DNA Sequence Design for Reliable DNA Computing by Using a Multiobjective Approach

J.M. Chaves-González and M.A. Vega-Rodríguez
University of Extremadura, Department of Computers and Communications Technologies, Cáceres, Spain
{jm, mavega}@unex.es

Abstract—In DNA computing, the design of reliable DNA sequences is one of the most important tasks, because sequences used for that purpose have to fulfill several biochemical restrictions to avoid undesirable reactions. These reactions usually cause incorrect computations, so it is necessary to avoid this circumstance by generating robust and reliable sets of sequences. In this paper, we consider six different conflicting design criteria to solve the problem by using the multiobjective fast non dominated sorting genetic algorithm (NSGA-II). Our proposal is compared against other sequence design methods, and our results show that our version of NSGA-II outperforms in reliability other existing techniques published in the literature.

Index Terms—DNA sequence design, multiobjective optimization, DNA computing, NSGA-II.

I. 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 have to be applied to 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], [4] because these approaches effectively consider different objectives without artificial adjustments which are needed by classical mono-objective methods when they are applied to multiobjective problems.

In this paper, we consider six different conflicting design criteria to generate reliable DNA sequences by using a version of the standard multiobjective fast non dominated sorting genetic algorithm (NSGA-II) [5]. Our proposal is compared against other relevant approaches, and as will be shown, it obtains more reliable sequences than other methods previously published in the literature.

The rest of this paper is organized as follows: Section II discusses related work. Section III describes very briefly the problem and the multiobjective formulation we have followed. The proposed metaheuristics is explained in Section IV. Section V is devoted to analyze the experiments carried out and also to compare our approach with other methods published in the literature. Finally, Section VI summarizes the conclusions of the paper and discusses possible lines of future work.

II. RELATED WORK

Most of the works published that describe the design of sequences for DNA computing consider the problem in terms of threshold-based constraints. In those works, sequences are designed by considering one or more restrictions that each pair of sequences has to fulfill according to a given threshold. Exhaustive and random searches are the simplest methods [6], [7], but they are not effective techniques because they use too much computational resources. Other strategies are based on templates [8], [9], [10], on the graph theory [11], or on classical evolutionary techniques such as simulated annealing [12]. Biologically inspired methods have been recently used to design reliable sequences. Thus, in [13], an approach based on in vitro evolution is used to find non-cross hybridizing DNA libraries. In [14], [15], authors take into account thermodynamic properties of DNA structures and sequence 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], evolutionary algorithms (EAs) can be considered the most widely used methods for designing reliable sequences. These methods make evolve one or more design criteria by using different evolutionary approaches. In particular, genetic algorithms are the most extended algorithms due to their simplicity [16]-[21]. But there are other evolutionary approaches which have been also applied to the problem. Studies based on swarm intelligence obtain interesting results, such as [22]-[25], which generate sequences by using particle swarm optimization, or [26]-[28], which use ant colony optimization. However, although those studies consider multiple design criteria, the proposed metaheuristics cannot be considered as multi-objective EAs, because in all cases, the problem is simplified and converted into a single-objective optimization problem by using a constrained weighted summation method. Only in references [29], [30] and [31] are proposed real multiobjective approaches. Results published in [31] and [29] are improved in [30] by using the algorithm (NACST/Seq). Sequences generated in that study represent the best results published so far, so we will compare our results against results provided by NACST/Seq [30] in Section V.
III. PROBLEM STATEMENT

DNA sequence design is crucial in many bio-molecular technologies, such as nanotechnology or DNA computing. The most important process to obtain the information stored in DNA sequences is the hybridization between a DNA sequence and its basepairing complement. However, undesirable hybridizations have to be avoided by properly designing DNA sequences, because illegal reactions cause potential errors that can be produced during the biological reaction and they have to be controlled in the design stage. Therefore, every DNA design criteria have to contribute to improving reliability. Design criteria are 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 criterion consists of generating sequences that are only allowed to form a duplex with its complement. The two most important 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. Continuity is the most extended criterion to predict secondary structures. This restriction counts the repeated run of identical bases. Regarding to the control of bio-chemical features, 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 sequence. 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 can control the occurrence of specific subsequences in the designed sequences.

