



# Improving DNA Computing through CRISPR based Model and Visual DNA Tool

A.M. El-Edkawy

Computer Science Department,  
Faculty of Computers and  
Information  
Mansoura University  
Egypt  
amr.eledkawy@mans.edu.eg

M.A. El-Dosuky

Computer Science Department,  
Faculty of Computers and  
Information  
Mansoura University  
Egypt  
mouh\_sal\_010@mans.edu.eg

Taher Hamza

Computer Science Department,  
Faculty of Computers and  
Information  
Mansoura University  
Egypt  
taher\_hamza@yahoo.com

## ABSTRACT

Since the advent of DNA computing field; there's debate about its ability to solve hard computational problems. A way to deal with the debate is to design a computational model for DNA computing and then studying its complexity power. The main contribution of this paper is to propose a DNA computing model that presents a usage of CRISPR in DNA computing field. The model has three basic operations which are merge, CRISPR and detect. The model is argued to be robust, as the three operations are robust, and to be the minimum model for DNA computing in terms of the number of operations. The proposed model has corresponded with a model inspired from the evolution of DNA sequences called Accepting Hybrid Network of Evolutionary Processors (AHNEPs). Based on that correspondence, the proposed model can be used to solve NP problems in polynomial time and PSPACE problems in polynomial space. The model is used to solve, the NP problem, the Hamiltonian Path Problem (HPP) in linear time. The limitations of DNA computing area could be avoided within the model. Also, the study presents "Visual DNA" which is a software that can simulate biochemical operations in DNA computing. Also, the software can make analysis for DNA sequences. So this simulation software will offer a useful tool for the DNA computing implementation because it will help in the analysis of input DNA sequences and in the prediction of output DNA sequences which would be helpful to avoid errors during the experimental process.

## General Terms

DNA Computing, Theoretical Computer Science, Computability Theory

## Keywords

Accepting Hybrid Network of Evolutionary Processors, CRISPR, DNA Computing, NP, PSPACE, Visual DNA, HPP.

## 1. INTRODUCTION

Silicon chips have been the core of the computers for over 50 years. As per Moore's Law, the number of electronic gadgets put on a chip has doubled every 18 months. Numerous have anticipated that Moore's Law will soon achieve its end, in light of the physical speed and scaling down restrictions of silicon chips so there's "No more of Moore's Law" [43].

DNA Computing is the execution of calculations utilizing natural particles, and DNA particularly, rather than silicon. DNA computing is a new and attractive improvement as interdisciplinary between molecular biology and computer science. It has developed lately, not just as an attractive innovation for information processing, yet additionally as a catalyst for learning exchange between biology, nanotechnology, and information processing. This region of research can possibly change our comprehension of the hypothesis and developments in computing.

The upsides of DNA over silicon consist of the following reasons [44]:

- Huge parallelism: calculations can be achieved simultaneously, instead of sequentially in silicon.
- Size: DNA Computers are much smaller than silicon ones.
- Storage density: a whole lot more data can be put away in a similar space quantity.

The fundamental part of information processing and the capability to address it on labs at the atomic level was first tended to by Adleman's experiment [1], which exhibited that the devices of lab molecular biology can be utilized to develop programs with DNA in vitro.

DNA computing methodologies can be achieved either in vivo (i.e. inside cell living things) or in vitro (inside test tubes). Tubes contain a finite number of DNA strands; where each strand is composed of a set of DNA bases which are adenine (A), cytosine (C), guanine (G), and thymine (T). Biologically, C attaches only to G and A attaches only to T on an complement strand of the DNA molecule. C is called the complement of G and vice versa. A is called the complement of T and vice versa [44].

DNA computing field first appeared for solving Hamiltonian Path Problem (HPP), as a famous example for an NP-complete problem, utilizing techniques of recombinant DNA [1]. Later, NP-complete problems solutions using DNA computing have been proposed, e.g. satisfiability (SAT) [2], 3-SAT [3], maximal clique [4]. The influence of high-density, parallel computation using molecules in solutions permits computing based on DNA molecules to solve problems, considered hard in computation like NP-complete problems, in polynomial time whereas a traditional Turing Machine (TM) needs exponential time [5, 6].

Nevertheless, besides the error susceptibility of biochemical operations, DNA computing suffers from exponential space problem [7]. The algorithm in [1] touches its space limit with an instance of the problem of size around 60 and of size 70 in the algorithm explained in [2], and that size of instances could be quickly solved by traditional computers. This raises the debate about whether DNA computing will take us to another place where a solution to all cases of hard problems that cannot be solved in traditional TM can be found.

A way to deal with the debate is to design a computational model for DNA computing, then analyzing the model by studying its power. The current models are built on several mixtures of some basic biochemical operations shown in the following list [8].

- Synthesize: Synthesizing the desired strand.
- Annealing: The solution is cooled to attach two complementary single strands together.
- Marking: Using hybridization to mark single strands, where complementary sequences transform them double-stranded by attaching to them.
- Melting: The solution is heated to have two complementary single strands from a double-stranded DNA.
- Merge: Mix test tubes contents in another one to realize unification of DNA bases.

- Amplifying: Utilizing Polymerase Chain Reaction (PCR) to Copy strands.
- Cutting: Utilizing restriction enzymes to cut strands at cutting sites.
- Substitution: Utilizing using PCR site-specific oligonucleotide mutagenesis to substitute, insert or delete sequences of DNA.
- Ligation: Utilizing ligase to attach strands that have compatible sticky ends.
- Extraction: Utilizing affinity purification to extract strands of the matched pattern.
- Length: Utilizing gel electrophoresis to use strands length as a way to separate them.
- Destroying: Destroying strands by exonucleases.
- Detecting: Reading out the tube contents.

