New constructive heuristics for DNA sequencing by hybridization*

Christian Blum and Mateu Yábar Vallès

ALBCOM, Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Barcelona, Spain
cblum@lsi.upc.edu, mateuyabar@gmail.com

May 17, 2006

Abstract

Deoxyribonucleic acid (DNA) is a molecule that consists of two complementary sequences of amino acids. Reading these sequences is an important task in biology, called DNA sequencing. However, large DNA molecules cannot be read in one piece. Therefore, existing techniques first break the given DNA molecules up into small fragments which can be read. One of these techniques is called the hybridization experiment. The reconstruction of the original DNA molecule from these fragments is a challenging problem from the computational point of view. In recent years the specific problem of DNA sequencing by hybridization has attracted quite a lot of interest in the optimization community. While most researchers focused on the development of metaheuristic approaches, work on simple constructive heuristics hardly received any attention. This is despite the fact that well-working constructive heuristics are often an essential component of succesful metaheuristics. It is exactly this lack of constructive heuristics that motivated the work presented in this paper. The results of our best constructive heuristic are comparable to the results of the best existing metaheuristics, while using less computational time.

1 Introduction

Deoxyribonucleic acid (DNA) is a molecule that contains the genetic instructions for the biological development of all cellular forms of life. Each DNA molecule consists of two (complementary) sequences of four different amino acids, namely adenine (A), cytosine (C), guanine (G), and thymine (T). In mathematical terms each of these sequences can be represented as a word from the alphabet $\{A, C, G, T\}$. One of the most important problems in computational biology consists in determining the exact structure of a DNA molecule, called *DNA sequencing*. This is not an easy task, because the amino acid sequences of a DNA molecule (henceforth also called DNA strands) are usually so large that they cannot be read in one piece. In 1977, 24 years after the discovery of DNA, two separate methods for DNA sequencing were developed: the chain termination method and the chemical degradation method.

*This work was supported by the Spanish CICYT project OPLINK (grant TIN-2005-08818-C04-01), and by the "Juan de la Cierva" program of the Spanish Ministry of Science and Technology of which Christian Blum is a post-doctoral research fellow.

Later, in the late 1980's, an alternative and much faster method called *DNA sequencing by hybridization* was developed (see [1, 15, 12]).

DNA sequencing by hybridization works roughly as follows. The first phase of the method consists of a chemical experiment which requires a so-called DNA array. A DNA array is a two-dimensional grid whose cells typically contain all possible DNA strands—called probes—of equal length l. For example, consider a DNA array of all possible probes of length $l = 3$:

GGT	**TGA**	**GCG**	**CTA**	**AAT**	**CCT**	**CTC**	**TTC**
GTC	**GTG**	**TTG**	**GGC**	**CGA**	**TTT**	**TCA**	**ATC**
GCT	**AGC**	**GGG**	**CCA**	**TAT**	**CGG**	**TAG**	**AAG**
GAA	**GGA**	**CGT**	**ACG**	**CTG**	**TGT**	**TAA**	**ATT**
TCT	**GAT**	**CAC**	**CAT**	**CAA**	**ACC**	**ATG**	**GTT**
CGC	**GCC**	**AGG**	**CTT**	**ATA**	**TCC**	**TGC**	**GTA**
AGA	**AAC**	**TTA**	**TGG**	**TCG**	**AGT**	**CAG**	**GAC**
CCG	**GCA**	**CCC**	**AAA**	**ACA**	**GAG**	**ACT**	**TAC**

After the generation of the DNA array, the chemical experiment is started. It consists of bringing together the DNA array with many copies of the amino acid sequence to be read, also called the DNA target sequence. Hereby, the target sequence might react with a probe on the DNA array if and only if the probe is a subsequence of the target sequence. Such a reaction is called hybridization. After the experiment the DNA array allows the identification of the probes that reacted with target sequences. This subset of probes is called the *spectrum*. Two types of errors may occur during the hybridization experiment:

1. **Negative errors:** Some probes that should be in the spectrum (because they appear in the target sequence) do not appear in the spectrum. A particular type of negative error is caused by the multiple existence of a probe in the target sequence. This cannot be detected by the hybridization experiment. Such a probe will appear at most once in the spectrum.

2. **Positive errors:** A probe of the spectrum that does not appear in the target sequence is called a positive error.

