J Heuristics (2007) 13: 209–225
DOI 10.1007/s10732-007-9014-6

Sequencing by hybridization: an enhanced crossover operator for a hybrid genetic algorithm

Carlos A. Brizuela · Luis C. González-Gurrola · Andrei Tchernykh · Denis Trystram

Received: 1 June 2005 / Revised: 26 May 2006 /
Accepted: 26 May 2006 / Published online: 25 April 2007
© Springer Science+Business Media, LLC 2007

Abstract This paper presents a genetic algorithm for an important computational biology problem. The problem appears in the computational part of a new proposal for DNA sequencing denominated sequencing by hybridization. The general usage of this method for real sequencing purposes depends mainly on the development of good algorithmic procedures for solving its computational phase. The proposed genetic algorithm is a modified version of a previously proposed hybrid genetic algorithm for the same problem. It is compared with two well suited meta-heuristic approaches reported in the literature: the hybrid genetic algorithm, which is the origin of our proposed variant, and a tabu-scatter search algorithm. Experimental results carried out on real DNA data show the advantages of using the proposed algorithm. Furthermore, statistical tests confirm the superiority of the proposed variant over the state-of-the-art heuristics.

Keywords Sequencing by hybridization · Hybrid genetic algorithm · Greedy crossover

C.A. Brizuela (✉) · A. Tchernykh
Computer Sciences Department, CICESE Research Center, Km 107 Carr. Tijuana-Ensenada, Ensenada, B.C., Mexico
e-mail: cbrizuel@cicese.mx

L.C. González-Gurrola
Instituto Tecnológico Superior de Santiago Papasquiaro, Km. 114 Carretera J. Guadalupe Aguilera-Guanaceví, Santiago Papasquiaro, Dgo. 34600 Mexico

D. Trystram
ID-Institut IMAG, Avenue Jean Kuntzmann, 38330 Montbonnot Saint Martin, 38041 Grenoble Cedex 9, France

1 Introduction

The DNA sequencing is one of the most important problems in molecular biology. It refers to the identification of an unknown short DNA sequence of $100 \leq n \leq 1000$ base pairs. There are two classical approaches to sequencing: the chemical one proposed by Maxam and Gilbert (1977), and the one involving gel electrophoresis by Sanger and Coulson (1978), widely used in sequencing laboratories. A relatively new approach denominated sequencing by hybridization (SBH) (Bains and Smith 1988), which consists of biochemical and computational parts, offers an interesting alternative. The biochemical part of this method has been already widely used for Single Nucleotide Polymorphism (SNP) analysis (Hirschhom et al. 2000) and its applicability to real sequencing problems depends mainly on the development of good algorithmic techniques for solving the computational part (Blazewicz et al. 2002a). For the biochemical part we are given an unknown DNA fragment, composed by a sequence of nucleotides from a set of four: A (adenine), C (cytosine), G (guanine), and T (thymine). In this set A and T are complements as well as C and G. From this sequence all oligonucleotides (short sequence of nucleotides) of length l, usually 8 to 10 nucleotides (Iduri and Waterman 1995) are detected, and compose the unknown sequence, when properly ordered and overlapped. The detection is performed using a full DNA chip (Southern 1988), or full DNA array, which contains all fragments (oligonucleotides) of length l, i.e. 4^l fragments. Copies of the unknown DNA sequence treated with a fluorescent substance are deposited on the DNA chip. During the biochemical phase, complementary fragments, of length l come together, i.e. hybridize. Two DNA fragments are complementary if at each position the nucleotide of one fragment is a complement of the corresponding nucleotide in the other fragment. After the hybridization and by reading a fluorescent image of the chip, we can obtain the spectrum, which is the set of oligonucleotides composing the unknown DNA sequence. Here, the second stage of SBH starts, i.e. given the spectrum and the length of the unknown DNA sequence, which can be estimated by using gel electrophoresis (Krebs and Dunaway 1996), find the order of oligonucleotides such that consecutive elements always overlap in $l - 1$ nucleotides. When the hybridization is performed without errors the spectrum includes all length l oligonucleotides of the unknown DNA sequence. However, many factors do not allow the experiment to be run error-free. There are two types of errors in the spectrum: if the spectrum does not include one or more oligonucleotides of the original DNA sequence, then we have an experiment with *negative errors*. On the other hand, if the spectrum includes oligonucleotides that are not present in the original sequence then we have an experiment with *positive errors*. Every repetition of an oligonucleotide within the original sequence becomes a negative error, since the hybridization cannot detect the number of occurrences of oligonucleotides in the sequence. The presence of negative errors forces the overlap between some neighboring oligonucleotides in the sequence to be of length less than $l - 1$. The presence of positive errors in the spectrum forces some oligonucleotides to be rejected during the reconstruction process. When the spectrum contains only negative errors, the problem can be approximately modeled as the *shortest common superstring problem* (SCS) (Blazewicz and Kasprzak 2003), which is NP-hard (Garey and

Johnson 1978). On the other hand, when the spectrum contains both types of errors, the problem can be represented as the *selective traveling salesman problem* (STSP) (Blazewicz et al. 1999) which is also known to be NP-hard (Garey and Johnson 1978).

