Using an Adaptive Memory Strategy to Improve a Multistart Heuristic for Sequencing by Hybridization

Eraldo R. Fernandes[1] and Celso C. Ribeiro[2]

[1] Department of Computer Science, Catholic University of Rio de Janeiro
Rua Marquês de São Vicente 225, 22453-900 Rio de Janeiro, Brazil
eraldoluis@inf.puc-rio.br

[2] Department of Computer Science, Universidade Federal Fluminense
Rua Passo da Pátria 156, 24210-240 Niterói, Brazil
celso@ic.uff.br

Abstract. We describe a multistart heuristic using an adaptive memory strategy for the problem of sequencing by hybridization. The memory-based strategy is able to significantly improve the performance of memoryless construction procedures, in terms of solution quality and processing time. Computational results show that the new heuristic obtains systematically better solutions than more involving and time consuming techniques such as tabu search and genetic algorithms.

1 Problem formulation

A DNA molecule may be viewed as a word in the alphabet {A,C,G,T} of nucleotides. The problem of DNA sequencing consists in determining the sequence of nucleotides that form a DNA molecule. There are currently two techniques for sequencing medium-size molecules: gel electrophoresis and the chemical method. The novel approach of *sequencing by hybridization* offers an interesting alternative to those above [8, 9].

Sequencing by hybridization consists of two phases. The first phase is a biochemical experiment involving a DNA *array* and the molecule to be sequenced, i.e. the *target sequence*. A DNA array is a bidimensional grid in which each cell contains a small sequence of nucleotides which is called a *probe*. The set of all probes in a DNA array is denominated a *library*. Typically, a DNA array represented by $C(\ell)$ contains all possible probes of a fixed size ℓ. After the array has been generated, it is introduced into an environment with many copies of the target sequence. During the experiment, a copy of the target sequence reacts with a probe if the latter is a subsequence of the former. This reaction is called *hybridization*. At the end of the experiment, it is possible to determine which probes of the array reacted with the target sequence. This set of probes contains all sequences of size ℓ that appear in the target sequence and is called the *spectrum*. An illustration of the hybridization experiment involving the target sequence ATAGGCAGGA and $C(4)$ is depicted in Figure 1. The highlighted cells are those corresponding to the spectrum.

AAAA	AAAC	AAAG	AAAT	AACA	AACC	AACG	AACT	AAGA	AAGC	AAGG	AAGT	AATA	AATC	AATG	AATT
ACAA	ACAC	ACAG	ACAT	ACCA	ACCC	ACCG	ACCT	ACGA	ACGC	ACGG	ACGT	ACTA	ACTC	ACTG	ACTT
AGAA	AGAC	AGAG	AGAT	AGCA	AGCC	AGCG	AGCT	AGGA	AGGC	AGGG	AGGT	AGTA	AGTC	AGTG	AGTT
ATAA	ATAC	ATAG	ATAT	ATCA	ATGC	ATCG	ATCT	ATGA	ATGC	ATGG	ATGT	ATTA	ATTC	ATTG	ATTT
CAAA	CAAC	CAAG	CAAT	CACA	CACC	CACG	CACT	CAGA	CAGC	CAGG	CAGT	CATA	CATC	CATG	CATT
CCAA	CCAC	CCAG	CCAT	CCCA	CCCC	CCCG	CCCT	CCGA	CCGC	CCGG	CCGT	CCTA	CCTC	CCTG	CCTT
CGAA	CGAC	CGAG	CGAT	CGCA	CGCC	CGCG	CGCT	CGGA	CGGC	CGGG	CGGT	CGTA	CGTC	CGTG	CGTT
CTAA	CTAC	CTAG	CTAT	CTCA	CTCC	CTCG	CTCT	CTGA	CTGC	CTGG	CTGT	CTTA	CTTC	CTTG	CTTT
GAAA	GAAC	GAAG	GAAT	GACA	GACC	GACG	GACT	GAGA	GAGC	GAGG	GAGT	GATA	GATC	GATG	GATT
GCAA	GCAC	GCAG	GCAT	GCCA	GCCC	GCCG	GCCT	GCGA	GCGC	GCGG	GCGT	GCTA	GCTC	GCTG	GCTT
GGAA	GGAC	GGAG	GGAT	GGCA	GGCC	GGCG	GGCT	GGGA	GGGC	GGGG	GGGT	GGTA	GGTC	GGTG	GGTT
GTAA	GTAC	GTAG	GTAT	GTCA	GTCC	GTCG	GTCT	GTGA	GTGC	GTGG	GTGT	GTTA	GTTC	GTTG	GTTT
TAAA	TAAC	TAAG	TAAT	TACA	TACC	TACG	TACT	TAGA	TAGC	TAGG	TAGT	TATA	TATC	TATG	TATT

