DNA Base-code Generation for Reliable Computing by Using Standard Multi-objective Evolutionary Algorithms

José M. Chaves González
Univ. Extremadura, Spain
Dept. Computers and Communications Technologies
Esc. Politécnica. Av. Universidad. 10003. Cáceres
jm@unex.es

Miguel A. Vega-Rodríguez
Univ. Extremadura, Spain
Dept. Computers and Communications Technologies
Esc. Politécnica. Av. Universidad. 10003. Cáceres
mavega@unex.es

ABSTRACT

Artificially generated DNA strands have to meet several complex bio-chemical constraints when they are used to solve any computational problem. In this context, DNA sequences have to satisfy several design criteria to prevent DNA strands from producing undesirable reactions which usually lead to incorrect computations. This study is focused on six different design criteria that ensure the reliability and efficiency of the operations performed with the generated DNA sequences. We have formulated DNA base-code generation as a multiobjective optimization problem in which there is not only a unique optimal solution, but a Pareto set of high-quality solutions. Reliable DNA sequences have been generated by using two well-known multiobjective approaches: the Non-dominated Sorting Genetic Algorithm II (NSGA-II) and the Strength Pareto Evolutionary Algorithm 2 (SPEA 2). We have performed experiments with three different-sized realistic datasets. Results show that the multiobjective algorithms developed are very appropriate for our problem, especially NSGA-II, which provides more reliable DNA sequences than other relevant approaches previously published in the literature.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization – *Global optimization*

General Terms

Algorithms

Keywords

DNA Sequence Design, Multiobjective Optimization, DNA Computing, Evolutionary Algorithms, NSGA-II, SPEA2

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.

1. INTRODUCTION

Deoxyribonucleic acid (DNA) computing refers to a computational model proposed by Adleman in 1994 [1]. This model uses DNA molecules as information storage and their biological reactions as processing operators. The hybridization between a concrete DNA sequence and its base-pairing complement is crucial for DNA computing, but undesirable hybridizations lead to incorrect computations, so DNA sequences that are supposed to be applied for reliable molecular computing have to be carefully designed [2]. The design of reliable sequences which generate specific duplexes during hybridization, while simultaneously avoiding other undesirable reactions, involves several conflicting design criteria which cannot be managed by traditional optimization techniques. Typical existing methods include non-exact approaches such as evolutionary algorithms, dynamic programming, and heuristic approaches [2]. However, a design based on multi-objective evolutionary algorithms (MOEAs) represents a more suitable design alternative [3] because these approaches effectively consider different objectives without the artificial adjustments which are needed in classical mono-objective methods when they are used to solve multiobjective problems. In this paper, we have modeled the problem by considering six different conflicting design criteria. These biochemical criteria are used to generate reliable DNA sequences by using adaptations of two well-known multi-objective algorithms: NSGA-II (fast non dominated sorting genetic algorithm [4]) and SPEA2 (strength Pareto evolutionary algorithm [5]). The developed proposals are compared against other relevant approaches, and as will be shown, we obtain more reliable sequences than other methods previously published.

The rest of this paper is organized as follows: Section 2 discusses a brief related work. Section 3 describes the problem and the multiobjective formulation we have followed. The metaheuristics developed are explained in Section 4. Section 5 is devoted to analyze the experiments carried out and also to compare our best approach with other methods published in the literature. Finally, Section 6 summarizes the conclusions of the paper and discusses possible lines of future work.

2. RELATED WORK

Most of works published in the literature related to the design of DNA sequences for DNA computing consider the problem in terms of threshold-based constraints. In those works, sequences are designed by considering one or more biological restrictions that every sequence has to fulfill according to a set of thresholds. Exhaustive and random searches are the simplest methods [6], [7],

but they are not very effective techniques because these searches use too much computational resources. Other strategies use templates [8], [9], [10], or are based on the graph theory [11], or apply classical evolutionary techniques such as simulated annealing [12]. On the other hand, biologically inspired methods have been recently used to design sequences that can be used in DNA computing. Thus, in [13], an approach based on in vitro evolution is used to find non-cross hybridizing DNA libraries. In [14], [15], authors consider thermodynamic properties of DNA structures and DNA free energies. However, biologically inspired algorithms have inherent difficulties, i.e., they cannot distinguish each DNA sequence in the library. On the other hand, according to the vast number of works published in the last years [16]-[31], we have to admit that evolutionary algorithms (EAs) have to be considered the most widely used methods for designing reliable sequences. These methods manage one or more bio-chemical design criteria by using different evolutionary approaches. In particular, genetic algorithms seem to be one of the most extended algorithms possibly due to their simplicity [16]-[21], but there are also other evolutionary approaches which have been applied to the problem. Thus, studies based on swarm intelligence obtain interesting results, such as the works in [22]-[25], which generate DNA sequences by using particle swarm optimization; or the works in [26]-[28], which use ant colony optimization for the same purpose. However, although those studies consider multiple design criteria, the proposed metaheuristics cannot be considered as multi-objective EAs, because in all those cases the problem is simplified and converted into a single-objective optimization problem by using a constrained weighted summation method. Only in [29], [30], [31], we find real multiobjective proposals. Results published in [29], [31] are improved in [30] by using the algorithm NACST/Seq. Sequences in that study represent the best results published so far, so we will compare our best results against the results provided by NACST/Seq. [30] in Section 5.

3. DNA BASE-CODE GENERATION

DNA base-code generation is crucial in many bio-molecular technologies, such as DNA computing, DNA sequencing or nanotechnology. In all those technologies, the most important process to obtain the information stored in DNA sequences is the hybridization created by a DNA sequence and its basepairing complement. However, undesirable hybridizations between sequences have to be avoided by properly designing of DNA sequences, because these illegal reactions cause potential errors that can be produced during the biological reaction. Therefore, interactions between artificially generated sequences have to be strictly controlled during the design stage. To fulfill this purpose, every DNA design criteria have to contribute to improving the reliability of the generated sequences. DNA base-code design criteria can be divided into four categories [30] according to their biological meaning: 1) avoiding inconvenient reactions; 2) controlling secondary structures; 3) controlling the biochemical characteristics of DNA sequences; and 4) restricting the composition of DNA sequences.

