Comparison of Two Approaches to Oligo Sets Optimization

Grzegorz Tomczuk[a] and Piotr Wąsiewicz[a]

[a]Institute of Electronic Systems, Warsaw

ABSTRACT

DNA computing 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. In this paper comparison of two such computer programs was provided. In our program a custom genetic algorithm with new hybrid operators was involved in creating a set of DNA chains. The second program written by Winfree makes random changes using a given set of short constant forbidden fragments.

Keywords: DNA sequence optimization, genetic algorithms, molecular computing, DNA computer

1. INTRODUCTION

Cooperation of computer scientists and genetic engineers developed a new information technology based on computing utilizing molecules during chemical reactions. Investigations indicate that some 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. This new methodology is called DNA computing.

First DNA computing method was invented by Adleman [1]. 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 [1–12]. It has been demonstrated that DNA computing is suitable for programming in logic [9, 10] and for solving NP-complete problems [9]. For example DNA computing molecular inference systems [3] 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 [5, 7, 8].

Single DNA chains of nucleotides made of four bases A,T,G,C have two different 5' and 3' ends. Such two single chains oriented in the opposite directions can connect and create double chains during process of hybridization (A, C from one chain is complementary with T, G from the second chain, respectivly, creating complementary pairs). Double chains consist of complementary base pairs. Due to this reaction the oligonucleotides may connect with each other during concatenation process called ligation forming longer DNA chains [13].

A sequence of such operations on DNA chains is called an algorithm. But in the typical DNA computing algorithm this sequence is determined by a model of DNA chains 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.

Both programs named Skalar and DNA-Design (by Winfree) are designed to optimize DNA sequences of given scheme single and double chains, which together can join and create two or three dimensional nanostructures, which may be useful in creating future powerful molecular, massively parallel computers. Optimization task requires searching for unintended complementary base pairs not included in the given scheme so called mismatches and eliminate them by inserting other base pairs.

Further author information:
Institute of Electronic Systems, Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland;
(Send correspondence to G.T.) G.T.: E-mail: gtomczuk@elka.pw.edu.pl; P.W.: E-mail: pwasiewi@elka.pw.edu.pl

2. THE SKALAR STRUCTURE

2.1. The Genetic Algorithm Description

Genetic algorithms [14] 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 [2] the given chain 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 chain set model, then it is an error and is called a mishybridization. Here a fitness of such the chain set is proportional to a value of all chain mishybridizations with itself and other chains. The task of the optimization was to find a DNA oligonucleotide chain set, which hybridizes only in one correct way defined in the input file Scheme.txt this means with a fitness value equal to zero.

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 chain 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 chain 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 chain 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-chain hybridizations consists in partial estimation of each hybridization case that can appear. In Fig. 1 it is shown how to process the analysis of chain complementarity. One chain 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.

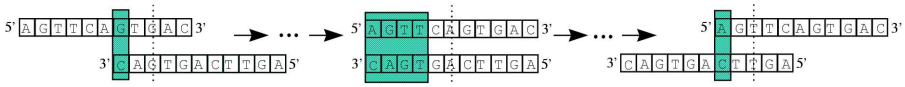

Figure 1. Chain complementarity analysis

The final value of all such hybridizations between two chains 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 chain, L_{s_2} - a number of nucleotides in the second chain, 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 - chain 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 chains, 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 chain set index k, $ppe_{i,j}$ - a threshold overflow value.

In general a fitness chain set value of the received solution consists of the all possible mishybridizations value sum. If this value is equal to zero, then all chains anneal to each other only in correct ways, and all energy constraint requirements are satisfied. Thus the fitness chain 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. $|\mathsf{G}{\equiv}\mathsf{C}|= 3$, $|\mathsf{A}{=}\mathsf{T}|= 2$. 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 chain set is obtained.

2.3. Classical Operators

The tournament selection is an operation of choosing chains with better fitness chain set value from the chain set population in order to create the next loop better chain population or perform some operations on them. First a defined tournament group of chains is chosen with uniform probability from the whole population. Second from this group one or two chains with the best fitness chain 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 chain: 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 chains selected from a generated with uniform probability chain 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 chain set value emerges directly from interactions among set chains. The optimization task for chains 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 chains with the longest mishybridizations. Such a strategy quickly profits by efficient solution quality improvement. However, the chain 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.