The time complexity for each of the above operations is  $O(1)$  because they can be implemented in a constant number of biochemical steps [45].

The main contribution of this paper is to propose a DNA computing model that presents a usage of CRISPR [9] in DNA computing field. The model has three operations which are merge, CRISPR and detect. The model is argued to be robust, as the three operations are robust, and to be the minimum model for DNA computing. The proposed model has corresponded with a model inspired from the evolution of DNA sequences called Accepting Hybrid Network of Evolutionary Processors (AHNEPs). Thus, the proposed model can solve NP problems in polynomial time and PSPACE problems in polynomial space. The model is used to solve, the NP problem, the Hamiltonian Path Problem (HPP) in linear time

Also, this study presents "Visual DNA" which is a software that can simulate biochemical operations in DNA computing. It can be used to make sure of the results of the biochemical operations before doing in vitro experiments. Also, the software can make analysis for DNA sequences. So this simulation software will offer a useful tool for the DNA computing implementation because it will help in the analysis of input DNA sequences and in the prediction of output DNA sequences which would be helpful to avoid errors during the experimental process.

The following parts of the paper are arranged as follows. Section 2 reviews the work done before in designing models for DNA computing and studying its power, it also provides an overview on AHNEP, and also presents the related work for the simulation software. Section 3 defines the proposed model and discusses the power of it. Section 4 explains Visual DNA tool. Section 5 concludes the paper.

## 2. Previous Work

### 2.1 DNA computing models

The following related studies demonstrate that the choice of possible biochemical operations, which are real or unreal, in the models affects the power and the robustness of the model.

In [10], a model named restricted model is proposed that's based on method explained in [1] where the model has only three operations that are merge, detect and extract using affinity purification. Also [10] studies another model named unrestricted model that's based on the three operations in the restricted model plus the operation of amplify using PCR. The study studies the complexity power of the models to be branching programs (BP) for the restricted model and to be nondeterministic branching programs (NBP) for the unrestricted model.

In [11], a model is proposed that have the operations of extraction, ligation, annealing, merge, length and detect and the model is characterized to have the complexity power of  $\Delta_2^P$  or equivalently  $P^{NP}$ .

In [12], a model called DNA-PASCAL model has twelve instructions depending on several operations where the reality of these operations have not been argued. They characterized their DNA-PASCAL model computational power to be one of the complexity classes P,  $\Delta_2^P$  or  $\Delta_3^P$ . Furthermore, in [13], a model called DNA-EC depending on equality checking principle. The model is similar to DNA-PASCAL model but the operations in it are more feasible. DNA-EC model has been characterized to simulate Universal Turing Machine.

In [14], a model based on the method used in [1] where the model has operations of anneal, amplify, length, detect and replaces the extract operation with substitution. The model is characterized to have the power of PSPACE.

In [15], the study presented two models of computation the first is called Parallel Association Memory (PAM) model and the other is called Recombinant DNA (RDNA) model. The latter is based on six recombinant DNA operations that's length, extract, anneal, melt, amplify, cut plus the operations of merge and detect. The models are characterized to have the power of PSPACE.

In [16, 17], a model that have five operations of the merge, detect, synthesize, anneal and length. The power of the computational model is between  $NC^k$  and  $SAC^{k+1}$  for  $k \geq 1$ . Furthermore, In [18], an expanded model of that in [16, 17] is presented to have eight operations that are synthesize, merge, length, detect, ligation, cut, ASM (Anneal, Separate by length, Melt) and PM (Polymerase, Melt). That model expands the power of the model in [16, 17] from NC to PSPACE.

There are other models of DNA computation of different types such as self-assembly computational models [19, 20], hairpin formation systems [21], sticker systems [22, 23], the DNA L systems [24] and splicing systems [25].

Of all the models studied, the model in [10] named the restricted model is the smallest in terms of the number of operations. A possible question is there smaller models for DNA computing, i.e., may one or more of the three operations be excluded or be changed by a more robust and powerful operation. Detect is the most efficient possible way for reading information from strands of DNA, and Merge is the only way for combining information. Extract is very susceptible to error, especially with large sequences. Then we ask; could we replace extract with another operation that's more robust and studying the effect of that change on the computational power.

CRISPR [9] is reliable and expands the proposed model to a higher computational power as shown in the subsequent section. So we argue that the model proposed by the paper is robust. Also, we argue that it's the minimum DNA computational model.

### 2.2 AHNEP

Network of Evolutionary processors (NEPs) [26] is an inspired model from the cell evolution via the evolution of the sequences of their DNA. It's inspired also from the fundamental design for distributed and parallel processing in logic flow diagrams [27] and connection machines [28]. The model comprises of many processors, where each one is located in a complete graph node and deals with data in that node. Also, each processor is evolutionary meaning that it can do point mutation operations in a sequence of DNA (substitution, insertion, deletion). The data in every node of the model is organized in sets of words that have many copies that are all parallel processed [29].

Hybrid networks of evolutionary processors (HNEPs), wherever every processor makes only single operation on a specific site in the words of the node. HNEPs can be language accepting devices (AHNEPs) [29] or language generating devices (GHNEPs) [30].

The configuration of AHNEP can be changed either by evolutionary step using point mutation operations or by communication step by sending strings to the connected node. The computation in AHNEP halts if the output node contains strings and is not empty and this case is called "accepting computation".

