# On the Far from Most String Problem, One of the Hardest String Selection Problems

**Daniele Ferone, Paola Festa, and Mauricio G.C. Resende**

**Abstract** This paper describes the *far from most string problem*, one of the computationally hardest string selection problems that has found its way into numerous practical applications, especially in computational biology and bioinformatics, where one is interested in computing distance/proximity among biological sequences, creating diagnostic probes for bacterial infections, and/or discovering potential drug targets.

With special emphasis on the optimization and operational research perspective, this paper studies the intrinsic properties of the problem and overviews the most popular solution techniques, including some recently proposed heuristic and metaheuristic approaches. Future directions are discussed in the last section.

**Keywords** Computational biology • Molecular structure prediction • String selection • Consensus • Combinatorial optimization • Metaheuristics

## 1 Introduction

The *far from most string problem* (FFMSP) is one of the *string selection* and *comparison problems*, also called *sequence consensus problems*.

Generally speaking, given a finite set of sequences, one is interested in finding their *consensus*, i.e., a new sequence that "agrees" as much as possible with all the given sequences. In other words, the objective is to determine a sequence called consensus, because it represents in some way all the given sequences.

D. Ferone • P. Festa (✉)
Department of Mathematics and Applications, University of Napoli FEDERICO II, Compl. MSA, Via Cintia, 80126 Napoli, Italy
e-mail: danieleferone@gmail.com; paola.festa@unina.it

M.G.C. Resende
AT&T Labs Research, Florham Park, NJ, USA
e-mail: mgcr@research.att.com



C. Vogiatzis et al. (eds.), *Dynamics of Information Systems*, Springer Proceedings in Mathematics & Statistics 105, DOI 10.1007/978-3-319-10046-3_7

The concept of being *representative* of a given set of sequences strongly depends on the specific objective pursued by the project of studying the information contained in the given sequences and their specific properties under experimentation. The most common objectives are listed in the following:

(i) the consensus is a new sequence whose total distance from all given sequences is minimum (*closest string problem*);
(ii) the consensus is a new sequence whose total distance from all given sequences is maximum (*farthest string problem*);
(iii) the consensus is a new sequence far from most of the given sequences (*FFMSP*).

String selection problems find thousands of applications in several and heterogenous fields, ranging from coding theory to molecular biology. A fundamental remark made by researchers in molecular biology regards the abstraction of the real three-dimensional structure of DNA and its representation as a unidimensional sequence of characters from an alphabet of four symbols. The same type of assumption involves also the protein represented as a sequence of characters from an alphabet of 20 symbols. As a result of the linear coding of DNA and proteins, many molecular biology problems have been formulated as computational and optimization problems involving strings and sequences, such as to rebuild long DNA sequences starting from overlapping fragments (fragment assembly), to compare two or more sequences looking for their similarities (strings coding the same function), and to look for patterns that occur with a certain frequency in DNA and/or protein sequences. Useless to say that many further targets can be pursued in molecular biology applications involving sequences. For example, another possible application arises in creating diagnostic probes for bacterial infections. Given a set of DNA sequences from a group of closely related pathogenic bacteria, the task is to find a substring that occurs in each of the bacterial sequences (as close as possible) without occurring in the host's DNA. Probes are then designed to hybridize to these target sequences, so that the detection of their presence indicates that at least one bacterial species is likely to be present in the host. Another biological application related to string selection and comparison problems is related to discovering potential drug targets. Given a set of sequences of orthologous genes from a group of closely related pathogens and a host (such as human, crop, or livestock), the goal is to find a sequence fragment that is more conserved in all or most of the pathogens' sequences but not as conserved in the host. Information encoded by this fragment can then be used for novel antibiotic development or to create a drug that harms several pathogens with minimal effect on the host. All these applications reduce to the task of finding a pattern that with some error occurs in one set of strings (closest string problem) and/or does not occur in another set (farthest string problem). The FFMSP can help to identify a sequence fragment that distinguishes the pathogens from the host, so the potential exists to create a drug that harms several but not all pathogens.

The remainder of this article is organized as follows. In Sect. 2, the FFMSP is described and its properties are analyzed. The most popular solution techniques

for this problem are surveyed in Sect. 3, along with the computational results obtained and analyzed in the literature. Concluding remarks and future directions are discussed in the last section.

## 2 Notation and Problem Description

Throughout this paper, the following notation and definitions will be used:

- An *alphabet* $\Sigma = \{c_1, c_2, \ldots, c_k\}$ is a finite set of elements, called *characters*.
- $s^i = (s^i_1, s^i_2, \ldots, s^i_m)$ is a sequence of length $m$ ($|s^i| = m$) on $\Sigma$ ($s^i_j \in \Sigma,\ j = 1, 2, \ldots, m$).
- Given two sequences $s^i$ and $s^l$ on $\Sigma$ such that $|s^i| = |s^l|$, $d_H(s^i, s^l)$ denotes their Hamming distance and is given by

$$d_H(s^i, s^l) = \sum_{j=1}^{|s^i|} \Phi(s^i_j, s^l_j), \tag{1}$$

  where $s^i_j$ and $s^l_j$ are the characters in position $j$ in $s^i$ and $s^l$, respectively, and $\Phi : \Sigma \times \Sigma \to \{0, 1\}$ is the predicate function such that

$$\Phi(a, b) = \begin{cases} 0, \text{ if } a = b; \\ 1, \text{ otherwise.} \end{cases}$$

- Given a set of sequences $\Omega = \{s^1, s^2, \ldots, s^n\}$ on $\Sigma$ ($s^i \in \Sigma^m, i = 1, 2, \ldots, n$) $d^{\Omega}_H$ denotes the Hamming distance among the sequences in $\Omega$ and it is given by

$$0 \leq d^{\Omega}_H = \min_{i,l=1,\ldots,n \mid i<l} d_H(s^i, s^l) \leq m. \tag{2}$$

Given a set of sequences $\Omega = \{s^1, s^2, \ldots, s^n\}$ on $\Sigma$ ($s^i \in \Sigma^m, i = 1, 2, \ldots, n$), the FFMSP consists in determining a string far from most of the strings in $\Omega$. This can be formally stated by saying that given a threshold $t$, a string $s \in \Sigma^m$ must be found maximizing the variable $x$ such that

$$d_H(s, s^i) \geq t,\ \forall\, s^i \in P \subseteq \Omega \text{ and } |P| = x, \tag{3}$$

or, equivalently

$$d^{P \cup \{s\}}_H \geq t, \text{ for } P \subseteq \Omega \text{ and } |P| = x. \tag{4}$$

For most consensus problems, Hamming distance (1) is used instead of any alternative measure (such as the editing distance) and biological reasons justifying this choice are very well described and motivated by Lanctot et al. in [24].

