Improving DNA Computing Using Evolutionary Techniques

Godar J. Ibrahim
Software Engineering
College of Engineering
Salahadin university-Erbil
Hawler, Kurdistan

Tarik A. Rashid
Software Engineering
College of Engineering
Salahadin university-Erbil
Hawler, Kurdistan

Ahmed T. Sadiq
Software Engineering
College of Engineering
Salahadin university-Erbil
Hawler, Kurdistan

***Abstract*—the field of DNA Computing has attracted many biologists and computer scientists as it has a biological interface, small size and substantial parallelism. DNA computing depends on DNA molecules' biochemical reactions which they can randomly anneal and they might accidentally cause improper or unattractive computations. This will inspire opportunities to use evolutionary computation via DNA. Evolutionary Computation emphasizes on probabilistic search and optimization methods which are mimicking the organic evolution models. The research work aims at offering a simulated evolutionary DNA computing model which incorporates DNA computing with an evolutionary algorithm. This evolutionary approach provides the likelihood for increasing dimensionality through replacing the typical filtering method by an evolutionary one. Thus, via iteratively increasing and recombination a population of strands, eliminating incorrect solutions from the population, and choosing the best solutions via gel electrophoresis, an optimal or near-optimal solution can be evolved rather than extracted from the initial population.**

Keywords—Parallel Computation; DNA Computation Algorithm; Evolutionary DNA Computing Algorithm

I. INTRODUCTION

Leonard Adleman was the first person to demonstrate that DNA computing could be utilized for computing in a laboratory environment and without using conventional computing devices. As a paradigm for his novel approach, he selected the Hamiltonian Path Problem (HPP) to obtain solutions via experimental DNA tests [1]. It is of interest to mention that further or more rapid progress would have been achieved if he had selected an easier problem. Subsequently, he selected the HPP on the use of conventional computers [2].This choice opened stimulating avenues and allowed other researchers to think more favorably of DNA computing. In doing so, he considered that the power of this technique was great and involved substantial prospects for parallel computing, as is feasible via operations with DNA. Adelman is now considered the founding father of DNA computing [3].

Currently, DNA computing is an interdisciplinary area in which ecologists, biologists, computer scientists, physical scientists, mathematicians, chemists, and other related specialists identify interesting problems that may be useful for the theoretical and practical sides of DNA computing [4]. DNA computers are essentially assortments of chosen DNA strands. The combinations of these strands imply solutions to a given problem that is to be solved. Tools are currently available to select the preliminary components and winnow candidate solutions.

The potential of parallel processing algorithms is substantial. DNA computing on large problems can involve parallel processing, given a preliminary arrangement and ample DNA. These challenging tasks are easily and effectively accomplished using DNA computing. In contrast, standard computers would require substantial parallelism and more hardware [2]. On the other hand, DNA computing is determined by DNA molecules' biochemical reactions, which they can unsystematically anneal and they might yield unsuitable computations. Therefore, the use evolutionary computation via DNA might be a good solution. Evolutionary computation emphasizes on optimization method and a search which is probabilistic. It mimics the organic evolution models via using operators such as recombination and randomized mutation or so called the progression of interaction between the formation of fresh generations and their assessments and the choice wherever a sole individual in a populace is affected by the surroundings and also by other individuals. It is considered that individuals that can perform better in such circumstances, they will have greater chances for surviving [5, 6].

The contribution of this work is to propose an evolutionary DNA algorithm based on the standard DNA algorithm as it is presented in this paper to solve the shortest path to increase the possibility of having an optimum solution and to improve the average cost of the final paths. In the next section, some of the techniques and approaches that have been used in the area of DNA computing are reviewed and outlined.

II. RELATED WORKS

DNA computing was primarily conceived by Leonard Adleman; in 1994, Adleman developed the idea of using DNA computing to address the HPP [1]. From the primary investigations and early tests by Adleman, innovations and progress emerged, e.g., a number of Turing machine devices were systematically developed and tested. Although the primary efforts to exploit this fresh methodology unveiled computationally complex problems, it was rapidly and conclusively shown that the methods were clearly inapplicable to this type of computational algorithm [3, 4, 7, 8].

In 1997, a research team led by Ogihara and Ray recommended the analysis of Boolean circuits, it is clear that Nand Boolean circuits cover only Nand gates. The research

team showed that the relationship between the logarithm of the maximum fan-out of the Boolean circuit and the runtime slow-down is proportionate, also they showed that the relationship among the product of the size and the maximum fan-out and the space complexity is proportionate [9, 10, 11, 7, 12, 13].

In 2002, researchers from Israel announced a computing device based on molecular programming. This computing device was constructed from DNA molecules and enzymes in contrast to the silicon materials of microchips and integrated circuits [14, 12]. Shapiro and his research team wrote in Nature that they had created a deoxyribonucleic acid computer [14]. The device was able to locate tumor-related entities within a cell, and the team was able to produce drugs that conferred cancer resistance whenever the disease was detected [14, 15].

Another study was conducted to determine SSCP sensitivity in detecting factor IX mutation [16]. The study investigated the blood of hemophilia B patients in Iran. Phenol, chloroform and other reagents were used to extract DNA. The gene regions were amplified using primer pairs and PCR. PCR fragments with improved flexibility were obtained. The study concluded that SSCP sensitivity was high in the system investigated [16].

In 2006, Dimitrova outlined and summarized examples for various DNA computing applications such as aqueous computing, molecular computing, DNA Turing machines, and the nascent field of synthetic biology [15]. In 2008, Abdulla successfully inserted a heuristic search in a DNA computing algorithm to generate better efficiency and flexibility. That study improved the DNA search technique by increasing the number of solutions and reducing the running time and memory capacity. The modifications of A* and alpha beta using DNA produced better results than ordinary A* and alpha beta algorithms [17].

Hari and his research team have suggested an advanced and efficient technique for addressing the problem of minimum vertex cover using a DNA computing algorithm. They suggested that a DNA computing algorithm could make it possible to address problems that are intractable on silicon computers. Nevertheless, the study stated that DNA computing algorithms are not feasible for solving simple problems due to their high degree of parallelism. Thus, computer scientists are required to design and elaborate more DNA computing algorithms [18]. The same research team has suggested that appropriate enzymes and legitimate approaches will dynamically shape biology to address more subtle problems and reduce the amount of error associated with the use of the algorithm.

The focus of this research paper is to incorporate an evolution strategy into DNA computing based only on the crossover operator and strategy parameter. This approach is expected to enhance or optimize the quality of the final solutions by increasing the size of the final solutions and also by evolving the most correct solutions to obtain an optimum or near optimum solution(s). This optimization could take place regardless of the increase in the time and memory capacity of the modified algorithm.

The structure of this paper is as follows. In the next section, DNA computing algorithms and operations are described. Then, a DNA interpretation for the problem of the shortest path is described in detail. Then after, the proposed evolutionary DNA computing method is established. Next, experimental results are described in detail. Finally, the conclusions are outlined. A glossary or list of terminology is presented in appendix A at the end of this paper.

III. DNA Computing Algorithm and Operations

Bioinformatics is now viewed as the study of information stored in DNA. The strings of letters correspond to combinations of the four bases A (adenine), T (thymine), G (guanine) and C (cytosine). These strings transmute information via convolution operations on the unit cell [18, 15]. DNA polymerase is the key enzyme. With a specified strand of DNA under suitable circumstances, this enzyme generates the complementary strand, another Watson-Crick DNA sequence in which the letter C stands opposite G, G stands opposite C, A stands opposite T and T stands opposite A. From the molecular sequence CATGTC, for example, the new molecular sequence GTACAG is created by DNA polymerase. DNA is regenerated by DNA polymerase. This capability allows cells to regenerate and eventually permits the investigator to make a replica of the original object of study or analysis. The replication of DNA via DNA polymerase is the most important life process. DNA polymerase is, in essence, a nano-machine. It links to DNA strands, it reads each base, and, as it passes, it writes the complementary information as a fresh, lengthening strand of DNA [18, 15].

