HYBRID METAHEURISTICS FOR THE FAR FROM MOST STRING PROBLEM

DANIELE FERONE, PAOLA FESTA, AND MAURICIO G.C. RESENDE

Abstract. Among the sequence selection and comparison problems, the *Far From Most String Problem* (FFMSP) is one of the computationally hardest with applications in several fields, including molecular biology where one is interested in creating diagnostic probes for bacterial infections or in discovering potential drug targets.

In this article, several hybrid metaheuristics are described and tested. Extensive comparative experiments on a large set of randomly generated test instances indicate that these randomized hybrid techniques are both effective and efficient.

1. The Far From Most String Problem (FFMSP)

The FFMSP is one of the so called *string selection and comparison problems*, that belong to the more general class of problems known as *sequences consensus*, where a finite set of sequences is given and 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. For the FFMSP, the objective is to find a sequence that is far from as many as possible sequences of a given set of sequences having all the same length.

To formally state the problem, the following notation is needed:

- 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 Σ ($s^i_j \in \Sigma,\ j = 1, 2, \ldots, m$);
- given two sequences s^i and s^l on Σ 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

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

Date: May 2013.

Key words and phrases. Computational biology, Molecular structure prediction, Protein and sequences alignment, Combinatorial optimization, Hybrid metaheuristics.

AT&T Labs Research Technical Report. Published in "Proceedings of Hybrid Metaheuristics 2013 (HM 2013)," Ischia, M.J. Blesa et al., (Eds.), *Lecture Notes in Computer Science*, vol. 7919, pp. 174-188, 2013.

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

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

The FFMSP consists in determining a sequence far from as many as possible sequences in the input set Ω. This can be formalized by saying that given a threshold t, a string s must be found maximizing the variable x such that

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

or, equivalently

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

Computational intractability of the general sequences consensus problem was first proved in 1997 by Frances and Litman [11] and in 1999 by Sim and Park [22]. Among sequences consensus problems, the FFMSP is one of the hardest from a computational point of view, as proved in 2003 by Lanctot et al. [18], who demonstrated that for sequences over an alphabet Σ with $|\Sigma| \geq 3$, approximating the FFMSP within a polynomial factor is NP-hard.

Given theoretical computational hardness results, polynomial time algorithms for the FFMSP can yield only solutions with no constant guarantee of approximation. In such cases, to find good quality solutions in reasonable running times and to overcome the inner intractability of the problem from a computational point of view, heuristic methods must be devised. The first attempt in this direction was done in 2005 by Meneses et al. [19], who proposed a heuristic algorithm consisting of a simple greedy construction followed by an iterative improvement phase. Later, in 2007 Festa [5] designed a GRASP and very recently in 2012, Mousavi et al. [20] devised a new function to be used in alternative to the objective function when evaluating neighbor solutions during the local search phase.

In this paper, we designed, implemented, and tested several pure and hybrid metaheuristics. The scope of the hybrid metaheuristics designing has been to combine the main characteristics of the pure metaheuristics themselves in the attempt to take advantage of their best properties in terms of computation time and solution quality.

The remainder of this article is organized as follows. In Section 2, we propose various randomized heuristics for finding approximate solutions of the FFMSP, based on the instantiation of several metaheuristics and their hybrids. Computational results are reported in Section 3. Concluding remarks and insights about further improvements of the proposed techniques are given in the last section.

2. Hybrid metaheuristics

In the last decades, a considerable amount of scientific papers has empirically shown that suitable combinations of concepts and characteristics from different metaheuristics can lead to the design of hybrid robust techniques that produce higher solution quality than the individual metaheuristics themselves, especially when solving difficult real-world combinatorial optimization problems.

Following this trend, to find good quality solutions to the FFMSP, we have considered several types of hybridizations. In particular, we have designed, implemented, and tested the following pure and hybrid multistart iterative heuristics:

- ◇ a pure GRASP, inspired by [5];
- ◇ a GRASP that uses Path-relinking for intensification;
- ◇ a pure 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.