The FFMSP is one of the computationally hardest sequence consensus problems. The intractability of the general sequence consensus problem was proved in 1997 by Frances and Litman [16] and in 1999 by Sim and Park [31]. In 2003, Lanctot et al. [25] demonstrated that for sequences over an alphabet $\Sigma$ with $|\Sigma| \geq 3$, approximating the FFMSP within a polynomial factor is NP-hard.

## 3 Several Alternative State-of-the-Art Algorithms for the FFMSP

Since polynomial time algorithms for approaching the FFMSP can yield only solutions with no constant guarantee of approximation, heuristic methods must be devised to find good-quality solutions in reasonable running times. This section overviews main heuristic/metaheuristic algorithms to efficiently find good suboptimal solutions for the FFMSP, starting from the first attempt done in 2005 by Meneses et al. [27] to the latter proposed techniques in 2013 by Ferone et al. [5, 6], who designed several pure and hybrid metaheuristics.

### *3.1 A Simple Heuristic Approach*

The first attempt in trying to obtain good approximate solutions in reasonable running times has been done by Meneses et al. [27], who in 2005 proposed a simple heuristic consisting of the following two phases:

Phase I: A *construction phase* that iteratively builds a feasible solution $s \in \Sigma^m$. Initially,

- for each position $j \in \{1, \ldots, m\}$, compute the set $V_j$ of characters appearing in that position in any of the $n$ strings in $\Omega$;
- for each character $c \in V_j$, compute the number of times that $c$ appears in the input on position $j$.

Then, for each position $j \in \{1, \ldots, m\}$, a string $s$ is iteratively built by choosing the character in $V_j$ that appears in the smallest number of strings.
For each position $j > 1$, check the effect that assigning a character to this position will have on previous assignments.

Phase II: A *local search phase* that starting from $s$ explores a suitably defined *neighborhood* of $s$ (a set of feasible solutions "close" to $s$) until a local optimum is found and returned as final solution.

```
algorithm iter-impr (c, s, 𝒩)
1   NoLocal := true;
2   while (NoLocal) do
3         /* exploration of the neighborhood N(s) */
4         if (∃ s̄ ∈ N(s) : c's̄ > c's, c's̄ ≥ c'y, ∀ y ∈ N(s)) then
5             s := s̄;
6         else NoLocal := false;
7         endif;
8   endwhile;
9   return (s);
end iter-impr
```

**Fig. 1** Local search procedure `iter-impr` for a maximization problem

Figure 1 shows the pseudo-code of the simplest local search procedure, known as *iterative improvement*. It takes as input a cost vector $c$, an initial solution $s$, and a predefined neighborhood function $N$. Starting from $s$, the iterative improvement procedure explores the neighborhood $N(s)$ looking for a better solution $\overline{s}$ in terms of objective function value. If such a solution exists, then the search continues from $\overline{s}$; otherwise, the procedure provides as output the current solution $s$ which is locally optimal with respect to the defined neighborhood.

Meneses et al. [27] proposed a 2-*exchange* local search procedure, whose basic step consists in randomly selecting a position $j \in \{1, \ldots, m\}$ and changing it to another character in $V_j$ selected at random.

## 3.2 A GRASP

Meneses et al.'s algorithm has been the first attempt in the design of heuristic approaches for the FFMSP. It is basically a greedy construction of a feasible solution followed by a local search procedure to generate a local optimal solution.

The second step along this research line has been to design a *multi-start* or *iterative process* as conceived in [26]. In general, in a multi-start technique, a solution is built either only once and usually at the beginning of the solution process or it is built at each iteration or "start" of the algorithm. In the first case, the unique solution built by the approach can be constructed applying any criterion, ranging from a pure greedy to a pure random strategy. Conversely, when a solution is built at each start of the algorithm, then a pure greedy strategy should not be followed since each time the pure greedy construction is invoked it would lead to the same solution or to a set of "close" solutions.

```
algorithm GRASP (f(·), 𝒩, Seed)
1  s_best:=∅; f(s_best):=−∞;
2  while stopping criterion not satisfied→
3        s:=GreedyRandomAdapBuild(Seed);
4        s:=LocalSearch(s, 𝒩, f(·));
5        if (f(s) > f(s_best)) then
6           s_best:=s;
7        endif
8  endwhile
9  return (s_best);
end GRASP
```

**Fig. 2** Pseudo-code of a generic GRASP for a maximization problem

The first multi-start iterative process designed for the FFMSP appeared in 2007 in [7], where a GRASP (greedy randomized adaptive search procedure) and a genetic algorithm have been proposed.

Originally proposed in the literature by Feo and Resende [3,4], in the last 20 years GRASP has been successfully applied to several computationally intractable combinatorial problems. The reader interested in a comprehensive study of GRASP strategies and variants is referred to the survey chapter by Resende and Ribeiro [29] and the more recent articles by Festa and Resende [11–13], as well as to the annotated bibliography of Festa and Resende [8–10].

Figure 2 depicts the pseudo-code of a generic GRASP heuristic for a maximization problem. GRASP is an iterative multi-start heuristic algorithm that for a certain number of iterations realizes two phases (loop while in lines 2–8): a construction phase (line 3) and a local search phase (line 4). Similarly to the semi-greedy heuristic proposed independently by [22], the basic GRASP construction phase starts from an empty solution and iteratively adds one element at a time to the partial solution under construction, ending up with a representation of a feasible solution. At each iteration, an element is randomly selected from a *restricted candidate list* (RCL), whose elements are among the best ordered, according to some greedy function that measures the (myopic) benefit of selecting each element. Once a feasible solution is obtained, the local search procedure attempts to improve it by producing a locally optimal solution with respect to a predefined neighborhood structure. Construction and local search phases are repeatedly applied until stopping criterion is met and the best local optimal solution found over all iterations is returned as output.