There have been many attempts at finding an efficient method to solve the computational part of the SBH. The problem can be solved in polynomial time (Pevzner 1989) when the biochemical phase is run error-free. The optimal solution can be obtained by reducing the problem to finding an Eulerian path in a directed graph. However, when errors are present, the problem becomes strongly NP-hard (Blazewicz and Kasprzak 2003), i.e., there is no known polynomial time algorithm for the SBH problem with errors. Exact and heuristic methods have been proposed assuming errors in the spectrum. A Tabu Search based algorithm for the latter case is proposed by Blazewicz et al. (2000). Later on, the same authors proposed a sophisticated heuristic (Blazewicz et al. 2002a) that improves their previous results. At the same time, a hybrid genetic algorithm (Blazewicz et al. 2002b) with a greedy crossover is proposed to effectively and efficiently deal with positive and negative errors. This genetic algorithm obtains better results than the Tabu Search algorithm (Blazewicz et al. 2000). Blazewicz et al. (1999) present a branch and bound method to deal with positive and negative errors. This algorithm achieves its best performance on instances with only positive errors, however, it has problems handling negative errors. In a recent work by Zhang et al. (2003), a new exact algorithm, based on the information of connected oligonucleotides, predecessor-successor relation, is employed to effectively handle both types of errors. Indeed, even in the presence of repetitions, this algorithm has exceptionally good performance. In Bui and Youseef (2004), a genetic algorithm is proposed to deal with both type of errors, and compared with the one in Blazewicz et al. (2002b). In a more recent work (Blazewicz et al. 2004), the Tabu search algorithm (Blazewicz et al. 2000) is enhanced by scatter search. The Tabu-Scatter-Search algorithm is compared with the results presented in Blazewicz et al. (1999, 2000, 2002b), and in all cases it obtains better results.

In this paper, a genetic algorithm[1] to deal with the computational part of the SBH problem is introduced. This algorithm is a variant of the hybrid GA (HGA) proposed by Blazewicz et al. (2002b). The variant achieves better similarity results, with respect to the original sequences for computationally hard instances, and shorter computation time.

The remainder of the paper is organized as follows. Section 2 describes the SBH problem. Section 3 introduces the proposed algorithm variant. Section 4 presents the experimental setup and results. Finally, Sect. 5 summarizes the paper and points out ideas for future research.

2 Problem statement

The input for the computational part of the SBH problem consists of a set $\mathbf{S} = \{s_1, s_2, \ldots, s_k\}$ of equal length (l) strings s_i over the alphabet $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$,

[1] This is a revised and extended version of the conference paper (Brizuela et al. 2004).

and a number n representing the length of the unknown sequence. Each s_i is always a fragment of the original sequence N, whenever the experiment is error-free ($|\mathbf{S}| = n - l + 1$). However, in general, s_i may represent a fragment that is not in the original sequence (positive errors). Furthermore, there may be fragments in the original sequence N that do not appear as a string s_i in $\mathbf{S}$ (negative errors). The problem is to find a sequence L of length no greater than n such that the number of used strings s_i is maximized, and therefore the differences between N and L is minimized. The justification for maximizing the number of used strings s_i's, is based on the assumption that most of the information from the hybridization experiment is correct. The SBH problem with positive and negative errors was proven to be strongly NP-hard (Blazewicz and Kasprzak 2003).

3 The proposed algorithm

Our Sequencing Genetic Algorithm (SGA) is based on the idea of genetic algorithms (Holland 1975; Goldberg 1989). The SGA is similar to the one proposed by Blazewicz et al. (2002b) in all genetic operators except for the crossover. The encoding method and the objective function are defined next.

3.1 Encoding

Each individual $\mathbf{i}$ is represented by a permutation of indices of oligonucleotides in the spectrum. Specifically, the adjacency-based coding (Grefenstette et al. 1988) is used: value i at locus j in the chromosome means that oligonucleotide i follows oligonucleotide j in the solution.

Therefore, feasible solutions are represented by subcycle free permutations, except for a single cycle of length $|\mathbf{S}|$. Figure 1 shows a feasible individual $\mathbf{i} = [4\ 2\ 3\ 0\ 1]$ for a given spectrum $\mathbf{S} = \{\text{CTG}, \text{ACT}, \text{GGA}, \text{GAC}, \text{TGA}\}$. In this chromosome, the number 4 at locus 0 indicates that the oligonucleotide at position 0 (CTG) in the spectrum, is followed by the oligonucleotide at position 4 (TGA) in the spectrum. In this way, following the indices in the spectrum, the sequence of oligonucleotides that individual $\mathbf{i}$ represents is: CTG, TGA, ACT, GGA, GAC $(0, 4, 1, 2, 3, 0)$. Notice that this solution defines a set of candidate sequences, not a single sequence, as it will become clear with the explanation of the fitness function computation.

Fig. 1 Adjacency based representation for the SBH problem, $|\mathbf{S}| = 5$

Algorithm 1 SGA

Input: **S** and n
Output: A candidate sequence L of length $|L| \leq n$

```
1  for i = 1 to Pop_Size
2     do Generate_Individual(i)
3        Evaluate Objective_Function(i)
4  do Select individuals by the SRM
5     Keep the best_individual found
6  While Num_Of_Iter without change in the Objective_Function(best_individual)
7     do Crossover with probability P_c
8        Apply generational replacement
9     for j = 1 to Pop_Size
10       do Evaluate Objective_Function(j)
11    do Select individuals to update the population
12       Keep the best_individual found
```

The algorithm input is given by the spectrum **S** and the length n of the original unknown sequence. *Pop_size* is the SGA population size. The function Generate_Individual(i) randomly and uniformly generates an individual. Subcycles must be avoided in the individuals to ensure the use of as many oligonucleotides in the spectrum as possible. Evaluate Objective_Function(i) computes the fitness for each individual i. The fitness of each individual is the maximum number of oligonucleotides that, overlapped in the order given by the chromosome, generates a sequence of nucleotides not longer than n, and with the maximum total overlap. P_c is the crossover probability.