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

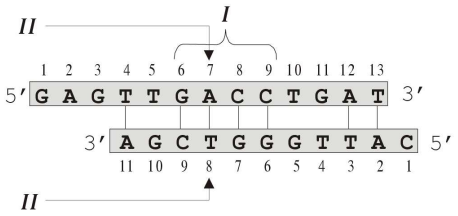

Figure 2. Base pair choosing in a longest mishybridization part selected for a division operator, I - a selected double part, II - chosen base pair nucleotides

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 chain 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 chain 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.

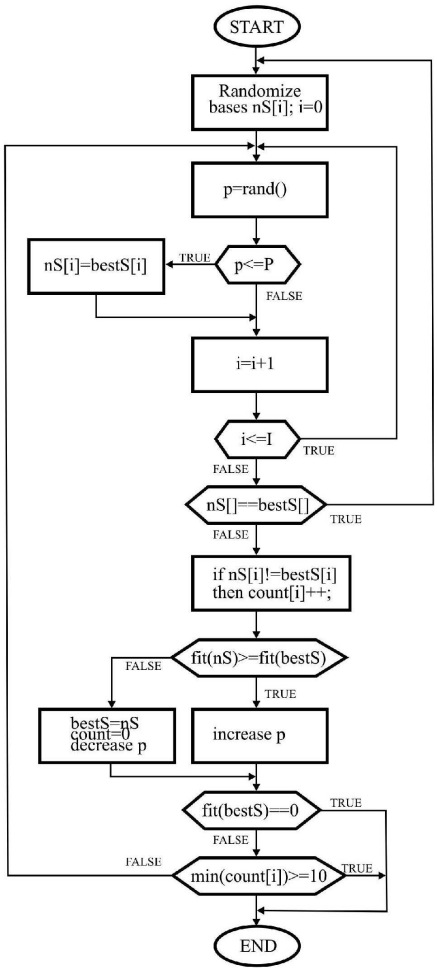

Figure 3. The block scheme of the DNA-Design algorithm.

Performing on each appropriate hybridization multi operators make more modifications, but it should be considered that only chains 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 chain pairs is not greater than the twofold chain number. Thus, $empeix_k$ is the k^{th} chain set quality measure. For *empeix*< 1 its change step is equal to 0.01, and for *empeix*≥ 1 equal to 0.1.

3. THE DNA-DESIGN STRUCTURE

In Winfree's DNA-Design a fitness function determines a number of unintended sequences, which are generated at the beginning. The program generates next solution by random changes of the best solution found in previous iterations. Algorithm execution will end when the solution with the fitness value equal to 0 is found or after a change level of the best solution exceeds a given value.

In DNA-Design the following algorithm was used as depicted in Fig. 3:

1. Set program parameters:
   - - $frac = 1/(2 * n)$ - nucleotide mutation probability,
   - - $decay = 0.95$ - nucleotide mutation probability change coefficient,
   - - $marked = zeros(1, n)$ - reset of probability change counters,
   - - $bestS$ - random set of variable with best solution.
2. Set a new solution by random - nS.
3. Choose with probability $1 - frac$ nucleotides, which are to be identical with $BestS$ nucleotides.
4. If $nS == bestS$ then $frac = frac/decay$, goto 2.
5. Increase nucleotide nS counters different from $bestS$ by one.
6. Evaluate a value of the fitness function for nS.
7. If nS is not better than $bestS$ then $frac = decay * frac$.
8. If nS is better than $bestS$ then $bestS = nS$, reset nucleotide change counters and $frac = frac/decay$.
9. If a value of fitness function for $bestS$ is equal to 0 or the least ferquently changed nucleotide was mutated ten times then STOP.
10. Goto 4.

The algorithm starts from random solution, creates in each iteration new solutions by random chosen nucleotide mutations with probability $frac$. The mutation probability increases if the new solution is not worse than the best one, otherwise it decreases.

4. OPTIMIZATION RESULTS

In Skalar the genetic algorithm tries to find optimal solution with an optimal value of fitness function, which in this case is proportional to the number of unintended complementary DNA sectors, which consist of unintended complementary base pairs creating together double fragments without any gap.

At the beginning exemplary scheme chain sequences are chosen by random. Eliminating of bad complementary joints is done by special operator, which replaces nucleotides of the longest unintended complementary fragments by such nucleotides which create the gap within complementary base pairs. Nucleotides A,T,G,C can be ordered in any sequence, but are not allowed to repeat too many times e.g. AAAA or GGG. Algorithm execution ends when the solution with the fitness value equal to 0 is found (no unintended double joints) or after a given iteration number.