A GRASP construction phase generally makes use of an adaptive greedy function for constructing the RCL and of a probabilistic selection criterion of a well-ranked element from the restricted list. In the case of the FFMSP, it is intuitive to relate the greedy function to the occurrence of each character in a given position. In fact, as in [27], for each position $j \in \{1, \ldots, m\}$, it is computed as the set $V_j$ of characters appearing in that position in any of the strings in $\Omega$ and then for each character $c \in V_j$, $g_j(c)$ is computed as the number of times that $c$ appears in the input on position $j$. Starting from an empty solution, at each construction iteration the choice of the next element to be added to the partial solution is determined by ordering all candidate characters in a candidate list $C$ with respect to the above defined greedy function. The probabilistic component of the GRASP here proposed is characterized by *randomly* choosing one of the best candidates in the list, but not necessarily the top candidate. As in any GRASP heuristic, the construction procedure is *adaptive*, because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element.

There are several different mechanisms to build the RCL. Typically, it can be limited by the number of elements (cardinality-based criterion) or by their quality (value-based criterion). If the cardinality-based criterion is chosen, then the cardinality of RCL is a priori fixed to some $p$ and the RCL is made up of those elements having the $p$ best greedy function values, while in the value-based case, the cardinality of RCL depends on a threshold parameter $0 \leq \alpha \leq 1$. In the GRASP proposed in [7], at each iteration $j \in \{1, \ldots, m\}$ of the construction procedure, RCL is formed by all possible candidates $y$ whose greedy function value $g_j(y)$ is better or equal to $\alpha \cdot g_j^*$, where $g_j^*$ is the best greedy function value. Note that the extreme case $\alpha = 0$ corresponds to a pure greedy strategy, while the extreme case $\alpha = 1$ is equivalent to a completely random strategy.

To realize the local search phase the 2-exchange procedure is used as in [27]. The procedure takes as input the solution $s = (s_1, \ldots, s_m) \in \Sigma^m$ built at the end of the GRASP construction phase and $\{V_j\}_{j=1,\ldots,m}$, where $V_j$ is the set of characters appearing in position $j$ in any of the strings in $\Omega$. Then, for each position $j = 1, \ldots, m$ and for each character $c \in V_j$, $c \neq s_j$, the 2-exchange procedure checks if the solution $\overline{s} = (s_1, \ldots, s_{j-1}, c, s_{j+1}, \ldots, s_m)$ obtained from $s$ exchanging the character in position $j$ is better than $s$ in terms of objective function value. Note that $\overline{s}$ is a neighbor of solution $s$ such that $d_H(s, \overline{s}) = 1$.

A local search may be implemented using either a *best improving* or a *first improving strategy*. The first improving strategy stops the current iteration as soon as an improving neighbor is found, while in the best improving strategy all neighbors are evaluated and the best among them is kept as new current solution from which to start the next iteration. In [7], both strategies have been implemented and, accordingly with results from the literature about these different possible strategies, the best improving produces better-quality solutions for most of the problem instances but in a higher amount of time as compared to the first improving strategy.

As any multi-start iterative heuristic, stopping criteria in a GRASP could be maximum number of iterations, maximum number of iterations without improvement of the incumbent solution, maximum running time, or solution quality at least as good as a given target value. In the GRASP for the FFMSP proposed in [7], the adopted stopping criterion has been a maximum number of iterations.

### *3.3 A New Solution Evaluation Function*

Starting from a given solution to the problem under studying, any local search procedure explores a suitable neighborhood set of solutions "close" to the starting solution and outputs a locally optimal solution with respect to the used neighborhood structure definition. In exploring the neighborhood set, a local search needs to evaluate many candidate solutions in order to compare them. The most used function to perform this evaluation is the objective function, so that a solution $s$ is better than a different solution $\overline{s}$ if and only if the objective function evaluated in $s$ produces a value strictly better than the value assumed by the objective function when evaluated in $\overline{s}$.

This way of investigating the neighborhood of a solution is basically a *steepest descend/ascend* process that presents serious drawbacks when the search landscape includes many local optimal solutions. Unfortunately, this is exactly the case of the FFMSP, because the set of possible objective function values is $\{0, 1, \ldots, n\}$ and is therefore rather small. To overcome this limit, Mousavi et al. [28] in 2012 proposed to use in the GRASP local search of Festa [7] an alternative solution evaluation function that takes into account both the classical objective function and the so-called *estimated Gain-per-Cost* heuristic function that expresses the likelihood of a solution to lead to better solutions with as few changes as possible.

The authors compared the original Festa's GRASP with their proposal on both randomly generated and real-world problem instances and as result of their experiments the variant of the GRASP they proposed overcomes in terms of solutions quality the original GRASP in all cases.

### *3.4 Hybrid and Pure Metaheuristics*

An issue from the heuristic/metaheuristic research community involves the idea of combining the main characteristics of pure metaheuristic frameworks in the attempt to take advantage of their best properties in terms of total running times and/or solution quality. Following this recent trend, Ferone et al. in [5] have designed the following pure and hybrid metaheuristics for finding good-quality solutions to the FFMSP:

- a pure GRASP, inspired by [7];
- a GRASP that uses Path-relinking for intensification;

