NACST/Seq: A Sequence Design System with Multiobjective Optimization

Dongmin Kim, Soo-Yong Shin, In-Hee Lee, and Byoung-Tak Zhang

Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
{dmkim,syshin,ihlee,btzhang}@bi.snu.ac.kr
http://bi.snu.ac.kr/

Abstract. The importance of DNA sequence design for reliable DNA computing is well recognized. In this paper, we describe a DNA sequence optimization system NACST/Seq that is based on a multiobjective genetic algorithm. It uses the concept of Pareto optimization to reflect many realistic characteristics of DNA sequences in real bio-chemical experiments flexibly. This feature allows to recommend multiple candidate sets as well as to generate the DNA sequences, which fit better to a specific DNA computing algorithm. We also describe DNA sequence analyzer that can examine and visualize the properties of given DNA sequences.

1 Introduction

Using the bio-molecules as basic computing or storage media, DNA computing wins the massive parallelism and some useful features such as the self-assembly. However, the chemical characteristics of materials involve some drawbacks in computing process.

Deaton and Garzon, for example, identified various types of errors those lead to false positives in Adleman's original techniques [1,2]. To overcome these drawbacks, they also gave a theoretical bound on the size of problems that can be solved reliably [3]. They introduced a new measure of hybridization likelihood based on Hamming distance and proposed a theory of error-preventing codes for DNA computing [4]. Since then, many researchers have proposed various algorithms and methods for the reliable DNA sequence design. These methods can be summarized into two different approaches.

One is the deterministic approach. Marathe et al. [5]. proposed a dynamic programming approach based on Hamming distance and free energy. Frutos et al. [6] proposed a template-map strategy to select a huge number of dissimilar sequences. Hartemink et al. [7] implemented the program "SCAN" to generate sequences for the programmed mutagenesis using an exhaustive search method. Feldkamp et al. also proposed another sequence construction system "DNASequenceGenerator" [8] using a directed graph.

M. Hagiya and A. Ohuchi (Eds.): DNA8, LNCS 2568, pp. 242–251, 2003.

A second approach to sequence design is to use evolutionary algorithms. Arita et al. [9] developed a DNA sequence design system using a genetic algorithm and a random generate-and-test algorithm. Tanaka et al. [10] listed up some useful sequence fitness criteria and then generated the sequence using simulated annealing technique with these criteria. Ruben et al. developed "PUNCH" [12] that employed a kind of genetic algorithm for the sequence optimization. And Deaton et al. proposed DNA based evolutionary search method [13].

The review above allows us to consider DNA sequence design as a numerical optimization problem given well-defined fitness measures. Based on the previous work [14,11], we formulate the DNA sequence design as a multiobjective optimization problem that is then solved by a genetic algorithm. This is implemented as a component (NACST/Seq) in the DNA computing simulation system, NACST (Nucleic Acid Computing Simulation Toolkit). NACST/Seq is especially useful in its ability to generate reliable codes and to provide the user with the flexibility of choosing optimal codes. The latter feature is attributed to the Pareto optimal set of candidate solutions produced by the multiobjective evolutionary algorithm. In addition, for the analysis and visualization of DNA sequence properties, we also developed NACST/Report, which is another component of NACST.

Section 2 describes the algorithm of NACST/Seq. Section 3 presents the software architecture of NACST. The sequence generation examples are shown in Section 4. And future work is discussed in Section 5.

2 Sequence Design by Multiobjective Evolutionary Optimization

As mentioned before, DNA sequence design can be considered as a numerical optimization problem. Moreover, it involves simultaneous optimization of multiple objectives. Most of the current sequence generation systems use the classical multiobjective optimization method (e.g. objective weighting) or single objective optimization method. But in many cases, there may not exist such a solution

Table 1. Objectives used by NACST/Seq.

Objective	Description
Similarity	similarity between two sequences
H-measure	degree of unexpected hybridization between two sequences
3'-end	"H-measure" in 3'-end of a sequence
GC Ratio	degree of difference with target G, C portion
Continuity	degree of successive occurrence of the same base
Hairpin	likelihood of forming secondary structure
Tm	degree of difference with target melting temperature

as the *best* with respect to all objectives involved. Therefore, it may be useful to recommend a set of solutions in which a solution is superior to the rest of all in one or more objectives by a multiobjective optimization algorithm. The fitness terms for sequence optimization are described in Table 1. These terms are originally summarized by Tanaka [10], we refine these fitness measures as more detailed numerical formulae for NACST/Seq [11]. With these seven objectives, the sequence optimization can be written formally as follows:

$$\begin{aligned}
&\Lambda = \{A, C, G, T\}, \Lambda^* \text{ denotes all possible sequences,}\\
&\mathcal{A}\ (\subset \Lambda^*) \text{ means the generated sequence pool,}\\
&\overline{x} \in \mathcal{A}, \quad f_i \in \{\, f_{GCratio}, f_{Tm}, f_{Hairpin}, f_{3'-end},\\
&\qquad\qquad f_{Continuity}, f_{H-measure}, f_{Similarity}\},\\
&\qquad \mathit{minimize}\ F(\overline{x}) = (f_1(\overline{x}), f_2(\overline{x}), \ldots, f_n(\overline{x})). \qquad (1)
\end{aligned}$$

NACST/Seq finds the sequence pool $\mathcal{A}$s, which satisfy that $F(\mathcal{A}), F(\mathcal{A}^{'})\ \exists i\ (i = 1, 2, \ldots, n)$ such that $f_i(\mathcal{A}) < f_i(\mathcal{A}^{'})$. The set of these $\mathcal{A}$s in each generation step is called the first Pareto front of that generation.

Fig. 1 shows a multiobjective genetic algorithm implemented in NACST/Seq.

The optimization algorithm is based on NSGA (nondominated sorting genetic algorithm) [15]. The original NSGA varies from simple genetic algorithms only in the selection operator, but we customized additionally the crossover and mutation operators to reflect the modified population architecture of NACST/Seq. In simple genetic algorithms, a population represents the feasible space in which each individual usually expressed through the bit string. But in NACST/Seq, each individual indicates a pool of sequences and it is indispensable to apply the evolutionary operators to each sequence for the evolution of whole population. Therefore, the crossover and mutation operators are divided into two steps - one is for individual level operation (step 1), the other belongs to sequence level (step 2). To improve the performance, we employ the elite strategy as shown in the last step in Fig. 1 and remove the sharing parameter used original nondominated sorting procedure by making the selection operator to work based on the rank of Pareto front. More detailed explanations can be found in [16].

3 NACST in Action

NACST consists of four independent components: NACST/Data, NACST/Seq, NACST/Report and NACST/Sim. NACST/Data allows the user to import and export the generated sequences and to edit functionality. NACST/Report shows the statistical figures of the sequences and plots graphs according to these data. NACST/Sim accomplishes DNA computing *in silico*.

In this paper, we focus on NACST/Seq with a brief description of NACST/Report. Before introducing NACST/Seq, we illustrate its design requirements. First, we require the fitness measures to be subdivided into fitness terms fine enough to consider the various characteristics of DNA sequences. Second, the

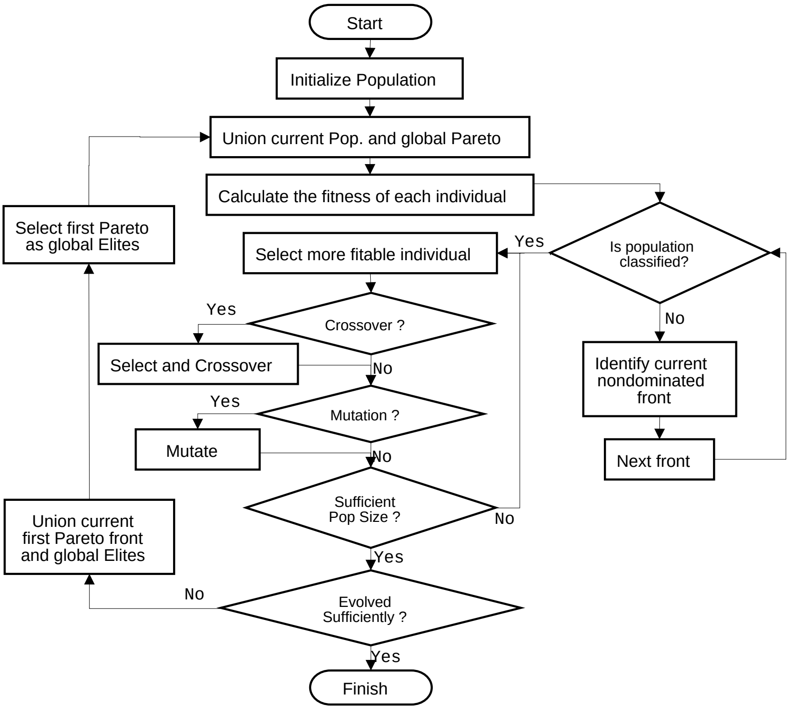

Start
Initialize Population
Union current Pop. and global Pareto
Calculate the fitness of each individual
Is population classified?
Yes
No
Select more fitable individual
Identify current nondominated front
Next front
Select first Pareto as global Elites
Crossover ?
Yes
Select and Crossover
No
Mutation ?
Yes
Mutate
No
Sufficient Pop Size ?
No
Yes
Union current first Pareto front and global Elites
Evolved Sufficiently ?
No
Yes
Finish