As any multistart iterative heuristic, stopping criteria in all the above listed techniques 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.

algorithm GRASP(t, m, Σ, $f_t(\cdot)$, $\{V_j(c)\}_{j\in\{1,\ldots,m\}}^{c\in\Sigma}$, Seed)
1 s_{best}:=$\emptyset$; $f_t(s_{best})$:=$-\infty$;
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($m$, Σ, $\{V_j(c)\}_{j\in\{1,\ldots,m\}}^{c\in\Sigma}$, $V_j^{\min}$, $V_j^{\max}$, Seed);
7 s:=LocalSearch(t, m, s, $f_t(\cdot)$, $\{\mathrm{RCL}_j\}_{j=1}^m$);
8 **if** $(f_t(s) > f_t(s_{best}))$ **then**
9 s_{best}:=s;
10 **endif**
11 **endwhile**
12 **return**(s_{best});
end GRASP

FIGURE 1. Pseudo-code of a GRASP for the FFMSP.

2.1. **A pure GRASP.** Each GRASP iteration consists of a construction phase [3, 4], where a solution is built in a *greedy*, *randomized*, and *adaptive* manner, and a local search phase which starts at the constructed solution and applies iterative improvement until a locally optimal solution is found. Repeated applications of the construction procedure yields diverse starting solutions for the local search and the best overall local optimal solution is returned as the result. The reader can refer to [8, 9, 10] for a study of a generic GRASP metaheuristic framework and its applications.

A complete solution is iteratively constructed in the construction phase, one element at a time. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list C with respect to a greedy function that measures the (myopic) benefit of selecting each element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best

```
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
```

FIGURE 2. Pseudo-code of the GRASP construction for the FFMSP.

candidates is called the *restricted candidate list* (RCL). For the FFMSP, in 2007 [5] a GRASP has been proposed to find suboptimal solutions for the FFMSP and Figure 1 depicts its pseudo-code, where $f_t : \Sigma^m \mapsto \mathbb{N}$ denotes the objective function to be maximized according to (3) and (4).

Figure 2 reports the pseudo-code of the construction procedure that iteratively builds a sequence $s = (s_1, \ldots, s_m) \in \Sigma^m$, selecting one character at time. The greedy function is related to the occurrence of each character in a given position. In fact, as in [5], for each position $j \in \{1, \ldots, m\}$ and for each character $c \in \Sigma$, we compute $V_j(c)$ as the number of times c appears in position j in any of the strings in Ω. The pure greedy choice would consist in selecting the character c with the lowest greedy function value $V_j(c)$. To define the construction mechanism for the RCL, let

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

Denoting by $\mu = V_j^{\min} + \alpha \cdot (V_j^{\max} - V_j^{\min})$ the cut-off value (line 3), where α is a parameter such that $0 \leq \alpha \leq 1$ (line 2), the RCL is made up by all characters whose value of the greedy function is less than or equal to μ (line 6). A character is then randomly selected from the RCL (line 9).

The basic step of the local search described in Figure 3 is slightly different from the one implemented in [5]. In our GRASP, it consists in investigating all positions $j \in \{1, \ldots, m\}$ (loop in lines 4–14) and changing the character in position j in the sequence s to another character in RCL_j. Instead, in [5] the position j and the new character in position j are selected at random. Moreover, the random selection of the new character in position j involves the set of all characters occurring in that position in all the given sequences in Ω.

The current solution is replaced by the first improving neighbor (lines 8–11). The search stops after all possible moves have been evaluated and no improving neighbor was found, returning a local optimal solution (line 16).

2.2. **A pure VNS.** Contrary to other metaheuristics based on local search methods, VNS [16] is based on the exploration of a dynamic neighborhood model. It explores increasingly distant neighborhoods of the current best found solution.

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

FIGURE 3. Pseudo-code of the GRASP local search for the FFMSP.