A. Mathematical model

After studying related publications, we have chosen six conflicting design criteria that will be optimized simultaneously with the aim of generating reliable DNA sequences: similarity, h-measure, continuity, hairpin, GC ratio and melting temperature. Every objective has to be minimized. Formally, the problem is described as follows.

Minimize \[ y = F(x) = (f_1(x), f_2(x), \ldots, f_d(x)) \] (1)

where \( f(x) \) are the objectives previously mentioned. A formal definition of each objective is given bellow.

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

\[
f_{\text{similarity}} (x, y) = \max_{g} (s_{\text{disc}} (x, \text{shift}(y, g, i)) + s_{\text{cont}} (x, \text{shift}(y, g, i))) \]

(2)

where \( x \) and \( y \) are parallel sequences and \text{shift} indicates a shift of sequence \( y \) by \( i \) bases and \( g \) gaps. \( s_{\text{disc}} \) is a real value between 0 and 1, and \( s_{\text{cont}} \) 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.

2) H-measure: This objective is similar to similarity, but instead of considering sequences in parallel, they are managed as complementary. 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).

\[
f_{h_{\text{measure}}} (x, y) = \max_{g} (h_{\text{disc}} (x, \text{shift}(y, g, i)) + h_{\text{cont}} (x, \text{shift}(y, g, i))) \]

(3)

where \( x \) and \( y \) are anti-parallel sequences and \text{shift} indicates a shift of sequence \( y \) by \( i \) bases and \( g \) gaps. \( h_{\text{disc}} \) and \( h_{\text{cont}} \) and the threshold have analogous values to the similarity measure.

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

\[
f_{\text{continuity}} (x) = \sum_{i=1}^{\max} \sum_{t \in \{A, C, G, T\}} (c_{ij}(x, i, t))^2 \]

(4)

where \( x \) is the sequence under study, \( \max \) is the difference between the length of the sequence and the threshold (\( T \)), \( c_{ij}(x, i) \) is equal to \( t \) if \( \exists \) \( t \) s.t. \( x_i \neq a \), \( x_{i+1} = a \) for \( 1 \leq j \leq T \), \( x_{i+1} \neq a \), and 0 otherwise. Continuity for all bases (A, T, C and G) is controlled in every sequence.

4) Hairpin: This restriction calculates the probability of secondary structures creation. 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 \( R_{\text{min}} \) bases as a loop and a minimum of \( P_{\text{min}} \) base pairs as a stem. 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, if more than half bases in the subsequence \( x_{ij} \ldots x_i \) hybridize to the
subsequence $x_{i+r} \ldots x_{i+p}$. The number of matches in these subsequences is defined as the penalty for this hairpin.

$$f_{	ext{pen}}(x) = \sum_{p=1}^{m} \sum_{r=1}^{p} \frac{1}{2} \sum_{i=1}^{\text{pinlen}(p,r,i)} \sum_{r=1}^{\text{pinlen}(p,r,i)}$$

where the function $\text{pinlen}(p,r,i) = \min(p+i, l-r+i-p)$ 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. For example, the GC% of the DNA sequence ATGATAGGCCGTGTA is 40 (6 out of 15).

6) Melting temperature, $T_m$: This measure predicts DNA thermal denaturation, which is a key factor for DNA computing. Both sequence and base composition are important determinants of DNA duplex stability. There are many ways to calculate this relevant feature, but we use the nearest neighbour (NN) model in this approach [31]. The mathematical description for this measure, calculated by using the NN method, is provided in (6).

$$T_m(x) = \Delta H^0 / \Delta S^0 + R \ln(|C_2|/4)$$

where $x$ is the DNA sequence studied, $R$ is a gas constant and $|C_2|$ is the total sequence concentration. $\Delta H^0$ and $\Delta S^0$ refer to predicted enthalpies and entropies. Those values were taken from [32].

