks have been removed from the words. Then, some rules have been initialized in order to give each Arabic letter its right weight as was explained in Table 1. One difficulty is caused by the two Arabic letters (التاء المربوطة) and (التاء المفتوحة), since the computer system cannot differentiate between these two Arabic letters; although these two letters have different weights. The letter (التاء المربوطة) has a weight of 7 and the letter (التاء المفتوحة) has a weight of 11. This problem has been solved by using the ASCII code, since the code of (التاء المربوطة) is (201) and the code of (التاء المفتوحة) is (202); therefore, the right weight value can be calculated for these letters. Another example of weight modifications, in the Ottoman writing the Arabic letter (الياء) is written as the Arabic letter (الالف المقصورة) so that the problem that appears is the differences in the weights of the letter (الياء) and the letter (الالف المقصورة) since the letter (الياء) has the weight of (6) but the letter (الالف المقصورة) has the weight of (1). This problem has been solved by adding (5) for the total weight of the word.

In order to apply the theory of Al-Refae'i in a computerized system, a genetic algorithm that solves the knapsack problem has been used to help in finding new successful results. The following algorithm shows the steps that were implemented in this research:

Step 1: select a number of (m) solutions randomly from the retrieved verses, where the selected verse has a value of '1' and the unselected one has a value of '0'.

Step 2: for each solution, calculate the total weight.

Step 3: for each solution, calculate the value of the solution; if the weight is divisible by 19 then the value is equal to the weight, otherwise, the value is equal to (-1*the weight).

Step 4: select the two best solutions that have the highest values.

Step 5: for each one of the two best solutions, if the weight of the solution is divisible by 19 and the value of the solution is equal to (19*the desired weight) then go to step 14.

Step 6: crossover the two selected solutions.

Step 7: for each offspring solution, calculate the weight of the offspring solution.

Step 8: for each offspring solution, calculate the value of the solution; if the weight is divisible by 19 then the value is equal to the weight, otherwise, the value is equal to (-1*the weight).

Step 9: select the two best solutions from the parents' solutions and the offspring's solutions.

Step 10: for the best solution, mutate a random bit.

Step 11: calculate the weight and the value of the mutated solution.

Step 12: select the two best solutions from the mutated one, and the two solutions from step 9.

Step 13: return to step 5.

Step 14: from the best solution, for each verse in the solution, search a word that has a weight equal to the desired weight.

Step 15: return the selected verses and the words from step 14.

Step 16: end.

In the following subsection an example of applying the algorithm to retrieve the full issue for the word Israel (اسرائيل) is explained.

8.1 The Full Issue of the Word Israel (اسرائيل)

First, the algorithm retrieves the related verses for the word (اسرائيل), as is shown in Table 2.

Table 2. The related verses of the word (اسرائيل)