The resemblance between the Turing machine and DNA polymerase is striking. It is known that Alan Turing created and designed a computer in the shape of a toy, which was later called the Turing machine. Originally, the device was intended to be conceptual and appropriate for precise mathematical examination.

Thus, it was meant to be truly simple, and Alan Turing fully succeeded. In one account, the Turing machine is described as a limited control device consisting of pairs of tapes. The limited control device moves sideways through the tape input, reading information. It moves sideways through the output of the tape, which reads and writes other data. It is noted that the limited control device is programmed with basic instructions; it would be easy to code programs for reading the letter strings (A, T, C, G) on the input of the tape and produce the complementary Watson-Crick string as the tape output. Thus, a Turing device can be used for coding different programs, e.g., to produce the complementary Watson-Crick strings or to play Chess. The key operations of DNA computing used in a DNA algorithm are defined below [19, 20, 17, 21, 15, 22, 23, 24, 25]:-

a) Watson-Crick pairing, as mentioned above, is pervasive; it is obvious that any DNA strand has its Watson-Crick complement. The pairing will occur when a DNA molecule encounters the complement of the original Watson-Crick strand. Later, both DNA strands will be annealed. Both strands join to produce the well-known double helix.

b) Polymerases allow the copying of information from one molecule to another. The DNA polymerase is able to generate strands of DNA complementary to the original Watson-Crick DNA string.

c) Ligases connect molecules. This concept can be illustrated with DNA ligase, which creates one single strand from two DNA strands. DNA ligase can be utilized in the cell to repair disruptions or breaks that can occur in strands of DNA. This phenomenon can be observed when skin cells are exposed to certain types of light.

d) Nucleases break down and thereby repress deoxyribonucleic and ribonucleic acids.

e) Gel electrophoresis serves to analyze and segregate DNA, RNA and protein macromolecules. It allows heterogeneous DNA molecules to be run on a gel with an electric current and identified.

f) Synthesis. Sequences of DNA can be written on paper and sent to a synthesis facility. The synthesis facility then returns a tube containing 1018 DNA molecules within a few days. Each of the molecules contains the specific sequence requested. Sequences of approximately 100 nucleotides in length can be synthesized dependably in the laboratory. The DNA is delivered in dry form in a narrow tube, and the molecules appear white in color.

IV. DNA Solution for the Sample of Network Diagram

Let a directed graph be given by *Graph= (Ver, Edg)*, where *Ver* is defined as a group of vertices and *Edg* is defined as a group of weights (See Fig. 1). A sequence of these vertices such as *Ver1, Ver2, Ver3... Vern* can define a basic path, as long as *Ver1, Ver2, Ver3... Vern* are considered distinct [2, 8, 25, 16]. The path length can obviously be calculated by adding up every edge weight in the path. Of course, considering walking from a specific source to a specific destination, the shortest simple path is the path that has the least weight among all of the paths in the graph. The objective is to minimize the total cost in the directed path. This is, of course, the simple shortest path going all the way from the key source to the destination. The shortest path problem is a network of several nodes and edges that are used to link all of the nodes, and the problem can be solved using the standard DNA algorithm.

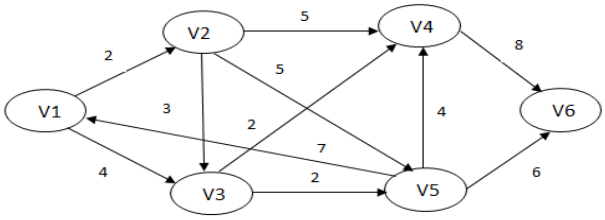

Fig. 1. Sample of Network Diagram

The standard DNA algorithm aims to handle the key DNA operation stages; these stages include the coding of the problem, which is performed in DNA, the production of random solutions, the amplification of the random DNA solutions using PCR, the elimination of repeated nodes using SSCP, and finally the sorting of the final solutions using electrophoresis (see Fig. 2).

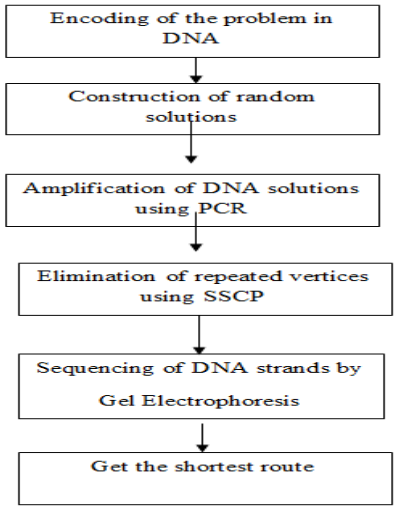

Fig. 2. Shows Block Diagram of Normal DNA Algorithm for Solving Shortest Path Problem

The key stages of the standard DNA algorithm can be described below [2, 16, 15, 26] :-

a) DNA problem representation (Coding)

It is obvious that in the graph, each vertex *Veri* must be linked to a specific palindromic 10-mer DNA sequence, which is represented via *Veri*. Therefore, every single edge *Veri* ➔ *Verj* (in the directed graph, a complementary oligonucleotide sequence of a 3' 5-mer of *Veri* is tracked via a complimentary sequence of a 5' 5-mer of *Verj)* can be combined.

b) Building of random pathways (Construction)

It can be said that every oligonucleotide coding vertex and every oligonucleotide coding edge must be combined to build random paths in the graph; tweaks or a tweak for each of the various sequences can be selected and are kept inside test tubes.

c) DNA path intensification using PCR (Amplification)

In this stage, the process of intensification is performed. It would begin with the source of the vertex and would end at the destination of the vertex. Note that 2 specific primers can be designed to target the source of the vertex and destination of the vertex, which will further amplify the PCR response.

d) Dismissal of repetitive vertices (Elimination)

SSCP is a simple and common practice of mutation detection. This process prevents nodes from reappearing within the strands of DNA. Basically, the PCR serves to proliferate and aggregate the region of interest. The subsequent DNA would be separated via electrophoresis to produce single-stranded molecules.

e) Sequencing of the DNA strands

At this stage, the strands achieved in stage 4 are sequenced. By reading the sequence, the weight of each strand is defined.

The desired solution is considered to be the path that has the least weight.

Applying the normal DNA algorithm to solve the shortest path problem, the above key operational stages can be expressed as follows:

a) DNA problem representation (Coding)

This can be performed via random synthesis of a palindromic 10-base strand of DNA for every single vertex, keeping in mind that *Veri* signifies the *ith* vertex (See Table 1). Thus, to represent every edge, *Veri* ➔ *Verj*, a palindromic 10-base strand of DNA can be synthesized, which is a sequence consisting of the 3' 5-mer complement of *Veri* and the 5' 5-mer sequence complement of *Verj*. The result of the graph (see Fig. 1) is shown in Table 2.

b) Building of random pathways (Construction)

In this stage, all sequences that correspond to both vertices *Veri* and edges *Veri* ➔ *Vj*, as synthesized in stage 1, must be stored in test tubes to allow ligation. Subsequently, ligation will occur when both sequences of the DNA *Ver1 (GCGAGATCTG)* and DNA Edge *Ver1*➔*Ver2 (TAGACCTTCA)* accidentally come into contact with each other.

