Placement of Unique Restriction Sites in Synthetic Genomes using Multi-Objective Optimization

Mahfuza Sharmin*†§, and M. Sohel Rahman *‡
*AℓEDA Group, Department of CSE,
Bangladesh University of Engineering and Technology, Dhaka-1000, Bangladesh
†sharmin@cse.buet.ac.bd, ‡msrahman@cse.buet.ac.bd
§Supported by a CodeCrafters-Investortools Research Grant for CSE, BUET (http://www.codecraftersintl.com/researchgrant.html)

***Abstract*—Placing unique restriction sites in synthetic genomes is computationally hard problem. This paper targets to apply variants of genetic algorithm to modify virus length genomes to introduce a large number of evenly spaced unique restriction sites while preserving their amino-acid sequence. From our experimental results we show that genetic algorithms, considering multiple objectives non-linearly, give better results than traditional heuristic methods and local search methods.**

***Keywords*-Synthetic biology, Restriction enzyme, Genetic algorithms, Multiobjective optimization**

I. Introduction

Synthetic biology is an emerging field in genetic engineering that involves redesigning of existing, natural biological systems for new purposes and the creation of novel artificial living things. The research here poses various newer algorithmic and computational problems. Viral genome synthesis, i.e., designing novel living organisms at the genetic level, is an important research area in this field.

Restriction enzymes are special enzymes that can recognize and cut specific nucleotide sequences in DNA molecules, i.e., the bacterium Hemophilus aegypticus produces HaeIII enzyme that cuts DNA wherever it finds the sequence 5'GGCC3'/3'CCGG5'. The pattern, GGCC/CCGG, is called the *restriction enzyme recognition site/restriction site/recognition site*. *Unique restriction enzyme* cuts the DNA at exactly one place. Due to their unambiguous recognition property, unique restriction enzymes are appealing to the scientists. That is why, in many cases, before experimenting the original sequence obtained from a living organism, such unique sites are artificially inserted and/or deleted in the DNA sequence keeping it functionally equivalent to the original one. To restructure the genome of an organism into a functionally equivalent sequence, it's amino acid sequence must be preserved. Each amino acid is encoded by a series of three adjacent bases, called *Codon*. Since there are more codons than encodable amino acids some degeneracy exist. Multiple codons representing the same amino acid are called *Synonymous codons*. For successful insertion and deletion of restriction sites, synonymous codons must be used while placing one codon in lieu of another.

Evolutionary Algorithms (EA) are popular approaches to solving single and multi-objective optimization problems [3]. These are search methods that take inspirations from natural phenomenon of selection and survival of the fittest in the biological world. They differ from the more traditional optimization techniques in that they employ a search involving a "population" of solutions instead of a single point. This paper presents some hybrid *Genetic algorithm*s to optimize the placement of unique restriction enzymes in viral genomes. In particular, here, we tackle the so called *Unique Restriction Site Placement Problem* (*URSPP*). Our experimental results indicate that the application of genetic algorithms in *URSPP* offers better outcome in various aspects. We believe that implementation of our approach will serve as a useful sequence design tool for researchers and practitioners in both industry and academia.

Designing synthetic genes by hand is a time-consuming and error-prone process. In the past, researchers used to send off their requirements of the separate steps of synthetic gene design to a black box provided by a gene synthesis company and let it use its proprietary programs to design genes. Now-a-days these syntheses are relatively inexpensive commercialized services. OriGene, GeneArt are some of the leading commercial vendors who continue to innovate with technologies to meet the growing demands of the researchers for gene synthesis. SiteFind is a free software tool that enables the user to introduce a novel restriction site into the mutation primers without changing the peptide sequence so that the site can be used as a marker for successful mutation [2]. GeneJax is a CAD tool for the parts extraction and visualization stages of the genome refactoring process [1]. In [7], the author proposes an algorithm for optimally removing restriction sites against sets of cutter sequences.