V#	Sura ID	Sura Name	Verse ID	Verse Description	Verse weight
V1	2	البقرة	40	يَبْنِي إِسْرَائِيلَ أَذْكُرُوا نِعْمَتِيَ الَّتِي أَنْعَمْتُ عَلَيْكُمْ وَأَوْفُوا بِعَهْدِي أَوْفِ بِعَهْدِكُمْ وَإِيَّيَ فَارْهَبُونَ	441
V2	2	البقرة	47	يَبْنِي إِسْرَائِيلَ أَذْكُرُوا نِعْمَتِيَ الَّتِي أَنْعَمْتُ عَلَيْكُمْ وَأَتَىٰ فَضَّلْتُكُمْ عَلَى الْعَالَمِينَ	343
V3	2	البقرة	49	وَإِذْ نَجَّيْنَكُمْ مِنْ آلِ فِرْعَوْنَ يَسُومُونَكُمْ سُوءَ الْعَذَابِ يُدَبِّحُونَ أَبْنَاءَكُمْ وَيَسْتَحْيُونَ نِسَاءَكُمْ وَفِي ذَلِكُمْ بَلَاءٌ مِنْ رَبِّكُمْ عَظِيمٌ	582
...
V118	61	الصف	6	وَإِذْ قَالَ عِيسَى ابْنُ مَرْيَمَ يَبْنِي إِسْرَائِيلَ إِنِّي رَسُولُ اللَّهِ إِلَيْكُمْ مُصَدِّقًا لِمَا بَيْنَ يَدَيَّ مِنَ التَّوْرَةِ وَأُبَشِّرُكُمْ بِالْمَسِيحِ الْمَسْمُومِ أَمْ كُنْتُمْ شَاكِرِينَ قَالُوا هَذَا سِحْرٌ مُبِينٌ	824
V119	61	الصف	14	يَا أَيُّهَا الَّذِينَ آمَنُوا كُونُوا أَنْصَارَ اللَّهِ كَمَا قَالَ عِيسَى ابْنُ مَرْيَمَ لِلْحَوَارِيِّينَ مَنْ أَنْصَارِي إِلَى اللَّهِ قَالَ الْحَوَارِيُّونَ نَحْنُ أَنْصَارُ اللَّهِ فَأَمَّا مَنْ بَدَّلَ اللَّهُ قَلْبَهُ فَإِنَّمَا يَمُرُّ بِالْحَرْبِ كَمَا يَمُرُّ بِالْحَرْبِ فَاتَّقُوا اللَّهَ إِنَّ اللَّهَ شَدِيدُ الْعِقَابِ	988

The second step generates random solutions by selecting verses randomly, then it calculates the solution weights and by using the genetic algorithm the most suitable solution will be selected, which appears in Table 3; the total weight is (703) which is divisible by the number 19 and return the number (37) which is the same weight of the word Alyahood (اليهود).

Table 3. The full issue of the word Israel (اسرائيل) that returns the word Alyahood (اليهود)

Sura ID	Sura Name	Verse ID	Verse Description	Verse weight
2	البقرة	135	وَقَالُوا كُونُوا هُودًا أَوْ نَصْرَىٰ تَهْتَدُوا فُلْ بَلْ مَلَّةٌ إِلَهُمَ حَنِيفًا وَمَا كَانَ مِنَ الْمُشْرِكِينَ	368
10	يونس	86	وَنَجِّنَا بِرَحْمَتِكَ مِنَ الْقَوْمِ الْكَافِرِينَ	167
26	الشعراء	197	أَوَلَمْ يَكُنْ لَهُمْ آيَةٌ أَنْ يَعْلَمَهُ عُلَمُوا بَنِي إِسْرَائِيلَ	168

8.2 Experimental Results

Many full issues have been revealed using the algorithm and return related terms or the same one such as the following examples:

- 1- The verses that are related to the word Yousef (يوسف) return it.
- 2- The verses that are related to the word Sirrat (صراط) return that word in addition to the word Tareeq (طريق).
- 3- The verses that are related to the word Alraheem (الرحيم) return both words Rahma (رحمة) and Aldhnob (الذنوب).

There are a lot of examples that have appeared during searches with the genetic algorithm solution for the knapsack problem. In addition, search about verses revolves around the same topic also return full issues as in the following examples:

- 1- Asma'a Yaom Al-Qiama (أسماء يوم القيامة)
- 2- Ma Ja'a Fe Al-Arsh (ما جاء في العرش)
- 3- Rahmat Allah Al-Wase'a (رحمة الله الواسعة)

There are also many other full issues that appear related to the topics of the verses. However, all the results indicate the main truth of the miracle in the Holy Quran, which also indicate that the Holy Quran is suitable for any place and at any time.