The fact that the earlier sequence ended via *ATCTG* and the latter sequence started via *TAGAC* is important because these two sequences are complementary. Thus, the annealing process will occur. If the resultant composite encounters a sequence that matches *Ver2 (GAAGTCAGTC)*, at that point, it may also be able to join the composite, as the earlier sequence complements the later starting sequence. It can be observed that composites can grow lengthwise when edges of DNA sequences are joined together via a vertex of DNA sequences. Accordingly, the paths can be expressed as follows:-

Ver3 ➔ Ver4 ➔
GTCACGACGAGCCATAGACC
Ver1 ➔ Ver2 ➔ Ver3 ➔ Ver5 ➔ Ver4 ➔ Ver6
GCGAGATCTGGAAGTCAGTCGTCACGACGAGTTCGTTTA GGCCATAGACCGATGAGAGTA
Ver2 ➔Ver3 ➔Ver5 ➔ Ver4
GAAGTCAGTCGTCACGACGAGTTCGTTTAGGCCATAGAC C
Ver1 ➔→ Ver2 ➔ Ver3 ➔ Ver5
GCGAGATCTGGAAGTCAGTCGTCACGACGAGTTCGTTTA G
Ver1 ➔ Ver3 ➔ Ver4 ➔ Ver6
GCGAGATCTGGTCACGACGAGCCATAGACCGATGAGAG TA
Ver1 ➔ Ver2 ➔ Ver3 ➔ Ver5 ➔Ver4 ➔ Ver6
GCGAGATCTGGAAGTCAGTCGTCACGACGAGTTCGTTTA GGCCATAGACCGATGAGAGTA
Ver1 ➔Ver2 ➔Ver5 ➔ Ver1 ➔ Ver3 ➔Ver5 ➔Ver6
GCGAGATCTGGAAGTCAGTCGTTCGTTTAGGCGAGATCT GGTCACGACGAGTTCGTTTAGGATGAGAGTA
Ver1 ➔ Ver2 ➔ Ver3 ➔Ver5 ➔Ver6
GCGAGATCTGGAAGTCAGTCGTCACGACGAGTTCGTTTA GGATGAGAGTA
Ver2 ➔ Ver3 ➔ Ver4 ➔ Ver6
GAAGTCAGTCGTCACGACGAGCCATAGACCGATGAGAG TA
Ver1 ➔ Ver3 ➔ Ver5 ➔ Ver4
GCGAGATCTGGTCACGACGAGTTCGTTTAGGCCATAGA CC
etc.

c) DNA path intensification using PCR (Amplification)

The generation of all paths that can hold both source and destinations (*Vers* and *Verd)* can be accomplished via DNA path intensification using PCR. This process occurs as follows:

I. If the source *Ver1* and the destination *Ver6* are selected, at that point, both primers that correspond to the source and destination (*Ver1 (GCGAGATCTG)* and *Ver6 (GATGAGAGTA)*) are added to the resulting solution. Then, the PCR can occur.

II. Additionally, the *GCGAGATCTG* primer would anneal to its target sequence that is established in the *Ver1 ➔Ver6 path*; a similar phenomenon will occur with the other primers.

Consequently, immediately after the accomplishment of PCR, every path from the *Ver1* source to the *Ver6* destination can be expressed as follows:-

Ver1 ➔Ver2 ➔Ver3 ➔Ver5 ➔ Ver6 ➔
GCGAGATCTGGAAGTCAGTCGTCACGACGAGTTCGTTTA GGATGAGAGTA
Ver1 ➔ Ver2 ➔ Ver3 ➔ Ver5 ➔ Ver4 ➔Ver6
GCGAGATCTGGAAGTCAGTCGTCACGACGAGTTCGTTTA GGCCATAGACCGATGAGAGTA
Ver1 ➔ Ver2 ➔ Ver5 ➔ Ver1 ➔ Ver3 ➔ Ver5 ➔Ver6
GCGAGATCTGGAAGTCAGTCGTTCGTTTAGGCGAGATCT GGTCACGACGAGTTCGTTTAGGATGAGAGTA
Ver1 ➔ Ver2 ➔ Ver4 ➔ Ver6
GCGAGATCTGGAAGTCAGTCGCCATAGACCGATGAGAG TA
Ver1 ➔Ver3 ➔ Ver5 ➔ Ver6
GCGAGATCTGGTCACGACGAGTTCGTTTAGGATGAGAG TA
Ver1 ➔Ver3 ➔ Ver5 ➔ Ver4 ➔ Ver6
GCGAGATCTGGTCACGACGAGTTCGTTTAGGCCATAGA CCGATGAGAGTA
Ver1 ➔ Ver2 ➔ Ver5 ➔ Ver6
GCGAGATCTGGAAGTCAGTCGTTCGTTTAGGATGAGAGT A

d) Dismissal of repetitive vertices (Elimination)

In this stage, the rule is obviously not to allow the vertices to reappear. In other words, repetition is not allowed; thus, nodes that are repeated for the second time will be dismissed.

The process of elimination can be performed via a single-stranded conformation polymorphism approach (SSCP). A hairpin structure can be formed via the series. It can hold reappearing nodes that can be connected to their corresponding split ends. The strands that hold hairpin loops will eventually be eliminated via the SSCP approach. Thus, the representation for all paths that have vertex repetition can be expressed as follows:-

Ver1 ➔ Ver2 ➔ Ver5 ➔ Ver1 ➔Ver3 ➔ Ver5 ➔ Ver6
GCGAGATCTGGAAGTCAGTCGTTCGTTTAGGCGAGATCT GGTCACGACGAGTTCGTTTAGGATGAGAGTA

It can be observed that with a path *Ver1 ➔ Ver2 ➔Ver5 ➔ Ver1 ➔ Ver3 ➔Ver5 ➔Ver6*, the vertices *Ver1* and *Ver5* are repeated. Thus, subsequently disregarding strands with loops, the achieved paths are expressed as follows:

Ver1 ➔ Ver2 ➔ Ver3 ➔ Ver5 ➔Ver6
GCGAGATCTGGAAGTCAGTCGTCACGACGAGTTCGTTTA GGATGAGAGTA
Ver1 ➔ Ver2 ➔Ver3 ➔ Ver5 ➔ Ver4 ➔Ver6
GCGAGATCTGGAAGTCAGTCGTCACGACGAGTTCGTTTA GGCCATAGACCGATGAGAGTA
Ver1 ➔ Ver2 ➔ Ver4 ➔Ver6
GCGAGATCTGGAAGTCAGTCGCCATAGACCGATGAGAG TA
Ver1 ➔Ver3 ➔Ver5 ➔ Ver6
GCGAGATCTGGTCACGACGAGTTCGTTTAGGATGAGAG TA
Ver1 ➔Ver3 ➔ Ver5 ➔Ver4 ➔Ver6
GCGAGATCTGGTCACGACGAGTTCGTTTAGGCCATAGA CCGATGAGAGTA
Ver1 ➔Ver2 ➔ Ver5 ➔ Ver6
GCGAGATCTGGAAGTCAGTCGTTCGTTTAGGATGAGAGT A

e) Sequencing of the DNA strands. In this stage, each weight path achieved in stage 4 is defined by interpreting the equivalent sequence.

Ver1 ➔ Ver2 ➔Ver4 ➔ Ver6
Overall weight for path is 15
Ver1 ➔Ver2 ➔Ver3 ➔ Ver5 ➔ Ver6
Overall weight for path is 13
Ver1 ➔ Ver2 ➔ Ver3 ➔ Ver5 ➔Ver4 ➔ Ver6
Overall weight for path is 19
Ver1 ➔Ver3 ➔Ver5 ➔Ver6
Overall weight for path is 12
Ver1 ➔Ver3 ➔Ver5 ➔ Ver4 ➔Ver6
Overall weight for path is 18
Ver1 ➔Ver2 ➔ Ver5 ➔ Ver6
Overall weight for path is 13

Note that *Ver1 ➔ Ver3 ➔ Ver5 ➔ Ver6* path produces 12; this is the least weight cost. Therefore, *Ver1 ➔ Ver3 ➔ Ver5 ➔ Ver6* is our solution to the problem path, which holds an equivalent DNA sequence that can be described as follows:

GCGAGATCTGGTCACGACGAGTTCGTTTAGGATGAGAG TA

V. Proposed Evolutionary DNA Computing

The Java programming language is used to design a simulation of DNA computing to solve the problem of the simple shortest path. The standard DNA algorithm is implemented; then, the cleaning stage is used to obtain the desired solutions randomly. However, the produced solutions that are not desirable are discarded through this stage. The shortest path is established; however, there are some restrictions that must be addressed:-

1) It is clear that the DNA algorithm generates random solutions that are governed by chance with respect to the DNA strands meeting each other or not; thus, the DNA algorithm might not produce all of the potential solutions as there is no evolutionary progress involved here to produce solutions via progress within considerable sequence populations arising from the DNA.