Following the literature on AHNEPs, many useful theoretical results appear that's shown below:

- For any TM recognizing a language  $L$  there's an AHNEP  $\Gamma$  accepting the same language  $L$  [31].
- For any nondeterministic  $TM(M)$ , recognizing a language  $L$  there's an AHNEP  $\Gamma$  accepting the same language  $L$ . Moreover, if  $M$  works within time  $f(n)$  then  $Time \Gamma(n) \in O(f(n))$  [29].
- $poly - time AHNEP \subseteq NP$  [32].
- $poly - length AHNEP \subseteq PSPACE$  [32].

- $PSPACE \subseteq poly - length AHNEP$  [32].
- $NP \subseteq poly - time AHNEP$  [32].
- $poly - time AHNEP = NP$  [29, 31, 32].
- $poly - length AHNEP = PSPACE$  [32].

### 2.3 DNA Computing Tools

Several studies tried to simulate DNA Computing operations. In [35] a simulation software for the DNA computing algorithm in [1] is proposed. In [36] a simulation software for the DNA computing solution of the elevator scheduling problem is proposed.

In [37, 38] a programming language for DNA computation is proposed based on the concept of strand displacement. In [39] a programming language for DNA computation is proposed based on the concept of chemical reaction networks. In [40] a query language for DNA computing called DNAQL is proposed. In [41] a software called Cello is proposed where you write a Verilog code that's automatically transformed for DNA sequences used for computation. In [42], the authors proposed a cloud-based tool called Genetic Constructor that can be used for designing and manipulating DNA fragments.

## 3. CRISPR Based Model

### 3.1 There Proposed Model Operations

There are three basic operations in the model. Merge and detect which are mentioned before and the third is CRISPR.

Merge: pour the contents of test tubes  $T1, T2, T3 \dots$  which are taken as input into another one which would be the output to realize unification of bases T. The operation is robust [18]. It can be written as  $T \leftarrow Merge(T1, T2, \dots)$

CRISPR: the operation targets and edits DNA robustly [9], deletes and appends sequences [33] and has been shown to be used in vitro efficiently [34]. The operation can be used in three ways to insert, delete or substitute DNA sequences.

CRISPR can be used to implement other DNA computing operations rather than the substitute operation thus assuring the universality of the proposed CRISPR based model.

It can be used as an extraction operation listed before. Arguing that designing that is possible, just add a tag to the Cas9 and pull the tag out. This use is named CRISPR-E.

Also, it can be used as length operation listed before. Arguing that designing that is possible by enumerating all the possible patterns of the length needed and using them as a parameter for the substitute operation. This use is named CRISPR-L.

Another usage it to operate as cut operation listed before. Arguing that designing that is possible just by making use of the Cas9 endonuclease enzyme for cutting. This use is named CRISPR-C.

CRISPR instructions can be formally defined as the following where T1 or T are given test tubes, T2 is the output test tube, x, y, u, v are strings in a DNA strand, N is an integer, S are given set of strings, and  $\beta\theta\beta1$  are two signs.

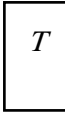


- Substitute (T, u, v): Given a tube T, for all strands that have string u; substitute the string u by string v and the output still in the same tube.

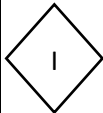
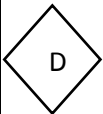
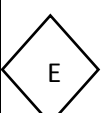
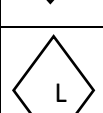
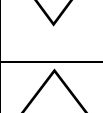
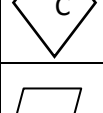
- Insert (T, x, y, u): Given a tube T, for all strands that have string x followed by string y; insert string u between x and y and the output still in the same tube.
- Insert (T, x, y, u): Given a tube T, for all strands that have string x followed by string y; delete string u between x and y and the output still in the same tube.
- CRISPR-E (T1, S, T2): eliminates all strands having a string in S from T1, and produces a test tube T2 with the eliminated strands;
- CRISPR-L(T1, N, T2): eliminates all strands with length N from T1, and produces a tube T2 with the eliminated strands;
- CRISPR-C(T,  $\beta\theta\beta1$ ): slices every strand having  $[\beta\theta\beta1]$  in T into diverse strands:  $[\dots\delta\beta\theta\beta1\gamma\beta\theta\beta1\alpha\dots] \rightarrow [\dots\delta\beta\theta], [\beta1\gamma\beta\theta], [\beta1\alpha\dots]$ ;

Detect: the operation takes a tube as input and outputs true or false. It returns true if T has one or more  $5' \rightarrow 3'$  strings; otherwise false is returned. The operation is robust [16]. It can be written as  $Detect(T)$ .

The instructions of the proposed model are listed below.

**Table 1** Proposed Model Instructions

Instruction	Symbol	Meaning	
$T$		A tube that's empty or containing DNA sequences.	
<i>Merge</i>		Mix test tubes contents in another one to realize unification of DNA bases.	
Operations	Substitute		Substitute a DNA sequence with another.

CRISPR	Insert		Insert DNA sequence.
	Delete		Delete DNA sequence.
	CRISPR-E		Extract strands of the matched pattern.
	CRISPR-L		Separate strands with a specific length.
	CRISPR-C		Cut strands at cutting sites.
Detect			Read out the tube contents.