9. Conclusion

The miracle of the number 19 in the Holy Quran is a newly discovered one by Al-Refae'i in his book about the biggest miracle. Al-Refae'i believes that each number in the Holy Quran has a related miracle; this means that the arrangement of each Sura of the Holy Quran including its verses and words is considered to be a code for one of the miracles in the Holy Quran. In addition, the repetition of certain words in the Holy Quran can also be considered as a miracle in the Holy Quran. The attention of Al-Refae'i has been focused on the miracle of the number 19, since there are many indicators suggesting that this number has a secret behind it in the Holy Quran. Al-Refae'i builds a new alphabet that gives each letter a weight according to the number of repetitions of this letter in the Holy Quran. Then, each verse in the Holy Quran has a specific weight depending on the letters in the Ottoman script for this verse. On this basis, Al-Refae'i has concluded that each case in the holy Quran has a class of verses with the same meaning and with a total weight divisible by the number 19. Many full issues have been found by Al-Refae'i and explained in his book. Finally, Al-Refae'i gives his perspective on this miracle and leaves the door open to the researchers following him in order to motivate them to find more miracles in the Holy Quran in the future.

In this study the researchers use an existing algorithm (a genetic algorithm) and modify its parameters to match the miracle of the number 19. The new algorithm has been applied to the problem and it achieved obvious success. Also, many new full issues have appeared in this study. Finally, this success indicates that the Holy Quran is a big miracle that is valid in any place and at any time.

Acknowledgment

First of all, the researchers would like to thank God whose grace let them accomplish this research. The researchers also want to express their deepest appreciation to Professor Asem Elsheikh who acted as a father and director during the research in addition to being a professor.

References

- Al-Kabbani, S. H. (2010). *Power of the Hands with Illustrations*. Retrieved January 20, 2013, from <http://nurmuhammad.com/Dwnlds/StepsonHandmeditation.pdf>
- Al-Kaheel, A. (2010). *The marvels of the number seven in the noble Qur'an*.
- Al-Kaheel, A. (2012). *Secrets of Quran Miracle_ some basic guidelines to numeric miracle*.
- Al-Refae'i, A. (2006). *The Biggest Miracle*. Syria: Dar Al-Khair.
- Alvesa, M., & Almeida, M. (2007). MOTGA_ A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem. *Computers and Operations Research*, 34, 3458-3470. <http://dx.doi.org/10.1016/j.cor.2006.02.008>
- Bhaskar, M., & Maheswarapu, S. (2011). A Hybrid Genetic Algorithm Approach for Optimal Power Flow. *TELKOMNIKA*, 9(2), 211-216.
- Bortfeldt, A., & Winter, T. (2008). *A Genetic Algorithm for the Two-Dimensional Knapsack Problem with Rectangular Pieces*. Retrieved November 3, 2012, from <http://www.fernuni-hagen.de/wirtschaftswissenschaft/download/beitraege/db425.pdf>