2) In case the DNA algorithm cannot produce all of the potential solutions, identification of the best solution is not assured.

3) The random generation solution size can be amplified; thus, the number of potential solutions could also be amplified. This process will allow us to obtain more final solutions; however, the downsides are that the search process will take more time and there is a greater need for memory capacity. Still, the best solution is not assured.

There must be other ways to enhance the standard normal DNA computing algorithm to obtain more diversity in the scope of the produced solutions, to create more correct solutions and also to obtain the best solution. In this regard, a method is suggested to conglomerate DNA computing with an evolutionary algorithm (Evolutionary DNA Computing). Evolutionary algorithm features are used in this research to produce solutions through progress within substantial sequence populations arising from the DNA. This evolutionary algorithm will increase the dimensionality of the system by replacing the customary filtering approach with an evolutionary approach. Thus, the best solutions might be obtained through iterative intensification, recombining strand populations, eliminating inappropriate solutions included in the population, and selecting the best solutions through gel electrophoresis instead of mining them from the preliminary population.

This proposed improvement of the algorithm has four modifying operations (See Fig. 3). Each operation has an effect on the algorithm, while they all share the same representation of the knowledge. The four operations are the following:-

1) Adding/Replacing the start/end of the PCR-dropped strands

The normal DNA algorithm is modified by adding/replacing the start/end of the PCR-dropped strands. The first level of enhancement is finished by obtaining all of the dropped PCR solutions, adding a start node to the beginning of the solution strand, and adding an end node to the end of the solution strand. By performing this operation, the PCR-dropped solution can address PCR solutions in such a manner that the chance of obtaining more final solutions will increase. Another level that is accomplished using the same function is to switch the start and end nodes of the dropped PCR solution strand by the desired start/end nodes. This step can add diversity to the solution space by increasing the number of PCR solutions and consequently increasing the chance of obtaining many solutions in the end. These 2 modifications are solely applicable to the Shortest Path problem. Details of the pseudo code snippet are shown below.

Replace and Add the start/end of the PCR solution:
Input: Dropped PCR solution
Output: Added PCR Test Tube Solution

For each dnaStrand in the PCR Test Tube
If dnaStrand.length > Size
*newStrand= **replaceStartEnd** (dnaStrand)*
Add newStrand to PCR Test tube
*newStrand= **addStartEnd** (dnaStrand)*
Add newStrand to PCR Test tube
End For
***replaceStartEnd** (dnaStrand)*
replacedStrand= dnaStrand.substring(Size, dnaStrand.length()-Size)
Return startNodeStrand+replacedStrand+endNodeStrand
***addStartEnd** (dnaStrand)*
Return startNodeStrand+ dnaStrand +endNodeStrand
replacedStrand= dnaStrand.substring(Size, dnaStrand.length()-Size)
Return startNodeStrand+replacedStrand+endNodeStrand
***addStartEnd** (dnaStrand)*
Return startNodeStrand+ dnaStrand +endNodeStrand

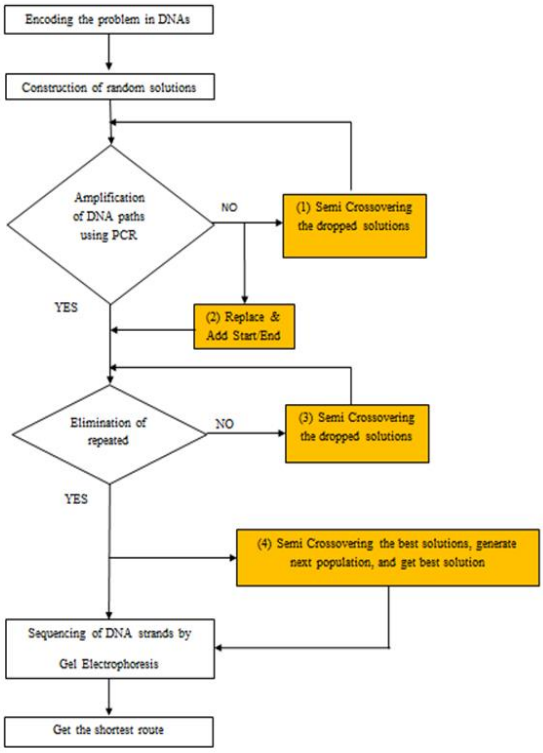

Fig. 3. The proposed technique

2) Crossovers Dropped in the PCR Solutions

Modifying the normal DNA algorithm by adding semi-crossovers of the dropped PCR strands involves applying another modification in the algorithm. The process works by obtaining the dropped PCR solutions and randomly selecting two nodes in the solution strand to be replaced by two random nodes obtained from the set of original nodes. This step is similar to a semi-crossover operation for increasing the possibility of obtaining more solutions. The output of this function is sent to the PCR function to obtain the correct PCR solutions and to send them to the next function, which is the SSCP function. The details of the pseudo code snippet are shown below.

PCR Semi-Crossover:
Input: Dropped PCR Solution TT, Nodes TT
Output: Added PCR Test Tube Solution

For each dnaStrand in PCR Dropped Test Tube
If dnaStrand.length > Size
While no termination do // two crossovers or break
Get random nodeS in dnaStrand
Get random NodeS from initial Test Tube
Replace nodeS with NodeS
Od
If isPCR (dnaStrand) // isPCR function already available
Add dnaStrand to PCR Test Tube
End If
End If
End For

3) Crossovers Dropped in SCCP Solutions

Modifying the normal DNA algorithm with semi crossovers of the dropped SSCP strands proceeds as follows:-

This modification is the same as the PCR Semi-Crossover but is applied on dropped SSCP solutions, and the difference is that the two random nodes are semi-crossed over, but not the start/end node, because there are already correct nodes in the solutions strand. The output of this function is sent to the SSCP function to obtain the correct SSCP solution from within the set. Details of the pseudo code snippet are shown below.

SSCP Semi-Crossover:
Input: Dropped SSCP Solution TT, Nodes TT
Output: Added SSCP Test Tube Solution
For each dnaStrand in the SSCP Dropped Test Tube
If dnaStrand.length > Size
While no termination do // two crossovers or break
Get random nodeS in dnaStrand but not start/end node
Get random NodeS from initial Test Tube
Replace nodeS with NodeS
Od
If isSSCP (dnaStrand) // isSSCP function already available
Add dnaStrand to SSCP Test Tube
End If
End For

4) Evolutionary SSCP

Modifying the normal DNA algorithm with the evolutionary approach:

The following is the real evolutionary improvement to the algorithm: the best SSCP solution and 10% of the other SSCP solutions (not the best ones) are taken from the SSCP solution list. A new generation is made from these selected solutions by crossing over one node of the solution randomly and checking if any solution is generated with a lower cost; it is used if no better solution is generated. The strategy parameter is tuned, and instead of crossing over only one node, two nodes are crossed over, and the evolution of the algorithm is evaluated. The generation of a new population from an initial population is continued until either a better solution is obtained or the termination criterion is met. Details of the pseudo code snippet are shown below.