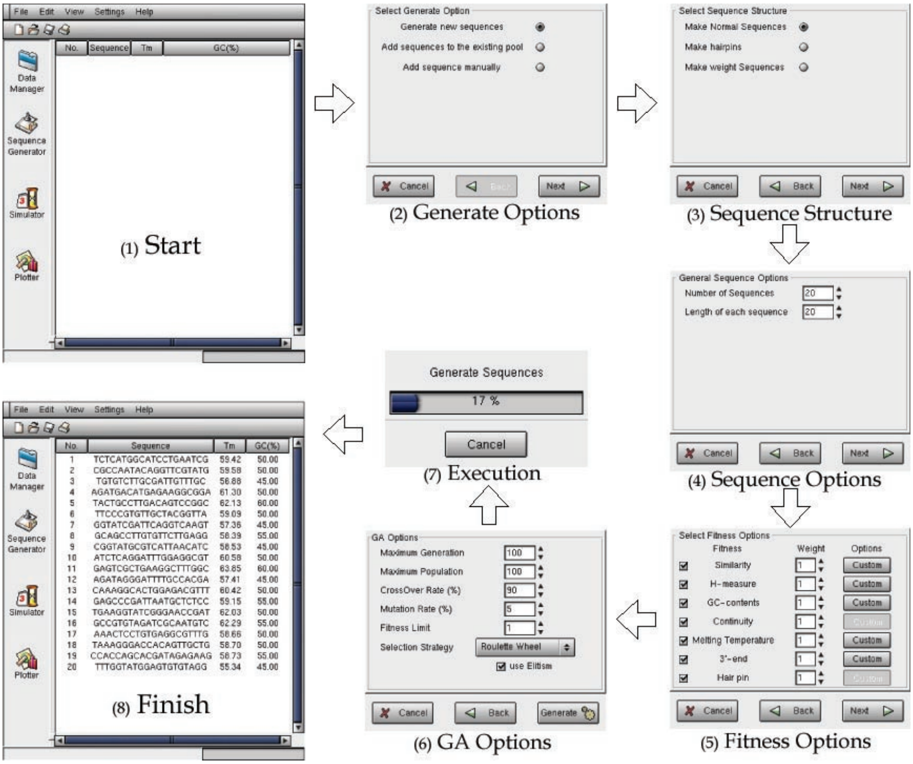
Fig. 1. Multiobjective genetic algorithm implemented in NACST/Seq.

sequence generation algorithm should be adaptable to various combinations of fitness. Third, users should be able to choose and combine objectives flexibly. Finally, the system should be able to show the generation result with the information that is sufficient to help users make decisions.

3.1 NACST/Seq

The sequence generation part of NACST, namely NACST/Seq, is implemented using C++ language in Linux platform and adopts a plug-in architecture that makes it possible to develop each fitness plug-in separately and to assure a future extension. In other words, we can add newly defined fitness plug-ins to NACST/Seq causing no alteration of pre-developed program texts and apply these plug-ins to the sequence generation process in run-time without the whole recompilation of the system.

The sequence generation steps are shown in Fig. 2. The first step is to select the generation option. In this step, the user can choose one option among "generate new sequences", "generate sequences and add those to an existing sequence pool" and "add a sequence to a pool manually". The first option means the generation of new sequences and new pools, the second option in this step



(2) Generate Options

Select Generate Option
- Generate new sequences ●
- Add sequences to the existing pool ○
- Add sequence manually ○

(3) Sequence Structure

Select Sequence Structure
- Make Normal Sequences ●
- Make hairpins ○
- Make weight Sequences ○

(4) Sequence Options

General Sequence Options
- Number of Sequences: 20
- Length of each sequence: 20

(5) Fitness Options

Select Fitness Options

	Fitness	Weight	Options
☑	Similarity	1	Custom
☑	H-measure	1	Custom
☑	GC-contents	1	Custom
☑	Continuity	1	Custom
☑	Melting Temperature	1	Custom
☑	3'-end	1	Custom
☑	Hair pin	1	Custom