The first category consists of generating sequences that are only allowed to form a duplex with its complement. The two most important bio-chemical design criteria for this restriction are: *similarity*, which calculates the inverse Hamming distance between two sequences; and *h-measure*, which tests the possibility of unintended DNA basepairing. Regarding to the second category, secondary structures are usually formed by the interaction of single stranded DNA. These structures include internal loop, bulge loop and hairpin. *Hairpin* penalty and *continuity* are the most extended criteria to predict secondary structures. Continuity criterion counts the repeated run of identical bases, and the hairpin penalty is used to avoid the formation of hairpin structures. Regarding to the control of bio-chemical features, the third category, the free energy and the *melting temperature* are the more reliable measures. The first measure indicates the required energy for breaking a duplex, and the second is defined as the temperature at which half of the DNA strands are in the double-helical state and half are in the random coil state [32]. *GC ratio* is a less precise measure, but it is very easy to calculate, so it is widely used. It indicates the percentage of guanine (G) or cytosine (C) in a DNA base-code. Finally, in some occasions, the composition of DNA sequences has to be restricted for some special purpose, such as to create a DNA subsequence for controlling proper reactions. This criterion is used to control the occurrence of specific subsequences in the designed sequences.

3.1 Mathematical formulation

After studying related publications with the work tackled in our paper, we have chosen six different conflicting design criteria that have to be optimized simultaneously with the aim of generating reliable DNA sequences that can be used for computation [30]. These bio-chemical criteria are: similarity, h-measure, continuity, hairpin, GC ratio and melting temperature. Each criterion is managed as an objective that has to be minimized. Formally, DNA base-code generation can be described as follows.

$$\text{Minimize } y = F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \tag{1}$$

where $f_i(x)$ are the objectives previously mentioned. A formal definition for each objective is provided below.

1) Similarity: This objective computes the similarity in the same direction of two given DNA sequences to keep each sequence as unique as possible. For a more complete comparison, we include position shifts and the target sequence is extended by adding its own sequence to the 3'-end with gaps. Moreover, we consider continuous (s_{cont}) and discontinuous (s_{disc}) similarities. The mathematical definition for this measure is described in (2).

$$f_{similarity}(x,y) = Max_{g,i}(s_{disc}(x, shift(y,g,i)) + s_{cont}(x, shift(y,g,i))) \tag{2}$$

Figure 1. Example of similarity calculation

where x and y are parallel sequences and *shift* indicates a shift of sequence y by i bases and g gaps. s_{disc} is a real value between 0 and 1, and s_{cont} is an integer between 1 and the length of the sequences. Finally, we have to indicate that similarities have to surpass a threshold that has to be established by experimentation to be considered. A simple example of this measure for a couple of sequences, without considering position gaps and shifts, can be observed in Figure 1, in which the continuous similarity is equal to 4 bases and the discontinuous similarity is equal to 6 bases.

2) H-measure: This criterion is similar to similarity, but instead of considering sequences in the same direction (from 5' to 3' end), they are managed as complementary, in opposite directions.

H-measure prevents cross hybridization between DNA strands. We consider elongated sequences with gaps for a more reliable measure. The mathematical definition is given in (3).

$$f_{h\_measure}(x,y) = Max_{g,i}(h_{disc}(x, shift(y,g,i)) + \\ + h_{cont}(x, shift(y,g,i))) \quad (3)$$

where x and y are anti-parallel sequences and *shift* indicates a shift of sequence y by i bases and g gaps. h_{disc}, h_{cont} and the threshold have analogous values to the similarity measure. An illustrative example of the h-measure calculation for two sequences in which a shift of 4 bases is considered is provided in Figure 2. In this case, the continuous h-measure is equal to 3 bases, and the discontinuous is equal to 4 bases.

Figure 2. Example of h-measure calculation

3) Continuity: This measure calculates the degree of successive occurrences of the same base in a sequence. The criterion prohibits consecutive runs of the same base over a given threshold. For example, if the threshold is 3, in the sequence AGGCAATAAAACGAAATGGGC, the third subsequence of adenines (A) violates the continuity and the others not. The mathematical definition for this measure is given in (4).

$$f_{continuity}(x) = \sum_{i=1}^{max} \sum_{a=>\{A,C,G,T\}} T(c_a(x,i),t)^2 \quad (4)$$

where x is the sequence under study, *max* is the difference between the length of the sequence and the threshold (T), $c_a(x,i)$ is equal to ε if ∃ ε s.t. $x_i \neq a$, x_i+1=a for $1 \leq j \leq \varepsilon$, $x_{i+\varepsilon+1} \neq a$, and 0 otherwise. Continuity for all bases is controlled in every sequence.

4) Hairpin: This measure calculates the penalty for secondary structures formation. For simplicity, it is calculated through the Hamming distance by considering the length of hairpin loop and the number of hybridized pairs. It is assumed that a hairpin has at least $R_{\min}$ bases as a loop and a minimum of $P_{\min}$ base pairs as a stem. A simple illustrative example can be observed in Figure 3. For a more reliable measure, it is also considered the penalty for formation of hairpins of various sizes at every position in the sequence. In (5) are considered hairpins with r-base loop and p-base pairs stem to be formed at position i in the sequence x, if more than half bases in the subsequence $x_{i-p}\ldots x_i$ hybridize to the subsequence $x_{i+r}\ldots x_{i+r+p}$. The number of matches in these subsequences is defined as the penalty for this hairpin.

$$f_{hairpin}(x) = \sum_{p=P\min}^{\max I} \sum_{r=R\min}^{\max R} \sum_{i=1}^{\max I} T(hp(x,p,r,i), \frac{pinlen(p,r,i)}{2}) \\ hp(x,p,r,i) = \sum_{j=1}^{pinlen(p,r,i)} bp(x_{p+i+j}, x_{p+i+r+j}) \quad (5)$$

where the function $pinlen$ $(p,r,i) = min(p+i, l-r-i-p)$ and denotes the maximum number of possible basepairs when a hairpin is formed at center $p+i+r/2$.

