Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM 2007)

Solving the Molecular Sequence Alignment Problem with Generalized Differential Evolution 3 (GDE3)

Saku Kukkonen
Department of Information Technology
Lappeenranta University of Technology
P.O. Box 20
FIN-53851 Lappeenranta, Finland
Email: saku.kukkonen@lut.fi

Sujit R. Jangam
Interdepartmental Biological Sciences
Northwestern University
2-100, Hogan, 2205 Tech Drive
Evanston - 60608, IL, USA
Email: s-jangam@northwestern.edu

Nirupam Chakraborti
Department of Metallurgical &
Materials Engineering
Indian Institute of Technology
Kharagpur 721302, India
Email: nchakrab@iitkgp.ac.in

Abstract— **Molecular sequence alignment is one of the most essential tools of the molecular biology. It permits to track changes and similarities between molecular sequences. In this paper the molecular sequence alignment problem is formulated suitable for an Evolutionary Algorithm (EA), and two problem instances are solved using Generalized Differential Evolution 3 (GDE3), which is a general purpose EA. Regardless of relatively large number of decision variables, the instances were solvable and results were comparable to those by sequence alignment solvers in comparison.**

I. Introduction

Alignment of two nucleic acid sequences is very important for the understanding of the functions of novel genes and the proteins encoded by them. It also helps to determine evolutionary relationships between the genes. The latter question is mostly answered by multiple sequence alignment, one method of doing which is progressive pairwise sequence alignment. In order to understand the function of a gene and/or the protein encoded by the gene, comparison of the gene sequence with a sequence of a gene of known function in a different species can be done and a putative function can be identified for the gene of interest. Thus, pairwise sequence alignment helps to predict the functions of novel genes within any species or genera. Particularly, these methods allow us to determine the similarity between genome segments of two or more organisms belonging to the same genera. Additionally, such techniques can also be used to study hosts of other processes such as molecular evolution, RNA folding, and gene regulation to name a few. These algorithms have also been used to determine homologies between proteins in order to predict structural and functional relationships. Various algorithms have been designed to carry out this task. These roughly fall into two categories: those that carry out global alignments and those that carry out local alignments. For pairwise sequence comparison, there is the well-known Smith and Waterman algorithm [1]. Algorithms using dynamic programming have been widely used for solving the sequence alignment problem. Additionally, various heuristic approaches have been developed to optimize accuracy and computational complexity such as ClustalW [2], T-Coffee [3], and BLAST [4]. Ant Colony Optimization (ACO) has also been used with Evolutionary Algorithms in sequence alignment [5], [6].

Evolutionary Algorithms (EAs) are search and optimization procedures that are motivated by the principles of natural evolution [7], [8]. Some fundamental ideas of natural genetics are borrowed and used artificially to construct search algorithms that are generally robust and require minimal problem information. They are developed in a way to simulate biological evolutionary process and genetic operations on chromosomes. It is now common knowledge that unlike the majority of the conventional search algorithms, an EA starts with a population of candidate solutions instead of a single one. In each step of a search, it emulates a number of biological processes, crossover, and mutation, for example to generate new and expectedly better progeny. The population-based approach of the EAs is very efficient for a global search. Unlike many gradient-based techniques they are also far less prone to get stranded in a local optima. Therefore, EAs are suitable for complex problems where the nature of the search space is not very clearly known *a priori*. Differential Evolution (DE) [9] is a relatively novel real-coded EA and it has been gaining popularity due to its good observed performance in practice. Its extension for constrained multi-objective optimization, Generalized DE, has been used here.

EAs have been used already earlier in multiple molecular sequence alignment [10], [11]. However, these methods have considered the alignment problem as a single-objective problem, whereas the nature of the problem is multi-objective as it will be described in Section III.

The rest of the paper is organized as follows: In Section II the concept of multi-objective optimization with constraints is handled briefly. In Section III, the molecular sequence alignment problem has been described in optimization point of view. Section IV describes the optimization method used to solve the problem. The problem formulation and experiments are reported in Sections V and VI. Finally, conclusions are drawn in Section VII.

II. Multi-Objective Optimization with Constraints

Many practical problems have multiple objectives and several aspects cause multiple constraints to problems. For ex-

ample, mechanical design problems have several objectives such as obtained performance and manufacturing costs, and available resources may cause limitations. Constraints can be divided into boundary constraints and constraint functions. Boundary constraints are used when the value of a decision variable is limited to some range, and constraint functions represent more complicated constraints, which are expressed as functions.

A mathematically constrained multi-objective optimization problem (MOOP) can be presented in the form [12, p. 37]:

$$\begin{array}{ll} \text{minimize} & \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_M(\vec{x})\} \\ \text{subject to} & (g_1(\vec{x}), g_2(\vec{x}), \ldots, g_K(\vec{x}))^T \leq \vec{0}. \end{array} \quad (1)$$

Thus, there are M functions to be optimized and K constraint functions. Maximization problems can be easily transformed to minimization problems and different constraints can be converted to form $g_j(\vec{x}) \leq 0$, Thereby the formulation in (1) is without loss of generality.

Typically, MOOPs are often converted to single-objective optimization problems by predefining weighting factors for different objectives, expressing the relative importance of each objective. Optimizing several conflicting objectives simultaneously without articulating the relative importance of each objective *a priori*, is often called Pareto-optimization. An obtained solution is Pareto-optimal if none of the objectives can be improved without impairing at least one other objective [12, p. 11]. If the obtained solution can be improved in such a way that at least one objective improves and the other objectives do not decline, then the new solution dominates the original solution. The objective of Pareto-optimization is to find a set of solutions that are not dominated by any other solution.

A set of Pareto-optimal solutions form a Pareto front, and an approximation of the Pareto front is called a set of non-dominated solutions, because the solutions in this set are not dominating each other in the space of objective functions. From the set of non-dominated solutions the decision-maker may pick a solution, which provides a suitable compromise between the objectives. This can be viewed as *a posteriori* articulation of the decision-makers preferences concerning the relative importance of each objective.

