Hybrid Genetic Approach to Oligo Sets Optimization

Piotr Wąsiewicz, Grzegorz Tomczuk, Jan J. Mulawka
Institute of Electronic Systems
Warsaw University of Technology
Nowowiejska 15/19, 00-665 Warsaw, Poland
{pwasiewi, gtomczuk}@elka.pw.edu.pl
{pwas,jml}@ise.pw.edu.pl

Abstract. DNA computing is a striking new information technology based on chemical reactions in tubes utilizing specially designed with a help of computer programs DNA polymers. This methodology provides new molecular mechanism for storing and processing information. DNA macrostructures are bases of specially designed algorithms realized by so called soft hardware applications. To obtain these structures a special DNA sequences design tool is required. A custom genetic algorithm with new hybrid operators was involved in creating a set of DNA strings. Changes in the input files and examples of string generation were introduced.

1 Introduction

In the new century a great progress in the area of nanoelectronics is expected. Traditional methodologies will soon meet their technological limits. Possibilities of silicon electronic chips further miniaturization are almost exhausted. Investigations indicate that different other chemical compounds can be used to store and process information on molecular scale. Especially organic substances used in biology are well suited for this purpose e.g. nucleic acids and proteins.

One of first steps in this direction was developed by cooperation of computer scientists and genetic engineers. As a result information technology based on computing utilizing molecules during chemical reactions has appeared. This new methodology is called DNA computing.

A single-strand DNA has a phospho-sugar backbone with two different, 5' and 3' ends and four bases Adenine, Thymine, Cytosine, Guanine denoted by the symbols A, T, C, and G, respectively. A double-strand DNA may be formed of oriented in the opposite directions two single strings due to hybridization or in other words annealing reaction, because A is complementary with T, and C is complementary with G. Due to this reaction the oligonucleotides may connect with each other during concatenation process called ligation forming longer DNA chains [17]. A sequence of such operations on DNA strings is called an algorithm. But in the typical DNA computing algorithm this sequence is determined by a model of DNA strings similar to the soft hardware specialized architecture driven here by heating, cooling and connected with them operations on DNA. Together the operation sequence and the model make computation possible.

First DNA computing method was invented by Adleman [9]. He demonstrated how to solve NP-complete combinatorial and graph problem of finding the Hamilton path that is very difficult to solve for conventional computers. His work began in this field further research, which was reported in many papers describing new DNA computing applications [1–15]. It has been demonstrated that DNA computing is suitable for programming in logic [10, 11] and for solving NP-complete problems [10]. For example DNA computing molecular inference systems [2] are a first step towards creation of the fifth computer hardware generation based on logic and the Prolog language structure. And multidimensional DNA computing leads in future to applications of molecular electronics in at least three dimensions. Emerging from it three dimensional molecular structures used in molecular computing algorithms enable creating special macromolecules conducting electrons. With such electronic molecules it will be possible to develop alternative architectures of extremely miniaturized computers. Thus, multidimensional DNA structures are bases for constructing alternative massively parallel computer architectures [4, 7, 8].

In order to have correct molecular computation results, the proper encoding, and model of DNA string set, and of course proper substrates such as enzymes are necessary to minimize the disadvantages of biological technology, especially extraction (5%), replication (0.001%), ligation and annealing errors [15]. Thus, it is necessary for large, complicated molecular systems to find at least optimal sequences of DNA strings in order to compute right solution without hybridization mismatches. To obtain proper encoding genetic algorithms were applied, what has been described in this paper.

2 The Genetic Algorithm Description

Genetic algorithms [18] belong to typical computation techniques that can be successfully applied to NP-hard optimization problems e.g. for optimizing functions with many local and one global optima. They consist of selections and different types of genetic operators that are repeated in cycles on population individuals.

In this problem the given string set with correct hybridizations as joints is like the function and the sequence of nucleotides is ascii coded. Each of these nucleotides is described by one ascii character.