B. Multiobjective definition

DNA sequence design problem can be naturally formulated as a constrained multiobjective evolutionary algorithm (MOEA). This consideration arises because two of the previously described objectives (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:

Minimize $f(x)$, where

$$i \in \{ \text{similarity, h-measure, continuity, hairpin} \}$$

subject to $g(x) = 0$, where $j \in \{ T_m, \text{GC ratio} \}$

IV. MULTIOBJECTIVE APPROACH

We have generated reliable DNA sequences which are suitable for molecular computing by using the biochemical criteria explained in the previous section and an adapter version of the multiobjective standard NSGA-II (fast non dominated sorting genetic algorithm [5]). This metaheuristics ranks the solutions based on the concept of dominance. One solution dominates another if its objectives have values as good as the objectives of the other solution and at least one objective presents better results [5]. NSGA-II finds non-dominated frontiers where solutions in that frontier set are not dominated by any solution. To distinguish between the qualities of two solutions which are ranked equally, the algorithm uses the crowding distance [5]. This measure forces NSGA-II to uniformly cover the frontier rather than bunching up several promising areas by trying to maintain diversity within the population of solutions. The pseudocode of the proposed algorithm is shown in Algorithm 1. Moreover, the algorithm uses the individual codification illustrated in Fig. 1 to represent the solutions provided by NSGA-II.

The individual which represents every solution is composed of a set of $n$ sequences, where $n$ is the number of sequences in the given problem. Each sequence includes the DNA sequence itself, the values of the four objectives considered — see equation (7), and the values of each constraint taken in consideration for each sequence in the set (Fig. 1). DNA strands are composed of $m$ nucleotides each. These nucleotides can be: adenine (A), thymine (T), guanine (G) and cytosine (C).

NSGA-II tries to improve current population of solutions, $P$, with an offspring, $Q$, in each iteration, by applying typical genetic operators such as tournament selection (lines 5-6), binary crossover operator ($Cr$, line 7) and random mutation operator ($f$, line 8). In our version of the algorithm, we create the descendant population, $Q$ (with $P$Size individuals), by using the parent population $P$ and the previously mentioned genetic operators. Then, the algorithm joins the two populations, $P$ and $Q$, to create a new population, $R$, with the size of $2 \times P$Size (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 $2 \times P$Size and there to have only $P$Size 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 [5] (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 generate reliable DNA sequences.
Algorithm 1 Pseudocode for our metaheuristics

1: \( P \leftarrow \text{generateParentPopulation}(P_{\text{Size}}) \)
2: \( \textbf{while} \ \text{not stop condition satisfied} \ \textbf{do} \)
3: \( \quad \text{/* Generating the offspring population, } Q */ \)
4: \( \quad \text{for } i=1 \text{ to } P_{\text{Size}} \ \textbf{do} \)
5: \( \quad \quad \text{ind1 } \leftarrow \text{tournamentSelection}(P) \)
6: \( \quad \quad \text{ind2 } \leftarrow \text{tournamentSelection}(P) \quad \text{// ind1 \neq \text{ind2}} \)
7: \( \quad \quad Q_i \leftarrow \text{recombination}(\text{ind1, ind2, } Cr) \)
8: \( \quad \quad Q_i \leftarrow \text{mutation}(Q_i, f) \)
9: \( \textbf{end for} \)
10: \( R \leftarrow \text{mergeSets}(P, Q) \)
11: \( R \leftarrow \text{fastNonDominatedSorting}(R) \quad \text{// } R = (F_1, F_2, \ldots) \)
12: \( P \leftarrow \emptyset \)
13: \( i = 1 \)
14: \( \textbf{while} \ |P| + |F_i| \leq P_{\text{Size}} \ \textbf{do} \)
15: \( \quad \text{crowdingDistanceAssignment}(F_i) \)
16: \( \quad P \leftarrow \text{mergeSets}(P, F_i) \)
17: \( \quad i = i + 1 \)
18: \( \textbf{end while} \)
19: \( P \leftarrow \text{sortByCrowdingDistance}(F_i) \)
20: \( P \leftarrow \text{mergeSets}(P, F_i) \quad \text{[1 : (P_{\text{Size}} - |P|]} \)
21: \( \textbf{end while} \)

V. EXPERIMENTS AND RESULTS

In this section, we describe the methodology followed, and we present the results obtained with our proposal by comparing them against other important results published.