Weak dominance relation $\preceq$ between two vectors is defined such that $\vec{x}_1$ weakly dominates $\vec{x}_2$, *i.e.*, $\vec{x}_1 \preceq \vec{x}_2$ iff $\forall i : f_i(\vec{x}_1) \leq f_i(\vec{x}_2)$. Dominance relation $\prec$ between two vectors is defined such that $\vec{x}_1$ dominates $\vec{x}_2$, *i.e.*, $\vec{x}_1 \prec \vec{x}_2$ iff $\vec{x}_1 \preceq \vec{x}_2 \wedge \exists i : f_i(\vec{x}_1) < f_i(\vec{x}_2)$. The dominance relationship can be extended to take into consideration constraint values besides objective values. A constraint-domination $\prec_c$ is defined here so that $\vec{x}_1$ constraint-dominates $\vec{x}_2$, *i.e.*, $\vec{x}_1 \prec_c \vec{x}_2$ iff any of the following conditions is true [13]:

- $\vec{x}_1$ and $\vec{x}_2$ are infeasible and $\vec{x}_1$ dominates $\vec{x}_2$ in constraint function violation space.
- $\vec{x}_1$ is feasible and $\vec{x}_2$ is not.
- $\vec{x}_1$ and $\vec{x}_2$ are feasible and $\vec{x}_1$ dominates $\vec{x}_2$ in objective function space.

The definition for weak constraint-domination $\preceq_c$ is analogous dominance relation changed to weak dominance in above definition. This constraint-domination is a special case of more general concept of having goals and priorities that is presented in [14].

```
ATCCTGACGGT
|||  ||  ||
CCT ACCGT
```

Fig. 1. Optimal alignment between DNA sequences ATCCTGACGGT and CCTACCGT.

III. The Molecular Sequence Alignment Problem

Deoxyribonucleic acid (DNA) sequences are typical molecular sequences and consist of four nucleotides, adenine (A), cytosine (C), guanine (G), and thymine (T), which identify DNA sequences [15]. For example, string ATCCTGACGGT could code one short DNA sequence. In nature, DNA strings are replicating, *e.g.*, during cell division. Sometimes errors occur in DNA replication and resulting strings differ from each other. The replicated DNA sequence might contain altered, missing, or additional bases compared to the original DNA sequence. Tracking these changes between two molecular sequences is known as the *molecular sequence alignment problem*, where maximal number of matching base pairs (bps) is tried to find by shifting sequences and adding minimal amount of gaps between bases. For example, this kind of alignment between sequences ATCCTGACGGT and CCTACCGT is shown in Fig. 1. The sequences contain seven matching base pairs and there is one gap added into the shorter sequence.

The alignment problem is a multi-objective optimization problem since it has two main goals: maximizing matching bases and minimizing number of gaps. Traditionally objectives have been combined to a single objective function and solved using a single-objective optimizer. However, this will provide only one alignment between sequences at one optimization run. Probably, there are several different ways to explain difference between sequences, *e.g.*, true changes might have caused less gaps and more mismatching nucleotides than predicted by maximizing matching bases. Multi-objective optimization is a way to find several alternatives. Especially, a multi-objective EA can provide these at one optimization run [16]. Besides several objectives, the problem might have constrains, *e.g.*, maximal number of gaps between two bases and maximal number of gaps in whole molecular sequence. In these cases, an optimization method for constrained multi-objective optimization is needed in order to solve the problem. Such kind of method is described in following Section.

IV. Generalized Differential Evolution

A. Differential Evolution

The DE algorithm [9], [17] was introduced by Storn and Price in 1995. The design principles of DE are simplicity, efficiency, and the use of floating-point encoding instead of binary numbers. As a typical Evolutionary Algorithm (EA), DE has a random initial population that is then improved using

selection, mutation, and crossover operations. Several ways exist to determine a stopping criterion for EAs but usually a predefined upper limit G_{max} for the number of generations to be computed provides an appropriate stopping condition. Other control parameters for DE are the crossover control parameter CR, the mutation factor F, and the population size NP.

In each generation G, DE goes through each D dimensional decision vector $\vec{x}_{i,G}$ of the population and creates the corresponding trial vector $\vec{u}_{i,G}$ as follows [18]:

$$
\begin{aligned}
&r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}, (\text{randomly selected,} \\
&\quad \text{except mutually different and different from } i) \\
&j_{rand} = \text{floor}\left(rand_i[0,1) \cdot D\right) + 1 \\
&\text{for}(j = 1; j \leq D; j = j + 1) \\
&\{ \\
&\quad \text{if}(rand_j[0,1) < CR \vee j = j_{rand}) \\
&\qquad u_{j,i,G} = x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\
&\quad \text{else} \\
&\qquad u_{j,i,G} = x_{j,i,G} \\
&\}
\end{aligned}
\tag{2}
$$

This is the most common DE version, DE/rand/1/bin. Both CR and F remain fixed during the entire execution of the algorithm. Parameter $CR \in [0, 1]$, which controls the crossover operation, represents the probability that an element for the trial vector is chosen from a linear combination of three randomly chosen vectors and not from the old vector $\vec{x}_{i,G}$. The condition "$j = j_{rand}$" is to make sure that at least one element is different compared to the elements of the old vector. The parameter F is a scaling factor for mutation and its value is typically $(0, 1+]$. In practice, CR controls the rotational invariance of the search, and its small value (*e.g.*, 0.1) is practicable with separable problems while larger values (*e.g.*, 0.9) are for non-separable problems. The control parameter F controls the speed and robustness of the search, *i.e.*, a lower value for F increases the convergence rate but also the risk of getting stuck into a local optimum. Parameters CR and NP have the same kind of effect on the convergence rate as F has.

After the mutation and crossover operations, the trial vector $\vec{u}_{i,G}$ is compared to the old vector $\vec{x}_{i,G}$. If the trial vector has an equal or better objective value, then it replaces the old vector in the next generation. This can be presented as follows in the case of minimization of an objective [18]:

$$
\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if} \quad f(\vec{u}_{i,G}) \leq f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases} . \tag{3}
$$

DE is an elitist method since the best population member is always preserved and the average objective value of the population will never get worse.