When an existing hybridization is not included in the string set model, then it is an error and is called a mishybridization. Here a fitness of such the string set is proportional to a value of all string mishybridizations with itself and other strings. The task of the optimization was to find a DNA oligonucleotide string set, which hybridizes only in one correct way defined in the input file Scheme.txt this means with a fitness value equal to zero.

2.1 The Algorithm Structure

The hybrid genetic algorithm has been applied in the task of choosing DNA sequences for molecular computing and executed on a defined in a file Scheme.txt string set. The program consists of the following genetic algorithm steps: 1. input data reading and beginning population creating; 2. setting the algorithm specific parameters; 3. all mishybridization and fitness string set value evaluating; 4. the tournament selection; 5. reproduction: putting the best solutions in the place of the worst ones and into the special elite group of the best ones, classical crossover and mutation, hybrid crossover and mutation; 6. all mishybridization and fitness string set value evaluating; 7. if the STOP condition is not satisfied, go to the point 4; 8. writing of the best solutions to the output files. The STOP condition is usually connected with a number of generations this means loops from the point 4 to the point 7 of the mentioned algorithm.

2.2 Evaluating of Optimization Results

Evaluating of all two-string hybridizations consists in partial estimation of each hybridization case that can appear. In Fig. 1 it is shown how to process the analysis of string complementarity. One string shifts sliding under another one (in this case its copy) one nucleotide by one nucleotide and all matching base pairs with complementary bonds A with T or G with C are called hybridizations. Starting from the double fragment beginning progressive base pairs searching is marked with gray color. A base pair G≡C value equals 3, and a base pair A=T value equals 2. In the first case from Fig. 1 all hybridizations (two pairs G≡C and two pairs A=T) have a value equal to 10. In the second case their value is equal to 6 (two G≡C base pairs), and there is no hybridization in the third case.

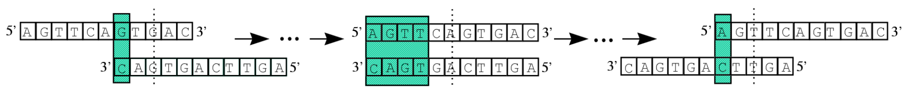

Figure 1: String complementarity analysis

The final value of all such hybridizations between two strings with indices i, j or self-hybridizations $(i = j)$ is denoted by:

$$e_{i,j} = \sum_{p=1}^{N} e_p \tag{1}$$

where: $N = L_{s_1} + L_{s_2} - 1$ - a number of hybridizations, L_{s_1} - a number of nucleotides in the first string, L_{s_2} - a number of nucleotides in the second string, e_p - a value of p^{th} hybridization.

It is worth mentioning that a relative error value, a relative error average value, a threshold overflow value are denoted by:

$$pe_{i,j} = \frac{e_{i,j}}{me_{i,j}}; \qquad xpe_{i,j} = \frac{\sum_{t=1}^{n} pe_{i,j,t}}{n}; \qquad ppe_{i,j} = \frac{pe_{i,j}}{empeix_k * xpe_{i,j}} \tag{2}$$

where: i, j - string indices, $pe_{i,j}$ - a relative value, $e_{i,j}$ - an absolute value, $me_{i,j}$ - a maximum value resulted from a sum of all possible errors, which can exist during annealing of the given strings, t - a test index, n - a number of all tests, $xpe_{i,j}$ - an average value, $pe_{i,j,t}$ - a relative value in the t^{th} test, $empeix_k$ - a threshold parameter with a string set index k, $ppe_{i,j}$ - a threshold overflow value.