5) GC content: This criterion indicates the percentage of bases C and G in the sequence x. This is important because the GC content affects to the chemical properties of DNA sequences. GC base pairs are more stable than AT base pairs because the GC pair is bound by three hydrogen bonds, while AT pairs are bound by two hydrogen bonds (see Figure 4). This fact makes high-GC-content DNA structures more tolerant of high temperatures. For example, if we consider the sequence ATGATAGGCGTTGTA, the GC% is 40 (6 out of 15).

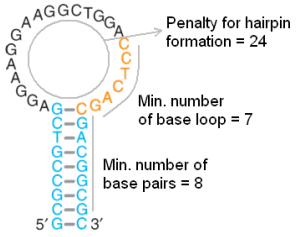

Figure 3. Example of hairpin penalty calculation

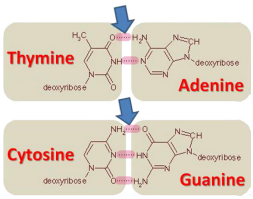

Figure 4. TA and CG base pairs

6) Melting temperature, T_m: This criterion predicts DNA thermal denaturation, which is a key factor for DNA computing. It is desirable that DNA sequences have similar chemical features for successful operations. Both sequence and base composition are important determinants of DNA duplex stability. DNA melting is the process by which double-stranded DNA unwinds and separates into single-stranded strands (breaking of hydrogen bonding between the bases. There are many ways to calculate this relevant feature, but we use the nearest neighbour (NN) model in this approach [32]. The mathematical description for this measure, calculated by using the NN method, is provided in (6).

$$Tm(x) = \Delta H^{\circ} / \Delta S^{\circ} + R \ln(|C_T|/4) \quad (6)$$

where x is the DNA sequence studied, R is a gas constant and $|C_T|$ is the total sequence concentration. ΔH° and ΔS° refer to predicted enthalpies and entropies. Those values were taken from [32].

3.2 Multiobjective definition

DNA sequence design problem can be naturally formulated as a constrained multiobjective evolutionary algorithm (MOEA), because reliability of sequences depends on several design criteria that have to be equally considered. Moreover, two of the previously described bio-chemical criteria (melting temperature and GC ratio) can be regarded as constraints that every sequence have to fulfill to be considered a valid sequence. Therefore, equation (1) is finally specified in the following terms:

$$\begin{aligned} &\text{Minimize } f_i(x), \text{ where} \\ &\quad i \in \{\text{similarity, h-measure, continuity, hairpin}\} \\ &\text{subject to } g_j(x) = 0, \text{ where } j \in \{T_m, \text{GC ratio}\} \end{aligned} \quad (7)$$

4. MULTIOBJECTIVE APPROACHES

We have generated reliable DNA base-codes which are suitable for DNA computing by using the bio-chemical criteria

explained in the previous section. These criteria are included in the fitness functions of the two MOEAs we have developed: NSGA-II [4] and SPEA2 [5]. Both metaheuristics ranks the solutions by using the concept of dominance. To this respect, in a multi-objective context, one solution dominates another if its objectives have values as good as the objectives of the other solution but at least one objective presents better results [4]. A very important part in the design of any metaheuristics is the individual representation. Figure 5 shows the data structure used to represent the solutions provided by our MOEAs. Each individual includes the necessary information to generate valid set of sequences that can be applied for DNA computing. Moreover, as our study is based on different instances, the length and the number of sequences included in the individual are not constant. The individual is formed by a set of *n* sequences. Each sequence includes the DNA strand used for the genetic operators to make the solution evolves in each iteration, and the values for each biochemical criterion taken in consideration (similarity, h-measure, continuity, hairpin, GC content and melting temperature). Moreover, DNA strands are composed of *m* nucleotides (or bases) each. These nucleotides can be: adenine (A), thymine (T), cytosine (C) and guanine (G). Finally, the quality of each solution is calculated through the average values for the four objectives considered (according to equation 7) for the *n* sequences.

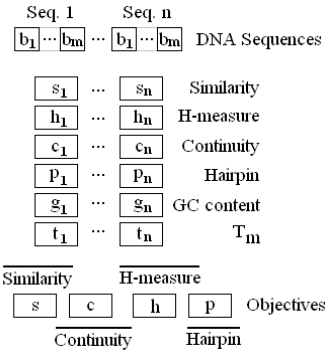

Figure 5. Data structure for the individual

4.1 NSGA-II for DNA base-code generation

The fast non dominated sorting genetic algorithm (NSGA-II, Algorithm 1) is a classical multiobjective approach created by Deb *et al*. [4] that tries to improve the current population, *P*, with an offspring, *Q*, by applying typical genetic operators such as tournament selection (lines 5-6), binary crossover (*Cr*, line 7) and random mutation (*f*, line 8). Crossover operator works in two levels. In the first level, our algorithm operates by using complete sequences of the individual. Thus, a descendant is generated with complete sequences of the two fathers selected in the selection stage. Then, the second level operates with a specific sequence, combining in the same sequence pieces of sequences from the two fathers selected. At the end of the crossover stage each sequence of a descendant can be either a complete sequence from one of the fathers or a new sequence formed with part of the sequences of the two fathers which are involved in the operation. On the other hand, mutation operator works for each sequence by exchanging a random base with another base which is also chosen randomly.

Thus, in our version of the algorithm, we create the descendant population, *Q* (with *PSize* individuals), by using the parent population *P* and the genetic operators described above. Then, the algorithm combines the original population, *P,* and the descendant population, *Q*, to create a new population, *R,* with the size of 2×*PSize* individuals (line 10). Next, a non-dominated sorting function is used to classify the population *R* in different Pareto fronts (line 11). The new population is generated from these fronts (first with F_1, then with F_2, and so on, lines 14-18). As the population size of *R* is 2×*PSize* and there have to be only *PSize* solutions in the final descendant population, not all elements in *R* will be in the new population. We apply elitism in our algorithm, so those fronts that do not fit in the new population will be removed. If there are solutions that belong to the last front and they cannot be added to the new population, the algorithm will choose the remaining by using the crowding distance [4] (lines 19-20). NSGA-II has become very popular in the last years, such that it has become a reference algorithm in the multiobjective field. This is the reason because we selected it to solve our multiobjective problem.