B. Generalized Differential Evolution

The first version of a Generalized Differential Evolution (GDE) extended DE for constrained multi-objective optimization, and it modified only the selection rule of the basic DE [13]. The basic idea in the selection rule of GDE is that the trial vector is selected to replace the old vector in the next generation if it weakly constraint-dominates the old vector. This means that the trial vector is required to dominate compared old population member in the constraint violation space or in the objective function space, or at least provide an equally good solution as the old population member. There was no explicit sorting of non-dominated solutions [16, pp. 40 – 44] during the optimization process or any mechanism for maintaining the distribution and extent of solutions. Also, there was no extra repository for non-dominated solutions.

The second version, GDE2, made a decision based on the crowdedness when the trial and old vector were feasible and non-dominating each other in the objective function space [19]. This improved the extent and distribution of the obtained set of solutions but slowed down the convergence of the overall population because it favored isolated solutions far from the Pareto-front until all the solutions were converged near the Pareto-front.

The third and latest version is GDE3 [20], [21], which is formally presented in (4). Notation CD means crowding distance [16, pp. 248–249], which approximates the crowdedness of a vector in its non-dominated set. Besides the selection, another part of the basic DE has also been modified. Now, in the case of feasible and non-dominating solutions, both vectors are saved for the population of next generation. Before starting the next generation, the size of the population is reduced using non-dominated sorting and pruning based on diversity preservation. The pruning technique used in GDE3 is based on crowding distance, which provides good crowding estimation in the case of two objectives. However, crowding distance fails to approximate crowdedness of solutions when number of objectives is more than two [21], and other crowdedness estimation technique – such as presented in [22] – should be used instead.

Because the constraint handling of GDE versions is based on dominance principle, it can be implemented in such a way that the number of function evaluations is reduced because not always all the constraints and objectives need to be evaluated, *i.e.*, inspecting constraint violations (even one constraint) is often enough to determine, which vector to select for the next generation [9]. This reduces number of needed constraint function evaluations, which is helpful in the case of many and/or computationally heavy constraint functions.

All the GDE versions handle any number of M objectives and any number of K constraints, including the cases where $M = 0$ (constraint satisfaction problem) and $K = 0$ (unconstrained problem). When $M = 1$ and $K = 0$, versions are identical to the original DE, and this is why they are referred as *Generalized* DEs.

V. Problem Formulation

As described in Section III, the molecular sequence alignment problem is to find minimal amount of gaps between consecutive bases in order to maximize number of matching base pairs between two molecular sequences. In problem coding, the number of gaps between consecutive nucleotides was coded with variables, each variable coding one gap.

$$
\begin{array}{l}
\text{Input : } D, G_{max}, NP \geq 4, F \in (0,1+], CR \in [0,1], \text{and initial bounds: } \vec{x}^{(lo)}, \vec{x}^{(hi)} \\
\text{Initialize : } \left\{ \begin{array}{l} \forall i \leq NP \wedge \forall j \leq D : x_{j,i,G=0} = x_j^{(lo)} + rand_j[0,1] \cdot \left(x_j^{(hi)} - x_j^{(lo)} \right) \\ i = \{1,2,\ldots,NP\}, j = \{1,2,\ldots,D\}, G = 0, m = 0, rand_j[0,1) \in [0,1), \end{array} \right. \\
\left\{ \begin{array}{l}
\text{While } G < G_{max} \\
\qquad \forall i \leq NP \left\{ \begin{array}{l}
\text{Mutate and recombine:} \\
r_1, r_2, r_3 \in \{1,2,\ldots,NP\}, \text{randomly selected,} \\
\qquad \text{except mutually different and different from } i \\
j_{rand} \in \{1,2,\ldots,D\}, \text{randomly selected for each } i \\
\\
\forall j \leq D, u_{j,i,G} = \left\{ \begin{array}{ll} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) & \\ \qquad \text{if} \quad rand_j[0,1) < CR \vee j = j_{rand} & \\ x_{j,i,G} \quad \text{otherwise} & \end{array} \right. \\
\text{Select :} \\
\vec{x}_{i,G+1} = \left\{ \begin{array}{ll} \vec{u}_{i,G} & \text{if} \quad \vec{u}_{i,G} \preceq_c \vec{x}_{i,G} \\ \vec{x}_{i,G} & \text{otherwise} \end{array} \right. \\
\\
\text{Set :} \\
\qquad m = m + 1 \\
\qquad \vec{x}_{NP+m,G+1} = \vec{u}_{i,G} \qquad \text{if} \left\{ \begin{array}{l} \forall j : g_j(\vec{u}_{i,G}) \leq 0 \\ \wedge \\ \vec{x}_{i,G+1} == \vec{x}_{i,G} \\ \wedge \\ \vec{x}_{i,G} \not\prec \vec{u}_{i,G} \end{array} \right.
\end{array} \right. \\
\qquad \left\{ \begin{array}{l}
\text{While } m > 0 \\
\qquad \text{Select } \vec{x} \in \{\vec{x}_{1,G+1}, \vec{x}_{2,G+1}, \ldots, \vec{x}_{NP+m,G+1}\} : \\
\qquad\qquad \left\{ \begin{array}{l} \forall i \quad \vec{x} \not\prec_c \vec{x}_{i,G+1} \\ \wedge \\ \forall (\vec{x}_{i,G+1} : \vec{x}_{i,G+1} \not\prec_c \vec{x}) \quad CD(\vec{x}) \leq CD(\vec{x}_{i,G+1}) \end{array} \right. \\
\qquad \text{Remove } \vec{x} \\
\qquad m = m - 1
\end{array} \right. \\
\qquad G = G + 1
\end{array} \right.
\end{array}
\tag{4}
$$

In addition, one variable coded transition between first nucleotides of sequences. This means that total number of variables was one smaller than the total number of nucleotides in sequences. For example, variable vector $\vec{x} = [-2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$ represents alignment shown in Fig. 1. First variable in the vector codes the transition between first nucleotides of sequences and its value is -2 since the first nucleotide of the first sequence locates two nucleotide locations to left from the first nucleotide of the second sequence in Fig. 1. After the first variable are gaps for the first sequence and then gaps for the second sequence coded into variables. The problem coding is illustrated in Fig. 2.