Figure 2 shows the way the objective function is computed. First, we start at position zero of the resulting cycle (0, 4, 1, 2, 3, 0). Then we join oligonucleotides, once the sequence has 3 oligonucleotides (CTG, TGA and ACT) and an $n' \leq 6 \leq n$, the next oligonucleotide (GGA) can be added. In this case, the resulting sequence with

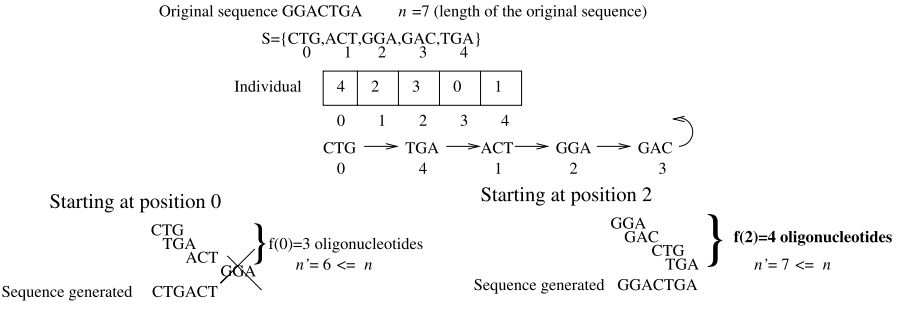

Fig. 2 Fitness function computation for individual **i** = [4 2 3 0 1] and **S** = {CTG, ACT, GGA, GAC, TGA}

4 oligonucleotides is CTGACTGGA $(0, 4, 1, 2)$. However, the last oligonucleotide (GGA) has to be eliminated since it makes the sequence to be of length (n') greater than n. Hence, the number of oligonucleotides used when starting at position 0 is 3. The process is repeated starting at each locus in the chromosome. This can be done given that the permutation of indices in the individual generates a cycle of length $|\mathbf{S}|$, and the last oligonucleotide can be considered to be adjacent to the first one. After the selection of all positions, the sequence employing the maximum number of oligonucleotides without violating the length restriction is chosen. For example, the sequence generated starting at position 2 $(2, 3, 0, 4)$ is GGACTGA. This sequence uses four oligonucleotides, being the maximum number this individual represents, and its length is exactly seven, that is n. Notice that for this particular case, the resulting sequence is identical to the original one, but this is not always the case. For each individual its fitness value is normalized, based on the maximum number of oligonucleotides in any valid sequence, and then linearly scaled, $f_{\text{new}} = (\frac{f}{(n-l+1)})k$, where f is the number of used oligonucleotides, and k the scaling factor.

The individuals are selected, lines 4 and 11 of Algorithm 1, according to the stochastic remainder method (SRM) with replacement (Goldberg 1989). The population for the next generation is constructed based on the already selected individuals. These selected individuals are randomly paired to undergo crossover; elitism is also applied (lines 5 and 12). The steps are repeated until a given number *Num_Of_Iter* of generations without improvement of the objective function value is reached.

3.2 Crossover operator

The proposed variant of the crossover operator works as follows. The best successor in the parents is always selected as long as a subcycle is not generated. Otherwise, the best successor not generating a subcycle is selected among the remaining oligonucleotides in the spectrum. Details for this operator are given below.

Algorithm 2 Crossover

Input: Two individuals (parents)
Output: One individual (child)

1 **set** the first oligonucleotide randomly (for the child)
2 **for** $i = 1$ to *Spectrum_Size* $- 2$
3 **if** it does not produce subcycle
4 **do** pick up the best overlapping oligonucleotide from the parents
5 **else**
6 **do** pick up the best overlapping oligonucleotide from the spectrum such that a subcycle is not introduced
7 **set** the last oligonucleotide

Spectrum_Size indicates the number of elements in the spectrum. The first oligonucleotide and its locus in the offspring are set randomly. For this randomly selected

oligo there are two possible successors (oligos): one in parent 1 and the other in parent 2. First we calculate the overlap generated by each of this successors and we take the one with the longest overlap, as long as it does not produce a subcycle. Otherwise, the best oligonucleotide from the spectrum is chosen. The best oligonucleotide is the one which produces the longest overlap with the previously selected one and that does not produce a subcycle. In all cases, if there is more than one best choice, the first found is chosen. Finally, the last oligonucleotide for completing a cycle of length $|\mathbf{S}|$ is set. Notice that the implementation of Step 3 implies that we search for the best overlapping oligo in the parents, and if it produces a subcycle then we go to the spectrum without verifying if the other parent produces a subcycle.

Figure 3 shows how this operation is performed. Suppose again that the spectrum is given by $\mathbf{S} = \{\text{CTG, ACT, GGA, GAC, TGA}\}$. Let us assume that the first oligonucleotide randomly selected is GAC, i. e., oligonucleotide number 3, and its randomly selected locus is 2 (Fig. 3A). Notice that for completing the cycle, oligonucleotide 2 will be the last one to be selected in the chromosome. We search in Parent 1 and in Parent 2 for the successors of oligonucleotide 3, which are 4 and 0, respectively (Fig. 3B). The best successor, that is the best overlapping oligonucleotide, is oligonucleotide 0 (Fig. 3C). Next, we look at the parents for the best successor of oligonucleotide 0, which is oligonucleotide 2, but this oligonucleotide generates a subcycle (Fig. 3D). In this case, we search in the spectrum for the best successor which is oligonucleotide 4 (Fig. 3E). For oligonucleotide 4, both successors in the parents generate a subcycle, so we search in the spectrum, given that oligonucleotide 2 will be the last one, we have only one option, oligonucleotide 1 (Fig. 3F). Finally, for completing the cycle, we select oligonucleotide 2 (Fig. 3G).

The previously proposed crossover (Blazewicz et al. 2002b) used a 80–20 rule. This means that from the first to the last generations 80% of the oligos come from