In general a fitness string set value of the received solution consists of the all possible mishybridizations value sum. If this value is equal to zero, then all strings anneal to each other only in correct ways, and all energy constraint requirements are satisfied. Thus the fitness string set value emerges from analyses of hybridizations and selfhybridizations evaluating. Mishybridizations have a value equal to a sum of neighboring complementary base pair values e.g. |G≡C|= 3, |A=T|= 2. The length of such fragments is determined by a mishybridization minimal length parameter. All correct (described in the scheme file or with a length less than the minimal mishybridization parameter) hybridization values equals zero. With neighboring complementary base pairs particular double fragments have sum values multiplied by their weight parameters placed in a special input file. The more neighboring nucleotide pairs, the greater parameter value. Additionally, interval, not complementary base pair values are often less than zero and are added to the mentioned error sum. They depend on potential hybridization arrangements. After adding all such modified sum values e_p the final fitness value of string set is obtained.

2.3 Classical Operators

The tournament selection is an operation of choosing strings with better fitness string set value from the string set population in order to create the next loop better string population or perform some operations on them. First a defined tournament group of strings is chosen with uniform probability from the whole population. Second from this group one or two strings with the best fitness string set value are chosen.

In the program two reproduction operator groups were implemented: the first classical one including typical crossover and mutation operators, the second hybrid one with division operators. Apart from these groups, an operation, that eliminates single fragments with the same type of nucleotides, was added e.g. for its parameter equal to 3 an exemplary string: ATGGGGGGGGGCTTG will be changed to the following one: ATGGxGGxGGGCTTG where: x - is a nucleotide of type A, T or C.

The crossover operation exchanges randomly chosen fragments between two strings selected from a generated with uniform probability string list. In this case two-point crossover is utilized. Two points at the ends or between nucleotides are randomly chosen and parts between them are exchanged. The mutation operation changes randomly with uniform distribution a chosen nucleotide type to another one.

2.4 Hybrid Custom Operators

Division operations were invented in order to speed up a process of optimization. A fitness string set value emerges directly from interactions among set strings. The optimization task for strings is to connect them with each other with the greatest probability in the way described in the scheme file Scheme.txt. The more double fragments in the correct places or the longer they are, the greater is probability of the connection. Thus, division operations eliminate or divide with the use of typical mutation redundant mishybridizations and selfmishybridizations in order to increase the previously mentioned probability.

Division operation is usually executed on strings with the longest mishybridizations. Such a strategy quickly profits by efficient solution quality improvement. However, the string modification depends on a chosen division operation. Operator *mut1* changes one nucleotide in one part of one mishybridization, *multimut1* - one nucleotide in one part of each mishybridization, *mut3* - each nucleotide in one part of one mishybridization, *multimut3* - each nucleotide in one part of each mishybridization, *mut5* - each nucleotide change in every part of one mishybridization, *multimut5* - each nucleotide change in every part of each mishybridization. Operation *hcros1* exchanges one base pair in one hybridization, *multihcros1* - one base pair in each hybridization, *hcros3* - base pair half in one hybridization, *multihcros3* - base pair half in each hybridization, *hcros5* - each base pair in one hybridization, *multihcros5* - each base pair in each hybridization.

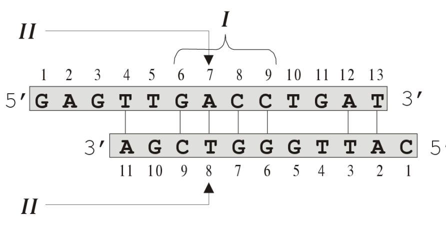

Figure 2: Base pair choosing in a longest mishybridization part selected for a division operator, I - a selected double part, II - chosen base pair nucleotides

Usually, a base pair, in which one chosen at random nucleotide is mutated, is selected by its position in the middle of the longest mishybridization part chosen by a division operation. The change of one nucleotide means that in the chosen from selected mishybridization base pair one of two nucleotides changes its kind e.g from A to G or C or T. The exchange means that base pair nucleotides exchange their places from one string to another and in the opposite direction e.g. in Fig. 2 in a base pair denoted by II a nucleotide A is moved to a nucleotide T place and T to an A place e.g. *hcross3* operates on a half of string nucleotides, *hcross5* on each nucleotide. Thus, on the contrary *hcross* operators are executed on the correct defined in the scheme file hybridizations or slightly possible mishybridizations.