Given the spectrum, the second phase of DNA sequencing by hybridization consists of the reconstruction of the target sequence from the spectrum. Let us, for a moment, assume that the obtained spectrum is perfect, that is, free of errors. In this case, the original sequence can be reconstructed in polynomial time with an algorithm proposed by Pevzner in [16]. However, as the generated spectra generally contain negative as well as positive errors, the perfect reconstruction of the target sequence is in general impossible.

1.1 DNA sequencing by hybridization

In order to solve the computational part of DNA sequencing by hybridization, one usually solves an optimization problem of which the optimal solutions can be shown to have a high probability to resemble the target sequence. In this work we consider the optimization problem that was introduced as a model for DNA sequencing by hybridization by Błażewicz et al. in [3]. In [7] it was shown that this model is NP-hard.[1] In fact, this optimization problem—outlined in the following—is a version of the *selective traveling salesman problem*.

[1] Note that also the other existing models for DNA sequencing with errors are NP-hard.

Henceforth, let the DNA target sequence be denoted by s_t. The number of amino acids of s_t shall be denoted by n (i.e., $s_t \in \{A, C, G, T\}^n$). Furthermore, the spectrum—as obtained by the hybridization experiment—is denoted by S. Remember that each $s \in S$ is an oligonucleotide (i.e., a short DNA strand) of length l (i.e., $s \in \{A, C, G, T\}^l$). In general, the length of any oligonucleotide s is denoted by $l(s)$. Let $G = (V, A)$ be the completely connected directed graph defined by $V = S$. To each link $a_{s,s'} \in A$ is assigned a weight $o_{s,s'}$, which is defined as the length of the longest DNA strand that is a suffix of s and a prefix of s'. A directed Hamiltonian path p in G is a directed path without loops. The length of such a path p, denoted by $l(p)$, is defined as the number of vertices (i.e., oligonucleotides) on the path. In the following we denote by $p[i]$ the i-th vertex in a given path p (starting from position 1). In contrast to the length, the cost of a path p is defined as follows:

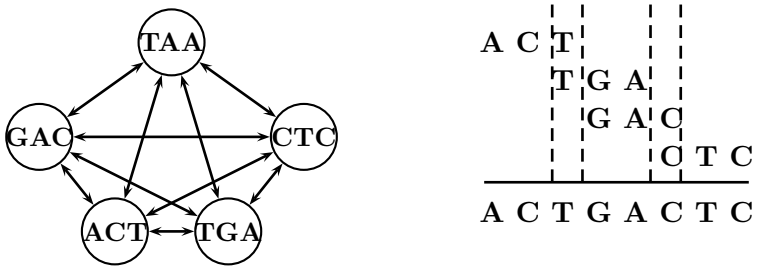
$$c(p) \leftarrow l(p) \cdot l - \sum_{i=1}^{l(p)-1} o_{p[i],p[i+1]} \tag{1}$$

The first term sums up the length of the olionucleotides on the path, and the second term (which is substracted from the first one) sums up the overlaps between the neighboring oligonucleotides on p. In fact, $c(p)$ is equivalent to the length of the DNA sequence that is obtained by the sequence of oligonucleotides in p. The problem of DNA sequencing by hybridization consists of finding a directed Hamiltonian path p^* in G with $l(p^*) \geq l(p)$ for all possible paths p that fulfill $c(p) \leq n$. In the following we refer to this optimization problem as *sequencing by hybridization (SBH)*.

As an example consider the target sequence $s_t =$ **ACTGACTC**. Assuming $l = 3$, the ideal spectrum is {**ACT,CTG,TGA,GAC,ACT,CTC**}. However, let us assume that the hybridization experiment provides us with the following faulty spectrum $S =$ {**ACT,TGA, GAC,CTC,TAA**}. This spectrum has two negative errors, because **ACT** should appear twice, but can—due to the characterisitcs of the hybridization experiment—only appear once, and **CTG** does not appear at all in S. Furthermore, S has one positive error, because it includes oligonucleotide **TAA**, which does not appear in the target sequence. An optimal Hamiltonian path in this example is $p^* = \langle$**ACT**,**TGA**,**GAC**,**CTC**$\rangle$ with $l(p^*) = 4$ and $c(p^*) = 8$. The DNA sequence that is retrieved from this path is equal to the target sequence (see Figure 1).