Let N_k, $k = 1, \ldots, k_{max}$, be a set of pre-defined neighborhood structures and let $N_k(s)$ be the set of solutions in the kth-order neighborhood of a solution s. In the first phase, a neighbor $s' \in N_k(s)$ of the current solution is applied. Next, a solution s'' is obtained by applying local search to s'. Finally, the current solution jumps from s to s'' in case the latter improved the former. Otherwise, the order of the neighborhood is increased by one and the above steps are repeated until some stopping condition is satisfied.

```
algorithm VNS(t, m, Σ, f_t(·), k_max, Seed)
1  s_best:=∅; f_t(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'':=LocalSearch(t, m, s', f_t(·), {RCL_j}_{j=1}^m);
7      if (f_t(s'') > f_t(s)) then
8        s:=s''; k:=1;
9        if (f_t(s'') > f_t(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
```

FIGURE 4. Pseudo-code of a VNS for the FFMSP.

```
algorithm Path-relinking(t, m, f_t(·), s, E, Seed)
1   ŝ:=Random(E, Seed);
2   f* := max{f_t(s), f_t(ŝ)}; s* := arg max{f_t(s), f_t(ŝ)};
3   s' := arg min{f_t(s), f_t(ŝ)}; ŝ:=s*;
4   Δ(s', ŝ):={i = 1, ..., m | s'_i ≠ ŝ_i};
5   while (Δ(s', ŝ) ≠ ∅) →
6         i* := arg max{f_t(s' ⊕ i) | i ∈ Δ(s', ŝ)};
7         Δ(s' ⊕ i*, ŝ) := Δ(s', ŝ) \ {i*};
8         s' := s' ⊕ i*;
9         if (f_t(s') > f*) then
10              f* := f_t(s'); s* := s';
11        endif;
12  endwhile;
13  return (s*);
end Path-relinking
```

FIGURE 5. Pseudo-code of a Path-relinking for the FFMSP.

In the case of the FFMSP, the kth-order neighborhood is defined by 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. The same local search strategy used within the pure GRASP algorithm described in Section 2.1 is used in the VNS heuristic, whose pseudo-code is reported in Figure 4.

2.3. **Path-relinking.** Path-relinking is a heuristic proposed in 1996 by Glover [12] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [13, 14, 15].

Starting from one or more elite solutions, paths in the solution space leading towards other guiding elite solutions are generated and explored in the search for better solutions. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions. At each iteration, all moves that incorporate attributes of the guiding solution are analyzed and the move that best improves (or least deteriorates) the initial solution is chosen.

Figure 5 illustrates the pseudo-code of the Path-relinking for the FFMSP. It is applied to a pair of sequences $(s, \hat{s})$, where s is a given input solution and $\hat{s}$ is a solution (*sufficiently different* from s – see Section 3) selected at random (line 1) from an elite set $\mathcal{E}$ of solutions that has a fixed size that does not exceed `MaxElite`. Their common elements are kept constant, and the space of solutions spanned by these elements is searched with the objective of finding a better solution. This search is done by exploring a path in the solution space linking the worst solution s' between s and $\hat{s}$ to the best one (line 3). s' is called the *initial solution* and $\hat{s}$ the *guiding solution*.

The procedure then computes (line 4) the symmetric difference $\Delta(s', \hat{s})$ between the two solutions as the set of components for which the two solutions differ:

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

Note that, $|\Delta(s', \hat{s})| = d_H(s', \hat{s})$ and $\Delta(s', \hat{s})$ represents the set of moves needed to reach $\hat{s}$ from s', where a move applied to the initial solution s' consists in selecting a position $i \in \Delta(s', \hat{s})$ and replacing s'_i with $\hat{s}_i$.

Path-relinking generates a path of solutions $s'_1, s'_2, \ldots, s'_{|\Delta(s',\hat{s})|}$ linking s' and $\hat{s}$. 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, the procedure examines all moves $i \in \Delta(s', \hat{s})$ from the current solution s' and selects the one which results in the highest cost solution (line 6), i.e. the one which maximizes $f_t(s' \oplus i)$, where $s' \oplus i$ is the solution resulting from applying move i to solution s'. The best move i^* is made, producing solution $s' \oplus i^*$ (line 8). The set of available moves is updated (line 7). If necessary, the best solution s^* is updated (lines 9–11). Clearly, the algorithm stops as soon as $\Delta(s', \hat{s}) = \emptyset$.