Algorithm 1 Pseudocode for our implementation of NSGA-II

```
 1: P ⇐ generateParentPopulation (PSize)
 2: while not stop condition satisfied do
 3:    /* Generating the offspring population, Q */
 4:    for i=1 to PSize do
 5:       ind1 ⇐ tournamentSelection (P)
 6:       ind2 ⇐ tournamentSelection (P)   // ind1≠ind2
 7:       Q_i ⇐ recombination (ind1, ind2, Cr)
 8:       Q_i ⇐ mutation (Q_i, f)
 9:    end for
10:    R ⇐ mergeSets (P, Q)
11:    R ⇐ fastNonDominatedSorting (R)   // R = (F_1, F_2, …)
12:    P ⇐ ∅
13:    i = 1
14:    while | P | + | F_i | ≤ PSize do
15:       crowdingDistanceAssignment (F_i)
16:       P ⇐ mergeSets (P, F_i)
17:       i = i + 1
18:    end while
19:    sortByCrowdingDistance (F_i)
20:    P ⇐ mergeSets (P, F_i [ 1 : (PSize – | P | ) ] )
21: end while
```

Algorithm 2 Pseudocode for our implementation of SPEA2

```
 1: P ⇐ generateRandomPopulation (PSize)
 2: A ⇐ ∅   //Archive (ArchiveSize)
 3: while not stop condition satisfied do
 4:    FitnessAssignment (P, A)
 5:    //EnviromentalSelection (A, P). Truncate A if necessary
 6:    A ⇐ NonDominatedIndividuals (A, P)
 7:    if | A | > ArchiveSize then
 8:       A ⇐ TruncateArchive (A)
 9:    endif
10:    //Generating new population
11:    for i=1 to PSize do
12:       ind1 ⇐ tournamentSelection (A)
13:       ind2 ⇐ tournamentSelection (A)   // ind1≠ind2
14:       P_i ⇐ recombination (ind1, ind2, Cr)
15:       P_i ⇐ mutation (P_i, f)
16:    end for
17: end while
```

4.2 SPEA2 for DNA base-code generation

The strength Pareto evolutionary algorithm 2 (SPEA2) is a population-based multiobjective algorithm created by Zitzler *et al*. in [5]. Algorithm 2 shows the pseudocode of our version of the

algorithm. A fitness value that is the sum of its strength raw fitness plus a density estimation is assigned to each individual (line 4). Then, the best solutions (non-dominated ones) of both, population and archive, are copied into a new population, truncating it with the aim of not exceeding the size of the population (lines 5-9). Moreover, SPEA2 uses also classical genetic operators, such as tournament selection (lines 12-13), binary crossover (line 13) and random mutation (line 14) to generate the next population at the end of every generation. Genetic operators used in our version of SPEA2 are the same of the ones used in our version of NSGA-II.

5. EXPERIMENTAL RESULTS

In this section, we describe the methodology followed, we present the results obtained with the algorithms developed and we compare the results obtained by the approach which obtains better results with other important results published in the literature.

Table 1. Parametric configuration, datasets and hypervolume reference points

NSGA-II configuration	
Crossover factor (*Cr*)	0.1
Mutation probability (*f*)	0.5
Parent selection	Binary tournament
SPEA2 configuration	
Archive size (*ArchiveSize*)	*PSize*/2
Crossover factor (*Cr*)	0.3
Mutation probability (*f*)	0.5
Parent selection	Binary tournament
Number of sequences and nucleotides per sequence for each data set used	
Sequences proposed in [33]	
Number of seqs. (nucleotides per seq.)	7 (20)
Normalization reference points	Min = (0, 0, 0, 0)
	Max = (35, 10, 80, 110)
Sequences proposed in [34]	
Number of seqs. (nucleotides per seq.)	14 (20)
Normalization reference points	Min = (0, 0, 30, 80)
	Max = (30, 5, 150, 175)
Sequences proposed in [35]	
Number of seqs. (nucleotides per seq.)	20 (15)
Normalization reference points	Min = (0, 0, 50, 90)
	Max = (15, 5, 140, 190)

5.1 Methodology

In order to obtain reliable conclusions, we have performed our experiments for the two algorithms developed with three different DNA datasets used for bio-molecular computing. The value of each parameter has been fixed after executing 30 independent runs to ensure statistical significance. All experiments were performed by using a Pentium 4 (2.8 GHz) with 1GB of RAM. The algorithm was compiled using *gcc* 4.4.5 compiler. In Table 1, we show the summary of the parametric configuration and details of the data sets included in our study as well as the normalization points used to calculate hypervolume indicator. Moreover, to perform a fair comparison with other authors [30], population sizes (*PSize* in Algorithms 1-2) and the stop condition for both MOEAs were taken from the studies we use for comparison: 3000 individuals and 200 iterations respectively [30]. The three datasets used in our study were proposed by different authors (Deaton *et al.* [33], Tanaka *et al.* [34], and Faulhammer *et al.* [35]) and they have been successfully applied to reliable DNA computing. Moreover, as our study is immersed in a multi-objective environment, the value for each parameter was established according to the quality of the Pareto front produced for each parameter. We have used the *hypervolume* (*HV*) indicator. This metrics evaluates the volume (in the objective function space) covered by members of a non-dominated set of solutions. Two reference points are required to calculate *HV*: r_{min} ($obj1_{min}$, $obj2_{min}$, $obj3_{min}$, $obj4_{min}$) and r_{max} ($obj1_{max}$, $obj2_{max}$, $obj3_{max}$, $obj4_{max}$), where each element of those points are the four objectives considered in our problem (similarity, h-measure, hairpin and continuity). Note that *HV* is not free from arbitrary scaling of objectives, that is to say, the value of this metrics may be distorted if the range of each objective function is different. Thus, before calculating this indicator, the objective function values have to be normalized. Normalization points for each DNA sequence set generated in our study are presented in Table 1. Therefore, after normalization, the ideal reference points are r_{min} = (0, 0, 0, 0) and r_{max} = (1, 1, 1, 1) for all datasets. Note that normalization points will be different for each dataset used.

5.2 NSGA-II and SPEA2 results

Multiobjective results provided by the algorithm developed are shown in Table 2. These results are given in terms of the average hypervolume calculated using data obtained from 30 independent runs for the three sets specified in Table 1. Moreover, in Table 2 we also indicate that our final results present differences which are statistically relevant, considering a confidence level of 95%, according to the ANOVA test. If we focus on the *HV* values provided in Table 2, we can notice that NSGA-II obtains better results than SPEA2 for all datasets (the smaller, the medium and the larger instance). Moreover, standard deviations are minimal, so we can estate that our results are statistically reliable.