**algorithm** GRASP $(m, \Sigma, f(\cdot), \{V_j(c)\}_{j\in\{1,\ldots,m\}}^{c\in\Sigma}$, Seed)
1 $s_{best}$:=$\emptyset$; $f(s_{best})$:=$-\infty$;
2 **for** $j = 1$ **to** $m\rightarrow$
3 $V_j^{\min}$:=$\min_{c\in\Sigma} V_j(c)$; $V_j^{\max}$:=$\max_{c\in\Sigma} V_j(c)$;
4 **endfor**
5 **while** stopping criterion not satisfied$\rightarrow$
6 $[s, \{\mathrm{RCL}_j\}_{j=1}^{m}]$:=GrRand $(m, \Sigma, \{V_j(c)\}_{j\in\{1,\ldots,m\}}^{c\in\Sigma}, V_j^{\min}, V_j^{\max}$, Seed);
7 $s$:=LocalSearch $(m, s, f(\cdot), \{\mathrm{RCL}_j\}_{j=1}^{m})$;
8 **if** $(f(s) > f(s_{best}))$ **then**
9 $s_{best}$:=$s$;
10 **endif**
11 **endwhile**
12 **return** $(s_{best})$;
**end** GRASP

**Fig. 3** Pseudo-code of a GRASP for the FFMSP

- a pure variable neighborhood search (VNS);
- a VNS that uses Path-relinking for intensification;
- a GRASP that uses VNS to implement the local search phase; and
- a GRASP that uses VNS to implement the local search phase and Path-relinking for intensification.

### 3.4.1 A Pure GRASP

The pure GRASP proposed in [5] has been inspired by the GRASP proposed in 2007 [7], but presents some different details in the design of its main ingredients. Figure 3 depicts its pseudo-code, where $f : \Sigma^m \mapsto \mathbb{N}$ denotes the objective function to be maximized according to (3) and (4).

As any GRASP framework and as the GRASP proposed in [7], also the GRASP proposed in [5] for the FFMSP proceeds for a certain number of iterations. At each iteration, it builds a solution sequence $s$, starting from which to look for a locally optimal solution with respect to a predefined neighborhood structure. In the following, the main ingredients of both construction and local search procedures are detailed.

The operations performed during the construction phase are described in Fig. 4, where a sequence $s = (s_1, \ldots, s_m) \in \Sigma^m$ is iteratively built, one character at each iteration. As described in [7], the greedy function is related to the occurrence of each character in a given position. In more detail, for each position $j \in \{1, \ldots, m\}$ and for each character $c \in \Sigma$, the number $V_j(c)$ of times $c$ appears in position $j$ in any of the strings in $\Omega$ is computed. Then, to build the RCL, let

```
function GrRand (m, Σ, {V_j(c)}_{j∈{1,...,m}}^{c∈Σ}, V_j^min, V_j^max, Seed)
1  for j = 1 to m→
2        RCL_j:=∅; α:=Random([0,1], Seed);
3        μ:=V_j^min + α·(V_j^max − V_j^min);
4        for all c ∈ Σ→
5            if (V_j(c) ≤ μ)) then
6                RCL_j:=RCL_j∪{c};
7            endif
8        endfor
9        s_j:=Random (RCL_j, Seed);
10 endfor
11 return (s, {RCL_j}_{j=1}^m);
end GrRand
```

**Fig. 4** Pseudo-code of the GRASP construction for the FFMSP

$$V_j^{\min} = \min_{c \in \Sigma} V_j(c), \qquad V_j^{\max} = \max_{c \in \Sigma} V_j(c).$$

Denoting by $\mu = V_j^{\min} + \alpha \cdot (V_j^{\max} - V_j^{\min})$ the cutoff value (line 3), where $\alpha$ is a parameter such that $0 \leq \alpha \leq 1$ (line 2), the RCL is built by selecting as its members all characters whose greedy function value is not greater than $\mu$ (line 6). Once the RCL is built, character is then randomly selected from it (line 9) (Fig. 4).

The GRASP local search proposed in [5] and presented in Fig. 5 analyzes all positions $j \in \{1, \ldots, m\}$ (loop in lines 4–14) and changes the character in position $j$ in the sequence $s$ to another character in $\mathrm{RCL}_j$. During the local search process, the current solution is replaced by the first improving neighbor (lines 8–11). The search in the neighborhood of the current solution stops after all possible moves have been evaluated and no improving neighbor of the current solution was found, returning a local optimal solution (line 16).

### 3.4.2 A Pure VNS

In [5], a pure VNS has been designed for the FFMSP. Contrary to other metaheuristics based on local search methods, given a solution $s$, VNS [21] is based on the exploration of increasingly distant neighborhoods $N_k(s)$, $k = 1, \ldots, k_{\max}$, until some stopping condition is satisfied. Figure 6 reports the pseudo-code of a VNS for the FFMSP, as proposed in [5].

At any VNS iteration a solution $S$ is built at random (line 3). Then, in loop lines 4–13, increasingly distant neighborhoods $N_k(s)$, $k = 1, \ldots, k_{\max}$, are explored by applying the same local search strategy used within the pure GRASP

```
function LocalSearch (m, s, f(·), {RCL_j}_{j=1}^{m})
1   max:=f(s); change:=.TRUE.;
2   while (change)→
3         change:=.FALSE.;
4         for j = 1 to m→
5             temp:=sj;
6             for all c ∈RCLj→
7                 sj:=c;
8                 if ( f(s) > max) then
9                     max:=f(s);temp:=c;change:=.TRUE.;break;
10                endif
11            endfor
12            sj:=temp;
13        endfor
14  endwhile
15   return(s);
end LocalSearch
```

**Fig. 5** Pseudo-code of the GRASP local search for the FFMSP

```
algorithm VNS (m, Σ , f(·), k_max, Seed)
1   s_best :=∅; f(s_best):=−∞;
2   while stopping criterion not satisfied→
3         k:=1; s:=BuildRand(m,Σ, Seed); /* pure randomly */
4         while (k ≤ k_max)→
5             s′:=Random (N_k(s), Seed);
6             s″:=locsearch (m, s′, f(·), {RCL_j}_{j=1}^{m});
7             if (f(s″) > f(s)) then
8                 s:=s″; k:=1;
9                 if (f(s″) > f(s_best)) then s_best:=s″;
10                endif
11            else k:=k+1;
12            endif
13        endwhile
14  endwhile
15  return (s_best);
end VNS
```

**Fig. 6** Pseudo-code of a VNS for the FFMSP

algorithm described in Sect. 3.4.1. For the FFMSP, the $k$th-order neighborhood $N_k(s)$ is the set of all sequences that can be derived from the current sequence $s$ by selecting $k$ positions $j_1, \ldots, j_k$ and changing $s_{j_1}, \ldots, s_{j_k}$ with a character in $\mathrm{RCL}_{j_1}, \ldots, \mathrm{RCL}_{j_k}$, respectively.