Performing on each appropriate hybridization multi operators make more modifications, but it should be considered that only strings with $ppe > 1$ are corrected by the division operators. The *empeix* parameter is changed from time to time during program execution in order to assure that a number of string pairs is not greater than the twofold string number. Thus, $empeix_k$ is the k^{th} string set quality measure. For $empeix < 1$ its change step is equal to 0.01, and for $empeix \geq 1$ equal to 0.1.

3 The Input Scheme File Description

The methodology of creating input files was introduced in [1]. The string set scheme consists of two basic sections. Their description succession is obligatory during writing of the scheme input file. At the beginning of each section there is a header name in square brackets. Apart from these sections a section [POPULATION] may appear at the scheme file beginning. This section cancels computer generation of string sets starting population and allows its reading from the file Population.txt, so it permits searching of suboptimal solution from the given point in the search space.

- [SEQUENCES]
  In this section the lengths of the given strings with their names are written here. Each particular string is described in a separated new line. All strings from the set have to be described here. The string description scheme is the following:

  $$<\textit{string-name}\text{—}5'\text{-}3'> \sqcup <\textit{string-length}>,$$

  where a sign $\sqcup$ means a space e.g.: AC2 20

- [JOINTS]
  In this section the strings names with their indices should be put in order to describe their complementary fragments. The description scheme is the following:

  $<$*string-name*$>$ $\sqcup$ = $\sqcup$ $<$*string-complementarity*$>$ [$\sqcup$ + $\sqcup$ $<$*string-complementarity*$>$],

  where:

  $<$*string-complementarity*$>$: $<$*beginning-nucleotide-number*$>$ $\sqcup$ − $\sqcup$ $<$*ending-nucleotide-number*$>$ $\sqcup$ $<$*string-name*$>$ $\sqcup$ $<$*beginning-nucleotide-number*$>$

  e.g.: DC1 = 7 - 16 AC2 4 + 29 - 42 DC3 3

  Indices joint with a sign − are attached to the string from the left side of the sign =. The beginning index, which appears before the sign + or at the line end, belongs to the string put at its left side, but on the right side of the sign =. The string indices increase from the 5' end to the 3' end. The sign $\sqcup$ denotes a sign of space.

- [ASSIGNMENTS]
  The constant fragments of DNA sequences are introduced here.

  <string-name—5′-3′ > ⊔ *<beginning-nucleotide-number>* ⊔ = ⊔ *<nucleotide-letter>* [*<nucleotide-letter>*],

  where *<nucleotide-letter>*: A|G|T|C e.g. DC1 15 = CTGCAG means that the DC1 string has five nucleotides CTGCAG in the positions from 15 to 19.

- [ENERGY]
  In this section there are placed additional constraints connected with the capacity of the G≡ C base pairs in double strings or G, and C nucleotides in single strings. For double fragments the description scheme is the following:

  <string-name—5′-3′ > ⊔ *<beginning-nucleotide-number>* ⊔ *<string-name—5′-3′ >* ⊔ *<beginning-nucleotide-number>* ⊔ = ⊔ <G≡ C *capacity in* %>

  This described by the beginning nucleotide numbers hybridization has to be mentioned in the section [JOINTS], so the beginning nucleotides from this section and from the section [JOINTS] have to be the same. For single strings the description scheme is the following:

  <string-name—5′-3′ > ⊔ *<beginning-nucleotide-number>* ⊔ = ⊔ <G≡ C *capacity in* %>

Both last scheme sections require that there is no common part among defined double or single fragments. The practical use of some mentioned description schemes is described in the next points.