- Bottaci, L. (2001). *A Genetic Algorithm Fitness Function for Mutation Testing*. Retrieved November 3, 2012, from <http://www2.hull.ac.uk/science/pdf/workshop.pdf>
- Bryant, K., & Benjamin, A. (2000). Genetic Algorithms and the Traveling Salesman Problem. Retrieved November 3, 2012, from <http://www.math.hmc.edu/seniorthesis/archives/2001/kbryant/kbryant-2001-thesis.pdf>
- Carlos, B. P., Darnes, V. A., Josefa, S. G., Saul, L. S., & Mireya, T. V. (2010). A Solution to Multidimensional Knapsack Problem Using a Parallel Genetic Algorithm. *International Journal of Intelligent Information Processing*, 1(2), 47-54. <http://dx.doi.org/10.4156/ijiiip.vol1.issue2.5>
- Chakaborty, R. (2010). *Genetic Algorithm and modeling _ soft computing*. Retrieved May 9, 2010, from http://www.myreaders.info/08_Fundamentals_of_Genetic_Algorithms.pdf
- Chekuri, C. (2009). *The Knapsack Problem*. Retrieved November 3, 2012, from http://www.cs.illinois.edu/class/sp09/cs598csc/Lectures/lecture_4.pdf
- Dizdar, D., Gershkov, A., & Moldovanu, B. (2009). *Revenue Maximization in the Dynamic Knapsack Problem*. Retrieved November 3, 2012, from <http://pluto.huji.ac.il/~alexg/pdf/knapsack.pdf>
- Djannaty, F., & Doostdar, S. (2008). A Hybrid Genetic Algorithm for the Multidimensional Knapsack Problem. *Int. J. Contemp. Math. Sciences*, 3(9), 443-456.
- El-Mihoub, T., Hopgood, A., Nolle, L., & Battersby, A. (2006). Hybrid Genetic Algorithms A Review. *Engineering Letters*, 13(2).
- Erben, W., & Konstanz, H. (2006). *Genetic Algorithms*. Retrieved November 3, 2012, from <http://www.cs.nott.ac.uk/~ajp/courses/g5baim/files/GAsErben.pdf>
- Gallardo, J., Cotta, C., & Fernandez, A. (2001). *Solving the Multidimensional Knapsack Problem Using an Evolutionary Algorithm Hybridized with Branch and Bound*. Retrieved May 9, 2012, from http://www.lcc.uma.es/~afdez/Papers/LNCS_HM2005.pdf
- Garg, P., Shastri, A., & Agarwal, D. (2005). An Enhanced Cryptanalytic Attack on Knapsack Cipher using Genetic Algorithm. *World Academy of Science, Engineering and Technology*, 12, 355-358.
- Gilbert, J., & Eppstein, M. (2002). *Case for Codons in Evolutionary Algorithms*. Retrieved November 3, 2012, from <http://www.waset.org/journals/waset/v12/v12-71.pdf>
- Gimeno, R., & Nave, J. (2006). *Genetic algorithm estimation of interest rate term structure*. Madrid: Imprenta del Banco de Espana.
- Han, K. H., & Kim, J. H. (2001). *Analysis of Quantum-Inspired Evolutionary Algorithm*.
- Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1999). The Compact Genetic Algorithm. *IEEE transactions on evolutionary computation*, 3(4), 287-297. <http://dx.doi.org/10.1109/4235.797971>
- Hassan, S. A. (2011). *Mathematical Miracle of Number - 19 in the Holy Quran 57 – Illustrations from the Holy Quran (19 x 3 = 57)*. Retrieved January 20, 2013, from <http://xa.yimg.com/kq/groups/5308962/357005157/name/Holy+Quran+and+Imam+Mahdi.pdf>