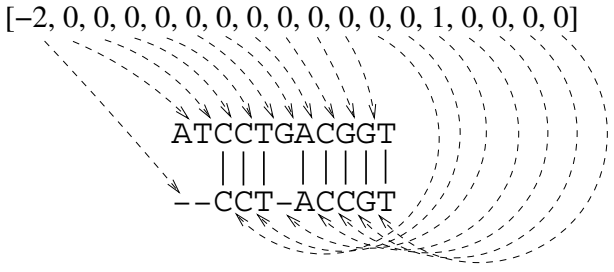

Fig. 2. Illustration of coding between decision variables and a sequence alignment.

The problem formulation contained also constraints. The total number of gaps was limited to be at most 50% from the number of nucleotides in a sequence, and the number of gaps between consecutive bases was limited to be at most 10% from the number of nucleotides in a sequence. Transition between sequences was limited according to number of nucleotides in sequences and minimal number of gaps was (naturally) limited to be 0. The whole alignment problem for molecular sequences

S_1 and S_2 had following formulation:

$$\begin{array}{ll} \text{Maximize} & \text{number of matching base pairs} \\ \text{Minimize} & \sum_{i=2}^{|S_1|+|S_2|-1} x_i \\ \\ \text{Subject to} & \sum_{i=2}^{|S_1|} x_i \leq 0.5|S_1| \\ & \sum_{i=|S_1|+1}^{|S_1|+|S_2|-1} x_i \leq 0.5|S_2| \\ & x_i \leq 0.1|S_1|, i = 2, \ldots, |S_1| \\ & x_i \leq 0.1|S_2|, i = |S_1|+1, \ldots, |S_1|+|S_2|-1 \\ & -|S_1|+1 \leq x_1 \leq |S_2|-1 \\ & x_i \geq 0, i = 2, \ldots, |S_1|+|S_2|-1, \end{array} \quad (5)$$

where x_i is ith variable from a decision variable vector $\vec{x}$ and $|S|$ is a number of nucleotides in molecular sequence S.

Boundary constraint violations can be handled different ways. One way is to reflect violating variable values back from the violated boundary using following rule before the selection operation of GDE3:

$$u_{j,i,G} = \begin{cases} 2x_j^{(lo)} - u_{j,i,G} & \text{if} \quad u_{j,i,G} < x_j^{(lo)} \\ 2x_j^{(up)} - u_{j,i,G} & \text{if} \quad u_{j,i,G} > x_j^{(up)} \\ u_{j,i,G} & \text{otherwise} \end{cases}, \quad (6)$$

where $x_j^{(lo)}$ and $x_j^{(up)}$ are lower and upper limits respectively for a decision variable x_j. This boundary constraint violation method was used in experiments.

Fig. 4 illustrates how a trial vector is generated using the problem coding and DE operators.

VI. Experiments

Three DNA sequences were taken from a public GenBank database of the National Center for Biotechnology Information [23]. Code names of these sequences are AY216995, AY217003, and Z36014, and they contain 1551, 1178, and 1710 bases, respectively (sequences are shown in Appendix). Two alignment problem instances were solved: AY216995 vs. Z36014 with 3260 variables and AY217003 vs. Z36014 with 2887 variables. Relatively large number of integer variables and non-linear functions make the instances challenging for a general purpose optimizer.

Problems and GDE3 optimizer were implemented in the C programming language. All the variables are internally real numbers and their value was rounded to nearest integer prior evaluation of objectives and constraints. Variables coding gaps were uniformly initialized to range $[0, 2]$.

The alignment problem instances were tried to solve using different control parameter combinations $CR \in \{0.0, 0.1, \ldots, 1.0\}$, $F \in \{0.1, 0.3, 0.5\}$, and $NP \in \{100, 500, 1000\}$ with a couple of repetitions. Maximum number of generations was 100 000 in all the cases. With the both problem instances it was observed that $F = 0.5$ is too large value to have converged solutions in number of generations used. Also, $CR = 0.0$ value should not be used for the same reason. For the smallest population size (100), F should be 0.3, for the larger population sizes, also $F = 0.1$ was usable parameter value for the first problem instance.

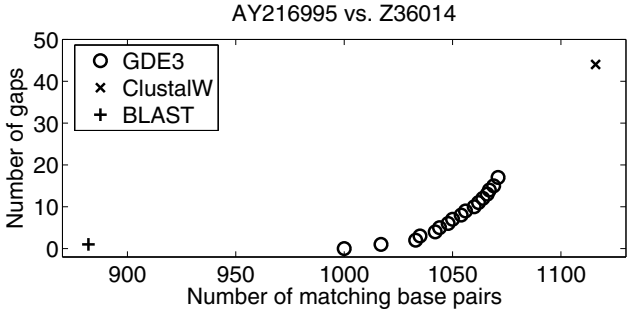

Fig. 3. Solutions for the AY216995 vs. Z36014 alignment problem instance.

It appeared that the crossover parameter should be in the value range $0.1 \leq CR \leq 0.7$, smaller values providing better convergence in general. There is little difference in results between population sizes 500 and 1000, thus it appears that population size 500 is already big enough for the problem instances.

Fig. 3 shows the best obtained set of solutions for the AY216995 vs. Z36014 problem instance. The control parameters values of GDE3 were $CR = 0.6$, $F = 0.1$, and $NP = 500$. Results obtained with ClustalW and BLAST are also shown[1]. Maximal number of matching base pairs was 1116 with ClustalW. Corresponding number of gaps was 44 and largest number of gaps between consecutive bases was seven. BLAST gave the worst solution, which had 882 matching base pairs and one gap. GDE3 found 17 different solutions with different number of matching base pairs and gaps. Maximal number of matching base pairs with GDE3 was 1071 and corresponding number of gaps was 17. Any consecutive bases did not have more than one gap. Apparently, solutions with more gaps between consecutive bases were not found because of the relatively small initialization range. DE is capable to advance search outside of the initialization range, but this would had needed larger F value and/or larger population size. Probably better choice would had been to use larger initialization range. All these options would slow down convergence.

