DNA Sequence Optimization Using Constrained Multi-Objective Evolutionary Algorithm

In-Hee Lee
Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University
Seoul, Korea
ihlee@bi.snu.ac.kr

Soo-Yong Shin
Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University
Seoul, Korea
syshin@bi.snu.ac.kr

Byoung-Tak Zhang
Biointelligence Laboratory
School of Computer Science and Engineering
Seoul National University
Seoul, Korea
btzhang@bi.snu.ac.kr

Abstract- Generating a set of the good DNA sequences needs to optimize multiple objectives and to satisfy several constraints. Therefore, it can be regarded as an instance of constrained multi-objective optimization problem. We apply the controlled elitist non-dominating sorting genetic algorithm with constrained tournament selection to this problem. First, multi-objective approach and constrained multi-objective approach are compared in terms of the effectiveness in finding feasible the solutions. Then the performance is evaluated by comparing with the good sequences published in literature.

1 Introduction

The problem of generating an independent DNA sequence set is very important for the efficient DNA-related experiments. Especially, it is one of the main issues of DNA microarray design and DNA-based computing. The independent DNA sequence set means a set of DNA sequences which have minimal tendency of cross-hybridization and maximal difference among them. In addition, they must have the similar physical conditions such as length and melting temperature. Thus, it can be regarded as a combinatorial optimization problem, in which evolutionary algorithm has shown good performance. The objectives to be optimized are the tendency to hybridize among the set, the difference of sequences among the set, and other additional factors.

There exist many researches which tried evolutionary approaches such as genetic algorithm and simulated annealing for sequence optimization. Deaton *et al.* used genetic search based on Hamming distance to design sequences for Hamiltonian path problem [Deaton98]. Arita *et al.* developed a DNA sequence design system using a genetic algorithm with three fitness criteria [Arita00]. Recently, Tanaka *et al.* listed up some sequence criteria and generated sequences with a simulated annealing technique [Tanaka01]. Reben *et al.* developed a system called "PUNCH" (Princeton University Nucleotide Computing Heuristic) to optimize DNA sequences using genetic algorithm with simple 2D matrix representation [Reben01]. Tuplan *et al.* used a simple stochastic hill climbing method to search DNA word [Tuplan02]. Kim *et al.* also applied a genetic algorithm [Shin02, Shin03].

These works considered sequence generation problem as a single objective optimization. However, as will be shown in the next section, sequence generation problem consists of multiple objectives to be optimized and constraints to be fulfilled. Therefore, unifying all objectives and constraints into a single objective would fail to provide useful and diverse solutions.

The authors already have tried multiobjective evolutionary approach, but penalties for violating constraints were also used as objective functions [Shin02, Shin03]. In this case, a solution which has good objective value but bad penalty, and one that has better penalty but worse function value, non-dominate each other. This implies that an infeasible solution can be non-dominated by any other feasible solution. Therefore, they can waste the slots for feasible solutions in Pareto front, and slow down convergence.

In this paper, we applied the controlled elitist non-dominated sorting genetic algorithm (NSGA-II) with constraint handling technique - constrained domination - to sequence generation problem [Deb01a, Deb01b]. NSGA-II can handle any number of objectives as well as reduce multiple objectives to a dummy fitness function using non-dominated sorting procedure. And NSGA shows the good performance in many artificial functions and applications [Deb01b]. And unlike when using weighted summation, it does not require additional parameters such as weights. Therefore,we choose NSGA-II to solve sequence generation problem.

The rest of this paper is organized as follows. In section 2, more formal definition of sequence generation problem and its objectives and constraints will be given. Then the detail of our implementation and the experimental result will appear in section 3 and 4, respectively. Finally, the conclusion is drawn in section 5.

2 Sequence Generation Problem

In a sequence generation problem, a designer must generate a set of sequences which shows good quality in various perspective and satisfies every constraints. Thus, it can be classified as a constrained multi-objective optimization problem. In this problem, the objectives to be optimized are the quality measures of a sequence set, and the constraints are the requirements for each sequence. From now on, we refer the problem of generating a set of n independent se-

quences with length l as the (n, l) – sequence generation problem. The formal description of (n, l) – sequence generation problem is as follows:

$$\begin{array}{ll} \text{Optimize} & f_i(\mathbf{X}), \quad i = 1, \cdots, M \\ \text{subject to} & g_j(\mathbf{X}) = 0, \quad j = 1, \cdots, N \end{array}$$

where $\mathbf{X} = (X_1, X_2, \cdots, X_n)$, and $X_k \in \{A, C, G, T\}^l$, for $k = 1, \cdots, n$. M and N denote the number of quality measures and constraints, respectively.