II. Preliminaries

The problem we handle in this paper is known as the *Unique Restriction Site Placement Problem* (*URSPP*) in the literature. Biologically, the target is to fabricate a new plasmid sequence from a given viral plasmid sequence so that the new sequence contains maximum unique restriction

978-1-4673-2560-8/12/$31.00 ©2012 IEEE

sites from a given restriction enzyme set through minimal sequence editing with the added constraint that the placed sites are distributed as evenly as possible along the sequence. Combinatorially, given a text and a set of patterns, the goal is to construct a new text with patterns inserted as many as possible, editing of letters as less as possible and such that the maximum distance between two consecutive inserted patterns (considering all insertions) is minimum. In [4], both the decision and optimization version of the problem have been studied.

The decision version of *URSPP* problem has been proved to be NP complete and it has been shown that the optimization version can not be approximated within a factor of 3/2 [4]. The best known result for the optimization version is a polynomial-time 2-approximation algorithm [4]. Also a *Dynamic Programming* (*DP*) algorithm with an exponential running time of $O(n^2 2^r)$ and some heuristic algorithms have been proposed in [4]. Here, n is the number of events[1] and r is the number of unused restriction enzymes. The authors in [4] provided two types of approximate implementations of the DP algorithm because of its exponential running time and memory requirement, namely *forward* and *backward* implementation. The other heuristic versions are called *greedy* approach and maximal bipartite *matching* technique. Very recently, some popular local search techniques like hill climbing and variations thereof have been applied to solve URSPP [6]. In [6] the representation of candidate solution and quality assessment function are proposed and using two types of mutation algorithm significant optimization has been achieved. For assessing fitness of a solution the objectives are converted to a single one by weighted sum. However, for *URSPP* we are interested in optimizing all objectives individually rather than a single fitness. Also the objectives are non-linear and it is very likely that scaling/weights used for one problem instance (viral sequence) will not be applicable for another. Hence we concentrate to genetic approaches where objectives are focused seperately/independently.

III. Methodology

In this paper, we use the same representation and quality assessment as proposed in [6]. We also use the *point mutation* as our breeding operator. However, before starting the main algorithm, the given set of restriction enzymes are preprocessed in order to construct a special data structure called *Restriction Map*[4]. For each restriction enzyme, all the site occurrences are obtained from *Restriction Map* and all the possible insertion points with number of required base changes are listed using an $O(nm)$ time[2] algorithm.

[1]An event corresponds to either a location where a restriction enzyme is currently cutting or a place where an unused restriction enzyme can be inserted [4].

[2]Here, n is the length of the sequence and m is the length of the recognition site.

Each candidate solution is represented by a direct encoding of a fixed sized double vector as shown in Table I [6].

Table I
One Candidate Solution

Meaning of notation	RE_1,	RE_2,	...	RE_i,	...
Gene for RE's	g_1	g_2	...	g_i	...
Presence or Absence, $\mathcal{B}$	1,	0,	...	1,	...
Possible location, $\mathcal{L}$	l_1,	l_2,	...	l_i,	...

For breeding of new candidate solutions we use both recombination/crossover and mutation. For recombination we apply the standard two point crossover technique [3].

The objectives of URSPP are denoted as follows:

f_1 = number of 1's in the boolean sub-vector, $\mathcal{B}$

f_2 = number of positions where the original sequence differs from the synthesized sequence

f_3 = the maximum among the distances between two consecutive restriction sites

To avoid linear parsimony pressure, we plan to use variants of *tournament selection* to select highly qualified solution from a population. Basically, *tournament selection* is a non-parametric selection algorithm which returns the fittest ones among some t individuals picked at random, with replacement, from the population. The versions used for our algorithms are, *Multiobjective Lexicographic Tournament Selection (mlts)*, *Multiobjective Majority Tournament Selection (mmts)* and *Multiobjective Ratio Tournament Selection (mrts)*.

The genetic algorithms considered in this paper are basic Genetic Algorithm and Non-Dominated Sorting Genetic Algorithm. Among these basic GA is hybridized by a popular local improver, namely, hill-climbing. Basic GA little-by-little selects a few parents and generates children until enough children have been created. To breed, we begin with an empty population of children. We then select two parents from the original population, copy them, cross them over with one another, and mutate the results. This forms two children, which we then add to the child population. We repeat this process until the child population is entirely filled.