Fig. 1. Hybridization experiment involving the target sequence ATAGGCAGGA and all probes of size $\ell = 4$.

The second phase of the sequencing by hybridization technique consists in using the spectrum to determine the target sequence. The latter may be viewed as a sequence formed by all $n - \ell + 1$ probes in the spectrum, in which the last $\ell - 1$ letters of each probe coincide with the first $\ell - 1$ letters of the next. However, two types of errors may be introduced along the hybridization experiment. *False positives* are probes that appear in the spectrum, but not in the target sequence. *False negatives* are probes that should appear in the spectrum, but do not. A particular case of false negatives is due to probes that appear multiple times in the target sequence, since the hybridization experiment is not able to detect the number of repetitions of the same probe. Therefore, a probe appearing m times in the target sequence will generate $m - 1$ false negatives. The problem of sequencing by hybridization (SBH) is formulated as follows: given the spectrum S, the probe length ℓ, the size n and the first probe s_0 of the target sequence, find a sequence with length smaller than or equal to n containing a maximum number of probes. The maximization of the number of probes of the spectrum corresponds to the minimization of the number of errors in the solution. Errors in the spectrum make the reconstruction problem NP-hard [5].

An instance of SBH may be represented by a directed weighted graph $G(V, E)$, where $V = S$ is the set of nodes and $E = \{(u, v) \mid u, v \in S\}$ is the set of arcs. The weight of the arc (u, v) is given by $w(u, v) = \ell - o(u, v)$, where $o(u, v)$ is the size of the largest sequence that is both a suffix of u and a prefix of v. The value $o(u, v)$ is the *superposition* between probes u and v. A feasible solution to SBH is an acyclic path in G emanating from node s_0 and with total weight smaller than or equal to $n - \ell$. This path may be represented by an ordered node list $a =< a_1, \ldots, a_k >$, with $a_i \in S, i = 1, \ldots, k$. Let $S(a) = \{a_1, \ldots, a_k\}$ be the set of nodes visited by a path a and denote by $|a| = |S(a)|$ the number of nodes in this path. The latter is a feasible solution to SBH if and only if $a_1 = s_0$, $a_i \neq a_j$ for all $a_i, a_j \in S(a)$, and $w(a) \leq n - \ell$, where $w(a) = \sum_{h=1,\ldots,|a|-1} w(a_h, a_{h+1})$ is the sum of the weights of all arcs in the path. Therefore, SBH consists in finding a maximum cardinality path satisfying the above constraints.

The graph associated with the experiment depicted in Figure 1 is given in Figure 2 (a). The solution is a path visiting all nodes and using only unit weight arcs, since there are no errors in the spectrum. The example in Figure 2 (b) depicts a situation in which probe GGCA was erroneously replaced by probe GGCG, introducing one false positive and one false negative error. The new optimal solution does not visit all nodes (due to the false positive) and uses one arc with weight equal to 2 (due to the false negative).