1.2 Existing approaches

The first approach to solve the SBH problem was a branch & bound method proposed in [3]. However, this approach becomes unpractical with growing problem size. For example, the algorithm was only able to solve 1 out of 40 different problem instances concerning DNA target sequences with 200 amino acids within one hour. Another argument against this branch & bound algorithm is the fact that an optimal solution to the SBH problem does not necessarily provide a DNA sequence that is equal to the target sequence. Therefore, the importance of finding *optimal* solutions is not the same as for other optimization problems. Therefore, the research community has focused on heuristic techniques for tackling the SBH problem. Most of the existing approaches are metaheuristics such as evolutionary algorithms and tabu search techniques (for a general overview on metaheuristic methods see, for example, [9]). A list of the existing approaches for the SBH problem is given in Table 1.



(a) Completely connected directed graph.

(b) DNA sequence retrieval from a Hamlitonian path.

Figure 1: (a) The completely connected directed graph with spectrum S = {**ACT,TGA,GAC,CTC,TAA**} as the vertex set. The edge weights (i.e., overlaps) are not indicated for readability reasons. For example, the weight on the edge from **TGA** to **GAC** is 2, because **GA** is the longest DNA strand that is a suffix of **TGA** and a prefix of **GAC**. An optimal Hamiltonian path is $p^* = \langle$**ACT**,**TGA**,**GAC**,**CTC**$\rangle$. In (b) is shown how to retrieve the DNA sequence that is encoded by p^*. Note that $c(p^*) = 8$, which is equal to the length of the encoded DNA sequence.

.

Table 1: A list of approaches for the SBH problem.

Type of algorithm	**Identifier**	**Publication**
Constructive heuristic	LAG	Błażewicz et al. [3], 1999
Constructive heuristic	OW	Błażewicz et al. [2], 2002
Evolutionary algorithm	EA1	Błażewicz et al. [8, 6], 2002
Evolutionary algorithm	EA2	Endo [13], 2004
Evolutionary algorithm	EA3	Brizuela et al. [10], 2004
Evolutionary algorithm	EA4	Bui and Youssef [11], 2004
Tabu search	TS	Błażewicz et al. [4], 2000
Tabu search / scatter search hybrid	TS/SS	Błażewicz et al. [5, 6], 2004
GRASP-like multi-start technique	GRASP	Fernandes and Ribeiro [14], 2005

1.3 Motivation and organization of the paper

Despite the fact that well-working constructive heuristics are often the basis for well-working metaheuristics, only two constructive heuristics exist. Both approaches were proposed by Błażewicz and colleagues; the first one is a look-ahead greedy (LAG) technique that was proposed in [3], and the second one called OW was proposed in [2] (see Table 1). Our motivation is to develop new types of constructive heuristics that can possibly lead to the development of better metaheuristic approaches.

The organization of the paper is as follows. In Section 2 we describe our constructive heuristics, and in Section 3 we conduct an experimental evaluation of these heuristics and compare them to the best techniques from the literature. Finally, in Section 4 we offer conclusions and an outlook to the future.

Algorithm 1 The LAG heuristic

1: **input:** A graph G, and the length of the target sequence n
2: $\hat{S} \leftarrow S$
3: $s^* \leftarrow$ Choose_Initial_Oligonulceotide($\hat{S}$)
4: $p \leftarrow \langle s^* \rangle$
5: **while** $c(p) \leq n$ **do**
6: $\quad \hat{S} \leftarrow \hat{S} \setminus \{s^*\}$
7: $\quad s^* \leftarrow \text{argmax}\{o_{p[l(p)],s} + o_{s,\text{suc}(s)} \mid s \in \hat{S}\}$
8: $\quad$ Extend path p by adding s^* to its end
9: **end while**
10: **output:** DNA sequence s that is obtained from p

2 New constructive heuristics

The first constructive heuristic that we propose is a simple extension of the look-ahead greedy (LAG) heuristic proposed in [3]. Therefore, we first brievly outline the LAG heuristic.

LAG: The idea of LAG is to start the path construction in graph G (see Section 1.1 for the definition of G) with one of the probes of the spectrum, and to extend this path in a step-by-step manner by means of a look-ahead strategy. The way in which this is done is shown in Algorithm 1. In this algorithm—as well as in the other algorithms outlined in this section—the following notations are used:

$$\text{pre}(s) \quad \leftarrow \quad \text{argmax}\{o_{s',s} \mid s' \in \hat{S},\ s' \neq s\} \ , \tag{2}$$

$$\text{suc}(s) \quad \leftarrow \quad \text{argmax}\{o_{s,s'} \mid s' \in \hat{S},\ s' \neq s\} \ , \tag{3}$$