SSCP Evolution Strategy:
Input: SSCP Solution Test Tube, Nodes Test Tube
Output: Added SSCP Test Tube Solution
While no termination do // two crossovers or break
Get best solution and 10% random solutions
For i=0 to EdgesTT.length // number of generations
If (new generation worse than previous one)
Set crossinOverNodes=2 // strategy parameter setting

Else
Set crossinOverNodes=1
makeNewGeneration*()*
Get best solution and 10% random solutions from new generation
End For
Get best solution from new generation
End While
makeNewGeneration*()*
While no termination do
makeSemiCrossOver*(currentStrand,Nodes TT) // function is already //defined*
add crossedOver strand to new population
End While

VI. Experimental Results

It is clear that the evolutionary SSCP improves the algorithm by evolving the solution into a better solution. In the next first, second or third generation, a better solution is generated by semi-crossovers of the nodes in the current generation; occasionally, this occurs in later generations. A better indication of the evolutionary improvement is that in some iteration, a solution is given even when there is no solution found in the normal algorithm.

An improved DNA computing algorithm for solving complex optimization problems is presented in this research study. The algorithm not only shows whether a solution exists but also provides more possible solutions; hence, the likelihood of obtaining the optimum solution is increased. The proposed algorithm might be extended to solve other optimization problems. This will be shown in the test result tables. To improve the algorithm, the focus is on the generation of more solutions rather than decreasing the running time or memory capacity, as current computers have sufficient CPU speed and memory capacity. The variables to be used are defined in the results of the Shortest Path Problem; all of the results in this section are supported by tables and charts to display the intermediate and final results, with statistical curves that represent the comparison between the standard DNA and improved DNA algorithms. The data tables of the results of the DNA Algorithm and Evolutionary DNA Algorithm for solving the Shortest Path problem are found below. Table 3 shows the solution of the standard DNA algorithm for SPP, which is the basic result to be compared with the results generated by the improved DNA algorithm.

Table 4 shows the first improved solution of SPP by the DNA algorithm. As explained in the previous sections, the algorithm is improved by working on the dropped solution at the PCR operation; the first step's desired start node will be added to the beginning of the strand, and the end node is added to the end of the strand. At the second step, the beginning node and end node of the strand will be replaced by the desired start and end nodes. It is clear in the table and by comparing with the previous table that the number of PCR solutions is dramatically increased (which increases the SSCP solutions automatically), which results in having more final solutions and a better average cost for the final paths.

By embedding the crossover operation in the PCR operation, as observed below the table, the number of PCR and SSCP is increased, which again results in an increased number of final solutions and the improvement of the average cost of the final paths, but the percentage of improvement is less than that of the previous modification. By embedding the SSCP crossover in the SSCP operation, the percent increase in the SSCP solution is even lower; hence, it has a smaller number of final solutions and therefore does not have good improvement in the average cost of the final paths. Until now, replacing Start/End in the PCR operation generates better results than the PCR and SSCP crossover operations. Even combining the PCR and SSCP does not yield good results (Tables 5, 6 and 7).

As is shown in Tables 8, 9 and 10, by Replacing Start/End in the PCR Operation with the PCR and SSCP crossover, the number of final solutions and the average cost of the final paths is improved. Thus, replacing Start/End in the PCR operation improves the standard DNA algorithm much more than the PCR and SSCP crossover. Tables 11 and 12 show the results of the evolutionary SSCP, evolving the resulting SSCP solution through several generations until the best SSCP solution is obtained. Although the number of PCRs, SSCPs and final solutions is low, the average cost of the final route is good; this finding indicates that the performance of the algorithm can be increased by evolving the final solution rather than increasing the search space of the problem. By adding the supportive operations to the evolutionary SSCP operation, the result is improved. There are more solutions in the end; therefore, the possibility of having the optimum or near-optimum solution is increased, and the average cost of the final paths is improved as well (See Tables 13, 14, 15 and 16).

The effect of the modifications on the standard DNA algorithm and the optimization of the algorithm by obtaining more solutions and better results in the end are clearer when the above data tables are converted to many distinct charts. The section below shows the corresponding charts of the main factors of the problems to better highlight the improvement in the algorithm. Below, the figures of the DNA Algorithm and Evolutionary DNA Algorithm for solving the Shortest Path problem can be found. Fig. 4 shows the number of solutions versus the number of nodes for the DNA Algorithm and Evolutionary DNA Algorithm. Clearly, the number of solutions is increased with the proposed evolutionary techniques.

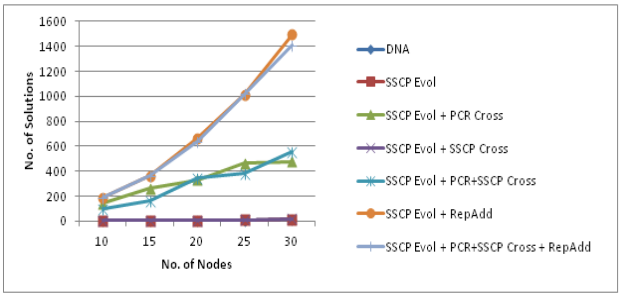

Fig. 4. shows Average No. of PCR Solutions; DNA Algorithm, SSCP Evol, SSCP Evol+PCR Corss, SSCP Evol+ SSCP Cross, SSCP Evol+PCR+SSCP Cross, SSCP Evol+PCR+SSCP Cross+RepAdd

Fig. 5 shows the number of PCR solutions versus the number of nodes for the DNA Algorithm and Evolutionary DNA Algorithm. The PCRCross+RepAdd produced more solutions than others

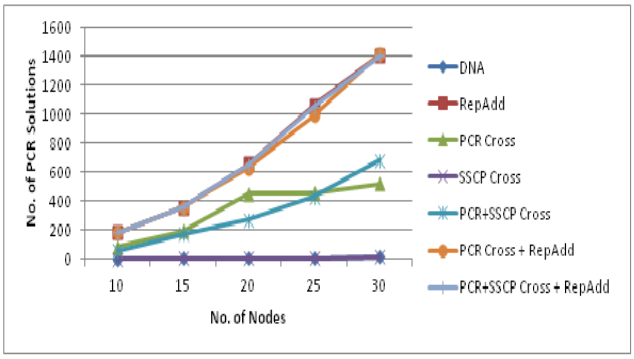

Fig. 5. Average No. of PCR Solutions; DNA Algorithm, RepAdd, PCR Corss, SSCP Cross, +PCR+SSCP Cross, PCR Cross+RepAdd, SSCP Cross+RepAdd

Fig. 6 shows the number of SSCP solutions versus the number of nodes for the DNA Algorithm and Evolutionary DNA Algorithm.

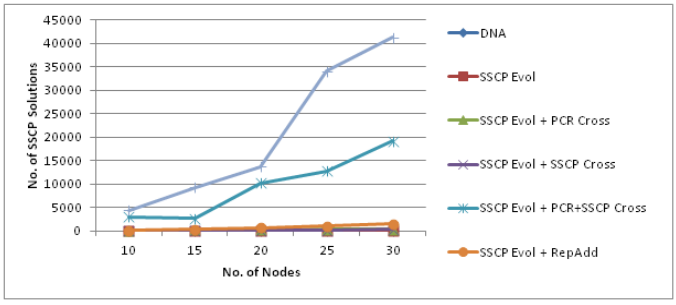

Fig. 6. shows, Average No. of SSCP Solutions; DNA Algorithm, SSCP Evol, SSCP Evol+PCR Corss, SSCP Evol+ SSCP Cross, SSCP Evol+PCR+SSCP Cross, SSCP Evol+PCR+SSCP Cross+RepAdd

Fig. 7 shows the number of SSCP solutions versus the number of nodes for the DNA Algorithm and Evolutionary DNA Algorithm.