(6) GA Options

GA Options
- Maximum Generation: 100
- Maximum Population: 100
- CrossOver Rate (%): 90
- Mutation Rate (%): 5
- Fitness Limit: 1
- Selection Strategy: Roulette Wheel
- ☑ use Elitism

Generate Sequences

17 %

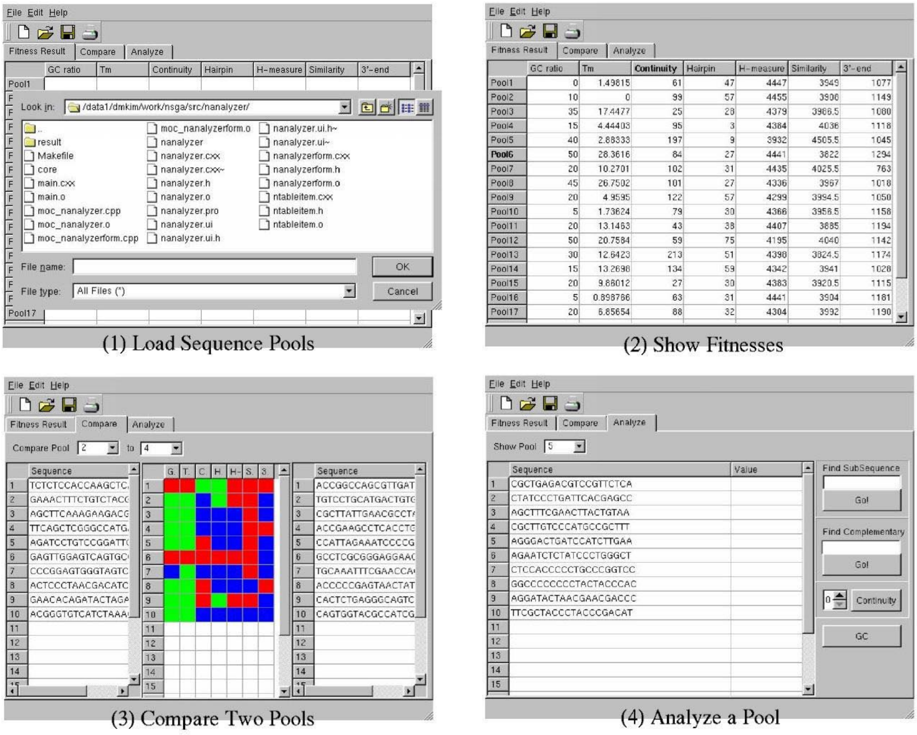
No.	Sequence	Tm	GC(%)
1	TCTCATGGCATCCTGAATCG	59.42	50.00
2	CGCCAATACAGGTTCGTATG	59.58	50.00
3	TGTGTCTTGCGATTGTTTGC	56.88	45.00
4	AGATGACATGAGAAGGCGGA	61.30	50.00
5	TACTGCCTTGACAGTCCGGC	62.13	60.00
6	TTCCCGTGTTGCTACGGTTA	59.09	50.00
7	GGTATCGATTCAGGTCAAGT	57.36	45.00
8	GCAGCCTTGTGTTCTTGAGG	58.39	55.00
9	CGGTATGCGTCATTAACATC	58.53	45.00
10	ATCTCAGGATTTGGAGGCGT	60.58	50.00
11	GAGTCGCTGAAGGCTTTGGC	63.85	60.00
12	AGATAGGGATTTTGCCACGA	57.41	45.00
13	CAAAGGCACTGGAGACGTTT	60.42	50.00
14	GAGCCCGATTAATGCTCTCC	59.15	55.00
15	TGAAGGTATCGGGAACCGAT	62.03	50.00
16	GCCGTGTAGATCGCAATGTC	62.29	55.00
17	AAACTCCTGTGAGGCGTTTG	58.66	50.00
18	TAAAGGGACCACAGTTGCTG	58.70	50.00
19	CCACCAGCACGATAGAGAAG	58.73	55.00
20	TTTGGTATGGAGTGTGTAGG	55.34	45.00

Fig. 2. Sequence generation process in NACST/Seq.

implies not a simple generation of sequence pools but a consideration of existing sequences in the generation of new sequences, and the last means that the user can add the existing sequence to a pool. The sequence structure, normal and hairpin, can be selected in the second option window. The option "normal" prevents the generation of self-complementary sequences, while the "hairpin" option acts vice versa. Because some DNA computing procedure needs to form the secondary structure intentionally [17]. Then, the general sequence option window appears. In this window, the number of sequences and the length of each sequence are adjusted. Next, the fitness option window provides the functionality of the combining and weighting the objectives. If the selected fitness needs additional arguments, the user can call up the sub-windows for tuning these arguments. For example, "Melting Temperature" needs the choice of the user between GC ratio and NN (nearest neighbor) methods. In addition, the oligo and Na+ concentration should be offered, if the user select the NN method. In this step, we can decide abundant properties of the generated sequences. Finally, the options for the genetic algorithm are determined. These include the generation number, the population size, the crossover and mutation rates. After execution of sequence generation, the main window shows the result sequences with their melting temperature and GC ratio.