- Hristakeva, M., & Shrestha, D. (2003). *Solving the 0-1 Knapsack Problem with Genetic Algorithms*. Retrieved from http://www.micsymposium.org/mics_2004/Hristake.pdf
- Hristakeva, M., & Shrestha, D. (2004). *Different Approaches to Solve the 0/1 Knapsack Problem*. Retrieved November 3, 2012, from http://www.micsymposium.org/mics_2005/papers/paper102.pdf
- Jarrar, B. (2001). *Numeric Miracles of the Holy Qur'an Chosen Examples*. Al-Bireh: Noon Center For Qur'anic Studies & Research.
- Julstrom, B. A. (2005). Greedy, Genetic, and Greedy Genetic Algorithms for the Quadratic Knapsack Problem. *GECCO'05*. Washington: USA.
- Jung, S. H. (2009). Selective Mutation for Genetic Algorithms. *World Academy of Science, Engineering and Technology*, 56, 478-481.
- Korejo, I., Yang, S., & Li, C. (2009). A Comparative Study of Adaptive Mutation Operators for Genetic Algorithms. *The VIII Metaheuristics International Conference*. Hamburg: Germany
- Kosorukoff, A. (2001). *Human Based Genetic Algorithm*. Retrieved November 3, 2012, from <http://personal.stevens.edu/~ysakamot/creativity/human%20ga.pdf>
- Kuilekov, M., Ziolkowski, M., & Brauer, H. (2003). Application of Genetic Algorithms to an Inverse Field Problem in Magnetic Fluid Dynamics. *Serbian journal of electrical engineering*, 1(1), 1-13. <http://dx.doi.org/10.2298/SJEE0301001K>
- Labeled, S., Gherboudj, A., & Chikhi, S. (2011). Modified Hybrid Particle Swarm Optimization Algorithm for Multidimensional Knapsack Problem. *International Journal of Computer Applications*, 34(2), 11-16.
- Lai, K. (2006). *The Knapsack Problem and Fully Polynomial Time Approximation Schemes (FPTAS)*. Retrieved November 3, 2012, from <http://math.mit.edu/~goemans/18434S06/knapsack-katherine.pdf>
- Liu, W., & Liu, G. (2012). Genetic Algorithm with Directional Mutation Based on Greedy Strategy for Large-scale 0-1 Knapsack Problems. *International Journal of Advancements in Computing Technology (IJACT)*, 4(3), 66-74. <http://dx.doi.org/10.4156/ijact.vol4.issue3.9>
- Martinjak, I., & Golub, M. (2006). *Comparison of Heuristic Algorithms in Functions Optimization and Knapsack Problem*. Retrieved May 9, 2012, from http://bib.irb.hr/datoteka/274886.Martinjak_Golub_IIS2006.pdf
- Mathew, T. V. (2002). Genetic Algorithm. Retrieved November 3, 2012, from http://www.civil.iitb.ac.in/tvm/2701_dga/2701-ga-notes/gadoc.pdf
- Matousek, R. (2002). *Hybrid genetic algorithm and knapsack problem in the Matlab environment*. Retrieved November 3, 2012, from http://dsp.vscht.cz/konference_matlab/matlab02/matousek.pdf
- Matthews, K. B. (2001). *Applying Genetic Algorithms to Multi-objective Land-Use Planning*. Retrieved November 3, 2012, from <http://www.macaulay.ac.uk/LADSS/papers/keith-thesis.pdf>
- Mishra, B. S. P., Addy, A. K., Dehuri, S., & Cho, S. B. (2005). *An Empirical Study on Parallel Multi-objective Genetic Algorithms _ 0/1 Knapsack Problem-A Case Study*. Retrieved May 9, 2012, from [http://sclab.yonsei.ac.kr/publications/Papers/IC/wcica_latex_sample\[1\].pdf](http://sclab.yonsei.ac.kr/publications/Papers/IC/wcica_latex_sample[1].pdf)
- Molina, C., Tinoco, R., & Bouallou, C. (2011). Adaptive random search method + genetic algorithms for research kinetics modeling _ CO2 absorption systems. *Chemical engineering transactions*, 25, 19-24.
- Nabil, E., Badr, A., & Farag, I. (2012). A membrane-immune algorithm for solving the multiple 0/1 knapsack problem. *Studia univ. Babeş-bolyai, informatica*, 57(1), 3-15.
- Nemati, H. (1994). *Genetic Algorithms*. USA: University of North Carolina at Greensboro.
- Paplinski, J. P. (2009). The Genetic Algorithm with Simplex Crossover for Identification of Time Delays. *Intelligent Information Systems*, 337-346.
- Penev, M. K. V. S. S. (2005). Genetic operators crossover and mutation in solving the TSP problem. *International Conference on Computer Systems and Technologies*.
- Puchinger, J., Raidl, G., & Pferschy, U. (2007). The Multidimensional Knapsack Problem _ Structure and Algorithms. Retrieved November 3, 2012, from www.informs.org/content/download/15204/.../onlineSupplement.tex