2.4. **Hybrid GRASP with Path-relinking.** Since GRASP iterations are independent of one another, it does not make use of solutions produced throughout the search. One way to add memory to GRASP is its hybridization with Path-relinking. In 1999 the first proposal of such a hybrid method was published by Laguna and Martí [17]. It was followed by several extensions, improvements, and successful applications [2, 6, 7].

Into the pure GRASP algorithm described in Section 2.1 we have integrated Path-relinking applied at each GRASP iteration to pairs $(s, \hat{s})$ of solutions, where s is the locally optimal solution obtained by GRASP local search and $\hat{s}$ 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 Figure 6.

The pool of elite solutions is originally empty (line 1). The best solution $\overline{s}$ found along the relinking trajectory is considered as a candidate to be inserted into this pool. If the pool is not full ($|\mathcal{E}| \leq$ `MaxElite`), the candidate is simply inserted. Otherwise, if the pool is full, the procedure `AddToElite` evaluates its insertion into $\mathcal{E}$. In more detail, if $\overline{s}$ is better than the best elite solution, then $\overline{s}$ replaces the worst elite solution. If the candidate is better than the worst elite solution, but not better than the best, it replaces the worst if it is *sufficiently different* (see Section 3) from all elite solutions.

2.5. **Hybrid GRASP with VNS.** As underlined in Subsection 2.2, until a stopping criterion is met, at each iteration VNS chooses a neighbor sequence s from the neighborhood of the current solution at random. In our hybrid GRASP with VNS, VNS is applied as local search and its starting solution is the sequence s output of the GRASP construction procedure.

2.6. **Hybrid VNS with Path-relinking.** As is the case for GRASP, VNS described in Section 2.2 also can be hybridized with Path-relinking, as intensification phase. At each VNS iteration Path-relinking is applied to pairs $(s, \hat{s})$ of solutions, where s is the locally optimal solution obtained by VNS and $\hat{s}$ is randomly chosen from the `MaxElite` high quality solutions found along the search.

2.7. **Hybrid GRASP with VNS and Path-relinking.** This hybrid procedure is simply obtained by replacing the local search phase of the GRASP procedure

```
algorithm GRASP+PR(t, m, Σ, f_t(·), {V_j(c)}^{c∈Σ}_{j∈{1,...,m}}, Seed, MaxElite)
1  s_best:=∅; f_t(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(t, m, s, f_t(·), {RCL_j}^m_{j=1});
9    if (iter ≤ MaxElite) then
10       E := E ∪ {s};
11       if (f_t(s) > f_t(s_best)) then s_best:=s;
13       endif
14   else
10       s̄:=Path-relinking(t, m, f_t(·), s, E, Seed);
15       AddToElite(E, s̄);
11       if (f_t(s̄) > f_t(s_best)) then s_best:=s̄;
13       endif
13   endif
11 endwhile
12 return(s_best);
end GRASP+PR
```

FIGURE 6. Pseudo-code of a hybrid GRASP with Path-relinking for the FFMSP.

with VNS and applying at the end of each major iteration Path-relinking as intensification procedure.

3. Experimental results

In this section, we present numerical results on computational experiments with the heuristics proposed in this article. We describe first the computer environment and the problem instances. Then, we describe implementation details and the used combination of values for the parameters of the heuristics. Finally, we report on the experimental evaluation of the different algorithms.

Our codes have been written in the *C language*, compilated with "*cc (GCC) 4.1.3 20070929 (prerelease) (Ubuntu 4.1.2-16ubuntu2)*", and ran on an "*Intel Core i7 Quad core @ 2.67 GHz RAM 6GB*" with *Linux* (*Ubuntu 11.10*) operating system.