3.2 NACST/Report

Another application of NACST is the analysis of sequence pools. Fig. 3 displays the analysis functions of NACST/Report.



Look in: /data1/dmkim/work/nsga/src/nanalyzer/

..
result
Makefile
core
main.cxx
main.o
moc_nanalyzer.cpp
moc_nanalyzer.o
moc_nanalyzerform.cpp
moc_nanalyzerform.o
nanalyzer
nanalyzer.cxx
nanalyzer.cxx~
nanalyzer.h
nanalyzer.o
nanalyzer.pro
nanalyzer.ui
nanalyzer.ui.h
nanalyzer.ui.h~
nanalyzer.ui~
nanalyzerform.cxx
nanalyzerform.h
nanalyzerform.o
ntableitem.cxx
ntableitem.h
ntableitem.o

File name:
File type: All Files (*)

	GC ratio	Tm	Continuity	Hairpin	H-measure	Similarity	3'-end
Pool1							
Pool17							

(1) Load Sequence Pools

	GC ratio	Tm	Continuity	Hairpin	H-measure	Similarity	3'-end
Pool1	0	1.49815	61	47	4447	3949	1077
Pool2	10	0	99	57	4455	3906	1149
Pool3	35	17.4477	25	28	4379	3966.5	1080
Pool4	15	4.44403	95	3	4384	4036	1118
Pool5	40	2.88333	197	9	3932	4505.5	1045
Pool6	50	28.3616	84	27	4441	3822	1294
Pool7	20	10.2701	102	31	4435	4025.5	763
Pool8	45	26.7502	101	27	4336	3967	1018
Pool9	20	4.9595	122	57	4299	3994.5	1050
Pool10	5	1.73624	79	30	4366	3956.5	1158
Pool11	20	13.1463	43	38	4407	3885	1194
Pool12	50	20.7584	59	75	4195	4040	1142
Pool13	30	12.6423	213	51	4398	3824.5	1174
Pool14	15	13.2698	134	59	4342	3941	1028
Pool15	20	9.88012	27	30	4383	3920.5	1115
Pool16	5	0.898766	63	31	4441	3904	1181
Pool17	20	6.85654	88	32	4304	3992	1190

(2) Show Fitnesses

Compare Pool 2 to 4

	Sequence
1	TCTCTCCACCAAGCTC
2	GAAACTTTCTGTCTACC
3	AGCTTCAAAGAAGACG
4	TTCAGCTCGGGCCATG
5	AGATCCTGTCCGGATT
6	GAGTTGGAGTCAGTGC
7	CCCGGAGTGGGTAGTC
8	ACTCCCTAACGACATC
9	GAACACAGATACTAGA
10	ACGGGTGTCATCTAAA
11	
12	
13	
14	

	G.	T.	C.	H.	H-	S.	3
1	🟥	🟥	🟩	🟩	🟥	🟥	🟥
2	🟩	🟩	🟦	🟩	🟥	🟥	🟦
3	🟩	🟩	🟦	🟦	🟦	🟥	🟦
4	🟩	🟩	🟦	🟦	🟦	🟥	🟥
5	🟩	🟩	🟥	🟦	🟦	🟥	🟦
6	🟥	🟥	🟥	🟥	🟥	🟥	🟦
7	🟦	🟩	🟦	🟦	🟦	🟥	🟦
8	🟩	🟩	🟥	🟦	🟦	🟦	🟥
9	🟩	🟩	🟥	🟩	🟥	🟥	🟦
10	🟩	🟩	🟦	🟦	🟦	🟦	🟦
11							
12							
13							
14							
15							

	Sequence
1	ACCGGCCAGCGTTGAT
2	TGTCCTGCATGACTGTG
3	CGCTTATTGAACGCCT
4	ACCGAAGCCTCACCTG
5	CCATTAGAAATCCCCG
6	GCCTCGCGGGAGGAA
7	TGCAAATTTCGAACCA
8	ACCCCCGAGTAACTAT
9	CACTCTGAGGGCAGTC
10	CAGTGGTACGCCATCG
11	
12	
13	
14	