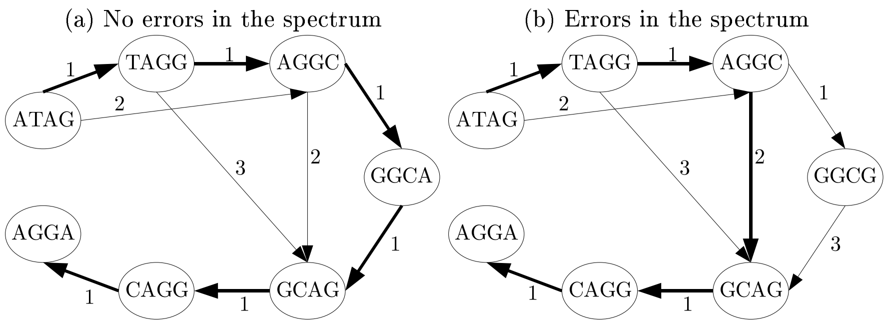


Fig. 2. Graphs and solutions for the target sequence ATAGGCAGGA with the probe size $\ell = 4$: (a) no errors in the spectrum, (b) one false positive error (GGCG) and one false negative error (GGCA) in the spectrum (not all arcs are represented in the graph).

Heuristics for SBH, handling both false positive and false negative errors, were proposed in [3, 4, 6]. We propose in the next section a new memory-based multistart heuristic for SBH, also handling both false positive and false negative errors. The algorithm is based on an adaptive memory strategy using a set of elite solutions visited along the search. Computational results illustrating the effectiveness of the new memory-based heuristic are reported in Section 3. Concluding remarks are made in the final section.

2 Memory-based multistart heuristic

The memory-based multistart heuristic builds multiple solutions using a greedy randomized algorithm. The best solution found is returned by the heuristic. An adaptive memory structure stores the best elite solutions found along the search, which are used within an intensification strategy [7].

The memory is formed by a pool Q that stores q elite solutions found along the search. It is initialized with q null solutions with zero probes each. A new solution a is a candidate to be inserted into the pool if $|a| > \min_{a' \in Q} |a'|$. This solution replaces the worst in the pool if $|a| > \max_{a' \in Q} |a'|$ (i.e., a is better than the best solution currently in the pool) or if $\min_{a' \in Q} dist(a, a') \geq d$, where d is a parameter of the algorithm and $dist(a, a')$ is the number of probes with different successors in a and a' (i.e., a is better than the worst solution currently in the pool and sufficiently different from every other solution in the pool).

The greedy randomized algorithm iteratively extends a path a initially formed exclusively by probe s_0. At each iteration, a new probe is appended at the end of the path a. This probe is randomly selected from the restricted candidate list $R = \{v \in S \setminus S(a) \mid o(u,v) \geq (1-\alpha) \cdot \max_{t \in S \setminus S(a)} o(u,t) \text{ and } w(a) + w(u,v) \leq n - \ell\}$, where u is the last probe in a and $\alpha \in [0,1]$ is a parameter. The list R contains probes with a predefined minimum superposition with the last probe in a, restricting the search to more promising regions of the solution space. The construction of a solution stops when R turns up to be empty.

The probability $p(u,v)$ of selecting a probe v from the restricted candidate list R to be inserted after the last probe u in the path a is computed using the superposition between probes u and v, and the frequency in which the arc (u,v) appears in the set Q of elite solutions. We define $e(u,v) = \lambda \cdot x(u,v) + y(u,v)$, where $x(u,v) = \min_{t \in S \setminus S(a)}\{w(u,t)/w(u,v)\}$ is higher when the superposition between probes u and v is larger, $y(u,v) = \sum_{a'' \in Q | (u,v) \in a''}\{|a''|/\max_{a' \in Q}|a'|\}$ is larger for arcs (u,v) appearing more often in the elite set Q, and λ is a parameter used to balance the two criteria. Then, the probability of selecting a probe v to be inserted after the last probe u in the path a is given by

$$p(u,v) = \frac{e(u,v)}{\sum_{t \in R} e(u,t)}.$$

The value of λ should be high in the beginning of the algorithm, when the information in the memory is still weak. The value of α should be small in the beginning, to allow for the construction of good solutions by the greedy randomized heuristic and so as to quickly enrich the memory. The value of α is progressively increased along the algorithm when the weight λ given to the superposition information decreases, to increase the diversity of the solutions in the list R.