Fig. 5 shows the best obtained set of solutions for the AY217003 vs. Z36014 problem instance. Corresponding control parameters values of GDE3 were $CR = 0.5$, $F = 0.3$, and $NP = 500$. All the methods found the same solution, which has 1167 matching base pairs and one gap. Besides, GDE3 found two other solutions from which one is better than found by ClustalW and BLAST. This alignment has 1168 matching base pairs and two gaps. Decision variable values for this solution were zero except $x_1 = 148$, $x_{1178} = 1$, and $x_{2458} = 1$.

Since ClustalW and BLAST are specialized to solve the molecular sequence alignment problem, they solved problem instances much faster than GDE3; ClustalW and BLAST give

[1]ClustalW tool used is at http://www.ebi.ac.uk/clustalw/ and BLAST tool is at http://www.ncbi.nlm.nih.gov/blast/bl2seq/wblast2.cgi (15.1.2007).

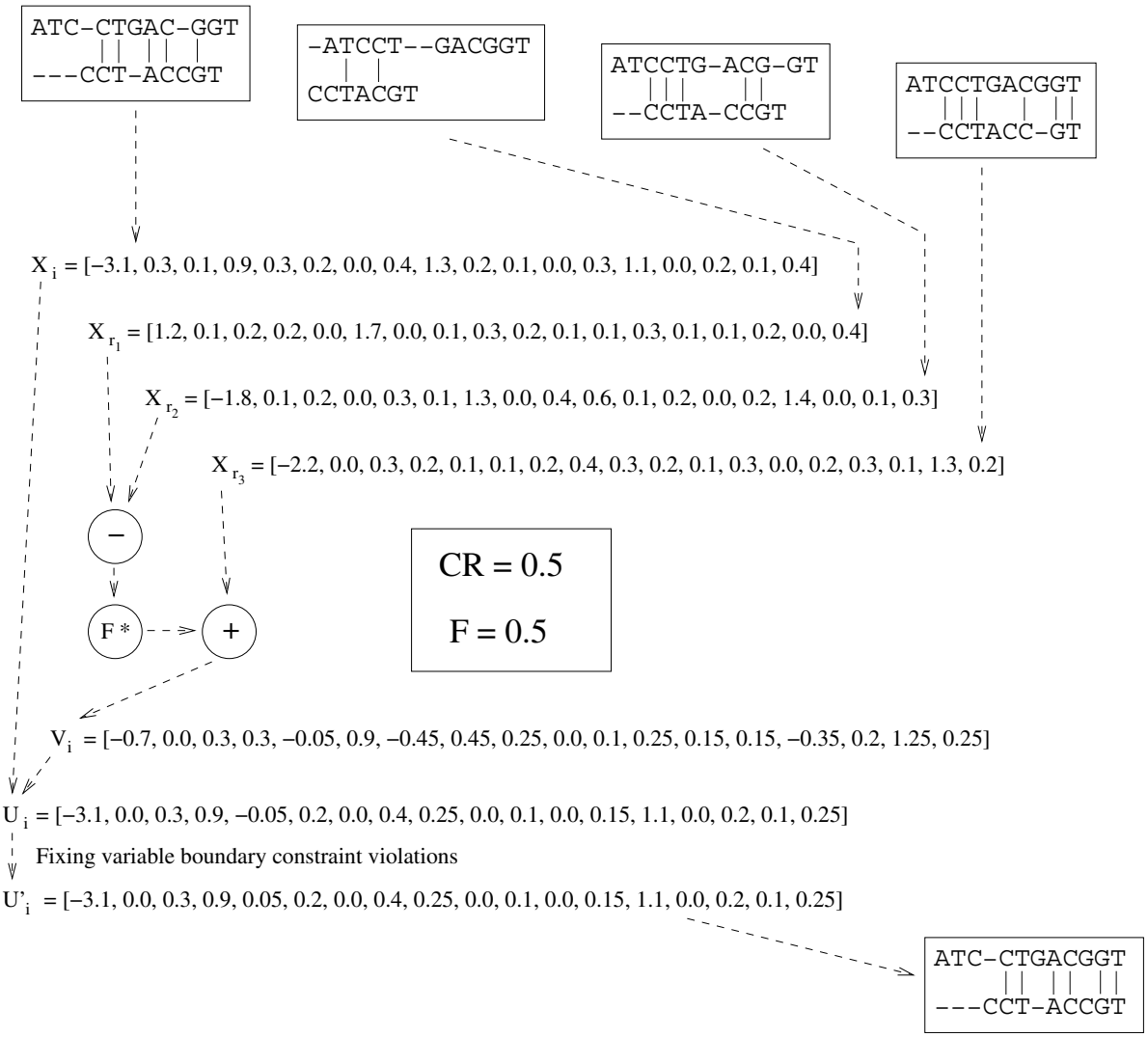

Fig. 4. Creation of a trial vector using the problem coding and DE operators.

a solution in seconds whereas 100 000 generations with a population of 500 individuals takes hours to evolve with GDE3. Still, it is notable that GDE3, a general purpose EA, was able to solve the problem instances despite large number of discrete decision variables.

Figs. 6 and 7 show convergence curves of objective values during GDE3 runs corresponding to Figs. 3 and 5. Curves represent the average objective values of the population through generations. For the AY216995 vs. Z36014 alignment problem instance in Fig. 6, the first objective value change most during 20 000 first generations and after this its value is improving very slowly. Corresponding number of gaps is reducing till 40 000 generations and after that it is increasing with the first objective. Also in the case of the AY217003 vs. Z36014 alignment problem instance in Fig. 7, the conflict between objectives cause non-monotonic convergence, this time to the curve of the first objective: the value of the first objective increases except when the number of generations is 30 000. The number of gaps is larger in Fig. 7 than in Fig. 6 during early generations because of larger F value – number of gaps is large at the beginning of generations also for the first problem instance when $F = 0.3$. Example about this is shown in Fig. 8, where the same control parameter values are used for the AY216995 vs. Z36014 alignment problem instance as for the AY217003 vs. Z36014 alignment problem instance in Fig. 7.