(3) Compare Two Pools

Show Pool 5

	Sequence	Value
1	CGCTGAGACGTCCGTTCTCA	
2	CTATCCCTGATTCACGAGCC	
3	AGCTTTCGAACTTACTGTAA	
4	CGCTTGTCCCATGCCGCTTT	
5	AGGGACTGATCCATCTTGAA	
6	AGAATCTCTATCCCTGGGCT	
7	CTCCACCCCCTGCCCGGTCC	
8	GGCCCCCCCCTACTACCCAC	
9	AGGATACTAACGAACGACCC	
10	TTCGCTACCCTACCCGACAT	
11		
12		
13		
14		
15		

(4) Analyze a Pool

Fig. 3. Functions in NACST/Report.

After the generation process described in the previous section, the results are saved in a file. Then, NACST/Report loads this results (Fig. 3-1). In fact, NACST/Report can load any sequence pool saved in its format and analyzes various aspects of the loaded sequence pools (Figs. 3-2,-3,-4)). In window 2 (Fig. 3-2), one can examine all sequence pools by the comparison of those fitness value measured through each objective used in optimization procedure. Window 3 (Fig. 3-3) provides the graphical representation of the superiority of fitness value in each sequence between two selected pools. At last, one can investigate the properties of a pool in window 4 (Fig. 3-4). For instance, it can highlight the position of a specific sub-sequence in a pool, find all complementary subsequences of user's input sequence, and mark all successive occurrence of the same base running over the threshold. These features can be put into practice, for example, predicting and analyzing the sequence properties in PCR experiments.

4 Working Examples

To demonstrate the working of the algorithm, we investigate some examples. First, We scored sequences in [1] using NACST/Seq. As we expected, 'Good codes' get better (lower) scores. For more details about this test can be found in [11].

Table 2. Fitness of the codes in Deaton's paper.

Sequence ($5' \rightarrow 3'$)	H-measure	3'-end	Similarity	Continuity	Hairpin	GC%	Tm
Good codes							
CTTGTGACCGCTTCTGGGGA	82	21	34	16	3	60	62.83
CATTGGCGGCGCGTAGGCTT	50	49	15	0	3	65	71.06
ATAGAGTGGATAGTTCTGGG	19	14	41	0	6	45	54.90
GATGGTGCTTAGAGAAGTGG	34	9	20	0	0	50	57.50
TGTATCTCGTTTTAACATCC	46	3	5	16	11	35	50.59
GAAAAAGGACCAAAAGAGAG	19	2	2	41	0	40	54.89
TTGTAAGCCTACTGCGTGAC	-	-	-	0	6	60	58.87
Bad codes							
ATCAGCTGGATTCATCTGAA	179	95	177	0	9	40	55.94
ATCAACAGAAATCCGCGGAA	139	67	74	9	6	45	61.64
ATCAGCTGAGGTCTGGTGAG	102	10	87	0	12	55	59.78
GTCCGCTGTATTCTCGTGAT	48	11	32	0	0	50	59.24
TTCAACTGTTTTCAGCTGTG	51	14	38	16	10	40	52.69
TTCACCTTTATTGAGCCGAA	29	12	20	9	9	40	53.83
TTCAGCCGATTTGCGGAGAA	-	-	-	9	19	50	61.06

And then, we generate the vertex sequences for the traveling salesman problem (refer to [11]) using all objectives. The genetic algorithm parameters used are: the population size is 200, the generation number is 1000, the crossover rates for the steps 1 and 2 are 0.97, and the mutation rate is 0.3. Fig. 4 shows the evolution of fitness values. As generation goes on, the algorithm finds the more suitable individuals for each objective. Especially, for hairpin, melting temperature, and GC ratio objectives, the *optimal* individuals are found in early steps, i.e. each fitness is zero. But, with respect to the average value of all objectives, which is usually regarded as a measure in single objective optimization methods, NACST/Seq shows relatively weak optimizing power. As a cause of this phenomenon, we conjecture that there exist some conflicts between objectives. That is to say, since one objective has the trade-off with other objectives, an individual optimized for one objective lost the fitness of other objectives reducing the influence of optimization in the average value of objectives. To confirm this connection of objectives, we repeat the generation process with the selected objectives considered as conflicting ones.