We sketch in Figure 3 the pseudo-code with the main steps of the memory-based multistart heuristic, in which N iterations are performed.

3 Numerical results

The memory-based multistart heuristic was implemented in C++, using version 3.3.2 of the GNU compiler. The `rand` function was used for the generation of pseudo-random numbers. The computational experiments were performed on a 2.4 GHz Pentium IV machine with 512 MB of RAM.

Two sets of test instances have been generated from human and random DNA sequences. Instances in group A were built from 40 human DNA sequences obtained from GenBank [2], as described in [4]. Prefixes of size 109, 209, 309, 409, and 509 were extracted from these sequences. For each prefix, a hybridization experiment with the array $C(10)$ was simulated, producing spectra with 100, 200, 300, 400, and 500 probes. Next, false negatives were simulated by randomly removing 20% of the probes in each spectrum. False positives were simulated

```
procedure MultistartHeuristic(S, s_0, ℓ, n)
    Initialize o, w, α, q, d, Q;
    a* ← null;
    for i = 1 to N do
        Set a ← (s_0);
        Build the restricted candidate list R;
        while R ≠ ∅ do
            Compute the selection probability for each probe v ∈ R;
            Randomly select a probe v ∈ R;
            Extend the current solution a by appending v to its end;
            Update the restricted candidate list R;
        end;
        Use a to update the pool of elite solutions Q;
        if |a| > |a*| then set a* ← a;
    end;
    return a*;
end;
```

Fig. 3. Memory-based multistart heuristic.

by inserting 20% of new probes in each spectrum. Overall, we have generated 200 instances in this group, 40 of each size. Instances in group R were generated from 100 random DNA sequences with prefixes of size 100, 200, ..., and 1000. Once again, 20% false negatives and 20% false positives have been generated. There are 100 instances of each size in this group, in a total of 1000 instances.

Preliminary computational experiments have been performed to tune the main parameters of the algorithm. The following settings were selected: $N = 10n$ (number of iterations performed by the multistart heuristic), $q = n/80$ (size of the pool of elite solutions), and $d = 2$ (minimum difference for a solution to be accepted in the pool). Parameters α and λ used by the greedy randomized construction heuristic are self-tuned. Iterations of this heuristic are grouped in 20 blocks. Each block performs $n/2$ iterations. In the first block, $\lambda = 100q$. In the second block, $\lambda = 10q$. The value of λ is reduced by q at each new block, until it is made equal to zero. The value of α is initialized according to Tables 1 and 2, and increased by 0.1 after every five blocks of $n/2$ iterations, until it is made equal to one.

Table 1. Initial values of α for the instances in group R.

n	100	200	300	400	500	600	700	800	900	1000
α	0.5	0.3	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.0

Two versions of the `MultistartHeuristic` algorithm described in Figure 3 were implemented: MS is a purely multistart procedure that does not make use of memory, while MS+Mem fully exploits the adaptive memory strategy described

Table 2. Initial values of α for the instances in group A.

n	109	209	309	409	509
α	0.5	0.3	0.2	0.1	0.1

in the previous section. To evaluate the quality of the solutions produced by the heuristics, we performed the alignment of their solutions with the corresponding target sequences, as in [4]. The *similarity* between two sequences is defined as the fraction (in percent) of symbols that coincide in their alignment. A similarity of 100% means that the two sequences are identical. Average similarities and average computation times in seconds over all test instances in group R for both heuristics are displayed in Figure 4. These results clearly illustrate the contribution of the adaptive memory strategy to improve the performance of the purely multistart heuristic.