### 3.4.3 Path-Relinking

Path-relinking is an intensification strategy exploring trajectories connecting elite solutions. It was proposed in 1996 by Glover [17] and later hybridized with tabu search and scatter search [18–20]. Generally speaking, starting from one elite solution, Path-relinking generates a path in the solution space leading towards another guiding elite solution. This path connecting the two solutions in the solution space is built by selecting moves that introduce in the initial solution attributes contained in the guiding solution. At each iteration, all moves are analyzed and the move that best improves (or least deteriorates) the initial solution is chosen. The scope of building this path is to explore it in the search for better solutions.

In [5], Path-relinking is applied to a pair of sequences $(\mathbf{s}, \hat{\mathbf{s}})$, where $\mathbf{s}$ is a given input solution and $\hat{\mathbf{s}}$ is a solution *sufficiently different* from $\mathbf{s}$ selected at random (line 1) from an elite set $\mathscr{E}$ of solutions that has a fixed size that does not exceed `MaxElite`.

$\mathbf{s}$ and $\hat{\mathbf{s}}$ have been retained sufficiently different if $|\Delta(\mathbf{s}, \hat{\mathbf{s}})| \geq \frac{m}{2}$, where $\Delta(\mathbf{s}, \hat{\mathbf{s}})$ is their the symmetric difference set and it is clearly defined as follows:

$$\Delta(\mathbf{s}, \hat{\mathbf{s}}) := \{i = 1, \ldots, m \mid \mathbf{s}_i \neq \hat{\mathbf{s}}_i\}. \tag{5}$$

Roughly speaking, $\Delta(\mathbf{s}, \hat{\mathbf{s}})$ is the set of components for which the two solutions differ. Once known and selected the sequences $\mathbf{s}$ and $\hat{\mathbf{s}}$, a path is explored in the solution space linking the worst solution $\mathbf{s}'$ between $\mathbf{s}$ and $\hat{\mathbf{s}}$ to the best one (line 3). $\mathbf{s}'$ is called the *initial solution* and $\hat{\mathbf{s}}$ the *guiding solution* (Fig. 7).

The procedure then computes (line 4) the symmetric difference $\Delta(\mathbf{s}', \hat{\mathbf{s}})$ between the two solutions as defined in Eq. (5), i.e., the set of moves needed to reach $\hat{\mathbf{s}}$ from $\mathbf{s}'$. A path of solutions $\mathbf{s}'_1, \mathbf{s}'_2, \ldots, \mathbf{s}'_{|\Delta(\mathbf{s}', \hat{\mathbf{s}})|}$ is generated linking $\mathbf{s}'$ and $\hat{\mathbf{s}}$. The best solution $s^*$ in this path is returned by the algorithm (line 13).

The path of solutions is computed in the loop in lines 5 through 12. This is achieved by advancing one solution at a time in a greedy manner. At each iteration, all possible moves $i \in \Delta(\mathbf{s}', \hat{\mathbf{s}})$ are examined. Then, the best move $i^*$ is made, producing solution $\mathbf{s}' \oplus i^*$ (line 8) and, if necessary, the best solution $\mathbf{s}^*$ is updated (lines 9–11 ). The algorithm stops as soon as $\Delta(\mathbf{s}', \hat{\mathbf{s}}) = \emptyset$.

### 3.4.4 Hybrid GRASP with Path-Relinking

Hybridizations of GRASP with Path-relinking have been proposed in the literature to incorporate into a general GRASP framework some sort of *memory mechanisms*.

```
algorithm Path-relinking(m, f(·),s,ℰ,Seed)
1   ŝ:=Random(ℰ,Seed);
2   f* := max{f(s), f(ŝ)}; s* := argmax{f(s), f(ŝ)};
3   s′:= argmin{f(s), f(ŝ)}; ŝ:=s*;
4   Δ(s′, ŝ):=:={i=1,...,m |s′_i≠ŝ_i};
5   while (Δ(s′, ŝ)≠∅) →
6         i* := argmax{f(s′ ⊕ i) | i ∈ Δ(s′,ŝ)};
7         Δ(s′ ⊕ i*,ŝ) := Δ(s′,ŝ) \ {i*};
8         s′ := s′ ⊕ i*;
9         if(f(s′) > f*) then
10           f* := f(s′); s* := s′;
11        endif
12  endwhile
13  return (s*);
end Path-relinking
```

**Fig. 7** Pseudo-code of a path-relinking for the FFMSP

The first attempt along this direction has been done by Laguna and Martí [23] in 1999. This attempt has been successful followed by several further extensions and improvements [2, 11, 13–15].

In [5], the authors have integrated Path-relinking into the pure GRASP algorithm described in Sect. 3.4.1. In more detail, at each GRASP iteration, Path-relinking is applied to a pair $(\mathbf{s}, \hat{\mathbf{s}})$ of solutions, where $\mathbf{s}$ is the locally optimal solution obtained by GRASP local search and $\hat{\mathbf{s}}$ is randomly chosen from a pool with a limited number `MaxElite` of high-quality solutions found along the search. The pseudo-code for the proposed GRASP with Path-relinking hybrid algorithm is shown in Fig. 8.

The pool $\mathscr{E}$ of elite solutions is originally empty (line 1) and it is then populated by the first `MaxElite` GRASP iteration local optima solutions. After `MaxElite` iteration, the best solution $\overline{s}$ found along the relinking trajectory is considered as a candidate to be inserted into $\mathscr{E}$. In more detail, the procedure `AddToElite` inserts $\overline{s}$ into the pool replacing the worst elite solution if $\overline{s}$ is better than the best elite solution or if it is better than the worst elite solution and *sufficiently different* (i.e., if $|\Delta(\overline{s}, \epsilon)| \geq \frac{m}{2}$, for all $\epsilon \in \mathscr{E}$) from all elite solutions.