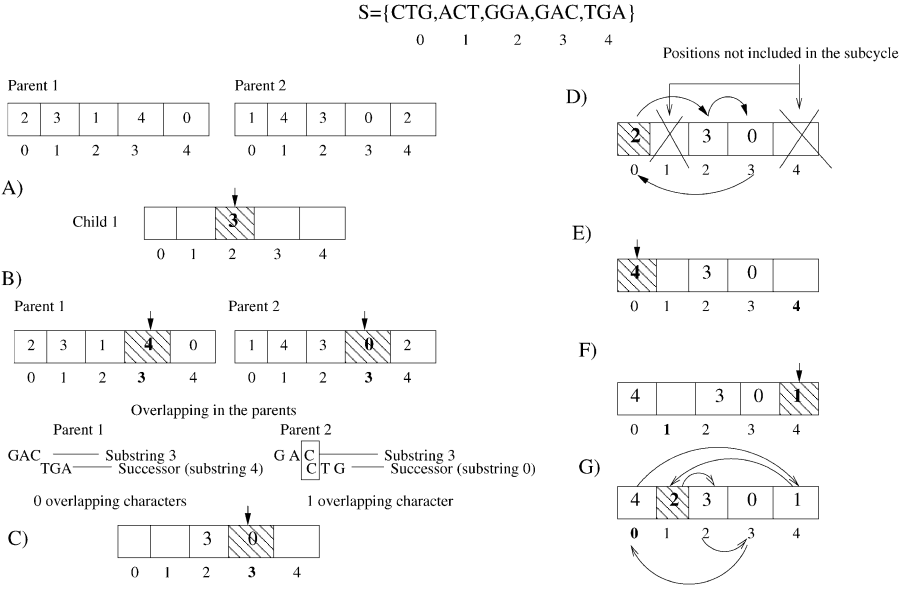

Fig. 3 Almost deterministic greedy crossover for $\mathbf{S} = \{\text{CTG, ACT, GGA, GAC, TGA}\}$

the parents and 20% from the spectrum. The main idea of what we propose here is to set this oligo's selection rule according to the quality of parents. At the beginning the quality of parents is low and the oligos are searched mainly in the spectrum. When the quality of parents increases the number of oligos coming from them also increases. In this way the algorithm can tune itself to a broader set of instances than it can do with the 80-20 rule. Another very important aspect of this idea is that the worst case computing time for a successful search in the parents takes O(1) while in the case of searching in the spectrum it takes $O(|\mathbf{S}|^2)$, where $|\mathbf{S}|$ is the spectrum cardinality.

4 Computational experiments: setup and results

In the computational experiments, the proposed algorithm SGA has been compared with two other meta-heuristic approaches for this problem: the hybrid genetic algorithm (HGA) (Blazewicz et al. 2002b), and the Tabu-Scatter search algorithm (TSS) (Blazewicz et al. 2004). These algorithms produce the best up to date results for SBH. They have been applied to real DNA sequences. The data used in these experiments are exactly the same as those used by Blazewicz et al. (2002b, 2004). These spectra have been derived from the DNA sequences coding human proteins (taken from GenBank, National Institute of Health, USA). Their accession numbers are given by D00726, D11428, D13510, X00351, X02160, X02874, X02994, X04772, X05299, X05451, X05908, X06537, X06985, X07820, X07982, X07994, X12654, X13440, X13452, X13561, X14322, X14618, X14758, X14894, X15005, X15610, X51408, X51535, X51841, X52104, X53799, X54867, X55762, X57548, X58794, Y00093, Y00264, Y00649, Y00651 and Y00711, respectively. Each spectrum was modified with the introduction of 40% of errors (20% negative and 20% positive). Given that $100 \leq |\mathbf{S}| \leq 500$, the spectrum instances contain from 40 to 200 errors (e.g. for $|\mathbf{S}| = 500$, 100 randomly chosen oligonucleotides are erased, and 100 randomly generated oligonucleotides are introduced in the spectrum). The spectra have been sorted alphabetically, thus no information about the original order of oligonucleotides from their original sequences is known (Blazewicz et al. 2002b). The length of oligonucleotides ($l = 10$) and the length of the sequences ($109 \leq n \leq 509$) have been chosen on the basis of real hybridization experiments (Caviani et al. 1994). For each spectrum size, 40 different instances are generated. The parameters for the instances are shown in Table 1.

Table 1 The instances used in the experiments

No. of instances	Length (n)	Spectrum size ($\|\mathbf{S}\|$)	Number of errors
40	109	100	40
40	209	200	80
40	309	300	120
40	409	400	160
40	509	500	200

The input instances are obtained in the following way:

1. Obtain a DNA sequence $\mathcal{N}$ of length n from GenBank (use the accession codes previously given).
2. Derive a set $\mathbf{P}$ of strings of equal length l from the sequence $\mathcal{N}$ shifting the characters one by one.
3. Modify $\mathbf{P}'$ randomly eliminating a given number of strings so that some of the resulting strings do not necessarily overlap in exactly $l-1$ characters.
4. Modify $\mathbf{P}$ randomly adding a given number of strings of length l to obtain $\mathbf{P}'$.
5. Ensure that $\mathbf{P}'$ is a free subset (i.e., there are not repeated oligos) which becomes the input $\mathbf{S}$ of the algorithm.

We show an example for this procedure:

$\mathcal{N} = \text{CACCGCATCGA}, n = 11, l = 4,$
$\mathbf{P} = \{\text{CACC, ACCG, CCGC, CGCA, GCAT, CATC, ATCG, TCGA}\}$
$\text{Error} = 20\%, \text{Number of strings to erase} = 0.20 * 8 = \lfloor 1.6 \rfloor = 1$
$\text{Index} = \text{rnd}[1 \ldots |\mathbf{P}|] = 4$
$\mathbf{P}' = \{\text{CACC, ACCG, CCGC, *, GCAT, CATC, ATCG, TCGA}\}$
$\mathbf{S} = \{\text{ACCG, ATCG, CACC, CATC, CCGC, GCAT, TCGA}\}$

To add 20% of positive errors we just need to add one oligo of length four, randomly generated.

The sequences produced by the algorithms have been compared with the original sequences using a pairwise alignment algorithm (Stoye 1998). The costs of alignment for two sequences are as follows: a mismatch (different nucleotides) brings a penalty of one point, a gap (an insertion, a nucleotide against a space) also brings a penalty of one point, and a match (the same nucleotides at a given position in both strings) brings no penalty (i.e., 0). Therefore, when two sequences have an alignment cost of zero they are identical (100% similar).