We have performed another experiment to further evaluate the influence of the adaptive memory strategy on the multistart heuristic. We illustrate our findings for one specific instance with size $n = 1000$ from group R. Figure 5 (a) displays the number of probes in the best solution obtained by each heuristic along 10000 iterations. We notice that the best solution already produced by MS+Mem until a given iteration is consistently better than that obtained by MS, in particular after a large number of iterations have been performed. Figure 5 (b) depicts the same results along 8.7 seconds of processing time. The purely multistart heuristic seems to freeze and prematurely converge to a local minimum very quickly. The use of the adaptive memory strategy leads the heuristic to explore other regions of the solution space and to find better solutions.

To give further evidence concerning the performance of the two heuristics, we used the methodology proposed by Aiex et al. [1] to assess experimentally the behavior of randomized algorithms. This approach is based on plots showing empirical distributions of the random variable *time to target solution value*. To plot the empirical distribution, we select a test instance, fix a target solution value, and run algorithms MS and MS+Mem 100 times each, recording the running time when a solution with cost at least as good as the target value is found. For each algorithm, we associate with the i-th sorted running time t_i a probability $p_i = (i - \frac{1}{2})/100$ and plot the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, 100$.

Since the relative performance of the two heuristics is quite similar over all test instances, we selected one particular instance of size $n = 500$ from group R and used its optimal value as the target. The computational results are displayed in Figure 6. This figure shows that the heuristic MS+Mem using the adaptive memory strategy is capable of finding target solution values with higher probability or in smaller computation times than the pure multistart heuristic MS, illustrating once again the contribution of the adaptive memory strategy. These results also show that the heuristic MS+Mem is more robust.

We have also considered the behavior of the heuristic MS+Mem when the number of errors and the size of the probes vary. The algorithm was run on randomly generated instances as those in group R, for different rates of false negative

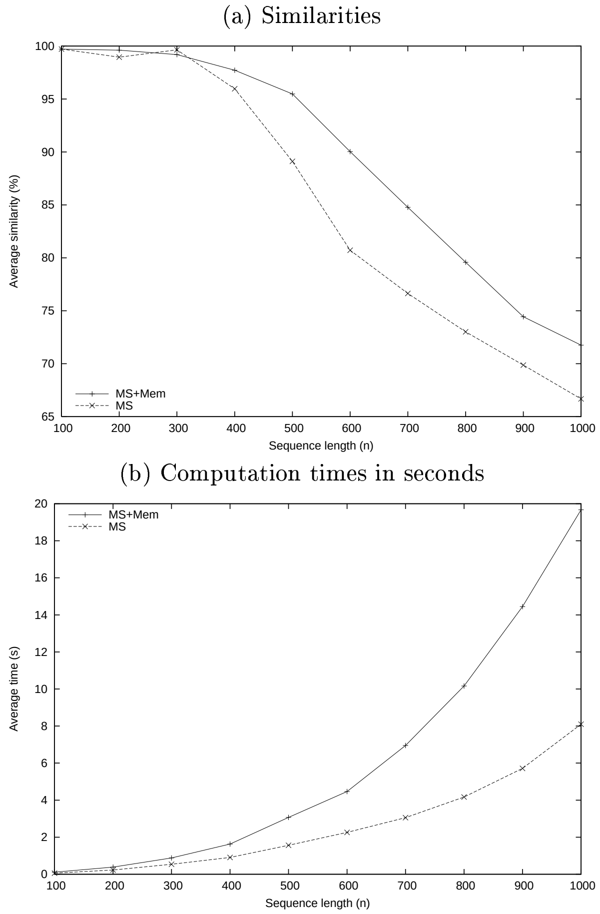
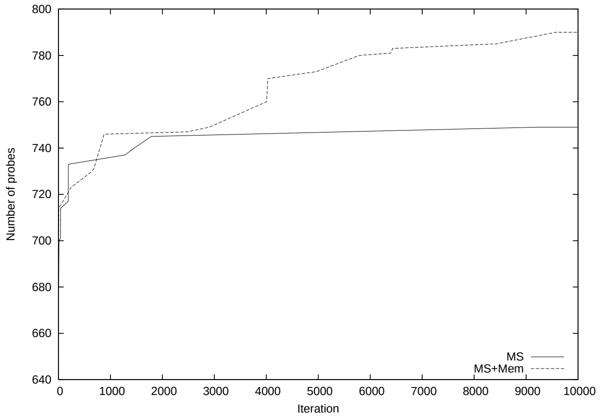