VII. Conclusions

The molecular sequence alignment problem with two instances has been solved using GDE3, a general purpose EA. Although problem is non-linear and it has relatively large number of integer variables, it was solvable. Obtained results were found comparable to those obtained with two problem solvers designed for the molecular sequence alignment problem. In

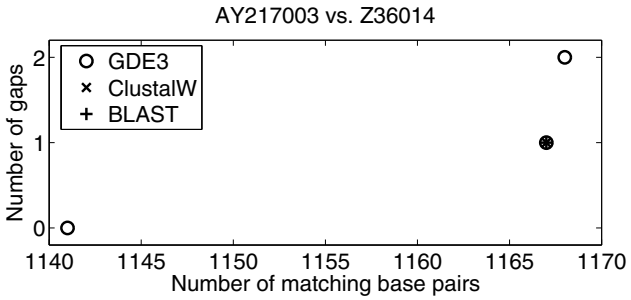

Fig. 5. Solutions for the AY217003 vs. Z36014 alignment problem instance.

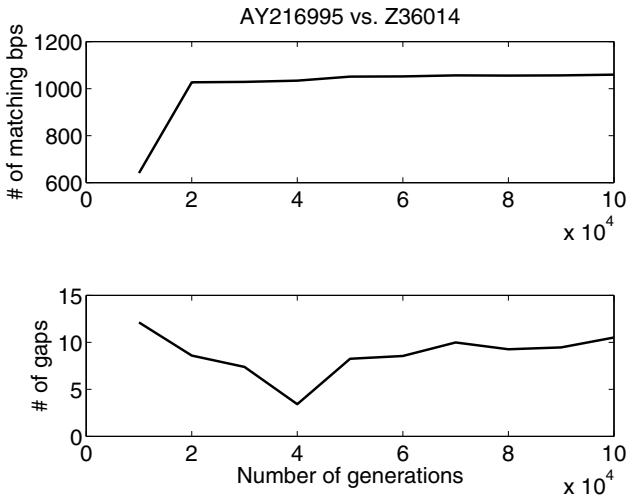

Fig. 6. Convergence curves of the objectives for the AY216995 vs. Z36014 alignment problem instance ($CR = 0.6$, $F = 0.1$, and $NP = 500$).

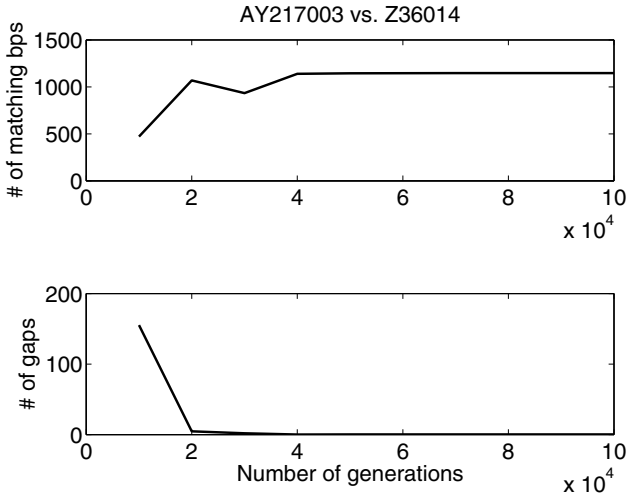

Fig. 7. Convergence curves of the objectives for the AY217003 vs. Z36014 alignment problem instance ($CR = 0.5$, $F = 0.3$, and $NP = 500$).

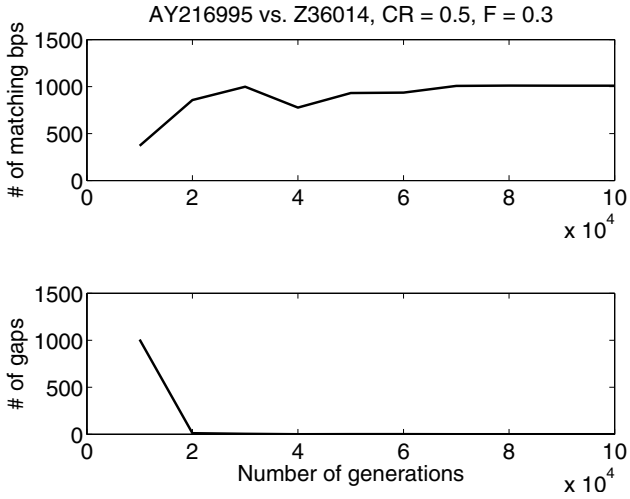

Fig. 8. Convergence curves of the objectives for the AY216995 vs. Z36014 alignment problem instance ($CR = 0.5$, $F = 0.3$, and $NP = 500$).

one instance, GDE3 found even better alignment than the other solvers in comparison.

ACKNOWLEDGMENTS

The first author gratefully acknowledges support from the East Finland Graduate School in Computer Science and Engineering (ECSE), the Academy of Finland, Technological Foundation of Finland, and Finnish Cultural Foundation.

REFERENCES

[1] T. F. Smith and M. S. Waterman, "Comparison of biosequences," *Advances in Applied Mathematics*, vol. 2, pp. 482–489, 1981.

[2] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice," *Nucleic Acids Research*, vol. 22, no. 22, pp. 4673–4680, Nov 1994.

[3] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: A novel method for fast and accurate multiple sequence alignment," *Journal of Molecular Biology*, vol. 302, pp. 205–217, 2000.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, pp. 403–410, 1990.

[5] S. R. Jangam and N. Chakraborti, "A novel method for alignment of two nucleic acid sequences using ant colony optimization and genetic algorithms," *Applied Soft Computing*, in press.

[6] Z.-J. Lee, S.-F. Su, C.-C. Chuang, and K.-H. Liu, "Genetic algorithm with ant colonyt optimization (ga-aco) for multiple sequence alignment," *Applied Soft Computing*, in press.

[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.

[8] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

[9] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin: Springer-Verlag, 2005.