Problem instances were generated at random. In the set of test instances, the sequence length m ranges from 300 to 800, the number n of sequences in Ω ranges from 100 to 200, and threshold t varies from $75\%m$ to $85\%m$. For each problem size, the algorithms were run for 100 random instances and average solution value was computed. The results obtained are summarized in Table 1, where for each problem type, in the first column the instance size (m, n, and t) is reported. The remaining columns report the average objective function values (z) obtained by each algorithm and the corresponding average running times (in seconds). We make the following observations:

TABLE 1. Average objective function values obtained by each algorithm and the corresponding average running times (in seconds).

	GRASP		GRASP+PR		GRASP+VNS+PR		VNS		VNS+PR	
n, m, t	z	Time	z	Time	z	Time	z	Time	z	Time
100, 300, 0.75	**100**	1.37	**100**	1.41	**100**	1.71	94.47	72.45	**100**	7.44
100, 300, 0.80	67.86	1.67	76.17	3.21	**76.68**	33.28	19.98	71.02	48.58	77.43
100, 300, 0.85	3.56	1.72	10.17	4.17	**12.23**	31.36	1.12	37.65	3.53	44.19
100, 600, 0.75	**100**	1.56	**100**	1.89	**100**	1.11	91.78	278.94	**100**	31.91
100, 600, 0.80	65.35	2.31	78.83	11.77	**80.47**	122.65	8.51	264.66	20.72	295.49
100, 600, 0.85	1.21	1.28	3.58	11.07	**4.36**	91.87	0.04	152.64	0.91	186.71
100, 800, 0.75	100	1.84	100	1.34	100	2.48	87.36	549.60	100	63.00
100, 800, 0.80	67.76	1.42	80.37	21.10	**82.30**	251.26	4.41	450.63	10.94	527.76
100, 800, 0.85	0.30	2.98	0.97	31.45	**2.51**	148.98	0.66	273.98	0.63	329.48
200, 300, 0.75	197.78	1.22	**200**	1.85	**200**	3.70	180.08	135.61	**200**	47.91
200, 300, 0.80	76.50	1.39	93.70	6.52	**94.45**	65.44	36.71	150.51	66.81	160.37
200, 300, 0.85	2.83	1.59	9.26	8.59	**11.12**	68.20	2.16	86.60	4.62	104.13
200, 600, 0.75	**200**	1.94	**200**	1.61	**200**	3.39	178.11	545.50	**200**	75.43
200, 600, 0.80	62.80	1.63	83.91	24.91	**86.17**	274.93	11.93	588.35	33.41	625.66
200, 600, 0.85	0.98	1.79	1.51	31.22	**2.38**	174.73	0.71	305.29	0.96	369.29
200, 800, 0.75	**200**	1.04	**200**	1.33	**200**	1.61	175.06	947.80	**200**	193.30
200, 800, 0.80	44.66	1.75	71.28	43.63	**72.44**	519.59	6.37	987.35	17.12	1102.47
200, 800, 0.85	0.86	1.55	1.93	59.20	**3.71**	311.81	0.15	544.21	0.49	659.02

- the stopping criterion for all algorithms was `MaxIterations` = 500 or the obtainment of an incumbent solution with objective function value $z = n$ (i.e., an optimal solution);
- the maximum order k_{max} in the dynamic neighborhood model used in VNS and in the hybrid VNS with Path-relinking and the hybrid GRASP with VNS and Path-relinking was set to 30;
- the maximum number `MaxElite` of elite solutions in the hybrid heuristics invoking Path-relinking as intensification procedure was set to 10;
- in Path-relinking, for inclusion of the candidate solution $\overline{s}$ in the elite set $\mathcal{E}$ when $\overline{s}$ is better than the worst elite set solution but not better than the best, $\overline{s}$ is inserted (replacing the worst solution) if it *sufficiently different* from all elite solutions, i.e. if $|\Delta(\overline{s}, \epsilon)| \geq \frac{m}{2}$, for all $\epsilon \in \mathcal{E}$;
- on all instances, the hybrid GRASP with VNS and Path-relinking found a better quality solution as compared to the competitor heuristics;
- the hybrid GRASP with Path-relinking found best results for 7 out of the 18 instances; hybrid VNS with Path-relinking found best results for 6 instances, while the pure GRASP and the pure VNS found the best solution for only 5 and 0 instances, respectively;
- at the expense of increased running times, the integration of Path-relinking in the pure metaheuristics was beneficial in terms of solution quality;
- looking at the objective function values achieved by GRASP with Path-relinking and GRASP with VNS and Path-relinking, at the expense of increased running times, the use of VNS in the local search phase of GRASP was beneficial.