4 String Cell Examples

Introduced in [7] the automata model is preserved in a one-dimensional tape, in which every cell contains a symbol from a finite alphabet. The next computation step is to copy an input tape contents with a help of a rule table to a new output tape. All cells are copied in parallel.

This model takes advantage of this fact that DNA strings annealing to each other can construct not only double strings, but also miscellaneous two or three-dimensional shapes. Depicted in Fig. 3 rule exchanges symbols like a crossover operator: $f(A, B) = B$, $g(A, B) = A$, where A is denoted by AGTCA and B is denoted by GTTAC.

Every rule defining symbols on the new tape is coded by the identical two-dimensional construction made from optimized specially sequences, but with unique single strings called usually sticky ends this means they can hybridize with complementary ones and for example are described as output signals by symbols (A, B) and as input signals by symbols (B, A).

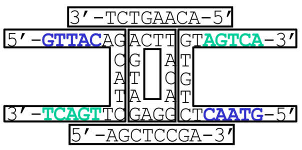

Figure 3: An exemplary automata rule $f(A, B) = B$, $g(A, B) = A$

In solution all rules connect with each other by their sticky ends forming multitape structure. Free rules attach with their complementary lower single string ends to formerly connected in a tape line other ones and denote with their upper sticky ends the next tape. A concatenation process called ligation follows hybridization. After melting characteristic long single strings are extracted and sequenced in order to read tape contents. Results are interesting by reason of utilizing only two operations on DNA: hybridization and ligation.

Such string cells sequences should be first optimized. The whole string set should be described in the input file. This operation looks like this: designer should put the chosen string set into the section [SEQUENCES], now each string has its length (a number of nucleotides); DNA strings from the section [SEQUENCES] correct complementary joints to the strings also from the section [SEQUENCES] are put into the section [JOINTS]; constant DNA fragments are added to the section [ASSIGNMENTS]. All errors in the file Scheme.txt e.g. overlapping double fragments or hybridizations going beyond defined single strings are detected and reported by the program during beginning input file reading.

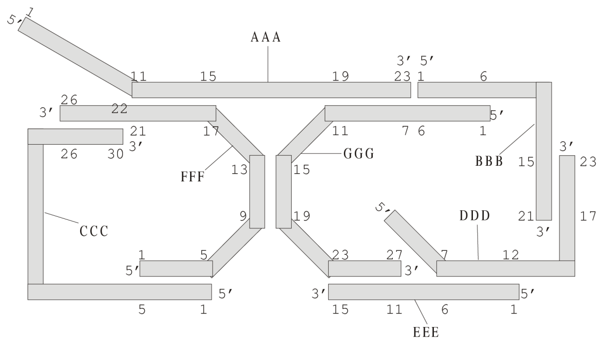

Figure 4: The exemplary DNA string set described in the file Scheme.txt from Example 1. There are illustrated strings with their names and lengths, numbers of nucleotides at the points of junctions and hybridizations.

4.1 Example 1

The exemplary DNA string set is illustrated in Fig. 4. The contents of the Scheme.txt input file are placed below. Its particular sections e.g. [JOINTS] were described at the previous point. The results of the string set from Fig. 4 sequence optimization are written as $5' - 3'$ DNA strings in the output file SchGroup.txt placed on the right side of the Scheme.txt.

Scheme.txt	SchGroup.txt
[SEQUENCES]	
AAA 23	
BBB 21	
CCC 30	
DDD 23	
EEE 15	
FFF 26	
GGG 27	TATTGGAGTCGAAGTACGTGAGAGTGC
[JOINTS]	
AAA = 11 - 15 FFF 17 + 19 - 23 GGG 7	AGGAAGAGAAACCCAACACGACT
BBB = 1 - 6 GGG 1 + 15 - 21 DDD 17	CCAATAAGGGGCAGCGTAGGT
CCC = 1 - 5 FFF 1 + 26 - 30 FFF 22	TTCAAACGACAGAAATGAATGGGAGCAGGC
DDD = 7 - 12 EEE 1	ACAGACATCGGAGTGGACCTACG
EEE = 11 - 15 GGG 23	TCCGATAAGCGCACT
FFF = 9 - 13 GGG 15	TTGAAAACACGTAAGGTGGGTGCCTG