2.1 Objective Functions

In general, the quality of a sequence set can be measured from three perspectives. From the following three perspectives, we formulate four objective functions.

In the first perspective, the sequences in the set must be different from each other as much as possible to prevent the sequences from hybridizing with wrong sequence. We formulate a minimization measure named *similarity* based on the Hamming distance measure including position shift. The similarity value for two sequences is defined by maximum value of Hamming distance with various position shifts. And for a sequence set, it is defined by the sum of values for two different pairs. Formally, it can be written as follows:

$$\begin{aligned} f_{similarity}(\mathbf{X}) &= \Sigma_{i \neq j} similarity_T(X_i, X_j), \\ similarity_T(X, Y) &= max_{-l \leq k \leq l} S_T(sh(X, k), Y), \\ similarity_T(X|\mathbf{X}) &= \Sigma_{i, X_i \neq X} similarity_T(X, X_i), \end{aligned}$$

where, $X, Y \in \mathbf{X}$, and $sh(X, k)$ denotes the shift of X by $|k|$ right (if $k < 0$, left). $S_T(X, Y)$ denotes the Hamming distance longer than threshold T between two sequences X, Y. $similarity_T(X|\mathbf{X})$ means the objective value of a sequence $X \in \mathbf{X}$ and is defined as the summation over other sequences in the set.

The decision variable space for (n, l) – sequence generation problem is $(4^l)^n = 4^{ln}$, and it grows exponentially as n and l grow. We try to get a hint of the surface for similarity, but due to the size of decision variable space, we could plot similarity for $(2, 5)$–sequence generation problem only. Setting 'A' as 0, 'C' as 1, 'G' as 2, and 'T' as 3, a sequence can be thought as a number of base 4. A sequence can be represented by decimal number corresponding to this number. We draw $f_{similarity}(\mathbf{X})$ for every pair $\mathbf{X} = (X, Y) \in \{A, C, G, T\}^5 \times \{A, C, G, T\}^5$. The global optimal value for similarity is 0. As can be seen in Fig. 1, there exist lots of local optima.

From the second point of view, the sequences in a set should interact – hybridize – to each other as little as possible. Garzon *et al.* suggested a measure called *H-measure* for this objective [Garzon97]. H-measure for a pair of sequences calculates the maximum number of bases that can be hybridized between the sequences. Since the independent DNA sequence set should be designed, H-measure is a minimization objective. As in the case of similarity, H-measure for a set of sequences is defined by sum of values

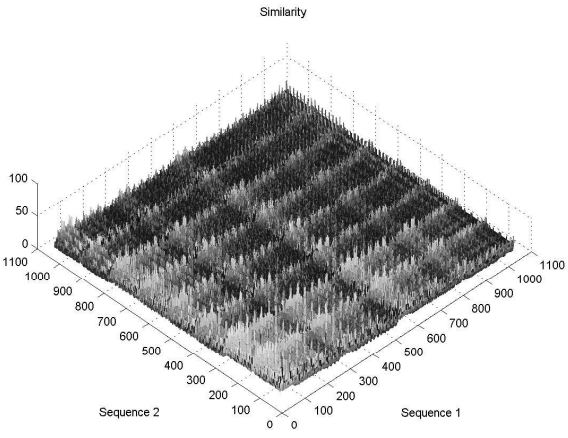


Figure 1: Fitness surface of similarity in case of $(2, 5)$ – sequence generation problem. Each sequence is represented by decimal number ('ATGC' as $0321_{(4)} = 57_{(10)}$).