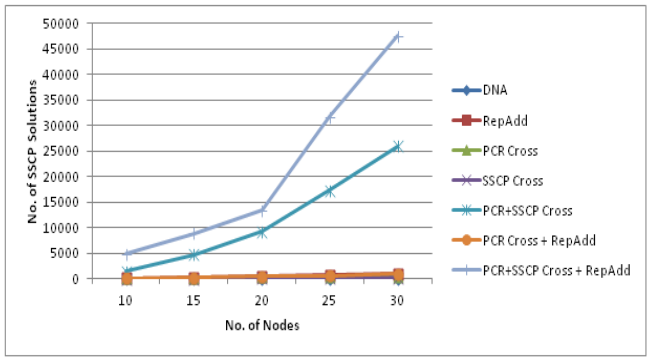

Fig. 7. shows – Average No. of SSCP Solutions; DNA Algorithm, RepAdd, PCR Corss, SSCP Cross, +PCR+SSCP Cross, PCR Cross+RepAdd, SSCP Cross+RepAdd

Fig. 8 shows the number of solutions versus the number of nodes for the DNA Algorithm and Evolutionary DNA Algorithm.

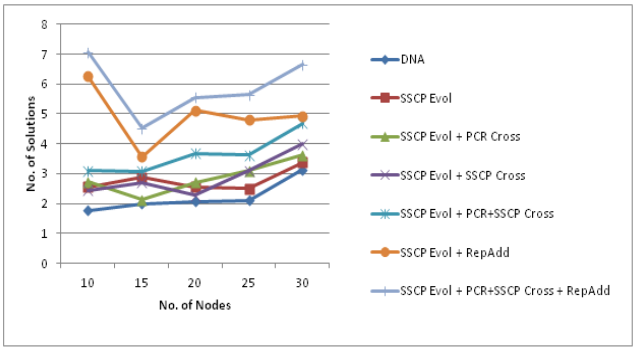

Fig. 8. Average No. of Final Solutions; DNA Algorithm, SSCP Evol, SSCP Evol+PCR Corss, SSCP Evol+ SSCP Cross, SSCP Evol+PCR+SSCP Cross, SSCP Evol+PCR+SSCP Cross+RepAdd

Fig. 9 shows the number of solutions versus the number of nodes for the DNA Algorithm and Evolutionary DNA Algorithm.

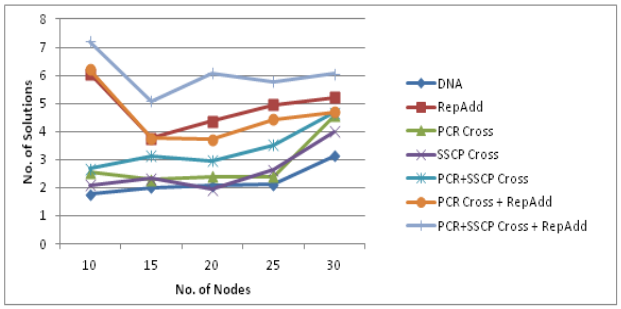

Fig. 9. Average No. of Final Solutions; DNA Algorithm, RepAdd, PCR Corss, SSCP Cross, +PCR+SSCP Cross, PCR Cross+RepAdd, SSCP Cross+RepAdd

Fig. 10 shows the Average path cost versus the number of nodes for the DNA Algorithm and Evolutionary DNA Algorithm. It is clearly seen that the average path cost using DNA algorithm is the most expensive among others.

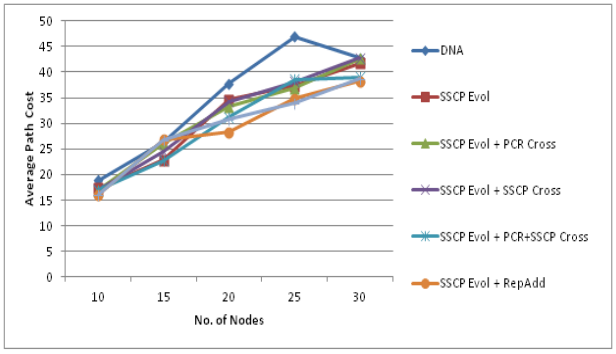

Fig. 10. Average Path Cost; DNA Algorithm, SSCP Evol, SSCP Evol+PCR Corss, SSCP Evol+ SSCP Cross, SSCP Evol+PCR+SSCP Cross, SSCP Evol+PCR+SSCP Cross+RepAdd

Fig. 11 shows the Average Path Cost versus the number of nodes for the DNA Algorithm and Evolutionary DNA Algorithm.

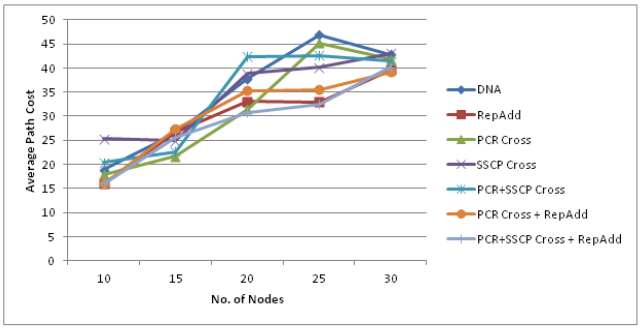

Fig. 11. Average Path Cost; DNA Algorithm, RepAdd, PCR Corss, SSCP Cross, +PCR+SSCP Cross, PCR Cross+RepAdd, SSCP Cross+RepAdd

TABLE I. SHOWS VERTICES SND THEIR COMPLIMENTS

COMPLIMENTS				
Ver1 =	*5'GCGAGATCTG3'*	*Comp: 3'CGCTCTAGAC5'*	*OR*	*5'CAGATCTCGC3'*
Ver2 =	*5'GAAGTCAGTC3'*	*Comp: 3'CTTCAGTCAG5'*	*OR*	*5'GACTGACTTC3'*
Ver3 =	*5'GTCACGACGA3'*	*Comp: 3'CAGTGCTGCT5'*	*OR*	*5'TCGTCGTGAC3'*
Ver4 =	*5'GCCATAGACC3'*	*Comp: 3'CGGTATCTGG5'*	*OR*	*5'GGTCTATGGC3'*
Ver5 =	*5'GTTCGTTTAG3'*	*Comp: 3'CAAGCAAATC5'*	*OR*	*5'CTAAACGAAC3'*
Ver6 =	*5'GATGAGAGTA3'*	*Comp: 3'CTACTCTCAT5'*	*OR*	*5'TACTCTCATC3'*

TABLE II. SHOWS THE REPRESENTATION OF EDGES SND THEIR CORRESPONDING WEIGHTS

Edge	***Weight***
Ver1 ➔Ver2= TAGACCTTCA	*2*
Ver1 ➔Ver3= TAGACCAGTG	*4*
Ver2 ➔Ver3= GTCAGCAGTG	*3*
Ver2 ➔Ver4= GTCAGCGGTA	*5*
Ver2 ➔Ver5= GTCAGCAAGC	*5*
Ver3 ➔Ver4= GACGACGGTA	*2*
Ver3 ➔Ver5= GACGACAAGC	*2*
Ver4 ➔Ver6= CGGTACTACT	*8*
Ver5 ➔Ver4= AAATCCGGTA	*4*
Ver5 ➔Ver6= AAATCCTACT	*6*

TABLE III. STANDARD DNA ALGORITHM

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	6248	55	2	2	2	19	13	27	13	0	0	0	0	0	0	0
15	23	26198	100	3	3	2	26	17	46	54	0	0	0	0	0	0	0
20	30	90071	171	4	4	3	38	16	88	172	0	0	0	0	0	0	0
25	39	170626	267	6	5	3	47	14	142	397	0	0	0	0	0	0	0
30	47	338264	386	15	8	3	43	30	290	1077	0	0	0	0	0	0	0