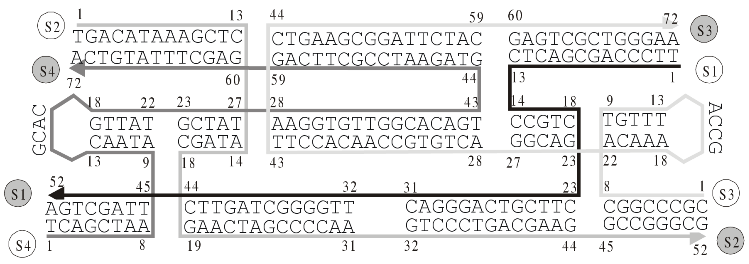

Figure 5: The exemplary DNA string set introduced in [8] and described in the file Scheme.txt from Example 2. There are illustrated strings with their names and lengths, numbers of nucleotides at the points of junctions and hybridizations.

4.2 Example 2

The exemplary DNA string set [8] is illustrated in Fig. 5. The contents of the Scheme.txt input file are placed nearby.

Scheme.txt contents

```
[SEQUENCES]
S1 52
S2 52
S3 72
S4 72
[JOINTS]
S1 = 1 - 13 S3 60 + 14 - 18 S3 23 + 19 - 44 S2 19 + 45 - 52 S4 1
S2 = 1 - 13 S4 60 + 14 - 18 S4 23 + 45 - 52 S3 1
S3 = 9 - 13 S3 18 + 28 - 43 S4 28 + 44 - 59 S4 44
S4 = 9 - 13 S4 18
[ASSIGNMENTS]
S3 14 = ACCG
S4 14 = GCAC
```

The results of the string set from Fig. 5 sequence optimization are written in the output file SchGroup.txt, which contents are following:

```
S1: TTCCCAGCGACTCCCGTCCTTCGTCAGGGACTTGGGGCTAGTTCTTAGCTGA
S2: TGACATAAAGCTCATAGCGAACTAGCCCCAAGTCCCTGACGAAGGCCGGGCG
S3: CGCCCGGCTTTGTACCGACAAAGACGGACTGTGCCAACACCTTCTGAAGCGG
    ATTCTACGAGTCGCTGGGAA
S4: TCAGCTAAATAACGCACGTTATGCTATAAGGTGTTGGCACAGTGTAGAATCC
    GCTTCAGGAGCTTTATGTCA
```

Table 1: Division operators test sets

The set index	Division operator sets for successive thresholds		
	empeix $\in \{0; 0.6\}$	*empeix* $\in \{0.6; 0.8\}$	*empeix* > 0.8
1	mut1(1)	mut1(1)	mut1(1)

2	mu1(0.9)
	multimut1(0.1)
		mut1(0.9)
		multimut1(0.1)
			mut1(0.9)
			multimut1(0.1)

3	mut1(1)	mut1(0.9)
		multimut1(0.1)
			mut1(0.75)
			mut3(0.2)
			hcross1(0.05)

4	mut1(0.6)
	mut3(0.2)
	hcross1(0.2)
		mut1(0.4)
		multimut1(0.2)
		mut3(0.2)
		multimut3(0.2)
			mut1(0.2)
			multimut1(0.2)
			mut3(0.2)
			multimut3(0.2)
			mut5(0.1)
			multimut5(0.1)

5 Optimization Results

The string set from the example 1 was used in optimization process of setting best operator parameter values. The genetic algorithm with a 30 individual population, a 2 individual elite group, a 3 individual tournament group and 50 computation generations was utilized. After exhaustive testing of different division operators together with different probability values from Tab. 1 the best configuration was found. It consists of division operators *mut* with probability 0.9 and *multimut* with 0.1 probability.