Table 2. Comparison between NSGA-II and SPEA2 results by using the average hypervolume of 30 independent runs

Dataset's authors Number of sequences (bases per sequence)	NSGA-II Mean±Std. dev.	SPEA2 Mean±Std. dev.	Statistically significant differences
Deaton et al. 7(20)	49.815%±0.007	41.872%±0.011	√
Tanaka et al. 14(20)	50.882%±0.021	44.321%±0.016	√
Faulhammer et al. 20(15)	53.512%±0.038	48.124%±0.023	√

We can conclude that our implementation of NSGA-II is able to explore the search space better than our version of SPEA2, and consequently, the sets of sequences generated with that MOEA will be of more quality attending to the four objectives considered (similarity, h-measure, hairpin and continuity). This means that sequences generated with NSGA-II will be more reliable for DNA computing than sequences generated with SPEA2.

5.3 Comparative results

In this section, the libraries of DNA sequences obtained by the best algorithm, NSGA-II, are carefully analyzed by comparing our results, in terms of the biochemical criteria under study, with the results provided in other relevant studies published in the literature. Our study is supported by reliable instances previously used for DNA computing. As described in Table 1, reference sequences were taken from Deaton *et al*. [33], Tanaka *et al*. [34], and Faulhammer *et al*. [35]. The best multiobjective study for the problem tackled is the work by Shin *et al.* [30], so our results are compared with that study (which uses the same data sets). However, the comparison is made by considering each objective separately, because unfortunately, neither in that study nor in other studies, multiobjective indicators (e.g. *hypervolume*), have been taken into account so far. Therefore, for each data set the quality of each design criterion is examined for a set of sequences

taken from the optimal Pareto front. Moreover, with the aim of performing a fair comparison with other studies, the same parametric adjustment for the biochemical constraints was established. Therefore, for *similarity* and *h-measure*, it was established lower limits for the continuous case equal to six nucleotides and 17% for the discontinuous case. For *continuity*, the threshold value was 2. *Hairpin* formation requires at least six basepairings and a six base loop. The *melting temperature* (Tm) was obtained by using the nearest neighbour model with 10nM DNA concentration and 1 M salt concentration. Finally, it is important to note that every value was decided empirically with biochemical background [30].

Such as was explained in [30], GC ratio and melting temperature were considered constraints for the problem with the same limits as the established in the literature. Thus, in [33] (Table 3) and [34] (Table 4) sequences have the GC ratio restricted to 50% and the melting temperature restricted between 46 and 53 degrees. On the other hand, in [35] (Table 5), the range of the GC ratio was established between 40% and 50% and the melting temperature between 31 and 39 degrees. Shin *et al.* [30] uses the same restrictions. The comparison between our results and those studies are given in Tables 3-5 and in Figures 6-8.

Table 3. Comparison of the DNA libraries generated in our study and other relevant studies for the 7(20) dataset

Seq. (5'→ 3')	C	P	H	S	Tm	GC
Sequences obtained in [33]						
ATAGAGTGGATAGTTCTGGG	9	3	55	64	52.6522	45
CATTGGCGGCGCGTAGGCTT	0	0	69	51	69.2009	65
CTTGTGACCGCTTCTGGGGA	16	0	60	63	60.8563	60
GAAAAAGGACCAAAAGAGAG	41	0	58	45	52.7111	40
GATGGTGCTTAGAGAAGTGG	0	0	58	54	55.3056	50
TGTATCTCGTTTTAACATCC	16	4	61	50	48.4451	35
TTGTAAGCCTACTGCGTGAC	0	3	75	55	56.7055	50
Sequences obtained in [30]						
CTCTTCATCCACCTCTTCTC	0	0	43	58	46.6803	50
CTCTCATCTCTCCGTTCTTC	0	0	37	58	46.9393	50
TATCCTGTGGTGTCCTTCCT	0	0	45	57	49.1066	50
ATTCTGTTCCGTTGCGTGTC	0	0	52	56	51.1380	50
TCTCTTACGTTGGTTGGCTG	0	0	51	53	49.9252	50
GTATTCCAAGCGTCCGTGTT	0	0	55	49	50.7224	50
AAACCTCCACCAACACACCA	9	0	55	43	51.4735	50
Sequences obtained with NSGA-II						
TCTTCTCTTCTCTCTCGGTC	0	0	63	34	47.506	50
CACACACACACACATACACC	0	0	37	54	48.902	50
AGAGAGGCGAGATGGAGAAA	0	0	48	47	50.044	50
AAGAAGGAAGCAGAGGCAGA	0	0	45	49	50.876	50
CAACCACACAACCACCACAA	0	0	34	58	51.205	50
CAACACACCACCAACCAACA	0	0	36	56	51.205	50
ATTAGGAGTTGAGGTTGGGG	0	0	57	35	48.734	50

In [33], Deaton *et al.* proposed a genetic algorithm to design good sequences for Adleman's graph. Shin *et al*., in [30], proposed NACST/Seq approach to improve those sequences. Results given in Table 3 and Figure 6 show that NSGA-II algorithm obtains DNA sequences with lower similarity (S) and h-measure (H) values, while providing minimal values for hairpin (P) and continuity (C). Every objective is to be minimized, as indicated in equation (7). This means that sequences obtained by our implementation have higher probability to hybridize with its correct complementary sequences. Furthermore, secondary structures are virtually prohibited because values for hairpin and continuity are reduced to zero. Moreover, ranges for melting temperature and GC ratio are also better, which means more stable sequences.

Results obtained in [34] generated sequences by using simulated annealing. In this case, the instance includes 20 DNA sequences of 20-mer oligonucleotides. NSGA-II is able to obtain better sequences by considering all restrictions (Table 4 and Figure 7). As is observed, continuity and hairpin values are again reduced to the minimal expression, avoiding the occurrence of secondary structures, while h-measure and similarity are also lower than in [34] and in [30].