**Fig. 1** Standard algorithm for solving HPP.

The proposed model can be used to solve the HPP as follows:

- Step 1: Generate random paths answer pool

We use the same encoding way used in [1] and shown in figure 2. The algorithm represented every city isolated, single-stranded DNA molecules, twenty bases in length, and every single conceivable way among cities as DNA molecules made out of the last ten bases of the leaving city and the initial ten bases of the arrival city. Merging the DNA strands with DNA ligase brought about all the conceivable arbitrary ways through the cities. This step can be performed as the following:

- (1-1) encoding 20-mer oligonucleotides representing cities in tube  $T_1$ ;
- (1-2) encoding 20-mer oligonucleotides representing edges in tube  $T_2$ ;
- (1-3) Merge ( $T_1, T_2$ );

After merging the random paths answer pool would be in  $T_1$ . Finishing step 1 is done in  $O(1)$  time steps since each operation have  $O(1)$  time steps.

- Step 2: Eliminate paths that don't start and end in the start and end vertex respectively

The second step can be done by implementing CRISPR-E instruction two times; the first to eliminate strands that don't start with the start vertex, and the other to eliminate strands that don't end in the end vertex. This step can be performed in the proposed model as the following where  $S_1$  is the string of the start vertex and  $S_2$  is the string of the end vertex:

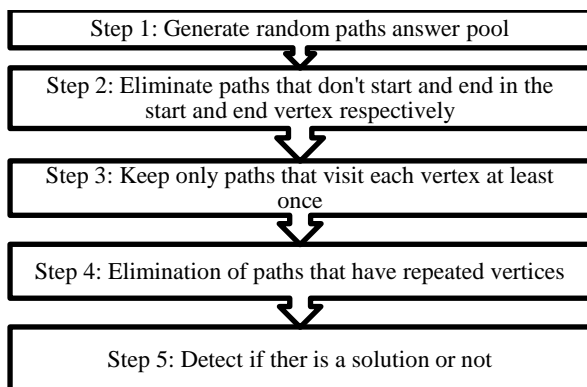
- (2-1) CRISPR-E ( $T_1, S_1, T_3$ );
- (2-2) CRISPR-E ( $T_3, S_2, T_4$ );

After these two steps; only paths that start in the start vertex and end in the end vertex would be in  $T_4$ . Finishing step 2 is done in  $O(1)$  time steps since each operation have  $O(1)$  time steps.

A program using the model is statements that have the following format:  $(T \leftarrow Merge(T_1 \leftarrow Insert(T_2, w), T_3, \dots))$ , that means one or more CRISPR operations (insert, delete, CRISPR-E, CRISPR-L, CRISPR-C or substitute) on tubes and then merge tubes. Note that tube  $T$  can be empty prior to the merging operation. Also with this form of statements we can make more changes or test more variables once by making parallel CRISPR operations on different test tubes. If in the end there's a tube has sequences satisfying problem  $p$  we say that the program solved  $p$ .

### 3.2 Solving HPP using CRISPR Based Model

The Hamiltonian Path Problem (HPP) in which there's a graph of vertices connected by edges and the problem is to find a path the visits every vertex in the graph exactly once. It's a famous example of NP problems. The standard solution is shown in the following figure.



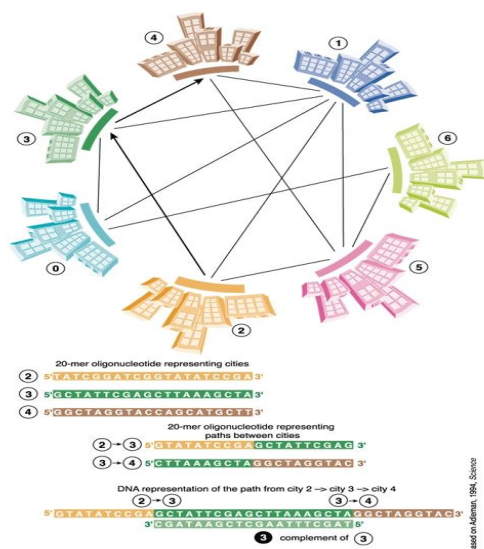


Fig. 2 Random Paths Generation [1].

- Step 3: Keep only paths that visit each vertex at least once

The third step can be done by implementing CRISPR-E instruction  $n-2$  times (where  $n$  is the number of vertices in the graph); where strands that don't have any of the vertices are eliminated. This step can be performed in the proposed model as the following where  $S_k$  is the string of the  $k$ th vertex:

For  $k=2$  to  $k=n-1$ .

$$(3-1) \text{ CRISPR-E } (T_{k+2}, S_k, T_{k+3});$$

After these three steps; the output would be in  $T_{n+2}$ . Finishing step 3 is done in  $O(n)$  time steps since each operation have  $O(1)$  time steps but the "For" clause makes this step takes  $O(n)$  time steps.

- Step 4: Elimination of paths that have repeated vertices

The fourth step can be implemented through CRISPR-L by eliminating the strands that don't have the length of the desired solution; that length is calculated depending on the encoding method of vertices and paths. In our encoding method, we encoded vertices and edges with twenty bases strands. If we have  $n$  vertices so the length of the desired solution would be  $40n$ . This step can be performed in the proposed model like the following:

$$(4-1) \text{ CRISPR-L } (T_{n+2}, 40n, T_{n+3});$$

After these four steps; the output solution would be in  $T_{n+3}$ . Finishing step 4 is done in  $O(1)$  time steps since each operation have  $O(1)$  time steps.

- Step 5: Detect if there is a solution or not.

The fifth step is done using detection operation which reads out the solution and it is implemented as follows:

$$(5-1) \text{ Detect } (T_{n+3});$$

After these five steps; the solution would be found in this step if it exists. Finishing step 5 is done in  $O(1)$  time steps since each operation have  $O(1)$  time steps.