Table 2: Results of the first experiment after 8 independent tests with 50 generations

The operator set index	The string set evaluation number	The fitness value average	The best fitness value
1	1322	12.778791	6.00000
1	1574	14.744251	4.00000
2	812	25.429571	5.00000
2	916	26.365063	4.00000
3	617	21.176756	5.00000
3	1274	24.923122	9.00000
4	950	38.321251	7.00000
4	1471	36.652889	5.00000

Operators hcros failed to obtain better results (Tab. 2). Additionally, the less maximum mishybridization parameter, the worse results are achieved.

Table 3: The second experiment results

Crossover probability	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Division probability	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
experiment 1st	3.08	2.79	2.83	2.63	2.79	3.60	3.69	3.74	3.62	4.08
experiment 2nd	2.82	2.93	2.97	3.31	2.98	2.90	3.35	3.54	3.74	4.12
experiment 3rd	2.98	2.87	2.97	3.11	3.44	3.66	3.68	3.49	4.24	3.93
experiment 4th	2.98	2.84	3.66	2.93	3.12	3.49	3.54	3.16	3.30	4.47
average:	2.97	2.86	3.11	3.00	3.08	3.41	3.57	3.49	3.73	4.15

In the second experiment classical mutation had a constant probability value equal to 0.05, but division operators had a variable probability value in the scope from 0.1 to 1, and also a crossover in the scope from 0 to 0.9. The best probability values found for division and crossover operators were 0.9 and 0.1 as is seen in Tab. 3.

In the third experiment described in Tab. 4 division and crossover operators with constant probability values equal to 0.9 and 0.1 was attached to the classical mutation with a variable probability value in the scope from 0 to 0.95 in the optimization process of the mentioned string set from the example 1. Thus, all given above exemplary DNA string sets were optimized with the use of division, crossover, and mutation operators with probability values equal to 0.9, 0.1, 0.05.

After setting optimal operator parameters it was noticed that the solution quality very often depends on the mishybridization minimal length value. The less mentioned parameter value, the greater final solution string set fitness value. Thus, with the decreasing minimal mishybridization length value the result quality becomes worse what is seen in Tab. 5.

Table 4: Results of the third experiment

Mutation probability	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45
The best fitness value	4.01	2.98	3.23	3.41	4.18	4.24	4.36	4.70	4.37	4.79

Mutation probability	0.50	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95
The best fitness value	4.53	4.30	4.06	4.20	4.42	5.02	5.19	4.34	4.59	4.51

Table 5: Hybrid genetic computation results with 50 generations in each test

minimal mishybridization length	fitness evaluation number	fitness average	best fitness value
2	1578	123.492592	80.000000
2	1183	109.499428	77.000000
3	683	15.959213	5.000000
3	974	15.645994	5.000000
4	14	8.328825	1.000000
4	32	6.558132	1.000000

Table 6: Random method computation results with 70000 random solutions in each test

minimal mishybridization length	fitness evaluation number	fitness average	best fitness value
2	48209	223.726105	142.000000
2	3634	224.002625	147.000000
3	25172	59.703358	24.000000
3	29664	59.783474	23.000000
4	11403	15.737105	1.000000
4	2152	15.463757	1.000000

It should be noted that all mishybridizations from $e_{i,j}$ with different lengths have different weight parameters defined in a special input file named frag.txt. The greater mishybridization length, the greater weigth parameter value.

The solution dependence on the minimal mishybridization length was also analysed with a random method. During computation in one test 70000 string sets was generated at random and the one with the best fitness value was chosen. In comparison with the hybrid method random computation has in general worse results except these one for the minimal wrong hybridization length value equal to 4 as is provided in Tab. 6.