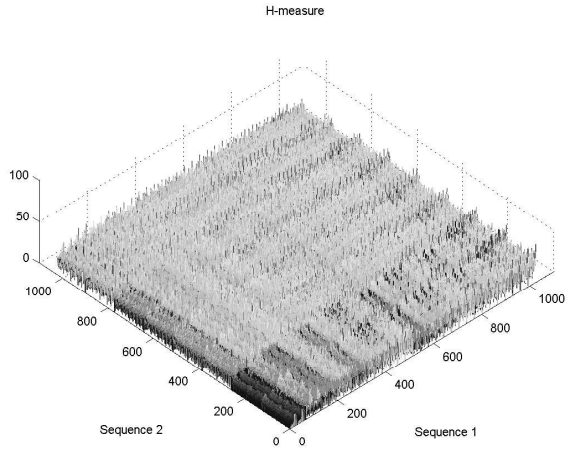


Figure 2: Fitness surface of H-measure in case of $(2, 5)$ – sequence generation problem. Each sequence is represented with a decimal number by the same way as in similarity.

for sequence pairs.

$$\begin{aligned} f_{H-measure}(\mathbf{X}) &= \Sigma_{i \geq j} hybrid_T(X_i, X_j), \\ hybrid_T(X, Y) &= max_{-l \leq k \leq l} H_T(sh(X, k), Y), \\ hybrid_T(X|\mathbf{X}) &= \Sigma_i hybrid_T(X, X_i), \end{aligned}$$

where, $H_T(X, Y)$ denotes the maximum number of base-pairs larger than threshold T that X and Y can produce. It should be minimized, also. And $hybrid_T(X|\mathbf{X})$ is defined in the same way as similarity. The plot of the fitness surface for H-measure in $(2, 5)$ – sequence generation problem is shown in Fig. 2. As in the case of similarity, $f_{H-measure}(\mathbf{X})$ is drawn for every pair $\mathbf{X} = (X, Y) \in \{A, C, G, T\}^5 \times \{A, C, G, T\}^5$. You can find many local optima.

The last perspective requires that the sequences should be the second structure-free. Since the unintended secondary structure lowers the efficiency of experiment, it should be avoided as much as possible. There are two

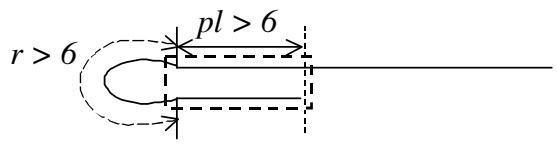


Figure 3: The hairpin structure.

known factors that cause the secondary structure. One is the continuous occurrence of the same base. And the other is the occurrence of the complementary substrings in a sequence. The former makes a strand to twist or bend, and the latter makes the strand to hybridize to itself. We define two minimization objective functions, *continuity* and *hairpin* for these factors respectively. Continuity is defined as follows,

$$
\begin{aligned}
f_{continuity}(\mathbf{X}) &= \Sigma_i \Sigma_{a\in\{A,C,G,T\}} con_{T,a}(X_i), \\
con_{T,a}(X) &= \Sigma_k^l (c_a(X,k))^2, \\
c_a(X,k) &= \begin{cases} o, & \text{if } \exists o > T, \text{ s.t. } x_{i-1} \neq a, \\ & x_{i+k} = a \text{ for } 1 \leq k \leq o, \\ & \text{and } x_{i+o+1} \neq a \\ 0, & \text{otherwise} \end{cases},
\end{aligned}
$$

and hairpin can be defined as below:

$$
f_{hairpin}(\mathbf{X}) = \Sigma_i hp(X_i),
$$

where, $hp(X)$ denotes the number of hairpin that X can form. Here, we assume that hairpin is formed when more than 6 bases separated by at least 6 base hybridize together as shown in Fig. 3. Therefore, this function is defined over the sequence whose length is longer than 18.

2.2 Constraints

For the reliable and efficient experiments, the decision maker requires each sequence to have similar physical and chemical properties. The most important properties are the number of bases 'G' and 'C' in the sequence and melting temperature. The melting temperature denotes the temperature where more than half of double strands start to break into single strands. These properties determine the experimental condition. To use a set of sequences in a experiment, the experimental conditions must match each other. Therefore, we take these properties as constraints for the (n,l)–sequence generation problem and define the penalty functions for each property.

$$
\begin{aligned}
g_{TM}(\mathbf{X}) &= \Sigma_i range(tm(X_i), Tm^L, Tm^U), \\
g_{GC}(\mathbf{X}) &= \Sigma_i range(gc(X_i), GC^L, GC^U), \\
range(t,l,u) &= \begin{cases} l-t & \text{if } t < l \\ t-u & \text{if } t > u \\ 0 & \text{otherwise} \end{cases},
\end{aligned}
$$

where, $tm(X)$ denotes the melting temperature of sequence X. Tm^U and Tm^L denote upper and lower limit for melting temperature, respectively. And $gc(X)$ denotes the number of 'G' and 'C' in X divided by l. GC^U and GC^L represent the upper and lower limit for $gc(X)$, respectively.