Fig. 4. Computational results obtained by heuristics MS+Mem and MS for the instances in group R.

and false positive errors: 0%, 10%, 20%, and 30%. Similarly, the algorithm was also run on randomly generated instances as those in group R with different probe sizes $\ell = 7, 8, 9, 10, 11$. Numerical results are displayed in Figure 7.

The memory-based multistart heuristic MS+Mem was compared with the tabu search algorithm (TS) in [4], the overlapping windows heuristic (OW) in [3], and the genetic algorithm (GA) in [6]. The numerical results are summarized in Tables 3 and 4, which depict the average similarities and the average computation times in seconds observed for each algorithm over the 40 instances with the same size in group A. The heuristic MS+Mem found much better solutions than the others. The alignments observed for the solutions produced by MS+Mem are systematically higher. The new heuristic MS+Mem is faster than TS and

(a) Best solutions along 10000 iterations



(b) Best solutions along 8.7 seconds of processing time

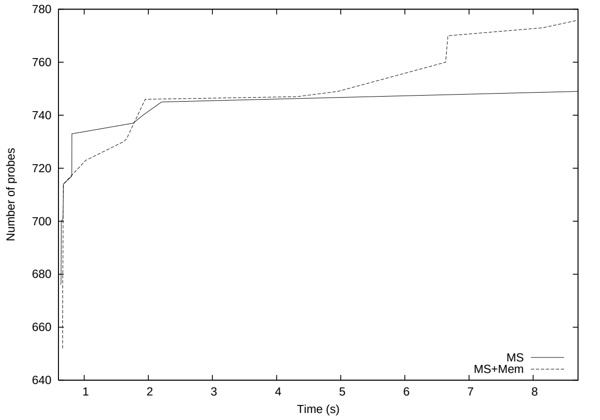


Fig. 5. Probes in the best solutions found by heuristics MS and MS+Mem for an instance with $n = 1000$ from group R.

competitive with GA (the results displayed for the overlapping windows heuristic were obtained on a CRAY T3E-900 supercomputer).

Further comparative results for the four algorithms are given in Table 5, in which we give the number of target sequences exactly reconstructed for each algorithm over the 40 instances with the same size in group A. The heuristic MS+Mem was able to reconstruct the 40 original sequences of size 109 and 209, and 39 out of the 40 instances of sizes 309, 409, and 509, corresponding to a total of 197 out of the 200 test instances in group A. The overlapping windows and the tabu search heuristics found, respectively, only 96 and 88 out of the 200 original sequences.

We also compared the new heuristic MS+Mem with the genetic algorithm for the instances in group R. Average similarities and average computation times in

Table 3. Average similarities for the instances in group A.

	n				
Algorithm	109	209	309	409	509
TS	98.6	94.1	89.6	88.5	80.7
OW	99.4	95.2	95.7	92.1	90.1
GA	98.3	97.9	99.1	98.1	93.5
MS+Mem	100.0	100.0	99.2	99.4	99.5

Table 4. Average computation times in seconds for the instances in group A.

	n				
Algorithm	109	209	309	409	509
TS	<1.0	5.0	14.0	28.0	51.0
OW	<1.0	<1.0	<1.0	<1.0	<1.0
GA	0.1	0.3	0.9	1.5	2.1
MS+Mem	0.1	0.4	0.8	1.6	3.0

Table 5. Target sequences exactly reconstructed for the instances in group A.