The parameters of the HGA have been set as follows. The population size is set to 50 individuals, $P_c = 1.0$, and the condition for termination is 20 iterations without improvement in the objective function. These values led to good quality solutions and short computation times (Blazewicz et al. 2002b). The parameters of the TSS have been set to values resulting in similar computation times to those needed by the HGA, as it is done in (Blazewicz et al. 2004). The parameters of the SGA are identical to those used by the HGA, $Pop\_Size = 50$, $Num\_Of\_Iter = 20$, and $P_c = 1.0$. The scaling factor for the SGA is set to $k = 1.5$ for all runs. The experiments have been performed on a PC with Athlon XP 2.0 GHz processor, 512 MB RAM, and Linux Mandrake 9.1 operating system. The SGA was implemented in ANSI C.

In Table 2, computational results of the HGA, TSS, and SGA algorithms are presented. All entries are average values over 30 runs, and at each run the algorithms were applied to 40 instances. We use five criteria to assess the algorithm quality: *average quality, optimum number, original sequences found, similarity score in points, and similarity score in percentage.* The *quality* is the number of oligonucleotides in the spectrum used for composing a solution. Notice that, the *optimal quality* is the number of proper (not corresponding to positive errors) oligonucleotides from the

Table 2 Results of SGA, HGA and TSS for $l = 10$. $Pop\_size = 50$, $Num\_of\_Iter = 20$

Spectrum size	100	200	300	400	500
Optimal quality	80	160	240	320	400
Average quality					
SGA	79.8	159.3	238.8	317.5	**396.7**
HGA	**79.9**	159.2	237.5	315.9	393.0
TSS	**79.9**	**159.7**	**239.1**	**317.7**	396.6
Optimum number					
SGA	35.4	24.9	21.9	15.0	13.9
HGA	39.7	28.3	20.8	11.0	5.4
TSS	**39.8**	**37.4**	**31.5**	**18.9**	**15.6**
Original sequences found					
SGA	28.3	22.0	16.1	10.6	11.5
HGA	**29.8**	21.2	13.8	6.5	3.7
TSS	**29.8**	**28.2**	**21.3**	**12.4**	**12.7**
Avg. similarity score (points)					
SGA	108.5	**206.4**	**299.8**	**370.7**	**434.7**
HGA	**108.6**	202.4	280.4	348.9	389.8
TSS	108.5	205.3	283.6	347.8	413.2
Avg. similarity score (%)					
SGA	99.5	**98.7**	**97.0**	**90.6**	**85.4**
HGA	**99.7**	96.8	90.7	85.3	76.5
TSS	99.6	98.2	91.7	85.0	81.1

spectrum used to construct the candidate sequence. The *optimum number* is the number of optimal solutions returned by the algorithms per run (out of 40), i.e. solutions that were constructed using the optimal number of proper oligonucleotides (optimal quality). This does not necessarily imply that solutions with the optimal number of oligonucleotides generate the original sequence. The *original sequences found*, another criterion for the quality of solutions, measures the number of times the original sequence (out of 30 runs) is found. In order to compute the *similarity*, we compare solutions generated by the algorithms with the original sequence by using a pairwise alignment algorithm (Stoye 1998). The *similarity score in points* is given by the original sequence length minus the alignment cost. The last criterion, *similarity score* in percentage, is just the previous criterion divided by the original sequence length n. A hundred percent of similarity means that the sequence generated by the algorithm is equal to the original one.

We can see that two methods (HGA and TSS) produce solutions of very high quality. However, the TSS algorithm is shown to be better than the HGA. The solutions obtained by the HGA have average qualities that range from 98.25% (393.0) to 99.87% (79.9) of the optimum, and by the TSS in the range from 99.15% (396.6) to 99.87% (79.9) of the optimum. The optimum number returned by the TSS algorithm is greater than the one returned by the HGA in all cases. For the first three

spectrum sizes, the TSS algorithm returns the optimal sequence more than 75% of the time. However, the optimum returned by one algorithm does not necessarily correspond to the original sequence, as it can be seen in the row of original sequences found. This fact makes us assume that some instances have more than one optimal solution, in terms of the objective function. For spectra of cardinality 500 there is a notable difference in the similarity score between these algorithms. Results produced by the SGA are also shown in this table. Average qualities range from 99.1% (396.7) to 99.75% (79.8) of the optimum values. These outcomes are generally better than those obtained by the HGA, and the difference becomes notable as the spectrum size increases (note that both GAs were executed with the same parameters). This can be seen in the optimum number and in the number of original sequences found. In contrast, the TSS algorithm obtains better results when compared to those generated by the SGA, but the difference decreases as the cardinality of the spectrum increases. In a very important criterion, the similarity score, the SGA obtains better results than the other two algorithms. Although the number of original sequences found by the TSS is greater than the ones obtained by the SGA, the latter provides better results in the similarity score criterion.

The difference in computation time between the SGA and the other algorithms, SGA is the fastest, allows us to improve the results obtained by the TSS, regarding the optimum number and the original sequences found. This improved performance is achieved by increasing, in the SGA, the population size and the number of iterations. Even with this increase in population size and number of iterations, the SGA remains the fastest algorithm.

Table 3 presents the results obtained by the SGA with the new set of parameters, $Pop\_size = 200$ and $Num\_of\_Iter = 40$. These solutions have average qualities that range from 99.5% to 99.87% of the optimum values. In this table we can see how as the spectrum size increases the SGA obtains more optima and more original sequences than the TSS. The similarity score is improved as well, being more than 90% in all cases. It is important to note that the similarity score produced by the SGA using the original parameters was also better than the one produced by the TSS.

Figure 4 shows computation times of the algorithms. It is clear that the SGA obtains better results than the other algorithms in much less computation time. All algorithms were run on the same machine. The differences in computation time, between SGA and HGA, are due to the number of times the algorithms go directly to the spectrum to search for the best successor. The SGA goes to the spectrum only 10% of the time.