A. Experimental methodology

The algorithm developed has been adjusted to obtain optimal results by performing a complete set of experiments. 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 I, we show the summary of the parametric configuration and details of the data sets included in our study.

Although there are not published results using standard multiobjective metrics, we have calculated Hypervolume (HV) indicator [36], because it is widely used in the multiobjective domain. HV measures the volume (in the objective function space) covered by members of a non-dominated set of solutions. The higher HV, the better results obtained. Our proposal have provided average hypervolumes of 49.815%±0.007 for the 7(20) instance, 50.882%±0.024 for the 14(20) instance, and 53.512%±0.038 for the 20(15) instance. These values are mean results of the hypervolumes obtained after 30 independent runs. As can be observed, \( \text{HV} \) grows with the complexity of the instance tackled and in every case the standard deviation is minimal. Moreover, note that this metrics is not free from arbitrary scaling of objectives, so the value of \( \text{HV} \) can be distorted if the range of each objective function is different. Thus, before calculating this metrics, we have to normalize the objective function values. In Table I we present the normalization points (\( \text{Min} \) and \( \text{Max} \) values) used for each instance. Therefore, our ideal reference point is \( r = (0, 0, 0, 0) \) for all data sets, since \( f_1 \) (similarity), \( f_2 \) (hairpin), \( f_3 \) (h-measure), and \( f_4 \) (continuity) have to be minimized.

| Number of sequences and nucleotides per sequence for each data set used |
|-----------------------------|-----------------------------|-----------------------------|-----------------------------|
| \( \text{Sequences proposed in [33]} \)                                      | \( \text{Sequences proposed in [34]} \)                                      | \( \text{Sequences proposed in [35]} \)                                      |
| Number of seqs. (nucleotides per seq.)                                     | 7 (20)                                                                    | 14 (20)                                                                   |
| Normalization reference points                                              | \( \text{Min} = (0, 0, 0, 0) \)                                             | \( \text{Min} = (0, 0, 30, 80) \)                                           |
| \( \text{Max} = (35, 10, 150, 190) \)                                        | \( \text{Max} = (30, 5, 150, 175) \)                                        | \( \text{Max} = (15, 5, 140, 190) \)                                        |

For comparison with other authors [30], we have used the same population size and stop condition as used by them. The population size was established to 3000 individuals, and the maximum number of generations was 200 iterations [30]. We have used three sets (Table I) of DNA sequences of different sizes proposed by different authors which have been used for reliable DNA computing. This fact ensures that our algorithm works with several types of instances which have been tested to be used for bio-molecular computing.

B. Comparative results

In this subsection, we analyze the quality of the sets of DNA sequences that are obtained by our proposal. We compare our approach with other relevant studies published. As described in Table I, reference sequences were taken from [33], [34], and [35]. Moreover, we compare our results with sequences generated by Shin et al. [50], which use a multiobjective approach with the same data sets. The comparison is made by considering each objective separately because unfortunately no studies have taken multiobjective indicators, such as hypervolume, into account so far.

Therefore, for each data set we examine the quality of each design criterion for a set of sequences taken from the optimal Pareto front generated by our NSGA-II. We established the same parametric adjustment for the biochemical constraints. Thus, for H-measure (H) and similarity (S), we set lower limits for the continuous case equal to six bases and 17% for the discontinuous case. For continuity (C), the threshold value was 2. Hairpin (P) formation requires at least six basepairings and a six base loop. The melting temperature (\( T_m \)) was calculated with 1 M salt concentration and 10nM DNA concentration.

Furthermore, the melting temperature and the GC ratio are considered constraints whose values were taken from the literature. For the results in [33] and in [34], sequences have the GC ratio restricted to 50% and the melting temperature between 46 and 53 degrees. On the other hand, for the work in [35], the range of the GC ratio is between 40% and 50% and the melting temperature
between 31 and 39 degrees. Shin et al. [30] uses the same restrictions. Comparative results are given in Figs. 2, 3 and 4, for the 7(20), 14(20) and 20(15) data sets respectively. Furthermore, in Table II, we show the comparison of sequences generated in [33], sequences generated in [30] and an example taken from the best Pareto front of the sequences generated by our approach.