- Sandurkar, S., & Chen, W. (1998). Gaprus - genetic algorithms based pipe routing using tessellated objects. Retrieved November 3, 2012, from <http://www.uic.edu/labs/idel/pdf/Sandurkar.pdf>
- Schmitt, L. M. (2001). Fundamental Study _ Theory of genetic algorithms. *Theoretical Computer Science*, 259, 1-61. [http://dx.doi.org/10.1016/S0304-3975\(00\)00406-0](http://dx.doi.org/10.1016/S0304-3975(00)00406-0)
- Senaratna, N. I. (2005). *Genetic Algorithms _ the Crossover-Mutation Debate*.
- Shopova, E. G., & Bancheva, N. G. (2006). BASIC _ A genetic algorithm for engineering problems solution. *Computers and Chemical Engineering*, 30, 1-17. <http://dx.doi.org/10.1016/j.compchemeng.2006.03.003>
- Sipper, M. (2009). Solving Knapsack Problems with Evolutionary Computation. Retrieved May 9, 2012, from <http://www.cs.bgu.ac.il/~orlovm/teaching/assignments/intro-2009a-evo-knapsack.pdf>
- Sivaraj, R., & Ravichandran, T. (2011). An Efficient Grouping Genetic Algorithm. *International Journal of Computer Applications*, 21(7), 38-42. <http://dx.doi.org/10.5120/2520-3424>
- Sriram, J. (2009). *KS-Solve _ Local Search for the Knapsack Problem*. Retrieved November 3, 2012, from <http://www.cs.dartmouth.edu/~afra/courses/44/w09/project/report/sriram-report.pdf>
- Stein, G., Chen, B., Wu, A., & Hua, K. (2004). Decision Tree Classifier For Network Intrusion Detection With GA-based Feature Selection. Retrieved November 3, 2012, from <http://dl.acm.org/citation.cfm?id=1167288&dl=ACM&coll=DL&CFID=104311455&CFTOKEN=25069829>
- Sudhakaran, M., & Raj, P. (2010). Integrating genetic algorithms and tabu search for unit commitment problem. *International Journal of Engineering, Science and Technology*, 2 (1), 57-69.
- Sugimoto, M. (2007). Distributed Cell Biology Simulations with the E-Cell System. Retrieved November 3, 2012, from <http://www.landesbioscience.com/pdf/ArjunanSugimoto.pdf>
- Sunanda, G., & Garg, M. (2009). GA implementation of the multi dimensional knapsack problem using compressed binary tries. *Advances in Computational Research*, 1(2), 43-46.
- Swetanisha, S., & Mishra, B. (2011). Evolutionary Multiobjective Genetic Algorithm to Solve 0/1 Knapsack Problem. *International Conference on Computer Engineering and Applications International Conference on Computer Engineering and Applications*. Singapore.
- Taskiran, G. K. (2010). An improved genetic algorithm for knapsack problems. Retrieved November 3, 2012, from <http://etd.ohiolink.edu/send-pdf.cgi/Kilincli%20Taskiran%20Gamze.pdf?wright1270598986>
- Tavares, J., Pereira, F., & Costa, E. (2005). *The Role of Representation on the Multidimensional Knapsack Problem by means of Fitness Landscape Analysis*. Retrieved November 3, 2012, from http://jorgetavares.files.wordpress.com/2008/04/2006_cec_1.pdf
- Thierens, T. (1999). Scalability Problems of Simple Genetic Algorithms. *Evolutionary Computation*, 7(4), 331-352. <http://dx.doi.org/10.1162/evco.1999.7.4.331>
- Uyar, S., Sariel, S., & Eryigit, G. (2004). A Gene Based Adaptive Mutation Strategy for Genetic Algorithms. *GECCO*, 271-281.
- Yang, S. (2002). Adaptive crossover in genetic algorithms using statistics mechanism. *Artificial Life*, 8, 182-185.
- Zorman, B., Kapfhammer, G., & Roos, R. (2002). Creation and Analysis of a JavaSpace-Based Distributed Genetic Algorithm. Retrieved May 9, 2012, from <http://cs.allegheeny.edu/~gkapfham/research/publish/DGAConference.pdf>