Table 4. Comparison of the DNA libraries generated in our study and other relevant studies for the 14(20) dataset

Seq. (5'→ 3')	C	P	H	S	Tm	GC%
Sequences obtained in [34]						
CGAGACATCGTGCATATCGT	0	7	143	124	59.6965	50
TATAGCACGAGTGCGCGTAT	0	3	137	130	62.1165	50
GATCTACGATCATGAGAGCG	0	4	135	126	56.1049	50
TCTGTACTGCTGACTCGAGT	0	9	163	124	56.5723	50
CGAGTAGTCACACGATGAGA	0	0	152	132	56.2894	50
AGATGATCAGCAGCGACACT	0	3	133	133	58.9724	50
TGTGCTCGTCTCTGCATACT	0	10	159	130	58.5736	50
AGACGAGTCGTACAGTACAG	0	0	152	134	54.9689	50
ATGTACGTGAGATGCAGCAG	0	0	139	121	57.8232	50
ATCACTACTCGCTCGTCACT	0	3	141	132	58.0122	50
TCAGAGATACTCACGTCACG	0	3	142	123	55.3226	50
GACAGAGCTATCAGCTACTG	0	3	129	124	54.565	50
GCTGACATAGAGTGCGATAC	0	0	130	133	56.5849	50
ACATCGACACTACTACGCAC	0	3	133	144	57.2186	50
Sequences obtained in [30]						
GTGACTTGAGGTAGGTAGGA	0	3	129	115	47.2490	50
ATCATACTCCGGAGACTACC	0	3	132	121	47.2304	50
CACGTCCTACTACCTTCAAC	0	0	128	121	47.4589	50
ACACGCGTGCATATAGGCAA	0	3	141	117	52.5401	50
AAGTCTGCACGGATTCCTGA	0	3	132	115	50.5497	50
AGGCCGAAGTTGACGTAAGA	0	0	132	116	51.0482	50
CGACACTTGTAGCACACCTT	0	0	132	123	50.2683	50
TGGCGCTCTACCGTTGAATT	0	0	135	116	52.0565	50
CTAGAAGGATAGGCGATACG	0	0	134	117	46.6253	50
CTTGGTGCGTTCTGTGTACA	0	0	140	116	50.5774	50
TGCCAACGGTCTCAACATGA	0	0	132	121	51.8587	50
TTATCTCCATAGCTCCAGGC	0	0	136	117	48.1017	50
TGAACGAGCATCACCAACTC	0	0	121	121	50.3351	50
CTAGATTAGCGGCCATAACC	0	0	127	119	47.6383	50
NSGA-II						
ACCTCGTCATCTACTTACCC	0	0	135	89	47.882	50
GGAAGGAGAAGGCCACATAA	0	0	113	110	49.083	50
CACCATAGCGCACACAACAA	0	0	122	97	51.396	50
GCAGACAACGGACGACAAAA	0	0	121	104	51.284	50
GGTGGTGTGTATGAGGTTGT	0	0	103	122	49.568	50
TCAACCGCCTTATTCTCCCT	0	0	128	96	50.664	50
TGTTGGTTGGTTGTGGTTGG	0	0	91	114	51.236	50
GTTGGCTTAGTGTCGTGCTT	0	0	114	109	50.775	50
TCTGCTCTGTTCTCCGTCTT	0	0	121	94	50.518	50
AGGCATTATAGGTGGTGGGA	0	0	104	128	50.13	50
AAGGAACGATAGAAGCGGGA	0	0	110	115	50.674	50
TCGCGTGTGTGGTTGATTGT	0	0	110	115	52.851	50
GGAGATGTAGAAGTGTAGGG	0	0	101	118	46.045	50
GAAGAGGAGGAGGAAGAAGA	0	0	97	109	47.455	50

Finally, in [35] sequences for the chess knight movement problem are presented. In that work, sequences with 20 15-mer were designed and then improved by sequences generated in [30] by using NACST/Seq. As in the previous cases, DNA strands generated by NSGA-II are more reliable for DNA computing

because they are more dissimilar and they avoid secondary structures. Comparative results are shown in Table 5 and Figure 8.

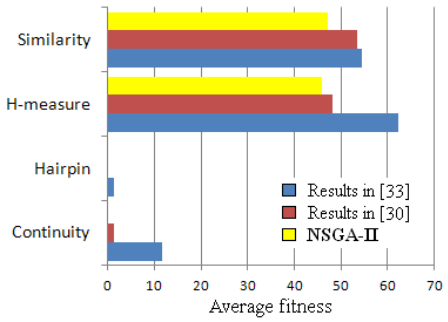

Figure 6. Average fitness results for our implementation of NSGA-II and other studies for the 7(20) dataset.

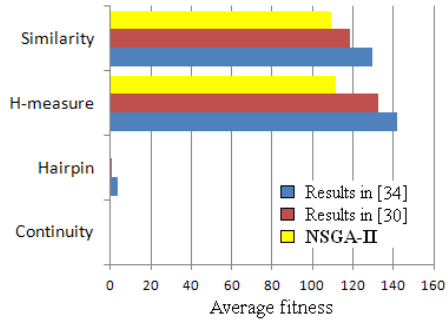

Figure 7. Average fitness results for our implementation of NSGA-II and other studies for the 14(20) dataset.

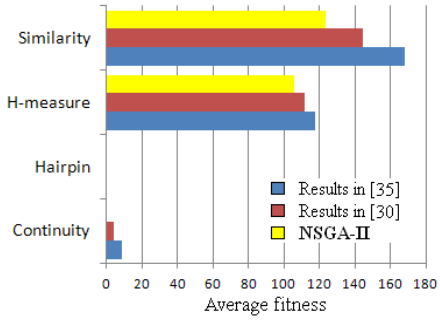

Figure 8. Average fitness results for our implementation of NSGA-II and other studies for the 20(15) dataset.

6. CONCLUSIONS AND FUTURE WORK

In this work, we present two MOEAs for the design of DNA sequences that can be applied to reliable molecular computing. The best approach, NSGA-II, obtains high-quality DNA libraries which simultaneously minimize similarity, h-measure, hairpin and continuity of each generated DNA strand in the set. Solutions are also restricted for a specific GC ratio and a certain melting temperature. We have used three different real-world instances proposed by three different authors to ensure the effectiveness of our study. These data sets include different number of sequences, number of bases per sequence and bio-chemical restrictions, and all of them have been successfully used for reliable DNA computing. After our study, we can conclude that our version of NSGA-II can generate better sequences (taking in consideration all the objectives and restrictions) than other approaches previously published in the literature.