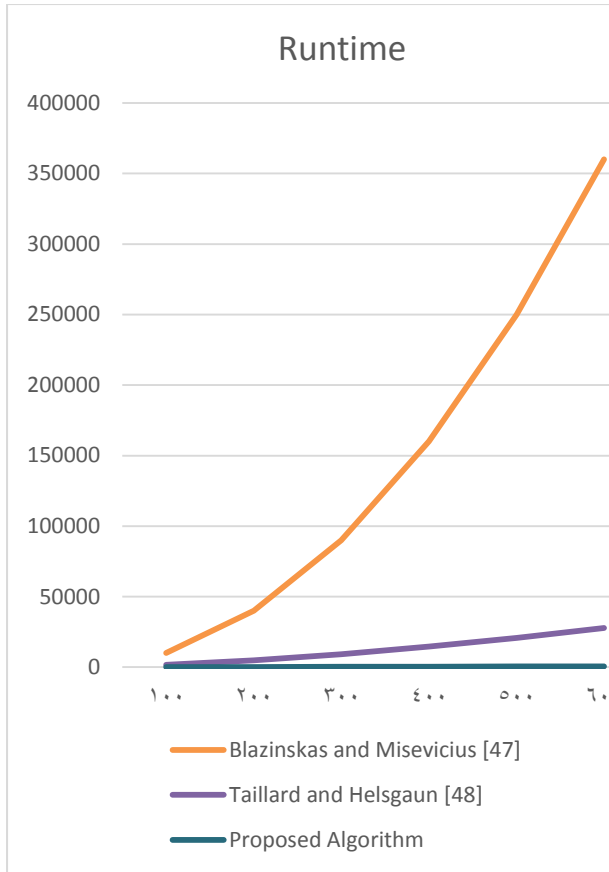
Also, finishing all the algorithm steps could be done in a finite number of operations. Because steps 1, 2, 4, 5 are finished in  $O(1)$ , step 3 is finished in  $O(n)$ . In conclusion, the solution of TSP for a graph with  $n$  vertices can be acquired in  $O(n)$  steps. The proposed algorithm is linear in time.

The following table compares between the proposed HPP algorithm using the CRISPR based model with previous HPP algorithms.

Table 2 Comparison between previous HPP solution algorithms and the proposed one.

Algorithm	Runtime	Deterministic
Brute Force	$O(n!)$	Yes
Eppstein [46]	$O(2^n/3)$	Yes
Blazinskas and Misevicius [47]	$O(n^2)$	No
Taillard and Helsgaun [48]	$O(n^{1.6})$	No
<b>Proposed Algorithm</b>	<b><math>O(n)</math></b>	<b>Yes</b>

The Table clarifies that the proposed algorithm is the fastest algorithm and gives the most accurate results. The following figure clarifies that.



**Fig. 3** Comparison between the proposed algorithm and previous ones solving HPP. The x-axis is the number of nodes in the HPP instance and the y-axis is the runtime.

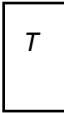
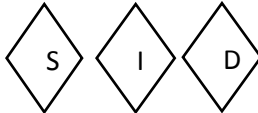

### 3.3 Correspondence between proposed model and AHNEP

By reference to the model of AHNEP; the simulation is possible between AHNEP and the proposed model if we consider the following:

- Simulating nodes that have processors with test tubes.
- Simulating words in the processors with DNA sequences in the test tubes.
- Operations that are important for the processor to be evolutionary all can be made using CRISPR operation.
- Communication between nodes can be simulated by the Merge operation.
- Accepting computation in AHNEP is simulated by Detect operation.

The following table shows the correspondence between the two models.

**Table 3** Correspondence between the proposed model and AHNEP

Proposed Model	AHNEP
	Processors
DNA sequences	Words
Parallel processing of DNA sequences	Parallel processing of words
	Operations in AHNEP are the same; substitute, insert, delete
	Edges making communication between nodes
<i>Detect(T)</i> returns true if <i>T</i> has one or more 5'→3' strings.	Accepting computation if the output node contains strings.

### Corollaries

Let the time complexity of programs in the proposed model is denoted by  $Time(CM)^1$  and the space complexity of programs in the proposed model is denoted by  $Space(CM)$ . Let  $P$  be a program in the proposed model that could be simulated by a language in AHNEP as shown before. With a return to the theoretical results on AHNEP already presented in the previous work section; the following corollaries hold:

- $poly - Time(CM) \subseteq NP$
- $poly - Space(CM) \subseteq PSPACE$
- $PSPACE \subseteq poly - Space(CM)$
- $NP \subseteq poly - Time(CM)$
- $poly - Time(CM) = NP$
- $poly - Space(CM) = PSPACE$

### 3.4 Comparison between CM and Previous Models

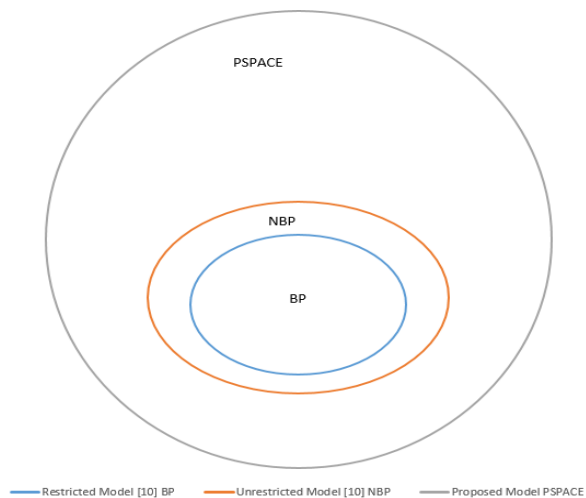
The following table shows a comparison between CM and previous models in terms of the higher solvable class of problems within the model, the number of operations in the model, it is practically applicable or not, and to what extent it is error-prone.