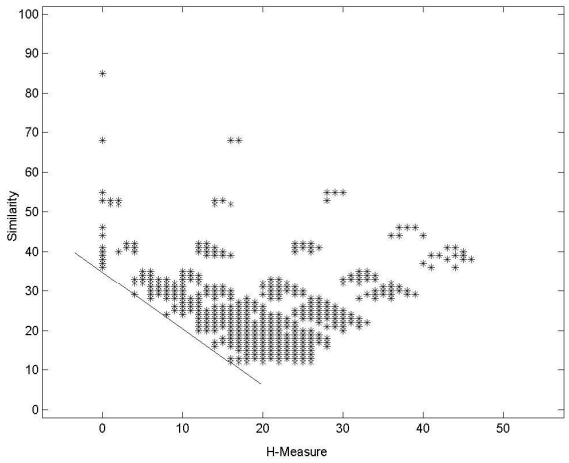


Figure 4: The relation between H-measure and similarity. The line shows trade-off surface.

2.3 Relationship Between Objectives

So far, we formulate the sequence generation problem as constrained multi-objective optimization problem with four objectives and two constraints. By the way, you can find a conflicting relation between similarity and H-measure. If a pair of sequences are very similar to each other, they will hardly hybridize to each other. Because, in most cases, a sequence can not be its complementary sequence at the same time. For the same reason, if a pair of sequences are complementary to each other, they will look very different in most cases. Therefore, it is difficult to minimize both H-measure and similarity. This conflicting relationship between H-measure and similarity is shown in Fig. 4 for $(2,5)$–sequence generation problem. Although it is not possible to achieve low value for both objectives, there exist several trade-off solutions even in a small problem as shown in Fig. 4. Using multi-objective evolutionary algorithm, we can find lots of such trade-off solutions simultaneously.

3 Sequence Generation Using Constrained Multi-objective Evolutionary Algorithm

As a first try of sequence generation using constrained multiobjective evolutionary algorithm, we used non-dominated sorting genetic algotithm with controlled elitism (NSGA-II) and constrained tournament selection. The details of our implementation will be explained in the following subsections.

3.1 Individual Chromosome

Each individual represents a set of sequences with given number and length. A DNA sequence consists of four bases. Therefore, each sequence can be represented by a number of base 4 or a quaternary string. And a set of sequences can be thought of as a set of these quaternary strings.

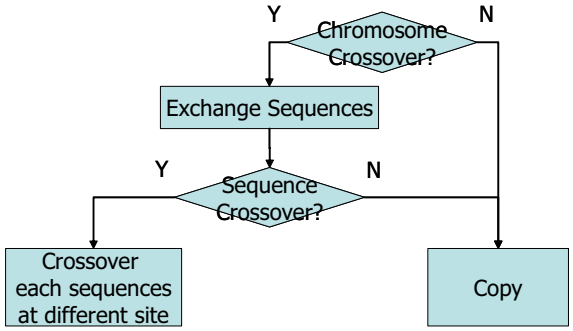


Figure 5: The two-stage crossover process.

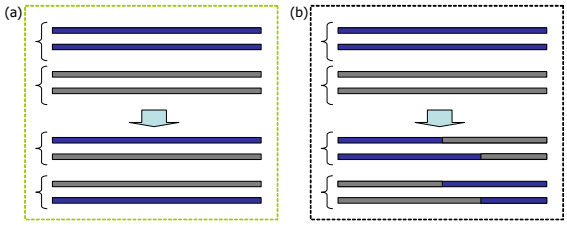


Figure 6: (a) The sequence set level (chromosome level) crossover. (b) The sequence level crossover.

3.2 Genetic Operators

The crossover operation of two chromosomes is composed of two stages. At the first stage, a sequence set level crossover occurs by exchanging sequences between two chromosomes. Then a sequence level crossover occurs. In this step, one-point crossover occurs to each sequence in two chromosomes. Fig. 5 and 6 show the two-stage crossover process. And the mutation changes each sequence by one base. As a selection operation, we use the constrained tournament selection. In constrained tournament selection, a solution that is feasible, has less penalty, or belongs to higher front becomes the winner. There are three cases in the constrained tournament [Deb01b].