### TABLE II
**Comparison of the Sequences in [33], Sequences in [30] and Sequences Obtained by Our Proposal**

<table>
<thead>
<tr>
<th>Seq. (5'→3')</th>
<th>C</th>
<th>P</th>
<th>H</th>
<th>S</th>
<th>Tm</th>
<th>GC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATAGAGGATAGTTCCTGGG</td>
<td>3</td>
<td>35</td>
<td>64</td>
<td>52.6522</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>CATTGGCCCGCTGAGCTCT</td>
<td>0</td>
<td>60</td>
<td>51</td>
<td>69.2099</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>CTGTTGACCGTCTGCGGAG</td>
<td>16</td>
<td>0</td>
<td>63</td>
<td>60.8563</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>GAAAGGGACAAAGGAGAG</td>
<td>41</td>
<td>0</td>
<td>45</td>
<td>52.7111</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>GATGTTCTGAGAAGTGGG</td>
<td>0</td>
<td>58</td>
<td>54</td>
<td>55.3056</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>TGATCTCCTGTTTACACCTC</td>
<td>16</td>
<td>4</td>
<td>61</td>
<td>48.4451</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>TTGTAAGCTCTACTGCGTAC</td>
<td>3</td>
<td>75</td>
<td>55</td>
<td>56.7055</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td><strong>Sequences obtained in [30]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTCTCTACATCCCTCCTCCT</td>
<td>0</td>
<td>43</td>
<td>58</td>
<td>46.6803</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>CTCTCTCTCTGCTCCTCCT</td>
<td>0</td>
<td>37</td>
<td>58</td>
<td>46.9393</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>TATCTTGTTGCTCCTTCT</td>
<td>0</td>
<td>45</td>
<td>57</td>
<td>49.1066</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>ATCTGTCCGTGGGTGTC</td>
<td>0</td>
<td>52</td>
<td>56</td>
<td>51.1380</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>TCTCTTTACGTTGGTTGGCT</td>
<td>0</td>
<td>51</td>
<td>53</td>
<td>49.9252</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>GTATCTCAGGCTCCTGT</td>
<td>0</td>
<td>55</td>
<td>49</td>
<td>50.7224</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>AACCCTGCAAACACACAACA</td>
<td>9</td>
<td>55</td>
<td>43</td>
<td>51.4735</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td><strong>Sequences obtained with NSGA-II</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCTTGTCTTCTCTCTGCTC</td>
<td>0</td>
<td>65</td>
<td>33</td>
<td>48.433</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>CACACACACACACACACACACAC</td>
<td>0</td>
<td>36</td>
<td>54</td>
<td>48.902</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>AGAGAGGAGAGATGGAGAGA</td>
<td>0</td>
<td>48</td>
<td>48</td>
<td>50.044</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>AAGAGGAGGAGGAGGAGGAGA</td>
<td>0</td>
<td>46</td>
<td>49</td>
<td>50.876</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>CAACACACACACACCAAAACAC</td>
<td>0</td>
<td>34</td>
<td>57</td>
<td>51.205</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>CAACACACACACACCAACAC</td>
<td>0</td>
<td>36</td>
<td>55</td>
<td>51.205</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>ATTAGGATTGAGGTTGGGG</td>
<td>0</td>
<td>57</td>
<td>35</td>
<td>48.734</td>
<td>50</td>
<td></td>
</tr>
</tbody>
</table>

Due to the limit in the number of pages, we are forced to summarize the results obtained with the other two instances in Figs. 3 and 4. 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 (Fig.3). 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].

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, sequences generated by our approach are more reliable for DNA computing because they are more dissimilar and they avoid secondary structures. Comparative results can be observed in Fig. 4.

### VI. CONCLUSIONS AND FUTURE WORK

In this work, we present NSGA-II for the design of DNA sequences that can be applied to reliable molecular computing. NSGA-II can obtain sets of sequences 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 approach. 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.

As future work, we are studying other multiobjective approaches based on swarm intelligence which we want to apply to the design of reliable DNA sequences.

REFERENCES