### 3.4.5 Hybrid GRASP with VNS

As underlined in Sect. 3.4.2, until a stopping criterion is met, VNS generates at each iteration a sequence $s$ at random. In the hybrid GRASP with VNS designed in [5], VNS is applied as local search and its starting solution is the sequence $s$ output of the GRASP construction procedure.

```
algorithm GRASP+PR (m, Σ, f(·), {V_j(c)}^{c∈Σ}_{j∈{1,...,m}}, Seed, MaxElite)
1  s_best:=∅; f(s_best):=−∞; ℰ := ∅; iter:=0;
2  for j = 1 to m→
3        V_j^min:=min_{c∈Σ} V_j(c); V_j^max:=max_{c∈Σ} V_j(c);
4  endfor
5  while stopping criterion not satisfied→
6        iter:=iter + 1;
7        [s, {RCL_j}^m_{j=1}]:=GrRand (m, Σ, {V_j(c)}^{c∈Σ}_{j∈{1,...,m}}, V_j^min, V_j^max, Seed);
8        s:=LocalSearch (m, s, f(·), {RCL_j}^m_{j=1});
9        if (iter ≤ MaxElite) then
10           ℰ := ℰ ∪ {s};
11           if (f(s) > f(s_best)) then s_best:=s;
13           endif
14       else
10           s̄:=Path-relinking (m, f(·), s, ℰ, Seed);
15           AddToElite (ℰ,s̄);
11           if (f(s̄) > f(s_best)) then s_best:=s̄;
13           endif
13       endif
11 endwhile
12 return (s_best);
end GRASP+PR
```

**Fig. 8** Pseudo-code of a hybrid GRASP with path-relinking for the FFMSP

### 3.4.6 Hybrid VNS with Path-Relinking

VNS described in Sect. 3.4.2 has been hybridized with Path-relinking applied as an intensification procedure at each VNS iteration. In the pair of solutions $(\mathbf{s}, \hat{\mathbf{s}})$ to which Path-relinking is applied, $\mathbf{s}$ is the locally optimal solution while $\hat{\mathbf{s}}$ is randomly chosen from the `MaxElite` high-quality solutions.

### 3.4.7 Hybrid GRASP with VNS and Path-Relinking

The main blocks of each iteration of this hybrid heuristic are the GRASP construction procedure, followed by VNS local search phase, and Path-relinking as intensification procedure. See [5] for a detailed description of the resulting hybrid algorithm.

### 3.4.8 Experimental Results and Recent Trends

In [5], the above described pure and hybrid metaheuristic approaches have been experimentally evaluated to determine which algorithm seems to be more effective to solve the FFMSP.

The objectives of the computational study were to compare the running times and the solution qualities achieved by the several alternative pure and hybrid algorithms when applied to solve FFMSP instances pseudo-randomly generated and characterized by several different sizes. In fact, in the set of test instances, the sequence length $m$ ranged from 300 to 800, the number $n$ of sequences in $\Omega$ ranged from 100 to 200, and threshold $t$ varied from $75\,\%m$ to $85\,\%m$. The stopping criterion for all algorithms was $\texttt{MaxIterations} = 500$ or the obtainment of an incumbent solution with objective function value $z = n$ (i.e., an optimal solution).

For each problem size, 100 random instances have been generated for each possible value of $t \in \{75\,\%m, 80\,\%m, 85\,\%m\}$. Each algorithm has been run on the 100 random instances and average solution values and the corresponding average running times (in seconds) were computed. For a detailed analysis of the experimental evaluation of the different algorithms, the reader can refer to [5]. Here, about the experiments carried by the authors, we report only the main conclusions which are the following:

- on all instances, better-quality solutions have been found by the hybrid GRASP with VNS and Path-relinking;
- at the expense of increased running times, both hybridizations of Path-relinking with all different metaheuristics and the use of VNS in the local search phase of GRASP have been beneficial in terms of solution quality.

Given the random component of each proposed algorithm and since their running times per iteration vary substantially, in [5] a further experiment has been carried, involving the empirical distributions of the random variable *time-to-target-solution-value* considering the following four random instances:

1. $n = 100, m = 300, t = 240$, and target value $\hat{z} = 0.70 \times n$ (Fig. 9a);
2. $n = 100, m = 300, t = 252$, and target value $\hat{z} = 0.12 \times n$ (Fig. 9b);
3. $n = 200, m = 300, t = 240$, and target value $\hat{z} = 0.40 \times n$ (Fig. 10a);
4. $n = 300, m = 300, t = 240$, and target value $\hat{z} = 0.28 \times n$ (Fig. 10b).

100 independent runs of each heuristic have been performed and the time taken to find a solution at least as good as the target value $\hat{z}$ has been saved. As in [1], to plot the empirical distribution, with the $i$th sorted running time ($t_i$) a probability $p_i = \frac{i-1/2}{100}$ is associated, and the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, 100$, are then plotted. About these further experiments, looking at Figs. 9 and 10, the authors concluded that, in a fixed amount of running time, the hybrid GRASP with Path-relinking has a higher probability than all competitors of finding a solution whose objective function value is at least as good as the target objective function value.