TABLE IV. IMPROVED DNA ALGORITHM USING REPLACING START/END AT PCR OPERATION

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	5892	55	186	139	28	16	13	53	7	52	0	186	47	0	0	0
15	23	23858	101	360	259	57	27	17	48	48	98	0	360	101	0	0	0
20	30	80279	169	661	498	101	33	16	91	165	164	0	661	163	0	0	0
25	38	164058	266	1064	699	165	33	14	157	712	259	0	1064	365	0	0	0
30	47	348691	381	1409	945	222	40	30	288	1219	367	0	1409	464	0	0	0

TABLE V. IMPROVED DNA ALGORITHM USING PCR CROSS OVER

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	6825	55	87	71	5	18	13	46	27	51	4159	84	16	0	0	0
15	23	27003	100	194	150	7	22	17	68	81	95	10268	189	43	0	0	0
20	30	91793	173	454	349	15	31	16	132	610	168	21817	449	104	0	0	0
25	39	173501	267	461	285	18	45	14	227	532	260	37273	455	176	0	0	0
30	47	348751	385	520	300	29	42	30	398	1246	363	54790	499	220	0	0	0

TABLE VI. IMPROVED DNA ALGORITHM USING SSCP CROSS OVER

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	5826	55	3	5	2	25	13	25	20	51	0	0	0	45	3	0
15	23	26540	100	3	13	2	25	17	48	212	96	0	0	0	42	11	0
20	30	87468	170	4	47	3	39	16	94	263	165	0	0	1	68	44	0
25	39	168514	264	7	70	3	40	14	148	544	256	0	0	2	248	65	0
30	47	364737	383	16	195	4	43	30	288	1081	367	0	0	5	656	184	0

TABLE VII. IMPROVED DNA ALGORITHM USING PCR+SSCP CROSS OVER

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	6098	55	60	1457	5	20	13	47	29	52	4290	58	15	173	1412	0
15	23	27694	99	175	4790	7	23	17	70	96	94	10275	172	51	693	4666	0
20	30	85079	168	272	9312	11	42	16	142	546	162	20665	267	105	2527	9145	0
25	39	166696	265	432	17259	18	43	14	252	621	257	36308	425	192	3903	17019	0
30	47	334954	383	686	26016	37	41	**33**	458	1309	367	55369	671	310	9978	25640	0

TABLE VIII. IMPROVED DNA ALGORITHM USING PCR CROSS OVER + REPLACING START/END AT PCR OPERATION

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	5911	55	187	137	28	16	13	30	19	52	0	187	50	0	0	0
15	23	23973	101	359	257	55	27	17	52	60	97	0	359	102	0	0	0
20	30	82685	169	634	475	98	35	16	95	354	163	0	634	158	0	0	0
25	39	167655	268	995	645	154	36	14	156	513	262	0	995	351	0	0	0
30	48	343511	386	1421	898	205	39	30	288	1008	370	0	1421	523	0	0	0

TABLE IX. IMPROVED DNA ALGORITHM USING SSCP CROSS OVER + REPLACING START/END AT PCR OPERATION

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	5786	55	186	4793	28	16	13	41	31	52	0	186	50	570	4657	0
15	23	25132	101	360	8347	59	27	17	68	320	98	0	360	94	2554	8081	0
20	30	84607	169	632	14160	100	32	16	120	391	163	0	632	161	4421	13689	0
25	39	164140	266	1010	30632	158	34	14	212	696	259	0	1010	342	7742	29964	0
30	47	346084	380	1397	41136	217	40	30	368	1237	364	0	1397	470	12694	40210	0

TABLE X. IMPROVED DNA ALGORITHM USING PCR+SSCP CROSS OVER + REPLACING START/END AT PCR OPERATION

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	6523	55	186	4856	29	16	13	71	50	52	0	186	50	429	4719	0
15	23	25262	100	358	8853	57	26	17	120	381	97	0	358	102	3012	8597	0
20	30	82588	170	652	13386	105	31	16	215	745	164	0	652	154	4783	12888	0
25	38	167798	264	1053	31748	162	33	14	369	1209	257	0	1053	360	9255	31055	0
30	47	338903	381	1404	47527	209	40	30	656	2245	366	0	1404	533	12219	46656	0

TABLE XI. IMPROVED DNA ALGORITHM USING EVOLUTIONARY SSCP

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	6383	55	3	32	3	17	13	1319	1285	51	0	0	0	0	0	11,400
15	23	25712	100	4	37	3	23	17	1721	664	95	0	0	1	0	0	16,847
20	30	88395	171	5	46	3	35	16	2093	876	165	0	0	0	0	0	15,270
25	39	169204	264	7	54	3	37	14	2847	1019	256	0	0	2	0	0	26,411
30	47	340939	387	16	69	4	42	30	4189	855	371	0	0	5	0	0	11,195

TABLE XII. IMPROVED DNA ALGORITHM USING EVOLUTIONARY SSCP + PCR CROSS OVER

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	6503	55	141	117	5	17	13	1190	1177	51	4416	138	52	0	0	48911
15	23	28113	101	261	223	7	26	17	1560	1208	96	10645	257	71	0	0	18559
20	30	88658	171	328	264	10	33	16	2022	477	165	21173	323	105	0	0	18237
25	39	167883	266	461	326	21	37	14	2619	705	258	36408	454	185	0	0	9635
30	47	340505	385	478	344	26	43	30	3720	795	370	56308	464	193	0	0	33182

TABLE XIII. IMPROVED DNA ALGORITHM USING EVOLUTIONARY SSCP + SSCP CROSS OVER

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	6308	55	2	31	2	17	13	1254	967	52	0	0	0	0	0	17900
15	23	25644	100	3	48	3	24	17	1537	1231	96	0	0	0	48	11	19387
20	30	92930	171	4	76	3	34	16	2130	459	166	0	0	1	36	32	18593
25	39	161309	266	8	125	4	38	14	2712	789	257	0	0	2	316	70	26349
30	47	329616	386	16	369	4	43	30	3500	1120	369	0	0	6	615	301	24955

TABLE XIV. IMPROVED DNA ALGORITHM USING EVOLUTIONARY SSCP + PCR+SSCP CROSS OVER

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	5839	55	97	3086	5	17	17	1324	1150	52	4577	94	32	431	2993	48161
15	23	28748	99	161	2665	7	23	17	1484	1068	96	11258	158	31	1226	2502	22009
20	30	86501	170	345	10233	13	31	16	2086	829	165	21433	340	113	2483	9961	16995
25	39	167654	268	382	12767	17	39	14	2653	1208	261	37471	377	146	3875	12480	9759
30	47	356121	383	555	19193	27	39	30	4100	683	362	54019	536	222	6353	18802	27878

TABLE XV. IMPROVED DNA ALGORITHM USING EVOLUTIONARY SSCP + REPLACE/ADD START/END AT PCR OPERATION

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	6089	55	186	166	29	16	13	1162	1496	52	0	186	49	0	0	24942
15	23	23191	101	361	285	53	27	17	1471	1091	98	0	361	110	0	0	17620
20	30	84220	170	666	541	106	28	16	2003	430	166	0	666	167	0	0	15522
25	39	158626	267	1014	697	152	35	14	2539	890	261	0	1014	367	0	0	28331
30	47	330718	381	1496	1007	211	38	30	2952	1201	365	0	1496	547	0	0	23650