Table 3 Results of the Sequencing Genetic Algorithm with $Pop\_size = 200$ and $Num\_of\_Iter = 40$

Spectrum size	100	200	300	400	500
Optimal quality	80	160	240	320	400
Average quality	79.9	159.7	239.4	318.4	398.0
Optimum number	39.7	33.1	30.3	23.1	21.5
Original sequences found	29.8	25.6	21.0	14.6	17.0
Avg. similarity score (points)	108.6	207.8	303.0	381.0	461.3
Avg. similarity score (%)	99.7	99.4	98.0	93.1	90.6

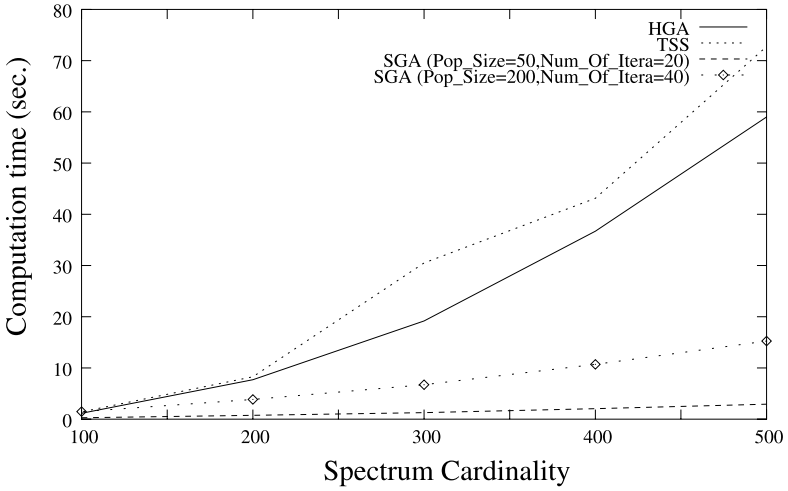

Fig. 4 Computation times for the HGA, TSS and Sequencing GA (with two different set of parameters)

To assess the statistical difference among the results of the HGA, TSS and the SGA, the non-parametric Kruskal–Wallis (Zar 1999) and Tukey multiple comparison (Zar 1999) tests were used. Given the importance of the application of these algorithms in real sequencing experiments, we considered applying these tests on the similarity score criterion. For a confidence level of $\alpha = 0.01$ the tests proved statistical difference among the SGA and the other algorithms for spectra cardinality greater than 200 oligonucleotides. It means that with a high level of confidence (99.99%), the differences in similarity scores of the SGA and those of the HGA and TSS algorithms are due to real differences in their performances, and not to random events. For spectra cardinality of 100 oligonucleotides, the similarity score of the algorithms are alike, i.e. the differences are not statistically significant.

Although for real hybridization experiments the original sequences have $109 \leq n \leq 509$ (Caviani et al. 1994), another series of experiments were performed in order to observe the behavior of the algorithms for larger sequences (709 and 1009 nucleotides). A total of 40 instances were derived for each sequence length (using $l = 10$). Their accession numbers (GenBank) are:

D00726, X05299, X07994, X12654, X51408, X51841, X52104, X57548, Y00093, Y00264, Y00649, AACZ02005042, AACZ02008280, AACZ02024680, AACZ02043728, AACZ02044283, AACZ02067430, AACZ02069750, AACZ02072337, AACZ02099914, AACZ02095742, AACZ02104255, AACZ02114766, AACZ02121490, AACZ02122970, AACZ02136257, AACZ02144575, AACZ02165291, AB170718, AB172116, AB172473, AB209268, DD222988, DD232731, NM_001034, NM_003199, NM_005860, NM_008709, NM_025481, and NM_033488, respectively.

The experiments where normalized to the performance of a PC with AMD Athlon (tm) 3.0 GHz processor, 512 MB RAM, and Linux Mandrake 10.2 operating system.

Table 4 Results of the HGA, TSS and the SGA applied to large spectrum cardinalities, and $l = 10$. *Pop_size* = 200 and *Num_of_Iter* = 40

	HGA	TSS	SGA	HGA	TSS	SGA
Spectrum size	700	700	700	1000	1000	1000
Optimal quality	560	560	560	800	800	800
Average quality	550.01	556.65	**556.66**	778.16	**793.37**	792.36
Optimum number	1.93	5.90	**11.13**	0.23	2.37	**2.9**
Original sequences found	1.73	2.43	**6.77**	0.00	0.23	**0.33**
Avg. similarity score (points)	478.88	500.82	**558.21**	571.02	646.15	**660.32**
Avg. similarity score (%)	67.54	70.64	**78.73**	56.59	64.04	**65.44**
Computation time (sec.)	62.21	150.98	**4.91**	131.79	351.58	**10.03**

Table 4 presents results produced by the algorithms over the mentioned data. All entries are average values over 30 runs, and at each run the algorithms were applied to 40 instances. The SGA was executed using the parameters *Pop_size* = 200 and *Num_of_Iter* = 40. The other algorithms maintain the same parameters used in the previous set of experiments. As it can be seen in this table, there is an important improvement in results (for all criteria) produced by the SGA over the ones produced by the other meta-heuristic approaches. Furthermore, the reduction in the computation time is significantly high.

If we have a DNA chip with longer oligos, say $l = 20$, and sequences of length, $119 \leq n \leq 1019$ we should expect a better result from the algorithms. This is because the overlap between consecutive oligos is bigger and the probability of maximum overlap between two non-adjacent oligos decreases as well as the probability of getting more than one optimum sequence. This is due to the known result (Southern et al. 1992) stating that for a fixed spectrum size the probability of getting unique (non repeated) oligos increases with the oligo's length.

A set of experiments are performed with $l = 20$, and $119 \leq n \leq 519$. These experiments are performed under the same conditions as those of Table 2. Table 5 shows the results, and as expected the solutions are of very high quality. The clear winner is the TSS in terms of solution quality. On the other hand, SGA is the fastest algorithm.