[10] C. Notredame and D. G. Higgins, "SAGA: sequence alignment by genetic algorithm," *Nucleic Acids Research*, vol. 24, no. 8, pp. 1515–1524, 2000.

[11] C. Zhang and A. K. C. Wond, "A genetic algorithm for multiple molecular sequence alignment," *Bioinformatics*, vol. 13, no. 6, pp. 565–581, 1997.

[12] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston: Kluwer Academic Publishers, 1998.

[13] J. Lampinen, "DE's selection rule for multiobjective optimization," Lappeenranta University of Technology, Department of Information Technology, Tech. Rep., 2001. [Online]. Available: http://www.it.lut.fi/kurssit/03-04/010778000/MODE.pdf, 15.2.2006

[14] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms–part i: A unified formulation," *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, vol. 28, no. 1, pp. 26–37, Jan 1998.

[15] J. M. Butler, *Forensic DNA Typing*. Elsevier, 2005.

[16] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, England: John Wiley & Sons, 2001.

[17] R. Storn and K. V. Price, "Differential Evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec 1997.

[18] K. V. Price, *New Ideas in Optimization*. London: McGraw-Hill, 1999, ch. An Introduction to Differential Evolution, pp. 79–108.

[19] S. Kukkonen and J. Lampinen, "An extension of Generalized Differential Evolution for multi-objective optimization with constraints," in *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, Birmingham, England, Sept 2004, pp. 752–761.

[20] ——, "GDE3: The third evolution step of Generalized Differential Evolution," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, Edinburgh, Scotland, Sept 2005, pp. 443–450. [Online]. Available: http://www.iitk.ac.in/kangal/papers/k2005013.pdf, 6.11.2006

[21] S. Kukkonen and K. Deb, "Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems," in *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, Vancouver, BC, Canada, July 2006, pp. 3995–4002.

[22] ——, "A fast and effective method for pruning of non-dominated solutions in many-objective problems," in *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX)*, Reykjavik, Iceland, Sept 2006, pp. 553–562.

[23] National Center for Biotechnology Information (NCBI), "GenBank overview," 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/Genbank/index.html, 9.11.2006

APPENDIX

The DNA sequences used in the experiments. The source of sequences is [23].

AY216995 (1551 bases):

```
TTTCTCAAGAAAGGTCTCGTGCCACGGCGCCATTGTGAAAATGCTTC
TTCCGTCAAGAAATGAGTCAAGGCCATGCCAATAAGATCTATCAATT
TTGACATCAGGCGACGCGGCCCTAGTCAAACTCTTCGTATAAATACC
ATGAGACAAGAAAGCAAAGGGATAGACATATTTCTCGAGCAATTATA
GTTTCCGTTTAGTGGGGCGGGTGCTAGAAGTATAATTCAAAAATTCT
GGTCTACTATATATCAACGCAACCTAACTATGTCTTCGCAAGCTATT
CCCGAAAGCCAAAAGGCAATCGTTTTTTATGAAACAAACGGTAAGCT
GGAATACAAGGACATTGCAGTCCCAGAGCCTAAAGCCAATGAAATTC
TAGTTCATGTCAAATATTCCGGTGTTTGTCACAGCGATTTACACGCA
TGGCGTGGCGATTGGCCATTCCAATTGAAATTTCCATTGATTGGTGG
GCATGAGGGTGCAGGTGTTGTTGTTAAGTTGGGCTCCAACGTCAAGG
GCTGGAAAGTCGGTGACTTGGCAGGTATCAAATGGTTGAACGGTACG
TGTATGTCCTGTGAATATTGTGAAGTTGGTAACGAATCTCAATGTCC
CCATTTGGATGGTACCGGCTTCACTCATGATGGTACTTTTCAAGAGT
ATGCAACTGCCGATGCTGTTCAGGCTGCTCATCTCCCAGAGAATGTC
GACCTTGCGGAAGTTGCCCCAATTTTGTGTGCAGGGGTTACAGTTTA
TAAAGCACTGAAAAGAGCCAATTTAATTCCCGGTCAATGGGTTACTA
TTTCTGGTGCTTGCGGAGGATTAGGTTCCCTGGCCATTCAGTATGCT
ACGGCTATGGGTTACAGGGTCATTGGTATTGACGAAGGGGAGACCAA
AAAACAGCTATTCGAAGGACTAGGTGGGGAAGTATTTATCGATTTCA
CAAAAGAAAAGGATATCGTTGGTGCCGTTATCAAAGCCACTGACGGT
GGTTCCCATGGTGTTATCAACGTGTCTGTTTCTGAAGCAGCCATCGA
AGCTTCAACGAGATACTGTAGACCCAACGGTACTGTTGTCTTGGTTG
GTATGCCGGCTCATGCTTATTGCAAATCCGACGTCTTCAATCAAGTT
GTTAAATCCATTTCCATTGTCGGATCTTGCGTTGGTAACAGAGCTGA
CACAAGGGAAGCTTTGGACTTTTTCGCCAGAGGATTGATTAAATCTC
CAATTCACTTGGCTAGCCTATCTGATGTTCCAGAGATTTTTGAAAAG
ATGGAAAAGGGTGAAATCGTTGGCAGATATGTTATTGACACTTCTAA
ATAATCTCTTTAACGAATGTAACGAATTGTATCTTTGATTTTATATA
ATCACATATTCTGCAAGTATGATCTTTTTTAACATGTATTTTTAAGA
AAGCGAATCTTTCTTACGTCCTTCGAAATTCTTATAAGCAAGACTGC
TTCGAGAATTTGCGTCATAAAACTCAGCACGCCCTGAGTATATATTA
GATTATCAGATGTATGATTCAGCACAGTACATTTTATTTGGGGTTGT
```