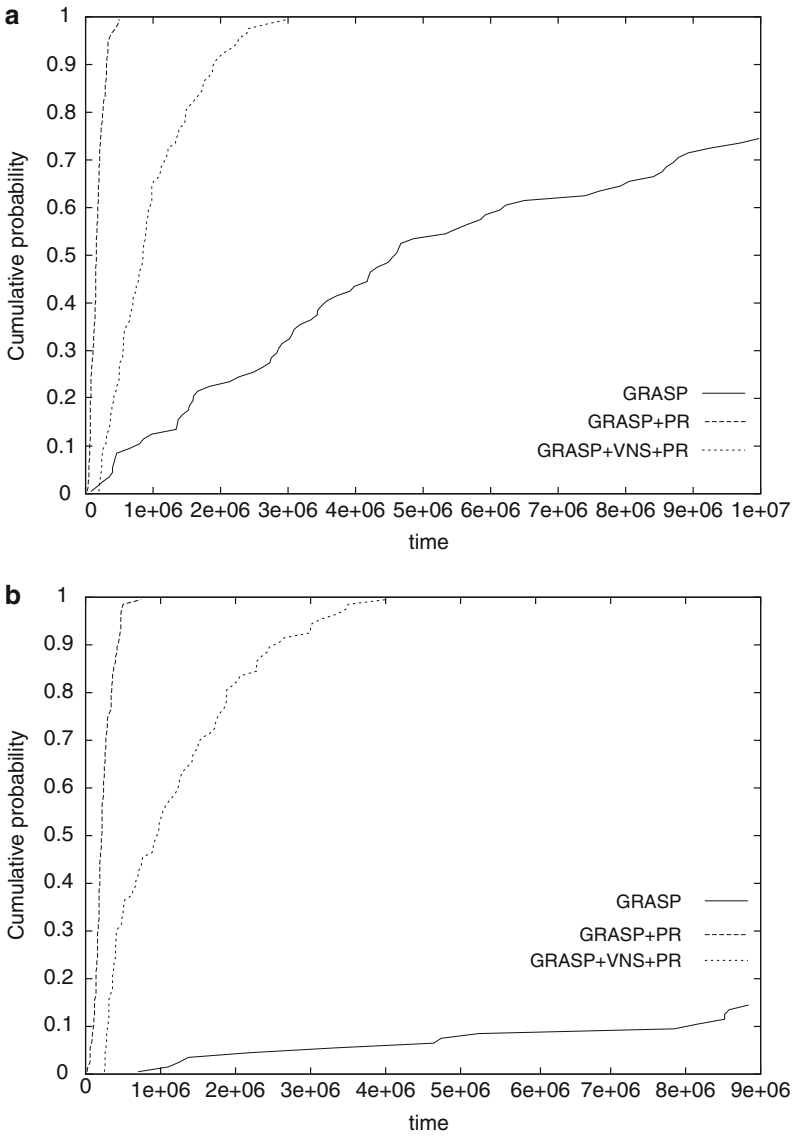


**Fig. 9** Time to target distributions comparing `GRASP`, `GRASP+PR`, and `GRASP+VNS+PR` (**a**) Random instance with $n = 100$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.70 \times n$ (**b**) Random instance with $n = 100$, $m = 300$, $t = 252$, and target value $\hat{z} = 0.12 \times n$

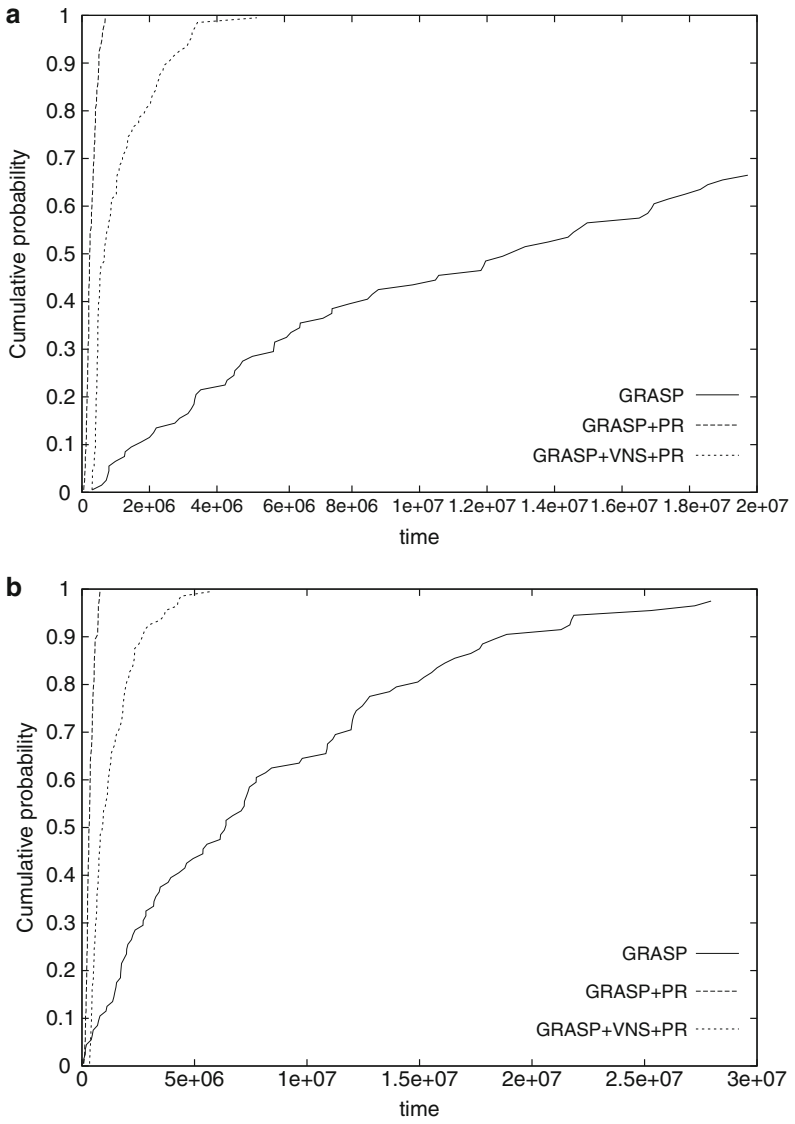


**Fig. 10** Time to target distributions comparing `GRASP`, `GRASP+PR`, and `GRASP+VNS+PR` (**a**) Random instance with $n = 200$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.40 \times n$ (**b**) Random instance with $n = 300$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.28 \times n$

## 4 Conclusions and Future Directions

The *FFMSP*—as well as all consensus problems—occurs as a problem and/or as a subproblem in many real-world scenarios, especially in computational biology and bioinformatics.

Among consensus problems, the FFMSP is one of the computationally hardest. For this reason, contrary to other problems in the same family, only very recently efficient, fast, and robust solution techniques have been designed and proposed in the scientific literature. With special emphasis on the optimization and operational research perspective, this paper surveyed the most popular solution techniques, starting from the first ingenuous and naive heuristic attempt by Meneses et al. [27] in 2005 and ending with recently proposed heuristic and metaheuristic approaches [5, 28] that appeared in 2012 and 2013.