	n				
Algorithm	109	209	309	409	509
TS	28	23	17	10	10
OW	28	20	21	13	14
GA	37	30	37	30	28
MS+Mem	40	40	39	39	39

Table 6. Target sequences exactly reconstructed for the instances in group R.

	n									
Algorithm	100	200	300	400	500	600	700	800	900	1000
GA	70	61	55	37	23	11	9	3	1	2
MS+Mem	79	74	83	72	58	52	24	14	11	3

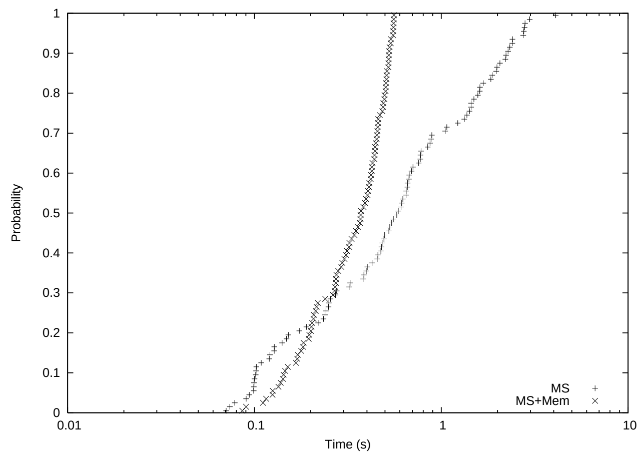


Fig. 6. Empirical probability distributions of time to target solution value for heuristics MS+Mem and MS for an instance of size $n = 500$ from group R.

seconds are shown in Figure 8. Table 6 depicts the number of target sequences exactly reconstructed by MS+Mem and the genetic algorithm over the 100 instances of each size in group R. Also for the instances in this group, the new heuristic outperformed the genetic algorithm both in terms of solution quality and computation times.

4 Concluding remarks

We proposed a multistart heuristic for the problem of sequencing by hybridization, based on an intensification strategy that makes use of an adaptive memory. The adaptive memory strategy makes use of a set of elite solutions found along the search. The choice of the new element to be inserted into the partial solution at each iteration of a greedy randomized construction procedure is based not only on greedy information, but also on frequency information extracted from the memory.

Computational results on test instances generated from human and random DNA sequences have shown that the memory-based strategy is able to significantly improve the performance of a memoryless construction procedure purely based on greedy choices. The memory-based multistart heuristic obtained better results than more involving and time consuming techniques such as tabu search and genetic algorithms, both in terms of solution quality and computation times.

The use of adaptive memory structures that are able to store information about the relative positions of the tasks in elite solutions seems to be particularly suited to scheduling problems in which blocks formed by the same tasks in the same order often appear in the best solutions.