where $\hat{S} \subseteq S$ and $s \in \hat{S}$ are given. In words, pre(s) is the best available predecessor for $s \in \hat{S}$, that is, the oligonucleotide that—as a predecessor of s—has the biggest overlap with s. Accordingly, suc(s) is the best available successor for $s \in \hat{S}$. In case of ties, the first one that is found is taken. In the original version of LAG as presented in [3], the function Choose_Initial_Oligonulceotide($\hat{S}$) chooses a random vertex for starting the path construction. However, in this paper we implemented this function as follows. First, set $S_{bs} \subset S$ is defined as the set of all oligonucleotides in S whose best successor is better or equal to the best successor of all the other oligonucleotides in S.

$$S_{bs} \leftarrow \{s \in \hat{S} \mid o_{s,\text{suc}(s)} \geq o_{s',\text{suc}(s')},\ \forall\, s' \in \hat{S}\} \tag{4}$$

Then, set $S^{wp} \subseteq S_{bs}$ is defined as the set of all oligonucleotides in S_{bs} whose best predecessor is worse or equal to the best predecessor of all the other oligonucleotides in S_{bs}:

$$S_{wp} \leftarrow \{s \in S_{bs} \mid o_{\text{pred}(s),s} \leq o_{\text{pred}(s'),s'},\ \ \forall\, s' \in S_{bs}\} \tag{5}$$

As starting oligonucleotide we choose the one (from S_{wp}) that is found first. The idea hereby is to start the path construction with an oligonucleotide that has a very good successor and at the same time a very bad predecessor. Such an oligonucleotide has a high probability to coincide with the start of the target DNA sequence s_t.

Algorithm 2 The FB-LAG heuristic

input: A graph G, and the length of the target sequence n
$\hat{S} \leftarrow S$
$s^* \leftarrow$ Choose_Initial_Oligonulceotide($\hat{S}$)
$p \leftarrow \langle s^* \rangle$
$\hat{S} \leftarrow \hat{S} \setminus \{s^*\}$
while $c(p) \leq n$ **do**
  $s_r \leftarrow \text{argmax}\{o_{p[l(p)],s} + o_{s,\text{suc}(s)} \mid s \in \hat{S}\}$
  $s_l \leftarrow \text{argmax}\{o_{\text{pre}(s),s} + o_{s,p[1]} \mid s \in \hat{S}\}$
  if $o_{p[l(p)],s_r} + o_{s_r,\text{suc}(s)} > o_{\text{pre}(s),s_l} + o_{s_l,p[1]}$ **then**
    Extend path p by adding s_r to its end
    $\hat{S} \leftarrow \hat{S} \setminus \{s_r\}$
  else
    Extend path p by adding s_l to its beginning
    $\hat{S} \leftarrow \hat{S} \setminus \{s_l\}$
  end if
end while
output: DNA sequence s that is obtained from p

LR-LAG: A simple extension of the LAG heuristic is obtained by allowing the path construction not only in forward direction but also in backward direction. We call this heuristic henceforth forward-backward lock-ahead greedy (FB-LAG) heuristic. At each construction step the heuristic decides (with the same criterion as LAG) to extend the current path either in forward direction or in backward direction (see Algorithm 2). A second change with respect to LAG concerns the implementation of function Choose_Initial_Oligonulceotide($\hat{S}$). As the path construction allows forward and backward construction it is not necessary to start the path construction with an oligonucleotide that has a high probability of being the beginning of the DNA target sequence. It is more important to start with an oligonucleotide that has a high probability of being part of the DNA target sequence:

$$s^* \leftarrow \text{argmax}\{o_{\text{pre}^2(s),\text{pre}(s)} + o_{\text{pre}(s),s} + o_{s,\text{suc}(s)} + o_{\text{suc}(s),\text{suc}^2(s)} \mid s \in S\} \ , \tag{6}$$

where $\text{pre}^2(s)$ denotes the best predecessor of the best predecessor of s (i.e., $\text{pre}(\text{pre}(s))$), and similar for $\text{suc}^2(s)$.