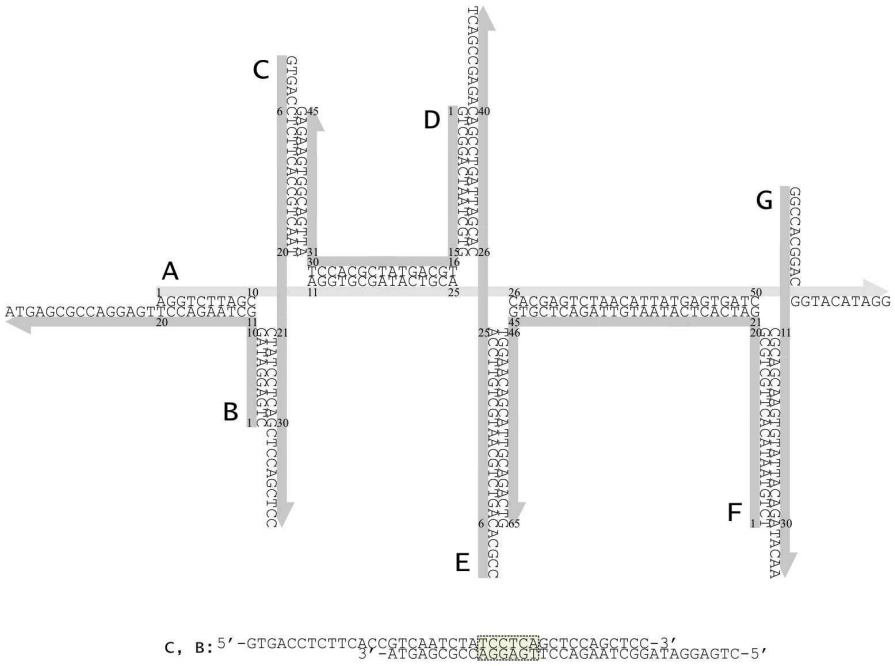

Figure 4. The optimized by Skalar first exemplary DNA chain set with the longest mishybridization depicted below.

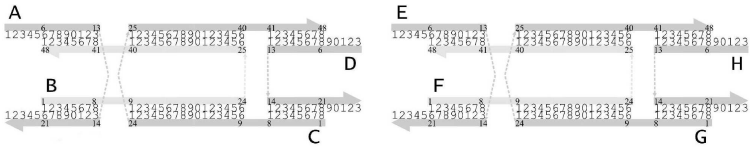

Figure 5. The second exemplary DNA chain.

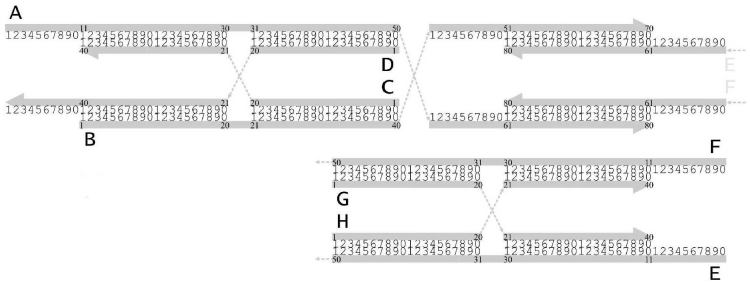

Figure 6. The third exemplary DNA chain.

During all tests Skalar parameters were the following: the number of population was equal to 30, mutation probability equals 0.05, hybrid operator probability equals 1 or 0.9, crossover probability - 0.15, the number of the best individual - 2, the number of individuals in the tournament selection was equal to 3 and two of them were always chosen.

4.1. The One Chain Set Scheme Comparison

The exemplary DNA chain set with optimized by Skalar DNA sequences is illustrated in Fig. 4.