- Infeasible – feasible: feasible solution wins.
- Infeasible – infeasible: one with less penalty wins.
- Feasible – feasible: if any one dominates the other, dominating one wins. If not, one with larger crowding distance wins.

We use the sum of penalties for each constraints as the penalty of a chromosome.

3.3 Constrained Multi-objective Evolutionary Algorithm

In [Shin03], controlled elitist non-dominated sorting genetic algorithm (NSGA-II) was used because it is one of the most suitable algorithm for the sequence generation problem. But the penalty functions were included as one of objective functions, and it may slow down the convergence to true Pareto optimal front. Since we treated a penalty function as one of objective functions, an infeasible solution may dominate feasible solution which has inferior objective values. Also, it is difficult to find a solution which is both feasible and non-dominated. Therefore, the infeasible solution may stay in the first front for a long time.

```
evolve(P)
    R = P ∪ Q
    evaluate(R)
    n = constrained-nondominated-sort(R)
    for each 1 ≤ k ≤ n
        if (all solution in front(R, k) is feasible)
            crowding-distance-assignment(front(R, k))
    P = controlled-parent-selection(R)
    Q = {}
    while |Q| < N
        p1 = constrained-tournament-selection(P)
        if (level1 crossover) then
            p2 = constrained-tournament-selection(P)
            if (level2 crossover) then
                Q = Q∪ {crossover2(p1,p2)}
            else
                Q = Q∪ {crossover1(p1,p2)}
            if (mutation) then mutate(Qp1, Qp2)
        else
            Q = Q ∪ {p1}
            if (mutation) then mutate(Qp1)
```

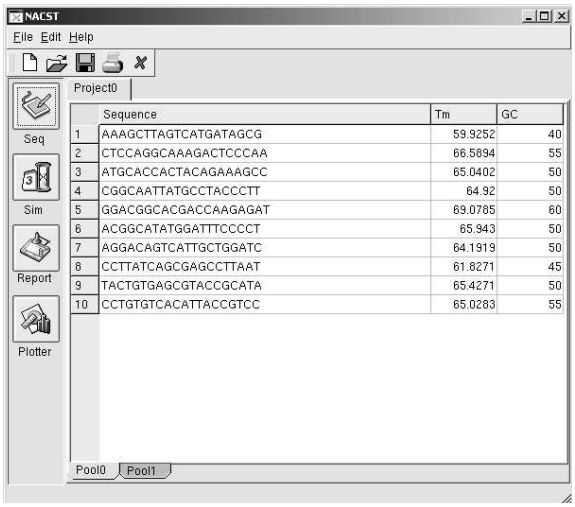
Figure 7: Main procedure for sequence generation.

To overcome this difficulty, a penalty function must be treated separately from other objective functions. Therefore, we include constrained tournament selection to handle the penalty functions. As explained in the previous subsection, constrained tournament selection favors the feasible solution over infeasible one and non-dominated one over the dominated. Therefore, the first front is always filled with solutions with least penalty or most non-dominated ones. And infeasible solutions are drived to feasible region.

The main procedure of sequence generation using controlled elitist NSGA-II with constrained tournament selection is shown in Fig. 7.

First, parent and offspring population are united and evaluated. In procedure constrained-nondominated-sort, the feasible solutions are always located in the better front than infeasible ones. Among the feasible solutions, the non-dominated ones belong to better front than those dominated. Among the infeasible solutions, those with the smaller penalty are located in the better front than those with the larger penalty. And among the feasible solutions, crowding distance is calculated. Then new parent population is formed by constrained tournament selection. From these new parents, new offsprings are generated by genetic operators such as two-stage crossover and mutation.

This algorithm is implemented to improve the sequence generating system 'NACST/Seq' (Nucleic Acid Computing Simulation Toolkit) [Shin03, Zhang98]. NACST/Seq is implemented in C++ using the Qt interface library on Linux platform. Fig. 8 shows the screenshot of NACST/Seq.



	Sequence	Tm	GC
1	AAAGCTTAGTCATGATAGCG	59.9252	40
2	CTCCAGGCAAAGACTCCCAA	66.5894	55
3	ATGCACCACTACAGAAAGCC	65.0402	50
4	CGGCAATTATGCCTACCCTT	64.92	50
5	GGACGGCACGACCAAGAGAT	69.0765	60
6	ACGGCATATGGATTTCCCCT	65.943	50
7	AGGACAGTCATTGCTGGATC	64.1919	50
8	CCTTATCAGCGAGCCTTAAT	61.8271	45
9	TACTGTGAGCGTACCGCATA	65.4271	50
10	CCTGTGTCACATTACCGTCC	65.0283	55

Figure 8: The screenshot of NACST/Seq and a result of example run.