SM: The idea of the sub-sequence merger (SM) heuristic (see Algorithm 3) is conceptionally quite different to the LAG and FB-LAG heuristics. Instead of constructing only one path, the heuristic starts with a set of $|S|$ paths, each of which only contains exactly one oligonucleotide $s \in S$, and then merges paths until a path of sufficient size is obtained. The heuristic works in two phases. In the first phase, two paths p and p' can only be merged if p' is the unique best successor of p, and if p is the unique best predecessor of p'. The heuristic enters into the second phase if and only if the first phase has not already produced a path of sufficient length. In the second phase, the uniqueness conditions are relaxed, that is, two paths p and p' can be merged if p' is among the best successors of p, and p is among the best predecessors of p'. The reason of having two phases is the following: The first phase aims to produce possibly error free sub-sequences of the DNA target sequence, whereas the second phase (which is more

Algorithm 3 The SM heuristic

input: A graph G, and the length of the target sequence n
$P \leftarrow \{\langle s \rangle \mid s \in S\}$
PHASE 1:
$stop =$ FALSE
for $overlap = l-1, \ldots, 1$ **do**
  while $\exists\, p, p' \in P$ **s.t.** $o_{p,p'} = overlap$ **&** $|S_{\text{suc}}(p)| = 1$ **&** $|S_{\text{pre}}(p')| = 1$ **&** $\text{suc}(p) = p'$ **&** $\text{pre}(p') = p$ **&** $stop =$ FALSE **do**
    Add path p' to the end of path p
    $P \leftarrow P \setminus \{p'\}$
    if $c(p) \geq n$ **then**
      $stop =$ TRUE
    end if
  end while
end for
PHASE 2:
for $overlap = l-1, \ldots, 1$ **do**
  while $\exists\, p, p' \in P$ **s.t.** $o_{p,p'} = overlap$ **&** $p' \in S_{\text{suc}}(p)$ **&** $p \in S_{\text{pre}}(p')$ **&** $stop =$ FALSE **do**
    Choose p and p' such that $l(p) + l(p')$ is maximal
    Add path p' to the end of path p
    if $c(p) \geq n$ **then**
      $stop =$ TRUE
    end if
  end while
end for
Let p be the path in P with maximal cost
$p^* \leftarrow$ Find_Best_Subpath(p)
output: DNA sequence s that is obtained from p^*

error prone due to the relaxed uniqueness condition) aims at connecting the sub-sequences produced in the first phase in a reasonable way.

In Algorithm 3, given two paths p and p', $o_{p,p'}$ is defined as $o_{p[l(p)],p'[1]}$, that is, the overlap of the last oligonucleotide in p with the first one in p'. In correspondence to the notations introduced in Equations 2 and 3, the following notations are used:

$$\text{suc}(p) \leftarrow \text{argmax}\{o_{p,p'} \mid p' \in P,\ p' \neq p\}\ , \tag{7}$$

$$\text{pre}(p) \leftarrow \text{argmax}\{o_{p',p} \mid p' \in P,\ p' \neq p\}\ . \tag{8}$$

Futhermore, $S_{\text{suc}}(p)$ is defined as the set of best successors of p, that is, $S_{\text{suc}}(p) \leftarrow \{p' \in P \mid o_{p,p'} = o_{p,\text{suc}(p)}\}$; and $S_{\text{pre}}(p)$ is defined as the set of best predecessors of p, that is, $S_{\text{pre}}(p) \leftarrow \{p' \in P \mid o_{p',p} = o_{\text{pre}(p),p}\}$. Finally, function Find_Best_Subpath(p) is implemented to retrieve from path p the longest sub-path (in terms of the number of oligonucleotides).

HSM: The hybrid sub-sequence merger (HSM) heuristic is obtained by combining the FB-LAG heuristic with the SM heuristic. This combination is based on the following observation: At

every stage of the SM heuristic, the FB-LAG heuristic can be applied to the problem instance that is obtained as follows. Given the current path set P of the SM heuristic, a spectrum $\hat{S}$ is created that contains the DNA sequences retrieved from the paths in P.[2] The result of the FB-LAG heuristic when applied to this problem instance can (of course) be regarded as a result for the original problem instance. It remains to specify at which stages of the SM heuristic the FB-LAG heuristic is applied. The first application of FB-LAG is the one to the original problem instance, that is, before the first phase of SM has started. Then, in the first as well as in the second phase of SM, FB-LAG is applied at the end of the respective for-loop (i.e., after line 12 and after line 21 in Algorithm 3). However, FB-LAG is only applied if the while-loop before was executed at least once. Note that in case the while-loop is not even executed a single time, the problem instance derived from the path set P has not changed since the previous application of FB-LAG. Finally, the output of HSM is the best result among the different applications of FB-LAG and the final result of SM.