Table 5. Comparison of the DNA libraries generated in our study and other relevant studies for the 20(15) dataset

Seq. (5'→ 3')	C	P	H	S	Tm	GC%
Sequences obtained in [35]						
CTCTTACTCAATTCT	0	0	114	162	36.1609	33.33
CATATCAACATCTTA	0	0	130	175	35.9744	26.67
ATCCTCCACTTCACA	0	0	107	187	41.6455	46.67
TTAAAATCTTCCCTC	25	0	121	159	35.4845	33.33
CTATTTATCCACACC	9	0	109	175	36.6125	40
GCTTCAAACAATTCC	9	0	117	154	41.2682	40
AACTCTCAAATTCAA	9	0	132	158	38.1608	26.67
CTAACCTTTACTTCA	9	0	120	178	35.2144	33.33
CATTCCTTATCCCAC	9	0	100	178	38.4742	46.67
CACCCTTTCTCCTCT	18	0	88	159	41.0117	53.33
TCCTCACATTACTTA	0	0	119	172	35.9641	33.33
ACTTCCTTTATATCC	9	0	116	168	33.0108	33.33
TTATAACAAACATCC	9	0	131	157	35.813	26.67
ACATAACCCTCTTCA	9	0	116	179	39.5534	40
ACCTTACTTTCCATA	9	0	118	174	34.8949	33.33
GTACATTCTCCCTAC	9	0	114	155	38.5983	46.67
CATAATCTTATATTC	0	0	131	175	30.7646	20
ATAATCACATACTTC	0	0	125	172	34.713	26.67
TCCACCAACTACCTA	0	0	104	159	41.5263	46.67
TTTTAAATTTCACAA	34	0	137	166	31.1984	13.33
Sequences obtained in [30]						
AAGAAAGGCGAAGAA	9	0	116	124	37.1454	40
CAACAAGAGCACATA	0	0	109	150	35.2693	40
CAACAGGACAAACGA	9	0	104	151	38.8994	46.67
AACCACCACTTCCTA	0	0	107	153	37.6383	46.67
TCTCTCACACATCTC	0	0	107	140	35.8936	46.67
AACAGCCTAACCGTA	0	0	115	146	38.7104	46.67
TGCATCCTTTCCTCT	9	0	125	121	38.3245	46.67
GGCATAACCACTCTT	0	0	124	143	37.2994	46.67
GAAGGCAGTCACTTA	0	0	136	123	37.0216	46.67
AAAAAGCACAGCTAC	25	0	108	147	36.4861	40
CCAAACAAACCGAGA	18	0	96	154	38.7244	46.67
AACGACAACGAACAA	0	0	97	160	38.8840	40
CACAACCTAACACCA	0	0	85	162	37.6240	46.67
CAATCCTTCTCGTTC	0	0	129	129	36.2532	46.67
CAACAAACAGGCTAC	9	0	105	159	37.2122	46.67
CCACTACATCTCTAA	0	0	117	157	32.0238	40
GGTTATCTATCTCCA	0	0	135	133	31.4527	40
CATCCACCTCAATTC	0	0	107	140	35.7989	46.67
AACTACGGACCTATT	0	0	123	131	34.4633	40
ACACCATAACAACAC	0	0	85	161	35.4422	40
NSGA-II						
CCTACAACCACAAAC	0	0	111	116	35.155	46.667
ACCACCGTACACAAA	0	0	112	127	38.87	46.667
AATTCGCCACACTTC	0	0	134	103	38.103	46.667
GAGAAGGAGAAGAAA	0	0	77	146	31.925	40
AACCAACCAACCAAC	0	0	97	109	38.009	46.667
GGAATAATGAAGGGG	0	0	96	133	33.464	46.667
GAGAAGAGAGAGAGA	0	0	86	150	32.892	46.667
GGAGATAACGGAAGA	0	0	99	137	34.734	46.667
GCTCTTCTTGGCTTT	0	0	153	85	37.493	46.667
TGTCTGGTGTGTGTT	0	0	130	103	38.729	46.667
GGTGTGTATGTGTGT	0	0	123	109	36.656	46.667
AACACGAAGACACAA	0	0	101	131	36.479	40
GTGTTGTGTTGTGTT	0	0	128	85	35.759	40
CTTGTTGCTTGCTTT	0	0	151	85	36.289	40
GAGGATAACAAGGGA	0	0	94	137	34.4	46.667
AAGGAAGGAAGGAAA	0	0	76	135	34.279	40
GAAGAGAGAGAGAGA	0	0	78	152	32.892	46.667
AGGACCGGAACAAAA	0	0	93	130	38.777	46.667
AAGAGGAAGAGAGAA	0	0	80	151	32.768	40
CCACCAACACCAAAA	0	0	101	121	38.016	46.667

As future work, we are studying other multiobjective approaches based on swarm intelligence that can be applied to the design of DNA sequences.

ACKNOWLEDGMENTS

This work was partially funded by the Spanish Ministry of Economy and Competitiveness, the ERDF (European Regional Development Fund), under the contract TIN2012-30685 (BIO project), and by the Government of Extremadura, with the aid GR10025 to the research group TIC015.

7. REFERENCES

[1] Adleman, L.M. Molecular computation of solutions to combinatorial problems. *Science* 266 (1994), 1021–1024.

[2] Brenneman, A and Condon, A. Strand design for biomolecular computation, *Theor. Comput. Sci.* 287 (2002), 39–58.

[3] Coello, C.A., Van Veldhuizen D.A and Lamont, G.B. Evolutionary Algorithms for Solving Multi-Objective Problems. Genetic Algorithms and Evolutionary Computation. *Kluwer* (2002).

[4] Deb, K. et al. A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Trans. on Evol Computation* 6 (2000), 182–197.

[5] Zitzler, E., et al.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Proceeding EUROGEN'02* (2002), 95-100.