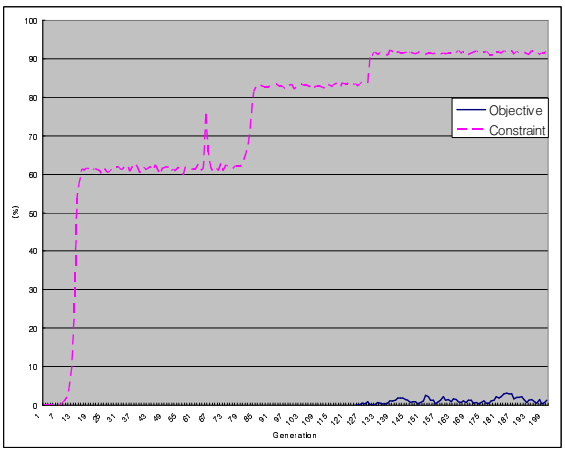


Figure 9: The ratio of feasible solutions over population during generations. The dashed line is the result when the constrained tournament selection is used. The other line is when the penalty is treated as an additional objective.

4 Experimental Result

4.1 Comparison on Constraint Handling Method

To demonstrate the effectiveness of constraint handling method in finding feasible solutions, we plot the ratio of the feasible solutions over population using two constraint handling methods. One method is not to treat the penalty for constraint violation as one of objectives and to handle it separately in selection. The other method is to think the penalty as one of objectives to be minimized and to handle it as additional objectives.

Fig. 9 shows the result in the $(7, 20)$–sequence generation problem. As objectives, H-measure, similarity, continuity, and hairpin are used, and the constraint is $g_{GC}(\mathbf{X})$ with $GC^L = GC^U = 50\%$. Except the constraint handling method, the same set of parameters are used: population size is 1000, maximum generation number is 200, crossover probability is 0.9, mutation probability is 0.01, and reduction rate for controlled elitist NSGA-II is 0.5. As you can see in Fig. 9, the constrained tournament selection finds feasible solution quickly and keeps them well. On the other hand, when the penalty is treated as the one of the objectives, infeasible solution is hardly removed from population, if it has good objective values. As a result, it becomes hard to insert a new feasible solution. As can be seen in Fig. 9, the feasible solutions could take up no more than 6% of population.

The proportion of each front in population is shown in Fig. 10 and 11. The lowest area denotes the portion of first front in the population. In both case, the first front take up 60 to 80% of the population. In Fig. 10, most of the first front member is feasible solution when compared with Fig. 9. On the other hand, in Fig. 11, most of the first front member is infeasible solution. And there is only one kind of feasible solutions in the first front. Therefore, it can be said that the constraint handling using constrained tournament selection is more effective.

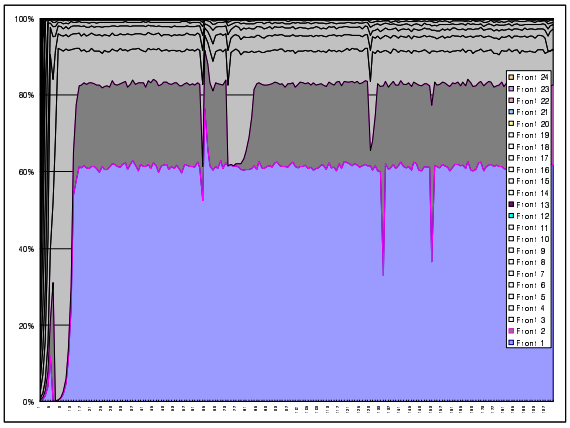


Figure 10: The proportion of each front in the population over generation when the constrained tournament selection is used.