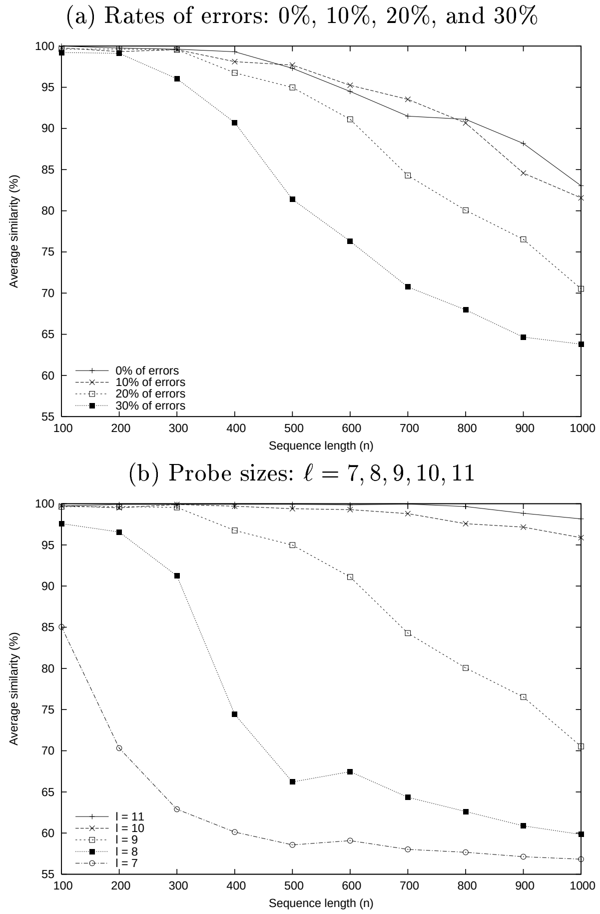


Fig. 7. Results obtained by the heuristic MS+Mem for instances with different rates of errors (a) and probe sizes (b).

References

1. R.M. Aiex, M.G.C. Resende, and C.C. Ribeiro. Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, 8:343–373, 2002.
2. D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, and D.L. Wheeler. Genbank: Update. *Nucleic Acids Research*, 32:D23–D26, 2004.
3. J. Blazewicz, P. Formanowicz, F. Guinand, and M. Kasprzak. A heuristic managing errors for DNA sequencing. *Bioinformatics*, 18:652–660, 2002.
4. J. Blazewicz, P. Formanowicz, M. Kasprzak, W. T. Markiewicz, and T. Weglarz. Tabu search for DNA sequencing with false negatives and false positives. *European Journal of Operational Research*, 125:257–265, 2000.
5. J. Blazewicz and M. Kasprzak. Complexity of DNA sequencing by hybridization. *Theoretical Computer Science*, 290:1459–1473, 2003.

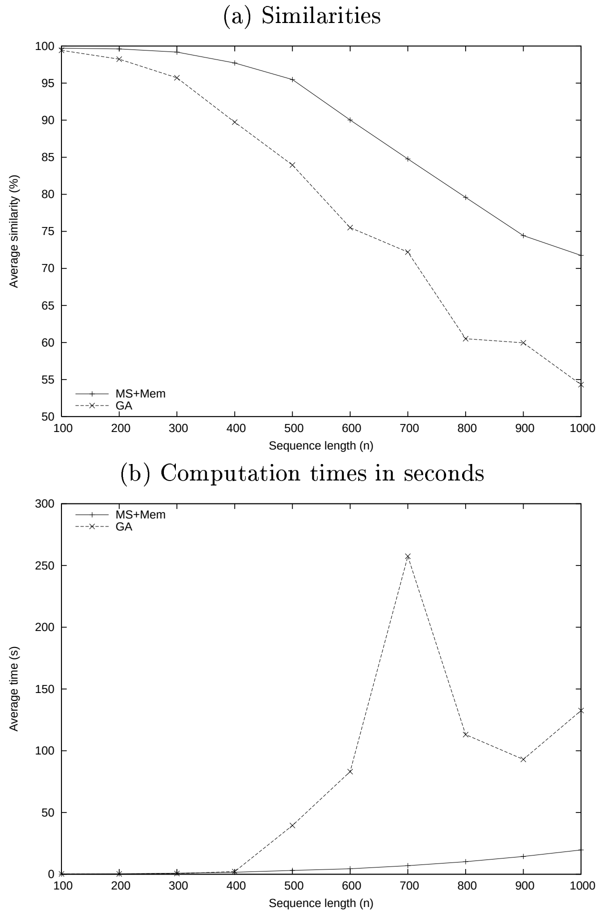


Fig. 8. Computational results obtained by the heuristic MS+Mem and the genetic algorithm (GA) for the instances in group R.

6. T.A. Endo. Probabilistic nucleotide assembling method for sequencing by hybridization. *Bioinformatics*, 20:2181–2188, 2004.
7. C. Fleurent and F. Glover. Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, 11:198–204, 1999.
8. P.A. Pevzner. *Computational molecular biology: An algorithmic approach.* MIT Press, 2000.
9. M.S. Waterman. *Introduction to computational biology: Maps, sequences and genomes.* Chapman & Hall, 1995.