Note that, when a potential insertion point for a recognition site of a given enzyme is picked we may accidentally create a recognition site for another enzyme. Similar situation may happen when an enzyme with multiple occurrences are turned on and all but the selected locations are deleted. To avoid this undesirable effect, each crossover and mutation is accompanied/followed by a validate operation. In the validation operation, if the creation of restriction site for so far inserted enzyme is detected, we choose next possible insertion point from potential insertion list. If no such possibility can be found, the presence bit of corresponding restriction enzyme is turned off.

IV. Simulation Results

Our experiments were conducted on a computer having 3GHz Intel Pentium 4 processor with 1GB DDR3 Memory.

We have run our experiments on some viral sequences obtained from the website of the National Center for Biotechnology Information and a set of 145 restriction enzymes from the REBase database [5]. Each algorithm with different types of tournament selection strategy is compared with previous heuristics with respect to the three objectives by taking average of multiple runs. To analyze the results, a paired two-sample t-tests has been performed and P-values are presented to indicate the statistical significance of results.

For space constraints, we present our results in a condensed form. The convention followed here is as follows. For each objective, the difference between the objective values obtained from the proposed algorithm and the objective value obtained from an existing algorithm respectively is computed ($\Delta f_i, 1 \leq i \leq 3$). Here, our aim is to have higher f_1 and lower f_2 and f_3. Therefore, in Δf_1 column, positive mean value is preferred which means that the proposed algorithm can insert larger number of restriction sites. On the other hand, negative mean values are preferable for Δf_2 and Δf_3. In the Tables, we also present the confidence interval (C.I.) around the mean value using a 95% confidence level.

From the performance of Genetic Algorithm (GA) against the heuristics of [4] differing the level of local improver, we have found that even with multiobjective tournament selection, GA can not outperform others[3]. In each of the cases GA offers better sequence with respect to f_3 which is our prime concern. The other two objectives, f_1 and f_2 are inversely related to each other. Therefore, as we increase the number of insertions of restriction sites using the local improver, the performance with respect to f_1 increases whereas the same with respect to f_2 decreases. Due to this conflicting dependency, multi-objective tournament selection alone can not provide very high quality solutions. To get even better results we move to another approach (as discussed in the following section) where separate concentration on objectives are more prominent.

A. *Non-Dominated Sorting Genetic Algorithm (*NSGA*)*

Here, we apply the idea of *Pareto domination*: Individual A pareto dominates individual B if A is at least as good as B in every objective and better than B in at least one objective. All individuals who can not pareto dominate each other forms pareto front of same rank. Pareto front of lowest rank gives the best individuals. So the fitness of ith individual is, $fitness(i) = \frac{1}{1+ParetoFrontRank(i)}$. In Non-Dominated Sorting Genetic Algorithm (NSGA), we try to find which candidate solution pareto dominates other and thus extract only the solutions comprising the pareto front. Table II and III presents comparison of *NSGA* with the heuristics of [4] and different local search techniques of [6], namely, Hill Climbing (HC) and Steepest Ascent Hill Climbing with (SAHC) or without (SAHCwR) Replacement respectively.

[3]Results are not presented due to space constraint.