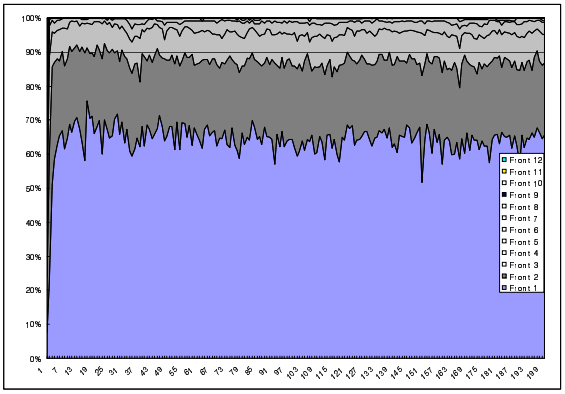


Figure 11: The proportion of each front in the population over generation when the penalty function is used as an objective.

Table 1: The fitness table for sequence sets generated by using two constraint handling methods. 'Tm' means the melting temperature of each sequence.

Sequence ($5' \rightarrow 3'$)	Continuity	Hairpin	H-measure	Similarity	Tm	GC%
Constrained tournament selection						
TCGCAACACGCAAAATAGGC	16	0	62	53	66.7797	50
AAGGGATCATCGGAGACAGA	9	0	60	58	64.56	50
ATTGGTTGCGGATGCTCGTA	0	0	62	59	66.9694	50
CGGGTTAATGTGCACGGATA	9	0	62	57	64.8338	50
GACTAAACCGACCTTGCACA	9	3	65	50	64.9101	50
AGTAATCGATGCCGGTTGTG	0	6	66	59	65.1012	50
TTGAAACGGGTCGTGTGTCA	18	5	63	60	66.9341	50
Total Fitness	61	14	220	198		
Penalty as additional objective						
ACCGTATCTCGGACTTCAGA	0	9	68	61	64.4817	50
TTGGTAGCTAGGGCACGTTT	18	0	63	54	66.2673	50
TCATAGCTCCAACTTCTCCC	9	0	58	70	63.2604	50
AGCCTGTCAGCTCTAATCTC	0	3	61	60	63.3041	50
TTGGTGTGCGGTCAAACGTA	9	10	64	53	67.2754	50
TCTCAGGGTCTTAGACCCTA	18	3	81	61	62.8872	50
CACTTCCAACTCGATCCCTA	9	0	57	63	63.2319	50
Total Fitness	63	25	226	211		

The generated sequences are summarized in Table 1. The upper table shows one of feasible solutions generated by using constrained tournament selection. And the lower table shows the only feasible solution found when $g_{GC}(\mathbf{X})$ is used as objective. The sequence set generated by constrained tournament selection dominates the other.

4.2 Comparison with Sequences in [Deaton96]

To show the effectiveness of the use of constrained multi-objective evolutionary algorithm, we compare the sequences published in other paper with those generated by the proposed algorithm.

Deaton *et al.* used a genetic algorithm to design good code words for Adleman's experiment [Deaton96]. This is an example of $(7, 20)$–sequence generation problem. The Hamming bound was used to be an upper bound on the number of good encodings. To compare the results, the fitness values of the sequences in [Deaton96] is re-calculated using NACST/Seq. In this problem, continuity, hairpin, H-measure, and similarity are used as objective function, and $g_{GC}(\mathbf{X})$ as constraint with $GC^L = 35\%$ and $GC^U = 65\%$. The sequences are compared in Table 2. In Table 2, the total fitness means the fitness value of sequence set. And the fitness values for each sequences are given in a row. The parameter setting of NACST/Seq is the same as in the previous subsection.

As can be seen in Table 2, the sequences generated using NSGA-II with constrained tournament selection dominates the other sequence set.

5 Conclusions

We implement the controlled elitist non-dominated sorting genetic algorithm with constrained tournament selection and two-stage crossover, and apply it to generate an optimized DNA sequence set. By comparing the result with other method, we show that the algorithm performs better than other method and has more diversity in the Pareto front.

Acknowledgments

This research was supported by the Molecular Evolutionary Computing (MEC) project of the Korean Ministry of Commerce, Industry and Energy, and the National Research Laboratory (NRL) Program from the Korean Ministry of Science and Technology, and the BK21-IT Program of the Korean Ministry of Education & Human Resources Development. The ICT at Seoul National University provided research facilities for this study.

Bibliography

[Arita00] Arita, M., Nishikawa, A., Hagiya, M., Komiya, K., Gouzu, H., and Sakamoto, K.,"Improving Sequence Design for DNA Computing," in *Proceedings of Genetic and Evolutionary Computation Conference 2000*, pp. 875–882, 2000.