TABLE XVI. IMPROVED DNA ALGORITHM USING EVOLUTIONARY SSCP + PCR+SSCP CROSS OVER + REPLACE/ADD START/END AT PCR OPERATION

V	E	LIG	RSG	PCRP	SSCP	GELP	AP	SP	RT	MC	drpPCR	pcrCross	pcrGen	drpSSCP	sscpCross	sscpGen	evolSSCPcross
10	18	5696	55	188	4477	30	16	13	1266	1399	52	0	188	48	855	4308	22460
15	23	24291	101	361	9325	58	27	17	1442	1187	98	0	361	99	1484	9029	15070
20	30	81129	169	637	13710	101	31	16	1959	462	165	0	637	163	5782	13195	13177
25	39	153987	267	1018	34081	151	34	14	2542	1165	261	0	1018	383	8935	33396	24580
30	47	347375	385	1410	41275	214	39	30	3079	1194	368	0	1410	473	13073	40279	21016

VII. CONCLUSIONS

In this paper, the fundamental ideas of DNA computing and evolutionary strategies are presented and elaborated. DNA computing is employed to resolve the shortest path problem. The results of the DNA computing algorithm are obtained and the performance of the algorithm is evaluated. Better results are thereby verified. Thus, a suggested Evolutionary DNA Algorithm was considered to take advantage of the Evolutionary Strategies by being embedded in the normal DNA Algorithm to optimize it and hence obtain better results. The optimization produces better results; this means that the number of solutions is increased; thus, the possibility of obtaining optimum solutions is increased as well. Additionally, because the evolutionary technique is used, the initial resulting solutions are evolved; hence, the average quality of the solution generation after generation is increased.

REFERENCES

[1] M. Adleman, " Molecular computation of solutions to combinatorial problems," Science, vol. 266, pp. 1021-1024, 1994.

[2] S. Hari, K. Rajeev and S. Vikas, "An approach towards the solution of NP-Complete Problem," Report and Opinion, vol. 3, no. 5, 2011.

[3] D. Boneh, C. Dunworth, J. Lipton and I. Sgall, "On the computational power of DNA," Discrete Applied Mathematics, vol. 71, pp. 79–94, 1996.

[4] L. Kari, G. Gloor and Y. Sheng, "Using DNA to solve the Bounded Post Correspondence Problem," Theoretical Computer Science, vol. 231, no. 2, pp. 192–203, 2000.

[5] G. Gautam and C. Biswanath, "A cascaded pairwise biomolecular sequence alignment technique using evolutionary algorithm," Information Sciences, 2014.

[6] C. Rudy, M. Buyong, N. Ruth and S. Amarda, "Mapping the Conformation Space of Wildtype and Mutant H-Ras with a Memetic, Cellular, and Multiscale Evolutionary Algorithm," PLoS Computational Biology; 2015.

[7] S. Junnarkar, "In Just a new Drops, A Breakthrough in Computing," New York Times; 1997.

[8] F. Mancini, "New perspectives on the Ising model," the June, vol. 45, no. 4, pp. 497-514, 2005.

[9] M. Ogihara M and A. Ray, "Simulating Boolean circuits on a DNA computer, Algorithmica, Published by Springer, vol. 25, pp. 239–250, 1999.

[10] T. Bäck, N. Kok and G. Rozenberg, "Evolutionary computation as a paradigm for DNA-based computing," In Landweber, L.F. and Winfree,E. (eds), Evolution as Computation. DIMACS workshop, Princeton, January 1999. Springer-Verlag, Heidelberg, Germany, pp. 15–40, 2003.

[11] T. Bäck, "Evolutionary Computation as a Paradigm for DNA-Based Computing, " Natural Computing Series; 2002.

[12] S. Lovgren, "Computer Made from DNA and Enzymes," National Geographic; 2003.

[13] H. Ahrabian and D. Nowzari, "DNA Simulation of Nand Boolean Circuits," AMO - Advanced Modeling and Optimization, vol 6, No. 2, pp 33-41, 2004.

[14] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar R. and E. Shapiro," An autonomous molecular computer for logical control of gene expression," Nature, vol. 429, no. 6990, pp. 423–429, 2004.

[15] N. Dimitrova,"Intelligent Algorithms in Ambient and Biomedical Computing," The Many Strands of DNA Computing, vol. 7, pp. 21-35, 2006.

[16] N. Nafiseh, Z.Sirous, L. Manigeh, K. Esmat, K. Morteza," Rapi and sensitive detection of point mutations and DNA polymorphisms in Factor IX Gene by using the Single Strand Conformation Polymorphism (SSCP)," the fourth biotechnology congress, Krmansha, Iran, 2005.

[17] S. Abdullah, "An improved DNA Computing approach using Heuristic Techniques, " Ph.D. Thesis, Computer Sciences Department, University of Technology Baghdad, Iraq, 2008.

[18] S. Hari S., K. Rajeev K. and S. Vikas, "An approach towards the solution of NP-Complete Problem," Report and Opinion, vol. 3, no. 5, 2011.

[19] M. Adleman, "Computing with DNA, The manipulation of DNA to solve mathematical problems is redefining what is meant by computation," Copyright Scientific American, Inc, 1998.

[20] M. Amos, "DNA Computing," Invited article for the Encyclopedia of Complexity and System Science, Springer, Manchester Metropolitan University, United Kingdom, 2008.

[21] R. Sekhar, "DNA Computing-Graph Algorithms. Department of Mathematics," Indian Institute of Technology, This work is partially supported by Com2MaC-KOSEF, Korea; 2010.

[22] G. Ibrahim, " Improving DNA Computing using Evolutionary Algorithms," Master Thesis. Software Engineering, College of Engineering, Salahaddin University, Hawler, Kurdistan, 2012.

[23] M. Yamamoto, N. Matsuura, T. Shiba, Y. Kawazoe and A. Ohuchi, "DNA Computing, Solutions of Shortest Path Problems by Concentration Control," Lecture Notes in Computer Science, Springer link, 2002, vol. 2340, pp. 203-212.

[24] J. Orlin, R. Ahuja, D. Simchi-Levi, S. Chopra, B. Golden and B. Kaku, " Networks and Flows," Discrete Mathematics and Its Applications, CRC Press; 1999.

[25] T. Ootaa and Y. Yasuib, "Toric Sasaki-Einstein manifolds and Heun equations, Nuclear Physics, Section B, vol. 742, no. 1–3, 2006.

[26] B. Datta and N Nilakantan, "Two-dimensional weak pseudomanifolds on eight vertices, " Proceedings of the Indian Academy of Sciences - Mathematical Sciences, vol. 112, no. 2, pp. 257-281, 2002.

[27] A. Simovici and C. Djeraba, "Advanced Information and Knowledge Processing, Mathematical Tools for Data Mining, Set Theory, Partial Orders, Combinatorics," Springer-Verlag London Limited, 2008.

APPENDIX A

AP	Represents the Average Path
DNA	Deoxyribonucleic Acid
drpPCR	Represents the number of dropped PCR solutions
drpSSCP	Represents the number of dropped SSCP solutions
E	Represents the number of network edges
EA	Evolutionary Algorithm
evolSSCPcross	Represents the number of Evolutionary SSCP Crossover operations
GA	Genetic Algorithms
GELP	Represents the number of Gel Electrophoresis solutions
GRS	Generate Random Solutions
HDNA	Heuristic Deoxyribonucleic Acid
HPP	Hamiltonian Path Problem
LIG	Represents the number of DNA Ligations
MC	Represents the Memory Capacity of the DNA Algorithm
MER	The length of the oligonucleotide is usually denoted by "mer" (from Greek meros, "part")
PCR	Polymerase Chain Reaction
pcrCross	Represents the number of PCR Crossover operations
pcrGen	Represents the number of PCR solutions generated by Crossover Operation
PCRP	Represents the number of PCR solutions
RNA	Ribonucleic Acid
RSG	Represents the number of Random Solutions Generated
SP	Represents the Shortest Path
SPP	Shortest Path Problem
SSCP	Single Strand Conformation Polymorphism, Represents the number of SSCP solutions
sscpCross	Represents the number of SSCP Crossover operations
sscpGen	Represents the number of SSCP solutions generated by Crossover Operations