AY217003 (1178 bases):

```
TTGTCTCTGATTGGAAGATACCTAAAAAAGTTATTTAACTACATATC
AACAAAATCAAAGCAACATGCCCTCGCAAGTCATTCCTGAAAAACAA
AAGGCTATTGTCTTTTATGAGACAGATGGAAAATTGGAATATAAAGA
CGTCACAGTTCCGGAACCTAAGCCTAACGAAATTTTAGTCCACGTTA
AATATTCTGGTGTTTGTCATAGTGACTTGCACGCGTGGCACGGTGAT
TGGCCATTTCAATTGAAATTTCCATTAATCGGTGGTCACGAAGGTGC
TGGTGTTGTTGTTAAGTTGGGATCTAACGTTAAGGGCTGGAAAGTCG
GTGATTTTGCAGGTATAAAATGGTTGAATGGGACTTGCATGTCCTGT
GAATATTGTGAAGTAGGTAATGAATCTCAATGTCCTTATTTGGATGG
TACTGGCTTCACACATGATGGTACTTTTCAAGAATACGCAACTGCCG
ATGCCGTTCAAGCTGCCCATATTCCACCAAACGTCAATCTTGCTGAA
GTTGCCCCAATCTTGTGTGCAGGTATCACTGTTTATAAGGCGTTGAA
AAGAGCCAATGTGATACCAGGCCAATGGGTCACTATATCCGGTGCAT
GCGGTGGCTTGGGTTCTCTGGCAATCCAATACGCCCTTGCTATGGGT
TACAGGGTCATTGGTATCGATGGTGGTAATGCCAAGCGAAAGTTATT
TGAACAATTAGGCGGAGAAATATTCATCGATTTCACGGAAGAAAAAG
ACATTGTTGGCGCTATAATAAAGGCCACTAATGGCGGTTCTCATGGA
GTTATTAATGTGTCTGTTTCTGAAGCAGCTATCGAGGCTTCTACGAG
GTATTGTAGGCCCAATGGTACTGTCGTCCTGGTTGGTATGCCAGCTC
ATGCTTACTGCAATTCCGATGTTTTCAATCAAGTTGTAAAATCAATT
TCCATCGTTGGATCTTGTGTTGGAAATAGAGCTGATACAAGGGAGGC
TTTAGATTTCTTCGCCAGAGGTTTGATCAAATCTCCGATCCACTTAG
CTGGCCTATCGGACGTTCCTGAAATTTTTGCAAAGATGGAGAAGGGT
GAAATTGTTGGTAGATATGTTGTTGAGACTTCTAAATGATCTTTTGT
AACGAAATTTGATGAATATATTTTTACTTTTTATATAAGCTATTTTG
TAA
```

Z36014 (1710 bases):

```
GTGACAATGAAATAATCAAATTGTGACATCTGCTGACGCGGGATCGT
TCCTTCGTATTGTCTAGATTGTAATCTATATAACATACTACGAATAT
AAAAGAGGGACTACAAGATATTTCTAGCGCAAACTACTGCTTTACTG
TCTCACAATGTCTCTGATTGGAAGATACCTAAGAAAATTATTTAACT
ACATATCTACAAAATCAAAGCATCATGCCTTCGCAAGTCATTCCTGA
AAAACAAAAGGCTATTGTCTTTTATGAGACAGATGGAAAATTGGAAT
ATAAAGACGTCACAGTTCCGGAACCTAAGCCTAACGAAATTTTAGTC
CACGTTAAATATTCTGGTGTTTGTCATAGTGACTTGCACGCGTGGCA
CGGTGATTGGCCATTTCAATTGAAATTTCCATTAATCGGTGGTCACG
AAGGTGCTGGTGTTGTTGTTAAGTTGGGATCTAACGTTAAGGGCTGG
AAAGTCGGTGATTTTGCAGGTATAAAATGGTTGAATGGGACTTGCAT
GTCCTGTGAATATTGTGAAGTAGGTAATGAATCTCAATGTCCTTATT
TGGATGGTACTGGCTTCACACATGATGGTACTTTTCAAGAATACGCA
ACTGCCGATGCCGTTCAAGCTGCCCATATTCCACCAAACGTCAATCT
TGCTGAAGTTGCCCCAATCTTGTGTGCAGGTATCACTGTTTATAAGG
CGTTGAAAAGAGCCAATGTGATACCAGGCCAATGGGTCACTATATCC
GGTGCATGCGGTGGCTTGGGTTCTCTGGCAATCCAATACGCCCTTGC
TATGGGTTACAGGGTCATTGGTATCGATGGTGGTAATGCCAAGCGAA
AGTTATTTGAACAATTAGGCGGAGAAATATTCATCGATTTCACGGAA
GAAAAAGACATTGTTGGTGCTATAATAAAGGCCACTAATGGCGGTTC
TCATGGAGTTATTAATGTGTCTGTTTCTGAAGCAGCTATCGAGGCTT
CTACGAGGTATTGTAGGCCCAATGGTACTGTCGTCCTGGTTGGTATG
CCAGCTCATGCTTACTGCAATTCCGATGTTTTCAATCAAGTTGTAAA
ATCAATCTCCATCGTTGGATCTTGTGTTGGAAATAGAGCTGATACAA
GGGAGGCTTTAGATTTCTTCGCCAGAGGTTTGATCAAATCTCCGATC
CACTTAGCTGGCCTATCGGATGTTCCTGAAATTTTTGCAAAGATGGA
GAAGGGTGAAATTGTTGGTAGATATGTTGTTGAGACTTCTAAATGAT
CTTTTGTAACGAATTTGATGAATATATTTTTACTTTTTATATAAGCT
ATTTTGTAGATATTGACTTTTTACGATTTATTTGTAACAATGAGAAT
TACTCCATTTCTGAACTTCAGTAAATAGCGAGTGATTCTGTACTTTG
CGAGAACCGGTGGACATTTGGTATTTTGCCTTACAAGAACAACCTAT
ACAAACGTTTCAATATCTAATTCTTTGTAATCCATTGTTTTACGAGA
CATATAATGTGATATATAGATGAACTTTACGTATAAAATGATATATT
TAAAACTAGCAACTGCGTGCGTAAGACAAACTGAAATAGGCCATTTA
CGGAAAAGAAATTTAATAATGTCGACTGGAAACTGAAACCAGGAGGA
GTAGAAATTGGTTAAATTGATTAGCTAAAATTTACTCGTTGTGGACA
GAGTTTGAGCCAAGCGGA
```