[Deaton96] Deaton, R., Murphy, R. C., Garzon, M., Franceschetti, D. R., and Stevens, S. E. Jr., "Good Encodings for DNA-Based Solutions to Combinatorial Problems," in *Proceedings of the Second Annual Meeting on DNA Based Computers*, pp. 247–258, 1996.

[Deaton98] Deaton, R., Garzon, M., Murphy, R. C., Rose, J. A., Franceschetti, D. R., and Stevens, S. E. Jr., "Reliability and Efficience of a DNA-based Computation," *Physical Review Letters*, vol. 80, no. 2, pp. 417–420, 1998.

[Deb01a] Deb, K., Pratap, A., and Meyarivan, T., "Controlled Elitist Non-Dominated Sorting Genetic Algorithm for Better Convergence," in *Proceedings of the*

Table 2: Comparison with the fitness of the good codes in [Deaton96]. The fitness is examined by NACST/Report.

Sequence ($5' \rightarrow 3'$)	Continuity	Hairpin	H-measure	Similarity	Tm	GC%
Deaton *et al*.						
ATAGAGTGGATAGTTCTGGG	0	3	55	64	59.8408	45
CATTGGCGGCGCGTAGGCTT	0	0	69	51	73.2554	65
CTTGTGACCGCTTCTGGGGA	16	0	60	63	69.0571	60
GAAAAAGGACCAAAAGAGAG	41	0	58	45	58.6868	40
GATGGTGCTTAGAGAAGTGG	0	0	58	54	62.3221	50
TGTATCTCGTTTTAACATCC	16	4	61	50	57.301	35
TTGTAAGCCTACTGCGTGAC	0	3	75	55	64.67	50
Total Fitness	82	10	218	191		
NACST/Seq						
CCGTTCCTCCACCTCCTTCC	0	0	38	61	68.855	65
GAGTAGAGGAGGAGAGGAGT	0	0	88	28	63.2937	55
GGTCCAGAGGTAGAGAGGTA	0	0	67	37	63.5948	55
CTCCTTATCCTCACTCTCTC	0	0	43	53	60.2153	50
CCTCTTCTCTCTTCTCTTCT	0	0	36	49	59.5948	45
CACACCACACACACCAACCA	0	0	46	42	67.851	55
TTCCTTCTTCCTTCTTCTTC	0	0	30	50	58.9543	40
Total Fitness	0	0	174	160		

First International Conference on Evolutionary Multi-Criterion Optimization (EMO-2001), pp.284–298, 2001.

[Deb01b] Deb, K., *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Ltd., 2001.

[Garzon97] Garzon, M., Neathery, P., Deaton, R., Murphy, R. C., Franceschetti, D. R., and Stevens, S. E. Jr., "A New Metric for DNA Computing," in *Proceedings of Genetic Programming 1997*, pp. 472–478, 1997, The MIT Press.

[Reben01] Ruben, A. J., Freeland, S. J., and Landweber, L., "PUNCH: an Evolutionary Algorithm for Optimizing Bit Set Selection," in *Proceedings of the 7th International Workshop on DNA-Based Computers*, pp. 260–270, 2001.

[Shin02] Shin, S.-Y., Kim, D., Lee, I.-H., and Zhang, B.-T., "Evolutionary Sequence Generation for Reliable DNA Computing," *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, vol. 1, pp.79–84, 2002.

[Shin03] Shin, S.-Y., Lee, I.-H., Kim, D., and Zhang, B.-T., "Multi-objective Evolutionary Optimization of DNA Sequences for Reliable DNA Computing," BI Technical Report, 2003.

[Tanaka01] Tanaka, F., Nakatsugawa, M., Yamamoto, M., Shiba, T., and Ohuchi, A., "Developing Support System for Sequence Design in DNA Computing," in *Proceedings of the 7th International Workshop on DNA-Based Computers*, pp. 340–349, 2001.

[Tuplan02] Tuplan, D. C., Hoose, H., and Condon, A., "Stochastic Local Search Algorithm for DNA Word Design," in *Proceedings of 8th International Workshop on DNA-Based Computers*, pp. 229–241, 2002.

[Zhang98] Zhang, B.-T. and Shin, S.-Y., "Molecular Algorithms for Efficient and Reliable DNA Computing," in *Proceedings of Genetic Programming 1998*, pp. 735–742, 1998, Morgan Kaufmann.