6 Conclusions

In this optimization approach hybrid genetic algorithms are utilized in optimization of string set nucleotide sequences. The mentioned program written in C++ is able to generate sequences of all possible string sets e.g. two dimensional sophisticated cellular automata or three dimensional structures suitable for more advanced DNA computing. With a quite new input data scheme very different string set models can be easily introduced with almost the same effort. In the paper criteria of evaluating string sets and constraints were represented. Input files, data flow, output data structures and configuration ways were mentioned. New hybrid genetic operators were described in detail.

After comparison with random methods and experiments with several string sets it should be mentioned that the program hybrid method is very efficient in optimizing and very often better or more universal than other approaches. Result quality dependency on utilized operators and different input parameters was analyzed. The performed tests prove, that for complex string sets with many single fragments or many double ones the best results are obtained only with use of new hybrid division operators. The new way of creating input string scheme files allows optimization of whatever string set that can be designed with as many as needed strings, double fragments and sticky ends.

Acknowledgments

This work was supported by the KBN Grant No 8T11F00816.

References

[1] J.J. Mulawka, P. Wąsiewicz, K. Piętak, Virus-enhanced genetic algorithms inspired by DNA computing, *LNAI - Subseries LNCS* **1609** (1999) 527–537.

[2] P. Wąsiewicz et al., The Inference Based on Molecular Computing, *Cybernetics and Systems: An Int. J.*, Taylor & Francis, vol. **31/3** (2000) 283-315.

[3] P. Wąsiewicz, J.J. Mulawka, Molecular Genetic Programming, *Soft Computing*, Springer, **5(2)** (2001).

[4] A. Dydyński, P. Wąsiewicz, Realization of Molecular Neural Networks, submitted to IEEE (2002).

[5] G. Păun et al., *DNA Computing - New Computing Paradigms*, Springer-Verlag Berlin (1998).

[6] *The Bibliography of Molecular Computation and Splicing Systems*, at http://-liinwww.ira.uka.de/bibliography/Misc/dna.html .

[7] E. Winfree, X. Yang, N.C. Seeman, Universal computation via self-assembly of DNA. *2nd Annual Meeting on DNA Based Computers*, Princeton, *DIMACS* (1996) 1052–1798.

[8] T.H. LaBean, E. Winfree, J.H. Reif, Experimental progress in computation by self-assembly of DNA tilings. *5th DIMACS Workshop on DNA Based Computers*, MIT, **54** USA (1999) 123–140.

[9] L.M Adleman, Molecular comput. of solutions to combinat. problems, *Science* **266** (1994) 1021–1024.

[10] R.J. Lipton, *DNA* Solution of Hard Computational Problems, *Science* **268** (1995) 542–545.

[11] M. Ogihara et al, *Simulating Boolean Circuits On a DNA Computer*, TR 631, Univ. Rochester (1996).

[12] J.J. Mulawka, P. Borsuk, P. Węgleński, Implementation of the Inference Engine Based on Molecular Computing Technique, *Proc. IEEE Int. Conf. Evol. Comp. (ICEC'98)*, USA (1998) 493–498.

[13] E.B. Baum, Building an associative memory vastly larger than the brain, *Science* **268** (1995) 583–585.

[14] R. Nowak et al, Molecular associative memory built on DNA, submitted to *Soft Comp.* (2002).

[15] S. Roweis et al, On the reduction of errors in DNA computation, *J. of Comp. Biology*, **6(1)** (1999) 65-75.

[16] J.R. Heath et al, A Defect-Tolerant Computer Architecture: Opportunities for Nanotechnology, *Science* **280** (1999) 1716-1721.

[17] J. Sambrook et al, *Molecular Cloning. A Laboratory Manual.*, Cold Spring Harbor Press (1989).

[18] D.E. Goldberg, *Genetic Algorithms in Search, Optim. and Machine Learning,* Addison-Wesley (1989).

[19] Mulawka J.J., *Expert Systems (in Polish)*, WNT, Warsaw (1996).