3 Results

We implemented the 4 heuristics outlined in the previous section in ANSI C++ using GCC 3.2.2 for compiling the software. Our experimental results were obtained on a PC with Intel Pentium 4 processor (3.06 GHz) and 1 Gb of memory.

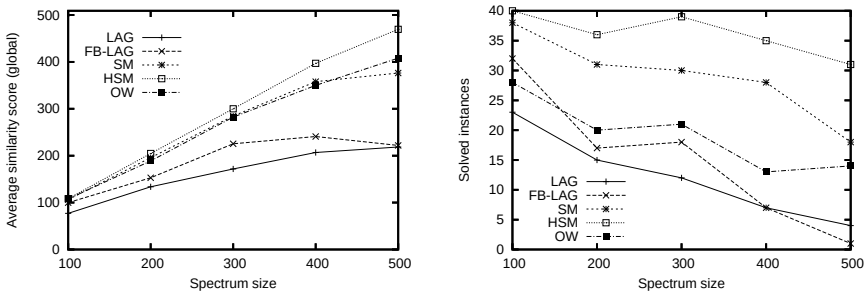
A wide-spread set of benchmark instances for DNA sequencing by hybridization was introduced by Błażewicz et al. in [3]. It consists of DNA target sequences coding human proteins obtained from GenBank, which is a database of genetic sequences provided by the National Institutes of Health, USA.[3] The instance set consists of 40 DNA target sequences of length 109, 209, 309, 409, and 509 (alltogether 200 instances). Based on real hybridization experiments, the spectra were generated with probe size $l = 10$. All spectra contain 20% negative errors as well as 20% positive errors. For example, the spectra concerning the DNA target sequences of length 109 contain 100 oligonucleotides of which 20 oligonucleotides do not appear in the target sequences.

We applied the 4 heuristics outlined in the previous section to all problem instances. The results are shown in Tables 2 to 5. Each table contains the results of the corresponding heuristic averaged over the 40 problem instances of each of the five different sizes. The second row of each table contains the average solution quality (i.e., the average number of oligonucleotides in the constructed paths). Remember that the optimization objective in the SBH problem is to maximize this value. The third table row provides the number (out of 40) of solved problem instances, that is, the number of instances for which a path of maximal length could be found.[4] The fourth and fifth table row provide average similarity scores obtained by comparing the computed DNA sequences with the DNA target sequences. The average scores in the fourth table row are obtained from the Needleman-Wunsch algorithm, which is an algorithm for global alignment. In contrast, the average scores that are displayed in the fifth table row are obtained by the application of the Smith-Waterman algorithm, which is an algorithm for local alignment. Both algorithms were applied with the following parameters: +1 for a match of oligonucleotides, -1 for a mismatch or a gap. Finally, the sixth table row

[2]Note that the oligonucleotides of such a spectrum might have different lengths.

[3]The database access keys for all DNA target sequences are provided in [3].

[4]Remember in this context that an optimal solution to the SBH problem does not necesarilly correspond to a DNA sequence that is equal to the target sequence.



(a) (b)

Figure 2: Comparison of all existing constructive heuristics concerning (a) the global average similarity score obtained, and (b) the number of optimally solved instances. The comparison concerns the instances of Błażewicz et al. [3].

provides the average computation times for solving one instance (in seconds).

From the results that are displayed in Tables 2 to 5 we can draw the following conclusions. First, the results of FB-LAG improve in general over the results of LAG. This means that it is beneficial to allow the path construction in two directions (forward as well as backward). Second, the results of the SM heuristic are clearly better than both the results of LAG and the results of FB-LAG. However, the best results are obtained by the HSM heuristic, which is a hybrid between FB-LAG and SM. Even for the largest problem instances, the HSM heuristic produces sequences with very high similarity scores. In order to provide a comparison of all existing constructive heuristics we added the OW heuristic to this comparsion. This comparison is shown graphically in Figure 2. The results clearly show that HSM is currently the best available constructive heuristic. Finally, in Figure 3 we present a comparison between HSM and the best available metaheuristic approaches from the literature. The results are surprising: HSM is clearly better than the 4 metaheuristic approaches EA1, EA4, TS, and TS/SS. Furthermore, the results of HSM are—except for the problem instance of target sequence size 509—comparable to the results of the best metaheuristic approach EA2. Taking into account the advantage in computation time (i.e., HSM needs not even half a second to compute its results for the largest problem instances, while the meta-heuristics need between several seconds and several minutes) the HSM heuristic seems to be a good choice even when compared to metaheuristic approaches.