Table II
COMPARISON OF NON-DOMINATED SORTING GENETIC ALGORITHM WITH EXISTING HEURISTICS FOR PHAGE VIRUS

Heuristic	Mutation Type		Δf_1	Δf_2	Δf_3
Greedy	1	mean	9.615	-7.58	-126
		STD	6.038	11.13	77.3
		C.I.	4.319	7.96	55.3
		P Value	7E-04	0.06	6E-04
	2	mean	9.615	-6.38	-140
		STD	3.66	12.34	63.39
		C.I.	2.618	8.828	45.35
		P Value	2E-05	0.137	6E-05
Matching	1	mean	6.975	-11.5	-126
		STD	5.729	9.354	56.46
		C.I.	4.098	6.691	40.39
		P Value	0.004	0.004	6E-05
	2	mean	6.975	-10.3	-140
		STD	3.675	7.808	91.07
		C.I.	2.629	5.586	65.15
		P Value	2E-04	0.002	9E-04
Forward	1	mean	37.4	65.2	-222
		STD	6.535	7.052	217
		C.I.	4.675	5.045	155.3
		P Value	2E-08	3E-10	0.01
	2	mean	37.4	66.4	-236
		STD	4.169	7.792	256
		C.I.	2.982	5.574	183.2
		P Value	4E-10	6E-10	0.017
Backward	1	mean	37.5	65.9	-222
		STD	6.311	6.19	217
		C.I.	4.515	4.428	155.3
		P Value	2E-08	9E-11	0.01
	2	mean	37.5	67.1	-236
		STD	4.007	6.919	256
		C.I.	2.866	4.95	183.2
		P Value	3E-10	2E-10	0.017

Table III
COMPARISON OF NON-DOMINATED SORTING GENETIC ALGORITHM WITH LOCAL SEARCH TECHNIQUES FOR PHAGE VIRUS

Heuristic	Mutation Type		Δf_1	Δf_2	Δf_3
HC	1	mean	3.2	-19.9	-115
		STD	4.917	13.7	152.2
		C.I.	3.517	9.799	108.8
		P Value	0.07	0.001	0.041
	2	mean	5.7	-15.9	-101
		STD	4.448	14.47	134.2
		C.I.	3.182	10.35	96.02
		P Value	0.003	0.007	0.042
SAHC	1	mean	3.4	-21.8	-109
		STD	8.072	18.55	157.8
		C.I.	5.774	13.27	112.8
		P Value	0.216	0.005	0.056
	2	mean	8.1	-14	-112
		STD	3.143	13.09	197.3
		C.I.	2.248	9.364	141.1
		P Value	2E-05	0.008	0.107
SAHCwR	1	mean	3.7	-15.6	-106
		STD	5.774	17.35	61.5
		C.I.	4.131	12.41	43.99
		P Value	0.073	0.019	4E-04
	2	mean	9.6	-7.7	-107
		STD	5.91	11.43	105.4
		C.I.	4.228	8.178	75.42
		P Value	6E-04	0.062	0.011

Table IV
COMPARISON OF NON-DOMINATED SORTING GENETIC ALGORITHM WITH EXISTING HEURISTICS FOR POLIO VIRUS

Heuristic	Mutation Type		Δf_1	Δf_2	Δf_3
Greedy	1	mean	5.855	-10.2	-39.2
		STD	2.606	12.61	31.41
		C.I.	1.865	9.017	22.47
		P Value	6E-05	0.031	0.003
	2	mean	7.455	-9.58	-30.7
		STD	3.672	12.47	44.13
		C.I.	2.627	8.918	31.57
		P Value	1E-04	0.038	0.056
Matching	1	mean	6.335	-9.27	-37
		STD	3.589	8.162	32.98
		C.I.	2.568	5.839	23.59
		P Value	3E-04	0.006	0.006
	2	mean	7.935	-8.67	-28.5
		STD	3.898	12.63	46.26
		C.I.	2.788	9.035	33.09
		P Value	1E-04	0.058	0.083
Forward	1	mean	7.3	0.3	-47.2
		STD	3.129	9.19	84
		C.I.	2.238	6.574	60.09
		P Value	4E-05	0.92	0.109
	2	mean	8.9	0.9	-38.7
		STD	3.107	12.59	72.41
		C.I.	2.223	9.007	51.8
		P Value	8E-06	0.826	0.125
Backward	1	mean	4.2	0.2	-49
		STD	8.867	8.967	89.14
		C.I.	6.343	6.414	63.77
		P Value	0.168	0.945	0.116
	2	mean	5.8	0.8	-40.5
		STD	7.33	12.2	72.47
		C.I.	5.244	8.727	51.84
		P Value	0.034	0.84	0.111