Another set of exploratory experiments were performed for $n = 719$ and 1019. The results are shown in Table 6. The average values for HGA and TSS are computed over 10 runs only, due to the large computation time they require. SGA averages are computed over 30 runs. It is clear that for $l = 20$, a spectrum cardinality of 1000 is not yet a threshold for the algorithms' performance degradation. An interesting open question is to know the threshold, in terms of n, for each algorithm when $l = 20$. As we also expect, the relative performance, considering quality of solution, of SGA is improving as $|\mathbf{S}|$ increases. Again, when the issue is computation time then SGA is the clear winner. We can predict with much confidence that our SGA will outperform the other two, in terms of solution quality for larger values of $|\mathbf{S}|$.

4.1 Discussion

The reason for success of the proposed algorithm is conjectured to be the quasi-deterministic choices it makes when selecting successors in the crossover operator.

Table 5 Results of the SGA, HGA and TSS for $l = 20$. *Pop_size* $= 200$ and *Num_of_Iter* $= 40$

Spectrum size	100	200	300	400	500
Optimal quality	80	160	240	320	400
Average quality					
SGA	79.87	159.55	239.56	319.37	399.32
HGA	**79.99**	159.96	239.86	319.71	399.65
TSS	**79.99**	**159.98**	**239.98**	**319.78**	**399.82**
Optimum number					
SGA	35.06	26.90	26.50	21.96	23.73
HGA	**39.96**	38.63	35.50	31.36	30.23
TSS	**39.96**	**39.93**	**39.96**	**39.46**	**39.63**
Original sequences found					
SGA	23.46	19.13	20.10	18.86	18.16
HGA	**25.00**	22.16	21.26	21.73	19.03
TSS	24.96	**22.90**	**24.00**	**26.63**	**24.80**
Avg similarity score (%)					
SGA	99.51	99.62	99.77	99.76	99.81
HGA	**99.57**	**99.74**	99.79	**99.80**	**99.82**
TSS	99.53	99.73	**99.80**	99.33	99.57
Computation time (sec.)					
SGA	**0.42**	**1.12**	**2.01**	**3.14**	**4.45**
HGA	3.11	15.50	38.79	73.39	117.28
TSS	1.09	9.01	38.06	56.86	96.13

Table 6 Results of the SGA, HGA and TSS for large spectrum cardinalities, and $l = 20$. *Pop_size* $= 200$ and *Num_of_Iter* $= 40$

	HGA	TSS	SGA	HGA	TSS	SGA
Spectrum size	700	700	700	1000	1000	1000
Optimal quality	560	560	560	800	800	800
Average quality	558.38	**559.47**	558.77	797.24	**799.60**	797.82
Optimum number	19.40	**36**	17.20	15.30	**36.5**	16.23
Original sequences found	14.10	**25.80**	13.2	12.70	**27**	16.46
Avg. similarity score (points)	**702.72**	697.99	697.48	990.88	**995.05**	980.36
Avg. similarity score (%)	**97.74**	97.08	97.00	97.24	**97.65**	96.21
Computation time (sec.)	124.93	180.85	**5.55**	267.54	443.38	**10.22**

The only random part in the whole process is in the selection of the initial locus and allele (Step 1 of Algorithm 2). This operator exploits the problem structure by using a deterministic greedy procedure which allows the SGA to select the best successor for

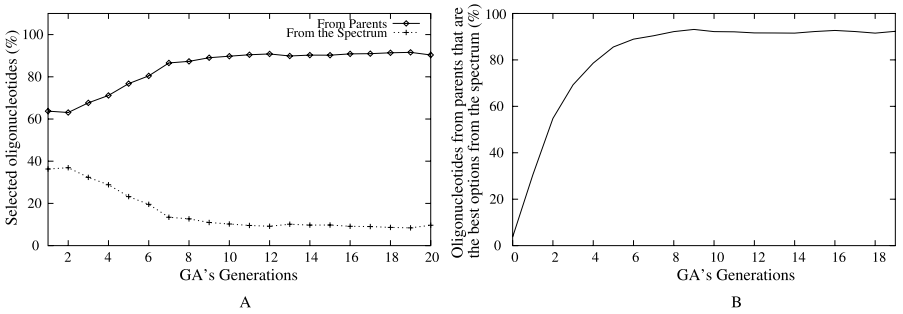

Fig. 5 **A** Source from where the crossover operator selects the oligonucleotides. **B** Percentage of oligonucleotides taken from parents that actually are the best options in the spectrum

a given oligonucleotide. The use of this operator makes the selection of the successors faster, and selecting the best successor helps to improve the solution quality.

In order to have a clearer idea of the performance of the greedy crossover and its influence on the construction of good solutions, the source from where the oligos come is analyzed. The criteria are measured considering a single run of a randomly selected instance. The larger the number of times the oligos are selected from the parents the faster the algorithm becomes. This is because searching the best oligo from the spectrum is computationally more expensive than searching it from the parents. Figure 5A shows the source (parents or spectrum) for selecting the next oligonucleotide in the crossover operator for constructing new individuals. It shows that the main source are the parents and that as the GA's population evolves (through generations) this source is selected approximately 90% of the time. We can see in this figure that the SGA can tune itself dynamically, i.e. the source for selecting oligos is not fixed as it was proposed by Blazewicz et al. (2002b).

Given one oligonucleotide, the application of a greedy strategy (Step 6 of Algorithm 2) guarantees the selection of the best successor in the spectrum. This does not necessarily happen if we select the next oligonucleotide from the parents. Therefore, it will be interesting to see what the differences are between oligonucleotides selected from the parents and oligonucleotides selected in a greedy strategy, considering the same previous oligonucleotide. Figure 5B shows the quality of oligonucleotides selected from the parents. After the first 10 generations, the oligonucleotides selected from the parents are almost (90%) the same as the ones selected if we would search in the spectrum for the best successors. This shows how high quality genetic information is passed from parents to children and this is a key point for determining the operator's performance given the short computation time needed to transfer this information.