**Table 4** Comparison between CM and Previous Models

<sup>1</sup>CM as an abbreviation for CRISPR based Model.

Model	Higher solvable class of problems	Number of operations	Practically Applicable	Error-Prone
Restricted Model [10]	<i>BP</i>	3	Yes	Extremely
Unrestricted Model [10]	<i>NBP</i>	3	Yes	Extremely
Boneh et al. Model [11]	$P^{NP}$	6	Yes	Extremely
DNA-PASCAL model [12]	<i>P</i>	12	No	Extremely
DNA-EC model[13]	<i>P</i>	12	No	Extremely
Beaver Model [14]	<i>PSPACE</i>	5	Yes	Extremely
RDNA model [15]	<i>PSPACE</i>	6	Yes	Extremely
Ogihara and Ray model [16, 17]	<i>NC</i>	5	Yes	Extremely
Dantsin and Wolpert model [18]	<i>PSPACE</i>	8	Yes	Extremely
<b>Proposed Model</b>	<i>PSPACE</i>	<b>3</b>	<b>Yes</b>	<b>Minimal</b>

**Fig. 4** The proposed model is the minimum model in the field with PSPACE complexity.

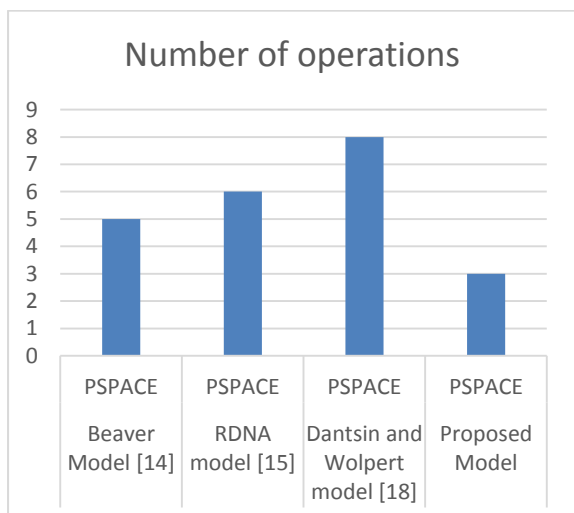


**Fig. 5** The proposed model complexity class versus other models with 3 operations.

In terms of the number of operations within the class, there are models in [10] that share the minimum number of operations with the proposed model but their complexity class of problems is too narrow and they also use error prone-operations [16]. Figure 5 clarifies that.

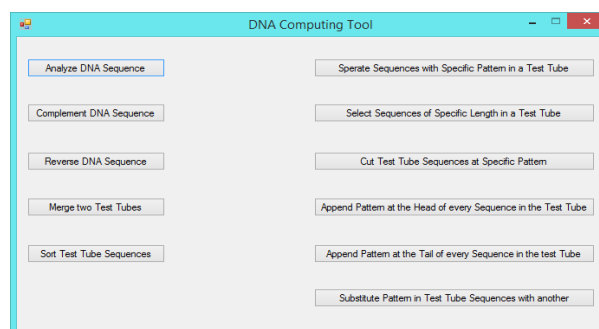
The proposed model has minimal error-prone because its operations are robust. Merge is robust [18], CRISPR is robust [9] and Detect is robust [18]. The proposed model is the minimum model in the field (just have three operations) with PSPACE complexity class of solvable problems.

In terms of the higher solvable class of problems, there are models [14, 15, 18] that share the same class with the proposed model but they use operations that are error-prone and that is an issue that faces DNA computing field in general [16]. Figure 4 clarifies that the proposed model is the minimum model in the field with PSPACE power complexity.



#### 4. Visual DNA

Visual DNA is a software used to implement operations of DNA computing discussed before in the introduction. Fig. 6 shows the main window of the software that enables the user to choose which DNA computing operation to use.



**Fig. 6** Main window for Visual DNA

When the user chooses to Analyze DNA sequence; a new window appears asking him for the sequence to be analyzed. The sequence analysis results show each nucleotide occurrence number and percentage in the sequence, the GC,



AT occurrence number, the melting temperature and the molecular weight of the sequence (Supplementary Fig. S1).

When the user chooses “Complement DNA Sequence” or “Reverse DNA Sequence”, a new window appears asking him to for the DNA sequence to be complemented or reversed consecutively. The result then is shown (Supplementary Figs S2, S3).

The other operations require the user to enter the contents of a test tube to make the operation on. The software takes the test tube as a text file where the new line in the text file means a new sequence in the test tube.

If the user chooses “Merge two Test Tubes” window; a new window appears asking for the contents of the two test tube contents then displaying the merge result of them (Supplementary Fig. S4).

If the user chose “Sort Test Tube Sequences”, a window appears asking for the test tube content then showing the shortest, longest and remaining strands in the test tube (Supplementary Fig. S5).

When the user chooses separate or select operation, a new window appears asking for the test tube contents and the pattern to separate sequences based on it or the length to select sequences based on it consecutively. The result then is shown (Supplementary Figs S6, S7).

When the user chooses cut, append head or append tail operation, a new window appears asking for the test tube contents and the pattern to cut sequences based on it, pattern to be added at the head of every sequence or at the tail of every sequence in the test tube consecutively. The result then is shown (Supplementary Figs S8-S10).

When the user chooses substitute operation, a window appears asking for the test tube contents and the pattern to be substitutes and the pattern to be substituted with then the result is displayed (Supplementary Fig. S11).