Table V
COMPARISON OF NON-DOMINATED SORTING GENETIC ALGORITHM WITH LOCAL SEARCH TECHNIQUES FOR POLIO VIRUS

Heuristic	Mutation Type		Δf_1	Δf_2	Δf_3
HC	1	mean	-9	-30.8	-9.1
		STD	3.528	12.42	44.9
		C.I.	2.524	8.882	32.12
		P Value	2E-05	3E-05	0.538
	2	mean	-4.9	-26.3	-1.2
		STD	5.216	13.33	48.25
		C.I.	3.732	9.538	34.52
		P Value	0.016	2E-04	0.939
SAHC	1	mean	-7.8	-20.7	-9.4
		STD	4.341	24.15	35.56
		C.I.	3.105	17.27	25.44
		P Value	3E-04	0.024	0.425
	2	mean	-6.3	-26.5	-12.6
		STD	3.86	10.29	51.17
		C.I.	2.761	7.359	36.61
		P Value	6E-04	2E-05	0.456
SAHCwR	1	mean	-9.9	-27.7	-27.1
		STD	4.04	14.61	37.82
		C.I.	2.89	10.45	27.05
		P Value	3E-05	2E-04	0.05
	2	mean	-5.8	-25	-7
		STD	3.553	14.83	38.52
		C.I.	2.542	10.61	27.56
		P Value	6E-04	5E-04	0.58

Table II shows that NSGA gives us better synthesized sequence than *greedy* and *matching* heuristics requiring lower number of base changes. Also the average number of inserted enzymes is higher and maximum gap is smaller. For example, against greedy heuristic and mutation type 1, the mean difference of f_1 (mean=9.615, STD=6.038, P Value=7E-04) is significantly greater than zero. A 95% confidence interval (C.I.=4.319) around the mean of Δf_1 is [5.296, 13.934]. Both the lowest and highest values of Δf_1 are positive i.e. the Δf_1 remains positive irrespective of the value of the variance. Similar inference holds for f_2 and f_3 except for f_2, a small part of the 95% confidence interval [0.38, -15.54] falls in the positive range. However, NSGA, is still costlier (in terms of base changes) than *forward* and *backward* implementation, though it is superior to those in terms of f_1 and f_3. Table III shows that NSGA is less costlier than local search techniques keeping a very good maximum gap between the restriction sites. The base changes are less in number as it intelligently inserts a slightly less number of enzymes than others. Tables IV and V focus on similar comparisons considering another virus.

V. CONCLUSION AND FUTURE DIRECTIONS

We have applied variants of GA to place large number of restriction sites in a viral genome without changing it's functionality. With current representation, the order of genes for enzyme insertion is fixed. Defining an appropriate notion to select a good candidate solution among all candidates of same rank might be a possible future work. Another future research issue can be the analysis of the influence of using Strength Pareto Evolutionary Algorithms in URSPP.

REFERENCES

[1] Anand, I., Kosuri, S., Endy, D.: Genejax: A prototype cad tool in support of genome refactoring (2006).

[2] Evans, P., Liu, C.: Sitefind: A software tool for introducing a restriction site as a marker for successful site-directed mutagenesis. BMC Mol. Biol. 6(22), 2005.

[3] Luke, S.: Essentials of Metaheuristics. Genetic Programming and Evolvable Machines, 12(3), 333-334 (2011).

[4] Montes, P., Memelli, H., Ward, C. B., Kim J., Mitchell, J. S. B., Skiena, S.: Optimizing Restriction Site Placement for Synthetic Genomes. Combinatorial Pattern Matching, 6129, 323-337 (2010).

[5] Roberts, R., Vincze, T., Posfai, J., Macelis, D.: Rebase-a database for dna restriction and modification: enzymes, genes and genomes. Nucl. Acids Res. 38, D234-D236 (2010).

[6] Sharmin, M., Afrin, M., Rahman, M.,: Local Search Techniques for Placing Unique Restriction Sites in Synthetic Genomes. In the proceedings of BICoB 13-18 (2012).

[7] Skiena, S.: Designing better phages. Bioinformatics 17, S253-S261 (2001).