5 Conclusions

We have presented an efficient and effective genetic algorithm for the computational part of the sequencing by hybridization problem. An almost deterministic greedy crossover operator is introduced to improve the solution quality and to reduce the

computational cost of a recently proposed hybrid genetic algorithm. Experimental results obtained on real DNA data show that the proposed algorithm can handle a large percentage of both positive and negative errors, yielding very high quality solutions. Computational experiments were performed to compare the algorithm with two other meta-heuristic approaches: a previous hybrid genetic algorithm and a tabu-scatter search algorithm. The new results are shown to be better than the previous ones. The algorithm achieved high similarity score (more than 90%), in average, in all instances with 100 to 500 oligonucleotides of lengths 10 and 20.

The main criterion for comparing these algorithms was the similarity score between the generated sequences and the original ones. Statistical tests applied to the results of these algorithms proved the superiority of the new variant of the hybrid genetic algorithm. The proposed algorithm does not use any additional information about spectra or original sequences, which could be derived from biochemical experiments, and may help to obtain better similarity scores. Furthermore, some ideas explaining the behavior of the crossover operator are given in order to get a better understanding of the rationale for success of the algorithm.

As a future research we will further explore the rationale for success of the algorithm as well as its robustness to larger values of positive and specially of negative errors.

Acknowledgements This research was partially supported by the Laboratorio Franco–Mexicano de Informática. The first author work was partially supported by CONACyT under grant C01-45811. The authors would like to thank Jacek Blazewicz and Marta Kasprzak for their valuable help in providing the test data and the executables for the HGA and TSS algorithms. The authors would also like to thank the anonymous referees for their comments that have been very helpful in writing this final version.

References

Bains, W., Smith, G.: A novel method for nucleic acid sequence determination. J. Theor. Biol. **135**, 303–307 (1988)

Blazewicz, J., Kasprzak, M.: Complexity of DNA sequencing by hybridization. J. Theor. Comput. Sci. **290**, 1459–1473 (2003)

Blazewicz, J., Formanowicz, P., Kasprzak, M., Markiewicz, W., Weglarz, J.: DNA sequencing with positive and negative errors. J. Comput. Biol. **6**, 113–123 (1999)

Blazewicz, J., Formanowicz, P., Kasprzak, M., Markiewicz, W., Weglarz, J.: Tabu search for DNA sequencing with false negatives and false positives. Eur. J. Oper. Res. **125**, 257–265 (2000)

Blazewicz, J., Formanowicz, P., Guinand, F., Kasprzak, M.: A heuristic managing errors for DNA sequencing. J. Bioinformatics **18**(5), 652–660 (2002a)

Blazewicz, J., Kasprzak, M., Kuroczycki, W.: Hybrid genetic algorithm for DNA sequencing with errors. J. Heuristics **8**, 495–502 (2002b)

Blazewicz, J., Glover, F., Kasprzak, M.: DNA sequencing-tabu and scatter search combined. INFORMS J. Comput. **16**(3), 232–240 (2004)

Brizuela, C.A., González, L., Romero, H.J.: An improved genetic algorithm for the sequencing by hybridization problem. In: Raidl, G. et al. (eds.) Applications of Evolutionary Computing. Lecture Notes in Computer Science, vol. 3005, pp. 11–20. Springer, Berlin (2004)

Bui, T.N., Youseef, W.A.: An enhanced genetic algorithm for DNA sequencing by hybridization with positive and negative errors. In: Deb, K. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference, Seattle, Washington, United States. Lecture Notes in Computer Science, vol. 3103, pp. 908–919. Springer, Berlin (2004)

Caviani, P., Solas, D., Sullivan, E., Cronin, M., Holmes, C., Fodor, S.: Light-generated oligonucleotide arrays for rapid DNA sequence analysis. Proc. Nat. Acad. Sci. USA **91**, 5022–5026 (1994)

Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness, 21st edn. Freeman, New York (1978)
Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
Grefenstette, J., Gopal, R., Rosmaita, B., Gucht, D.V.: Genetic algorithms for the travelling salesman problem. In: Grenfestette, J.J. (ed.) Proceedings of the First International Conference on Genetic Algorithms and their Applications, vol. 1, pp. 160–168. Erlbaum, Hillsdale (1988)
Hirschhom, J., Sklar, P., Lindblad-Toh, K., Lim, Y.-M., RuizGutierrez, M., Bolk, S., Langhorst, B., Schaffner, S., Wichester, E., Lander, E.: SBE-TAGS: An arraybased method for efficient single-nucleotide polymorphism genotyping. Proc. Nat. Acad. Sci. USA **97**, 12164–12169 (2000)
Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor (1975)
Iduri, R.M., Waterman, M.S.: A new algorithm for DNA sequence assembly. J. Comput. Biol. **2**(2), 291–306 (1995)
Krebs, J., Dunaway, M.: DNA length is a critical parameter for eukaryotic transcription in vivo. Mol. Cell. Biol. **16**(10), 5821–5829 (1996)
Maxam, A., Gilbert, W.: A new method for sequencing DNA. Proc. Nat. Acad. Sci. **74**, 560–564 (1977)
Pevzner, P.: l-tuple DNA sequencing: computer analysis. J. Biomol. Struct. Dyn. **7**, 63–73 (1989)
Sanger, F., Coulson, A.: The use of thin acrylamide gels for DNA sequencing. FEBS Lett. **87**, 107–110 (1978)
Southern, E.: United Kingdom patent application GB8810400 (1988)
Southern, E., Maskos, U., Elder, J.K.: Analyzing and comparing nucleic acid sequences by hybridization to arrays of oligonucleotides: evaluation using experimental models. Genomics **13**, 1008–1017 (1992)
Stoye, J.: Multiple sequence alignment with the divide-and-conquer method. Gene **211**(2), GC45–GC56 (1998)
Zar, J.H.: Biostatistical Analysis, 4th edn. Prentice-Hall, Englewood Cliffs (1999)
Zhang, J., Wu, L., Zhang, X.: Reconstruction of DNA sequencing by hybridization. J. Bioinform. **19**(1), 14–21 (2003)