In [6], as current and immediately future work, we are performing the following steps:

1. To better understand the practical behavior of the algorithms proposed in [5], we are analyzing the numerical results by applying a recently published tool designed by Ribeiro et al. [30] for characterizing stochastic algorithms' running times under the assumption that the running times of the algorithms follow exponential (or shifted exponential) distributions, as it is the case of our hybrid heuristics.
2. We are collecting and interpreting further numerical results, by applying the heuristics proposed in [5] on a larger dataset of instances, both randomly generated and taken from real-world applications of the problem.
3. We are designing some further variants of the approaches proposed in [5]. Three natural extensions are

   - to perform a post-optimization phase, consisting in performing Path-relinking among pairs of elite solutions;
   - to implement alternative Path-relinking strategies, known as backward, mixed, and randomized Path-relinking;
   - to integrate in the local search of the algorithms Mousavi et al.'s function [28].

In addition, thinking about future research it would also be interesting to propose some further metaheuristic approaches, analyzing the possibility of designing some sophisticated nature-inspired techniques, whose main ingredients could be ad hoc tuned for this problem.

## References


1. Aiex, R.M., Resende, M.G.C., Ribeiro, C.C.: Probability distribution of solution time in GRASP: an experimental investigation. J. Heuristics **8**, 343–373 (2002)
2. Canuto, S.A., Resende, M.G.C., Ribeiro, C.C.: Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks **38**, 50–58 (2001)
3. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. **8**, 67–71 (1989)
4. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. Global Optim. **6**, 109–133 (1995)
5. Ferone, D., Festa, P., Resende, M.G.C.: Hybrid metaheuristics for the far from most string problem. In: Proceedings of 8th International Workshop on Hybrid Metaheuristics, Lecture Notes in Computer Science, Springer, vol. 7919, pp. 174–188 (2013)
6. Ferone, D., Festa, P., Resende, M.G.C.: Hybrid metaheuristics for the far from most string problem. Technical Report, Department of Mathematics and Applications, University of Napoli FEDERICO II (2013)
7. Festa, P.: On some optimization problems in molecular biology. Math. Biosc. **207**(2), 219–234 (2007)
8. Festa, P., Resende, M.G.C.: GRASP: an annotated bibliography. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and Surveys on Metaheuristics, pp. 325–367. Kluwer Academic Publishers, Norwell (2002)
9. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP: part I: algorithms. Int. Trans. Oper. Res. **16**(1), 1–24 (2009)
10. Festa, P., Resende, M.G.C.: An annotated bibliography of GRASP: part II: applications. Int. Trans. Oper. Res. **16**(2), 131–172 (2009)
11. Festa, P., Resende, M.G.C.: Hybrid GRASP heuristics. Stud. Comput. Intell. **203**, 75–100 (2009)
12. Festa, P., Resende, M.G.C.: GRASP: basic components and enhancements. Telecommun. Syst. **46**(3), 253–271 (2011)
13. Festa, P., Resende, M.G.C.: Hybridizations of GRASP with path-relinking. Stud. Comput. Intell. **434**, 135–155 (2013)
14. Festa, P., Pardalos, P.M., Resende, M.G.C., Ribeiro, C.C.: Randomized heuristics for the MAX-CUT problem. Optim. Meth. Software **7**, 1033–1058 (2002)
15. Festa, P., Pardalos, P.M., Pitsoulis, L.S., Resende, M.G.C.: GRASP with path-relinking for the weighted MAXSAT problem. ACM J. Exp. Algorithmics **11**, 1–16 (2006)
16. Frances, M., Litman, A.: On covering problems of codes. Theor. Comput. Syst. **30**(2), 113–119 (1997)
17. Glover, F.: Tabu search and adaptive memory programing: advances, applications and challenges. In: Barr, R.S., Helgason, R.V., Kennington, J.L. (eds.) Interfaces in Computer Science and Operations Research, pp. 1–75. Kluwer Academic Publishers, New York (1996)
18. Glover, F.: Multi-start and strategic oscillation methods: principles to exploit adaptive memory. In Laguna, M., Gonzáles-Velarde, J.L. (eds.) Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, pp. 1–24. Kluwer Academic Publishers, Norwell (2000)
19. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
20. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Contr. Cybern. **39**, 653–684 (2000)
21. Hansen, P., Mladenović, N.: Developments of variable neighborhood search. In: Ribeiro, C.C., Hansen, P. (eds.) Essays and Surveys in Metaheuristics, pp. 415–439. Kluwer Academic Publishers, Norwell (2002)

22. Hart, J.P., Shogan, A.W.: Semi-greedy heuristics: an empirical study. Oper. Res. Lett. **6**, 107–114 (1987)
23. Laguna, M., Martí, R.: GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS J. Comput. **11**, 44–52 (1999)
24. Lanctot, J., Li, M., Ma, B., Wang, S., Zhang, L.: Distinguishing string selection problems. Information and Computation, Elsevier **185**(1), 41–55 (2003)
25. Lanctot, J., Li, M., Ma, B., Wang, S., Zhang, L.: Distinguishing string selection problems. Inf. Comput. **185**(1), 41–55 (2003)
26. Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. **21**, 498–516 (1973)
27. Meneses, C.N., Oliveira, C.A.S., Pardalos, P.M.: Optimization techniques for string selection and comparison problems in genomics. IEEE Eng. Med. Biol. Mag. **24**(3), 81–87 (2005)
28. Mousavi, S.R., Babaie, M., Montazerian, M.: An improved heuristic for the far from most strings problem. J. Heuristics **18**, 239–262 (2012)
29. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (eds.) State-of-the-Art Handbook of Metaheuristics. Kluwer Academic Publishers, Norwell (2002)
30. Ribeiro, C.C., Rosseti, I., Vallejos, R.: Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms. J. Global Optim. **54**, 405–429 (2012)
31. Sim, J.S., Park, K.: The consensus string problem for a metric is $NP$-complete. J. Discrete Algorithms. **1**(1), 111–117 (2003)