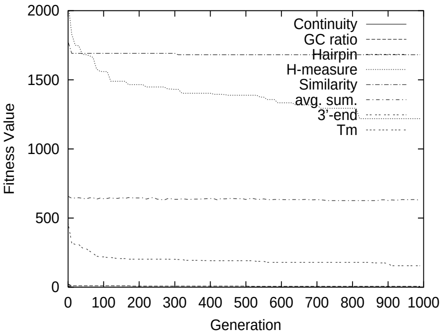


Fig. 4. Fitness values over generation.

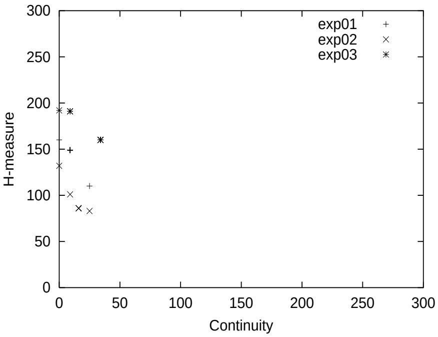


Fig. 5. Continuity vs. H-measure.

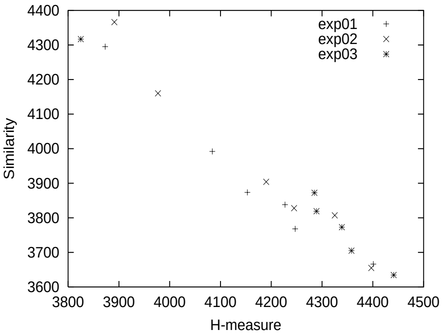


Fig. 6. H-measure vs. Similarity.

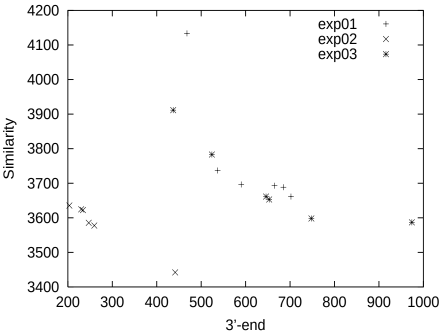


Fig. 7. 3'-end vs. Similarity.

Fig. 5 shows that there exist weak relations between continuity and H-measure, thus the algorithm works efficiently. While Fig. 6 and Fig. 7 depict other cases, the results confirm our expectation that there exist some conflicts between objectives. Because similarity has heavy inverse relations to H-measure and 3'-end, NACST/Seq explores multiple solutions between these two objectives instead of needless searching global one. This fact implies that there exist some difficulties (e.g. biased optimization) with classical approaches to the problem.

Table 3 shows the result of analysis in TSP vertex generation using window (2) in Fig. 3. NACST/Report can evaluate a sequence pool designed by other sequence generators or human experts. The values listed in the last row of Table 3 came from [11], we got those values with the single objective evolutionary algorithm using the sum of objectives. As shown in Table 3, NACST/Seq can find many alternative *better* sequence pools than that of the single objective optimization method.

Table 3. Fitness of the generated vertex sequences for TSP.

No.	GC%	Tm	Continuity	Hairpin	H-measure	Similarity	3'-end
NACST/Seq							
0	5	4.7827	86	32	2033	1982.5	291
1	35	0	45	24	2107	1903.5	255
2	35	11.6674	0	22	2081	1948.5	225
3	65	13.6196	61	0	2160	1852.5	282
4	55	24.5350	196	3	1223	2658.5	116
5	95	24.7548	131	22	2182	1683.0	304
6	70	26.2218	111	15	1435	2549.0	75
7	35	7.3607	43	3	1848	2054.0	204
8	30	0.4267	36	6	1957	1980.0	217
9	45	12.8784	18	7	1929	2021.5	182
Single-objective evolutionary algorithm with sum of fitness values							
0	35	4.6080	61	16	1383	2575.0	80

5 Discussion

In this paper, we described the evolutionary sequence generator called NACST/ Seq and NACST/Report, which were implemented as components of the DNA computing simulator, NACST (Nucleic Acid Computing Simulation Toolkit). We formulated sequence design as a multiobjective optimization problem and used a nondominated sorting procedure to generate the multiple candidate sequence pools. NACST/Seq can generate the promising DNA sequences and ensure that the user is able to choose more suitable sequences for the specific DNA experiments. NACST/Report provides the analysis and visualization of the sequence properties. This feature allows the user to investigate sequences *in silico* before real bio-chemical experiments.