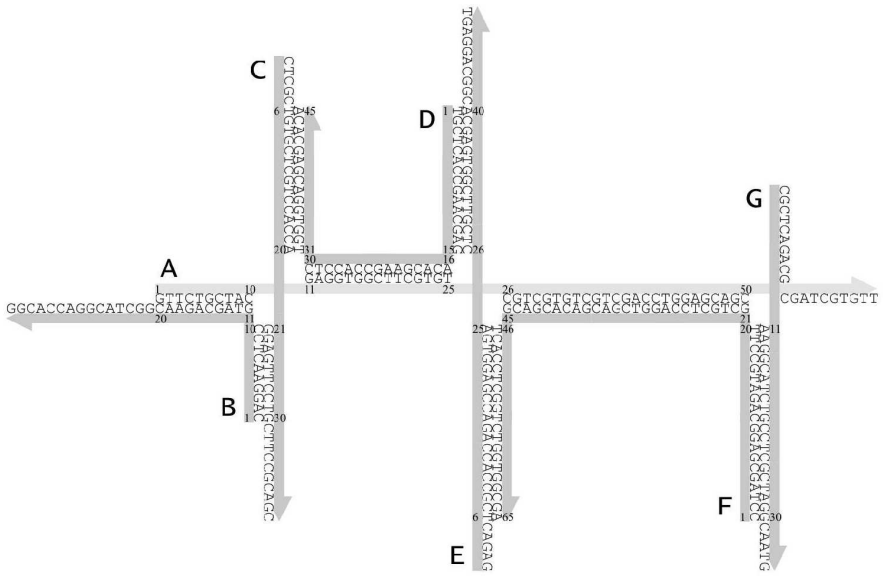

Figure 7. The optimized by DNA-Design first exemplary DNA chain set with the longest mishybridization depicted below.

Such chain sequences should be first optimized. The whole chain set should be described in the input file.

This operation looks like this in Skalar: designer should put the chosen chain set into the section [SEQUENCES], now each chain has its length (a number of nucleotides); DNA chains from the section [SEQUENCES] correct complementary joints to the chains 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 chains are detected and reported by the program during beginning input file reading.

The new way of creating input chain scheme files allows optimization of whatever chain set that can be designed with as many as needed chains, double fragments and sticky ends.

The contents of the Scheme.txt input file describing the exemplary set are placed below.

Scheme.txt contents

```
[SEQUENCES]
A 60
B 35
C 40
D 45
E 50
F 65
G 35
[JOINTS]
A = 1 - 10 B 11 + 11 - 25 D 16 + 26 - 50 F 21
B = 1 - 10 C 21
C = 6 - 20 D 31
D = 1 - 15 E 26
E = 6 - 25 F 46
F = 1 - 20 G 11
```

A structure of optimized set of chains is putting into DNA-Design in special format. It may by placed within a code of program or external file by putting several sequences of a double pairs indexes of a chain (an oligo), and length of intended match. It assures all joints of structure will be well defined. DNA-Design input file contents based on rules defined by Winfree in DNA-Design are the following:

DNA-Design input file contents

```
seq = ['NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN ' ...
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN ' ...
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN ' ...
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN ' ...
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN ' ...
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN ' ...
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN'];
[St, wc, eq] = constraints(seq, [1 1 2 20 10; 2 1 3 30 10; 3 6 4 45 15;
1 11 4 30 15; 4 1 5 40 15; 5 6 6 65 20; 1 26 6 45 25; 6 1 7 30 20], []);
```

Solutions obtained by Skalar are much better e.g. one 6-nucleotide mismatch is seen in Fig. 4 and there is no longer unintended complementary double fragments or with the same length ones, but in Fig. 7 more, even longer unintended double fragments are depicted. Winfree's program generates solutions containing sequences from nature or at least not from generated set of forbidden short single DNA sequences, so the solution space is a subset of the Skalar optimization space.

Table 1. The optimization results for both programs and all schemes

	Skalar	DNA-Design
Sch_1	1,33	34,33
Sch_2	7,33	98,66
Sch_3	0,33	37,33

4.2. The Two Chain Scheme Comparison

The mentioned first scheme and additional two schemes from Figs. 5 and 6 were used in comparison of program efficiency. The second one was obtained from the Winfree's program. The last one was created for more reliable testing.

The Skalar program reads schemes from the file called Scheme.txt, which contains chain names, lengths, and double joints. The Winfree's one has schemes written in the program code.

In order to ensure comparable results, a number of iterations in Skalar was denoted the following:

$li_S = li/licz_{pop}$

where: li_S - the number of iterations in Skalar, li - the number of iterations in DNA-Design, $licz_{pop}$ - genetic population size in Skalar.

Average values of the fitness function for the best solution from each scheme optimization (0 is the best solution) were given in Tab. 1.

5. CONCLUSIONS

After Skalar comparison with the well-known tool among DNA set designers called DNA-Design utilizing three chain sets it should be mentioned that the Skalar 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 the best results are obtained only with use of new hybrid division operators.

ACKNOWLEDGMENTS

This work was supported by the MolCoNet EU grant.

References

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

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

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

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

[5] P. Wąsiewicz, A. Dydyński, G. Tomczuk, J.J. Mulawka, A. Pluciennicza k, Molecular Neuron Realization, *WSEAS Transactions Journal on Biology and Biomedicine*, 1(1), 2004.

[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* pp.1052–1798, 1996.

[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 pp.123–140, 1999.

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

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

[11] 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 pp.493–498, 1998.

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

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

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