[6] Hartemink, A.J. et al. Automated constraint based nucleotide sequence selection for DNA computation. *Proc. 4th DIMACS Workshop DNA Based Comput.* (1998), 227–235.

[7] Penchovsky R., Ackermann, J. DNA library design for molecular computation. *J. Comput. Bio.* 10, 2 (2003), 215–229.

[8] Frutos, A.G, Thiel, A.J., et al. DNA computing at surfaces: Four base mismatch word design. *Proc. 3rd DIMACS Workshop DNA Based Comput.* (1997), 238.

[9] Arita, M. and Kobayashi, S. DNA sequence design using templates. *New Generation Comput.* 20 (2002), 263–277.

[10] Liu, W. Wang, S. et al. DNA Sequence Design Based on Template Strategy. *Journal of Chem. Inf. Comput. Sci.* 43 (2003), 2014–2018.

[11] Feldkamp, U., Saghafi, S., et al. DNA sequence generator–A program for the construction of DNA sequences. *Proc. 7th Int. Workshop DNA Based Comput.* (2001), 179–188.

[12] Tanaka, F., Nakatsugawa, M., et al. Developing support system for sequence design in DNA computing. *Proc. 7th Int. Workshop DNA Based Comput.* (2001), 340–349.

[13] Deaton, R., Chen, J. A PCR-based protocol for in vitro selection of noncrosshybridizing olgionucleotides. *Proc. 8th Int. Workshop DNA Based Comput.* (2002), 196–204.

[14] Deaton, R., Chen, J., et al. Asoftware tool for generating noncrosshybridization libraries of DNA oligonucleotides. *Proc. 8th Int. Workshop DNA Based Comput*. (2002), 252–261.

[15] Heitsch, C.E., Condon, A.E., and Hoos, H.H. From RNA secondary structure to coding theory: A combinatorial approach. *Proc. 8th Int. Workshop DNA Based Comput.* (2002), 215–228.

[16] Deaton, R., et al. Reliability and efficiency of a DNA-based computation," *Phy. Rev. Lett.* 80, 2 (1998), 417–420.

[17] Zhang, B.T. and Shin, S.H. Molecular algorithms for efficient and reliable DNA computing. *Proc. Gen. Program.* (1998), 735–742.

[18] Arita, M., Nishikawa, A., Hagiya, M., Komiya, K., Gouzu, H., and Sakamoto, K. Improving sequence design for DNA computing. *Proc. Genetic Evol. Comput. Conf. (GECCO)* (2000), 875–882.

[19] Ruben, A.J. Freeland, S.J. and Landweber, L. PUNCH: An evolutionary algorithm for optimizing bit set selection. *Proc. 7th Int.Workshop DNA Based Comput.* (2001), 260–270.

[20] Hongyan, Z. and Xiyu, L. Improved Genetic Algorithm for Designing DNA Sequences. *Proc. 2nd. Int. Symposium on Electronic Commerce and Security* (2009), 514–518.

[21] Zhang, Q., Wang, B., et al. DNA Word Set Design Based on Minimum Free Energy. *IEEE T. Nanobiosci.* 9, 4 (2010), 273–277.

[22] Xu, C., Zhang, Q., Wang B. and Zhang, R. Research on the DNA Sequence Design Based on GA/PSO Algorithms. *Proc. 2nd. Int. Conf. on Bioinformatics and Biomed. Eng.* (2008), 816–819.

[23] Khalid, N.K., Kurniawan1, T.B., et al. A Model to Optimize DNA Sequences Based on Particle Swarm Optimization. *Proc. 2nd Asia Int. Conf. on Modeling & Simulation* (2008), 534–539.

[24] Cui, G. and Li, X. The Optimization of DNA Encodings Based on Modified PSO/GA Algorithm. *Proc. Int. Conf. on Computer Design and Applications* (2010), 609–614.

[25] Ibrahim, Z., Khalid, N.K., et al. DNA Sequence Design for DNA Computation Based on Binary Particle Swarm Optimization. *Int. Jour. of Innovative Comp., Inform. and Control* 8, 5B (2012), 3441–3450.

[26] Kurniawan, T.B., Khalid, N.K., et al. Evaluation of Ordering Methods for DNA Sequence Design Based on Ant Colony System. *Proc. 2nd Int. Conf. on Modeling & Simulation* (2008), 905–910.

[27] Kurniawan, T.B. Khalid, N.K., et al. Sequence Design for Direct-Proportional Length-Based DNA Computing using Population-Based Ant Colony Optimization. *Proc. ICROS-SICE Int. Joint Conference* (2009), 1486–1491.

[28] Mustaza, S.M., Abidin, A.F.Z., et al. A Modified Computational Model of Ant Colony System in DNA Sequence Design. *Proc. IEEE Student Conf. on Research and Development* (2011), 169–173.

[29] Shin, S-Y., Kim, D-M., Lee, I.-H. and Zhang, B.-T. Evolutionary sequence generation for reliable DNA computing. *Proc. Congr. Evol. Comput. (CEC)* (2002), 79–84.

[30] Shin, S.-Y., Lee, I.-H., Kim, D. and Zhang, B.-T. Multiobjective Evolutionary Optimization of DNA Sequences for Reliable DNA Computing. *IEEE Transaction on Evolutionary Computation* 9, 2 (2005), 143–158.

[31] Wang, Y., Shen, Y., Zhang, X. and Cui, G. DNA codewords design using the improved NSGA-II algorithms. *Proc. 4th Int. Conf. on Bio-Inspired Computing* (2009), 48–52.

[32] Santa-Lucia, Jr. J. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc. Nat. Acad. Sci. U.S.A.* 95 (1998), 1460–1465.

[33] Deaton R, Murphy R.C, et al. Good encodings for DNA-based solutions to combinatorial problems. *Proc. 2nd Annual Meeting DNA Based Computation* (1996), 247–258.

[34] Tanaka, F., Nakatsugawa, M., et al. Toward a general-purpose sequence design sstem in DNA computing. *Proc. Congr. Evol. Comput. (CEC)* (2002), 73–78.

[35] Faulhammer, D., Cukras, A.R., et al. Molecular computation: RNA solutions to chess problems. *Proc. Natl. Acad. Sci. U.S.A.* 97 (2000), 1385–1389.