The work in progress is to build a simulation system (NACST/Sim) of DNA computing that considers the thermodynamics of DNA sequences. We also plan to verify the practical usefulness of the sequences generated by NACST/Seq with real bio-chemical experiments. Additionally, objectives functions will be refined for physical model of DNA fidelity.

Acknowledgement

This research was supported in part by the Ministry of Education under BK21-IT Program and the Ministry of Commerce through Molecular Evolutionary Computing (MEC) Project. The RIACT at Seoul National University provides research facilities for this study.

References

1. R. Deaton, R. C. Murphy, M. Garzon, D. R. Franceschetti, and S. E. Stevens Jr., "Good encodings for DNA-based solutions to combinatorial problems," in *Proceedings of the Second Annual Meeting on DNA Based Computers*, 1996.

2. R. Deaton, R. C. Murphy, M. Garzon, D. R. Franceschetti, and S. E. Stevens Jr., "Genetic search of reliable encodings for DNA based computation," Late-Breaking papers at the First Genetic Programming Conference, pp. 9-15, 1996.
3. R. Deaton, M. Garzon, J. A. Rose, D. R. Franceschetti, R. C. Murphy, and S. E. Stevens Jr., "Reliability and efficiency of a DNA-based computation," *Physical Review Letters*, vol. 80, no. 2, pp. 417-420, 1998.
4. M. Garzon, P. Neathery, R. Deaton, R. C. Murphy, D. R. Franceschetti, and S. E. Stevens Jr., "A new metric for DNA computing," in *Proceedings of Genetic Programming 1997.*, The MIT Press. pp. 472-478, 1997.
5. A. Marathe, A. E. Condon, and R. M. Corn, "On combinatorial DNA word design," in *Proceedings of 5th DIMACS Workshop on DNA Based Computers*, pp. 75-89, 1999.
6. A. G. Frutos, A. J. Thiel, A. E. Condon, L. M. Smith, and R. M. Corn, "DNA computing at surfaces: 4 base mismatch word design," in *Proceedings of 3rd DIMACS Workshop on DNA Based Computers*, pp. 238, 1997.
7. A. J. Hartemink, D. K. Gifford, and J. Khodor, "Automated constraint-based nucleotide sequence selection for DNA computation," in *Proceedings of 4th DIMACS Workshop on DNA Based Computers*, pp. 227-235, 1998.
8. U. Feldkamp, S. Saghafi, and H. Rauhe, "DNASequenceGenerator - A program for the construction of DNA sequences," in *Preliminary Proceedings of 7th international Workshop on DNA-Based Computers*, pp. 179-188, 2001.
9. M. Arita, A. Nishikawa, M. Hagiya, K. Komiya, H. Gouzu, and K. Sakamoto, "Improving sequence design for DNA computing," in *Proceedings of Genetic and Evolutionary Computation Conference 2000*, pp. 875-882, 2000.
10. F. Tanaka, M. Nakatsugawa, M. Yamamoto, T. Shiba, and A. Ohuchi, "Developing support system for sequence design in DNA computing," in *Preliminary Proceedings of 7th international Workshop on DNA-Based Computers*, pp. 340-349, 2001.
11. S.-Y. Shin, D. Kim, I.-H. Lee, and B.-T. Zhang, "Evolutionary sequence generation for reliable DNA computing," *Congress on Evolutionary Computation 2002,*, 2002. (Accepted)
12. A. J. Ruben, S. J. Freeland, and L. Landweber, "PUNCH: an evolutionary algorithm for optimizing bit set selection," in *Preliminary Proceedings of 7th International Workshop on DNA-Based Computers*, pp. 260-270, 2001.
13. R. Deaton, R. C. Murphy, J. A. Rose, M. Garzon, D. R. Franceschetti, and S. E. Stevens Jr., "A DNA Based Implementation of an Evolutionary Search for Good Encodings for DNA Computation," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pp. 267-272, 1997.
14. B.-T. Zhang and S.-Y. Shin, "Code optimization for DNA computing of maximal cliques," in *Advances in Soft Computing - Engineering Design and Manufacturing.*, Springer., 1999.
15. N. Srinivas, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 3, no. 2, pp. 221-248, 1995.
16. S.-Y. Shin, D. Kim, I.-H. Lee, and B.-T. Zhang, "Multiobjective evolutionary algorithms to design error-preventing DNA sequences," *Parallel Problem Solving from Nature VII*, 2002. (Submitted)
17. M. Hagiya, M. Arita, D. Kiga, K. Sakamoto, and S. Yokoyama, "Towards parallel evaluation and learning of Boolean μ-formulas with molecules," *DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 48, pp. 57-72, 1999.