4 Conclusions and outlook to the future

In this work we have proposed new constructive heuristics for the problem of DNA sequencing by hybridization. First, we extended an existing heuristic. Then, we proposed a conceptionally new heuristic that is based on merging shorter DNA strands into bigger ones until a DNA strand of sufficient size is obtained. Finally we proposed a hybrid between both types of constructive heuristics. The results of this hybrid method show that it is the best constructive heuristic available to date. Furthermore, the results of our hybrid method are comparable to the results of the state-of-the-art metaheuristic. Only concerning the biggest problem instances our hybrid method is slight disadvantages. On the other side, our constructive heuristic need less computation time.

Table 2: Results of LAG for the instances by Błażewicz et al. [3].

Spectrum size	100	200	300	400	500
Average solution quality	76.98	153.53	230.68	309.03	383.08
Solved instances	23	15	12	7	4
Average similarity score (global)	77.05	133.63	171.78	206.80	218.60
Average similarity score (local)	91.83	152.43	209.33	272.40	293.48
Average computation time (sec)	0.0035	0.016	0.037	0.076	0.13

Table 3: Results of FB-LAG for the instances by Błażewicz et al. [3].

Spectrum size	100	200	300	400	500
Average solution quality	78.38	155.70	234.95	310.03	386.20
Solved instances	32	17	18	7	1
Average similarity score (global)	99.78	153.03	225.45	241.00	221.83
Average similarity score (local)	102.38	174.15	253.63	284.58	290.13
Average computation time (sec)	0.0051	0.022	0.054	0.11	0.19

Table 4: Results of SM for the instances by Błażewicz et al. [3].

Spectrum size	100	200	300	400	500
Average solution quality	79.75	157.80	234.90	306.90	367.38
Solved instances	38	31	30	28	18
Average similarity score (global)	106.33	195.85	284.68	357.98	376.25
Average similarity score (local)	107.20	203.03	293.75	377.00	416.68
Average computation time (sec)	0.005	0.02	0.046	0.082	0.13

Table 5: Results of HSM for the instances by Błażewicz et al. [3].

Spectrum size	100	200	300	400	500
Average solution quality	80.00	159.68	239.90	319.38	398.88
Solved instances	40	36	39	35	31
Average similarity score (global)	108.40	204.78	300.00	396.90	469.55
Average similarity score (local)	108.70	206.85	305.35	399.85	479.88
Average computation time (sec)	0.012	0.048	0.11	0.21	0.35

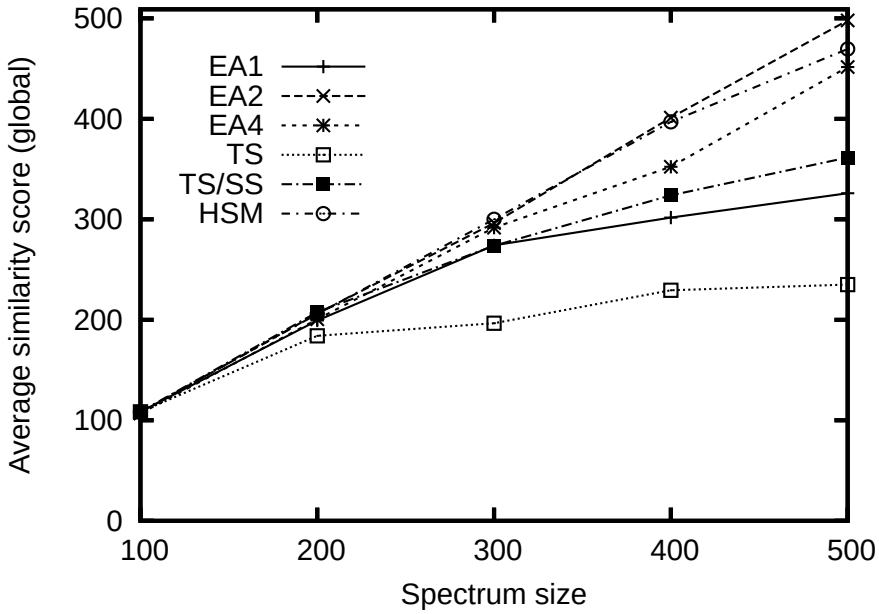


Figure 3: Comparison of HSM with all existing meta-heuristics except for EA3. The comparison is done concerning the global average similarity score obtained for the instances by Błażewicz et al. [3]. (b)

We believe that our new constructive technique (as implemented by the SM heuristic) can be used to develop metaheuristics that are superior to exisiting metaheuristics for DNA sequencing by hybridization. As a first step, existing metaheuristics might be applied to problem instances resulting from intermediate stages of the SM heuristic. This might improve their results and save much computation time.