Visual DNA software is associated with test files for DNA sequences and test tube content examples. The software is implemented using C# programming language within the .net framework 4.5.

Visual DNA is Source code is freely available at (<https://github.com/AmrEledkawy/Visual-DNA>) under the GPL license.

## 5. Conclusion and Future Work

In this paper, a usage for CRISPR in DNA Computing have been proposed in the study to propose a DNA computational model that can solve NP problems in polynomial time and PSPACE problems in polynomial space. The proposed model is robust and argued to be the minimum DNA computational model. The model is used to solve, the NP problem, the Hamiltonian Path Problem (HPP) in linear time.

The study presented “Visual DNA” which is a software that can simulate biochemical operations in DNA computing. It can be used to make sure of the results of the biochemical operations before doing in vitro experiments. Also, the software can make analysis for DNA sequences. So this simulation software will offer a useful tool for the DNA computing implementation because it will help in the analysis of input DNA sequences and in the prediction of output DNA sequences which would be helpful to avoid errors during the experimental process.

The future work for the simulation tool can be seen in three ways. First is by extending the functionality of the software to apply new operations. Second is to apply the software for complete simulation of different DNA computing based algorithms. Finally, trying to make it a standalone full programming language for sake of independence.

## References

- [1] L.M. Adleman, Molecular computation of solutions to combinatorial problems., Science. 266 (1994) 1021–1024. doi:10.1126/science.7973651.
- [2] R.J. Lipton, DNA Solution of Hard Computational Problems, Science (80-. ). 268 (1995) 542–545. doi:10.1126/science.7725098.
- [3] R.S. Braich, Solution of a 20-Variable 3-SAT Problem on a DNA Computer, Science (80-. ). 296 (2002) 499–502. doi:10.1126/science.1069528.
- [4] Q. Ouyang, DNA Solution of the Maximal Clique Problem, Science (80-. ). 278 (1997) 446–449. doi:10.1126/science.278.5337.446.
- [5] R. Impagliazzo, R. Paturi, F. Zane, Which Problems Have Strongly Exponential Complexity?, J. Comput. Syst. Sci. 63 (2001) 512–530. doi:10.1006/jcss.2001.1774.
- [6] K. Sakamoto, D. Kiga, K. Komiya, H. Gouzu, S. Yokoyama, S. Ikeda, H. Sugiyama, M. Hagiya, State transitions by molecules, BioSystems. 52 (1999) 81–91. doi:10.1016/S0303-2647(99)00035-0.
- [7] Hartmanis J. The structural complexity column. Bulletin of the EATCS (1989) 37:117-26.
- [8] Kari L. Laboratory techniques with potential use for computation. [online] Csd.uwo.ca. Available at: <https://www.csd.uwo.ca/~lila/biooop.html>.
- [9] Cong L, Ran FA, Cox D, Lin S, Barretto R, Habib N, Hsu PD, Wu X, Jiang W, Marraffini LA, Zhang F. Multiplex genome engineering using CRISPR/Cas systems. Science (2013) 339(6121):819-23.
- [10] Winfree E. Complexity of restricted and unrestricted models of molecular computation. DNA Based Computers (1995) 1:187-98.
- [11] D. Boneh, C. Dunworth, R.J. Lipton, J. Sgall, On the computational power of DNA, Discret. Appl. Math. 71 (1996) 79–94. doi:10.1016/S0166-218X(96)00058-3.
- [12] Rooß D, Wagner KW. On the power of DNA-computing. Information and computation (1996) 131(2):95-109.
- [13] Yokomori T, Kobayashi S. DNA-EC: a model of DNA computing based on equality checking. DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science (1999) 48:347-60.
- [14] Beaver D. Computing with DNA. Journal of Computational Biology (1995) 2(1):1-7.
- [15] Reif JH. Parallel molecular computation. InProceedings of the seventh annual ACM symposium on Parallel algorithms and architectures, ACM (1995) pp 213-223.
- [16] Ogihara, Ray, The Minimum {DNA} Computation Model and Its Computational Power, in: Unconv. Model. Comput., 1998. citeseer.nj.nec.com/ogihara97minimum.html.
- [17] Ogihara M. Relating the minimum model for DNA computation and Boolean circuits. InProceedings of the 1st Annual Conference on