References

[1] W. Bains and G. C. Smith. A novel method for nucleid acid sequence determination. *Journal of Theoretical Biology*, 135:303–307, 1988.

[2] J. Błażewicz, P. Formanowicz, F. Guinand, and M. Kasprzak. A heuristic managing errors for DNA sequencing. *Bioinformatics*, 18(5):652–660, 2002.

[3] J. Błażewicz, P. Formanowicz, M. Kasprzak, W. T. Markiewicz, and J. Weglarz. DNA sequencing with positive and negative errors. *Journal of Computational Biology*, 6:113–123, 1999.

[4] J. Błażewicz, P. Formanowicz, M. Kasprzak, W. T. Markiewicz, and J. Weglarz. Tabu search for DNA sequencing with false negatives and false positives. *European Journal of Operational Research*, 125:257–265, 2000.

[5] J. Błażewicz, F. Glover, and M. Kasprzak. DNA sequencing—Tabu and scatter search combined. *INFORMS Journal on Computing*, 16(3):232–240, 2004.

[6] J. Błażewicz, F. Glover, and M. Kasprzak. Evolutionary approaches to DNA sequencing with errors. *Annals of Operations Research*, 138:67–78, 2005.

[7] J. Błażewicz and M. Kasprzak. Complexity of DNA sequencing by hybridization. *Theoretical Computer Science*, 290(3):1459–1473, 2003.

[8] J. Błażewicz, M. Kasprzak, and W. Kuroczycki. Hybrid genetic algorithm for DNA sequencing with errors. *Journal of Heuristics*, 8:495–502, 2002.

[9] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

[10] C. A. Brizuela, L. C. González, and H. J. Romero. An improved genetic algorithm for the sequencing by hybridization problem. In G. R. Raidl, S. Cagnoni, J. Branke, D. Corne, R. Drechsler, Y. Jin, C. G. Johnson, P. Machado, E. Marchiori, F. Rothlauf, G. D. Smith, and G. Squillero, editors, *Proceedings of the EvoWorkshops – Applications of Evolutionary Computing: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC*, volume 3005 of *Lecture Notes in Computer Science*, pages 11–20. Springer Verlag, Berlin, Germany, 2004.

[11] T. N. Bui and W. A. Youssef. An enhanced genetic algorithm for DNA sequencing by hybridization with positive and negative errors. In K. Deb, R. Poli, W. Banzhaf, H.-G. Beyer, E. K. Burke, P. J. Darwen, D. Dasgupta, D. Floreano, J. A. Foster, M. Harman, O. Holland, P. L. Lanzi, L. Spector, A. Tettamanzi, D. Thierens, and A. M. Tyrrell, editors, *Proceedings of the GECCO 2004 – Genetic and Evolutionary Computation Conference*, volume 3103 of *Lecture Notes in Computer Science*, pages 908–919. Springer Verlag, Berlin, Germany, 2004.

[12] R. Drmanac, I. Labat, R. Brukner, and R. Crkvenjakov. Sequencing of megabase plus DNA by hybridization: Theory of the method. *Genomics*, 4:114–128, 1989.

[13] T. A. Endo. Probabilistic nucleotide assembling method for sequencing by hybridization. *Bioinformatics*, 20(14):2181–2188, 2004.

[14] E. R. Fernandes and C. C. Ribeiro. Using an adaptive memory strategy to improve a multistart heuristic for sequencing by hybridization. In S. E. Nikoletseas, editor, *Proceedings of WEA 2005 – 4th International Workshop on Experimental and Efficient Algorithms*, volume 3503 of *Lecture Notes in Computer Science*, pages 4–15. Springer Verlag, Berlin, Germany, 2005.

[15] Y. P. Lysov IuP, V. L. Florentiev, A. A. Khorlin, K. R. Khrapko, and V. V. Shik. Determination of the nucleotide sequence of DNA using hybridization with oligonucleotides. a new method. *Doklady Akademii nauk SSSR*, 303:1508–1511, 1988.

[16] P. A. Pevzner. l-tuple DNA sequencing: Computer analysis. *Journal of Biomulecular Structure and Dynamics*, 7:63–73, 1989.