- Genetic and Evolutionary Computation, Morgan Kaufmann Publishers Inc. (1999) Volume 2 pp 1817-1821.
- [18] Dantsin E, Wolpert A. A robust DNA computation model that captures PSPACE. *International Journal of Foundations of Computer Science*. (2003) 14(05):933-51.
- [19] L. Adleman, Towards a mathematical theory of self-assembly, *Univ South. Calif. Tech Rep.* 00-722 (2000) 12. <http://lims.mech.northwestern.edu/students/bernheisel/repository/adleman00toward.pdf>.
- [20] Winfree E, Eng T, Rozenberg G. String tile models for DNA computing by self-assembly. In *International Workshop on DNA-Based Computers*, Springer Berlin Heidelberg. (2000) pp 63-8.
- [21] K. Sakamoto, Molecular Computation by DNA Hairpin Formation, *Science* (80-. ). 288 (2000) 1223-1226. doi:10.1126/science.288.5469.1223.
- [22] Păun G, Rozenberg G, Salomaa A. DNA computing: new computing paradigms. Springer Science & Business Media. (2005).
- [23] S. Roweis, E. Winfree, R. Burgoyne, N. V Chelyapov, M.F. Goodman, P.W. Rothmund, L.M. Adleman, A sticker-based model for DNA computation, *J. Comput. Biol.* 5 (1998) 615-29. doi:10.1089/cmb.1998.5.615.
- [24] Salomaa A. DNA complementarity and paradigms of computing. In *International Computing and Combinatorics Conference*, Springer Berlin Heidelberg. (2002) pp 3-17.
- [25] Păun G, Salomaa A. DNA computing based on the splicing operation. *Mathematica Japonica*.(1996) 43(3), 607--63.
- [26] E. Csuhanj-Varjú, V. Mitrana, Evolutionary systems: A language generating device inspired by evolving communities of cells, *Acta Inform.* 36 (2000) 913-926.
- [27] De Errico L, Jesshope C. Towards a new architecture for symbolic processing. In *Proceedings of the sixth international conference on Artificial intelligence and information-control systems of robots*, World Scientific Publishing Co, Inc. (1995) pp 31-40.
- [28] W.D. Hillis, The Connection Machine, *Sci. Am.* 256 (1987) 108-115.
- [29] M. Margenstern, V. Mitrana, M.J. Pérez-Jiménez, Accepting hybrid networks of evolutionary processors, *Lect. Notes Comput. Sci.* 3384 (2005) 235-246. doi:10.1007/b136914.
- [30] Martín-Vide C, Mitrana V, Pérez-Jiménez MJ, Sancho-Caparrini F. Hybrid networks of evolutionary processors. In *Genetic and Evolutionary Computation Conference*, Springer Berlin Heidelberg. (2003) pp 401-412.
- [31] F. Manea, C. Martín-Vide, V. Mitrana, A universal accepting hybrid network of evolutionary processors, in: *Electron. Notes Theor. Comput. Sci.*, 2006: pp. 95-105. doi:10.1016/j.entcs.2005.09.024.
- [32] F. Manea, M. Margenstern, V. Mitrana, M.J. Pérez-Jiménez, A new characterization of NP, P, and PSPACE with accepting hybrid networks of evolutionary processors, *Theory Comput. Syst.* 46 (2010) 174-192. doi:10.1007/s00224-008-9124-z.
- [33] Zhang L, Jia R, Palange NJ, Satheka AC, Togo J, An Y, Humphrey M, Ban L, Ji Y, Jin H, Feng X. Large genomic fragment deletions and insertions in mouse using CRISPR/Cas9. *PLoS One*. (2015) 10(3):e0120396.
- [34] Liu Y, Tao W, Wen S, Li Z, Yang A, Deng Z, Sun Y. In vitro CRISPR/Cas9 system for efficient targeted DNA editing. *Mbio*. (2015) 6(6):e01714-15.
- [35] Baskiyar, Sanjeev. "Simulating DNA computing." *International Conference on High-Performance Computing*. Springer, Berlin, Heidelberg, 2002.
- [36] Muhammad, M. S., et al. "A Simulation Software for DNA Computing Algorithms Implementation." *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering* 4.12 (2010): 1819-1826.
- [37] Phillips, Andrew, and Luca Cardelli. "A programming language for composable DNA circuits." *Journal of the Royal Society Interface* 6.Suppl 4 (2009): S419-S436.
- [38] Lakin, Matthew R., and Andrew Phillips. "Compiling DNA strand displacement reactions using a functional programming language." *International Symposium on Practical Aspects of Declarative Languages*. Springer, Cham, 2014.
- [39] Chen, Yuan-Jyue, et al. "Programmable chemical controllers made from DNA." *Nature nanotechnology* 8.10 (2013): 755.
- [40] Brijder, Robert, Joris JM Gillis, and Jan Van den Bussche. "The DNA query language DNAQL." *Proceedings of the 16th International Conference on Database Theory*. ACM, 2013.
- [41] Nielsen, Alec AK, et al. "Genetic circuit design automation." *Science* 352.6281 (2016): aac7341.
- [42] Bates, Maxwell, et al. "Genetic Constructor: An online DNA design platform." *ACS synthetic biology* 6.12 (2017): 2362-2365.
- [43] Theis, Thomas N., and H-S. Philip Wong. "The End of Moore's Law: A New Beginning for Information Technology." *Computing in Science & Engineering* 19.2 (2017): 41-50.
- [44] El-Seoud, Samir Abou, Reham Mohamed, and Samy Ghoneimy. "DNA Computing: Challenges and Application." *International Journal of Interactive Mobile Technologies* 11.2 (2017).
- [45] S. Soo-Yong, Z. Byoung-Tak, J. Sung-Soo, Solving traveling salesman problems using molecular programming, *Proc. 1999 Congr. Evol. Comput. (Cat. No. 99TH8406)*. 2 (1999) 994-1000. doi:10.1109/CEC.1999.782531.
- [46] Eppstein, D. The traveling salesman problem for cubic graphs. In *Proceedings of the 8th Workshop on Algorithms and Data Structures. Lecture Notes in Computer Science*, vol. 2748. Springer-Verlag, New York, (2003) 307-318.
- [47] Blazinkas, A. and Misevicius, A. Generating high quality candidate sets by tour merging for the traveling salesman problem. In *Tomas Skersys, Rimantas Butleris, and Rita Butkiene, editors, Information and Software Technologies: 18th International Conference Proceedings, ICIST 2012, Kaunas, Lithuania, September 13-14, 2012*, pages 62-73. Springer, Berlin, Heidelberg, 2012.
- [48] Taillard, Éric D., and Keld Helsgaun. "POPMUSIC for the travelling salesman problem." *European Journal of Operational Research* 272.2 (2019): 420-429.