Given the random component of each proposed algorithm and since their running times per iteration vary substantially, we have performed two further experiments.

TABLE 2. Average objective function values obtained by each algorithm after 90 seconds of computation.

n, m, t	GRASP	GRASP+PR	GRASP+VNS+PR	VNS	VNS+PR
$100, 300, 0.75$	**100**	**100**	**100**	94	**100**
$100, 300, 0.8$	71.07	**79.61**	78.12	23.43	49.54
$100, 300, 0.85$	6.41	**13.18**	11.86	1.02	3.27
$100, 600, 0.75$	**100**	**100**	**100**	91.79	**100**
$100, 600, 0.8$	70.24	**80.13**	78.05	6.38	11.29
$100, 600, 0.85$	2.73	**4.98**	4.48	0.03	0.12
$100, 800, 0.75$	**100**	**100**	**100**	85.18	**100**
$100, 800, 0.8$	70.07	**82.64**	79.45	3.71	8.42
$100, 800, 0.85$	1.17	**1.84**	1.65	0	0.03
$200, 300, 0.75$	199.81	**200**	**200**	179.34	**200**
$200, 300, 0.8$	81.75	**100**	95.11	34.71	61.67
$200, 300, 0.85$	4.82	**11.90**	11.03	2.32	3.70
$200, 600, 0.75$	**200**	**200**	**200**	172.41	**200**
$200, 600, 0.8$	66.23	**88.49**	80.31	10.25	19.10
$200, 600, 0.85$	1.03	**2.42**	1.73	0.09	0.97
$200, 800, 0.75$	**200**	**200**	**200**	164.01	194.45
$200, 800, 0.8$	49.87	**73.08**	62.36	4.23	8.17
$200, 800, 0.85$	0.08	**0.21**	0.17	0.06	0.85

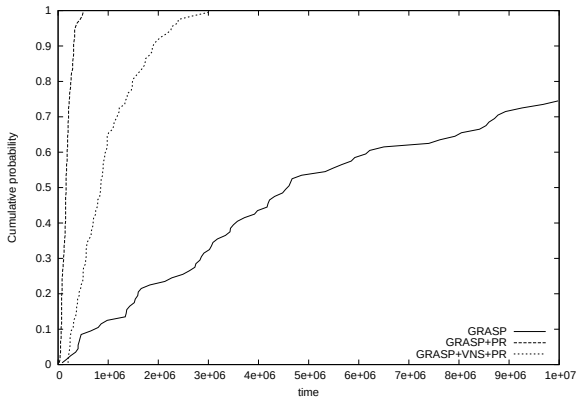
First, on the same set of randomly generated problem instances and by using the same values of the parameters for the algorithms, we have run them for a given fixed amount of time, set to 90 seconds. For each problem type, Table 2 reports the average objective function values (z) obtained by each algorithm. It is still evident that the integration of Path-relinking in the pure metaheuristics is beneficial in terms of solution quality. Moreover, in a given fixed amount of computation time the number of iterations performed by the hybrid GRASP with Path-relinking is higher than that performed by the hybrid GRASP with VNS and Path-relinking. This implies that GRASP with VNS and PR performs a smaller number of samplings of the solution space with the conclusion that in this scenario, the hybrid GRASP with PR found better quality solutions.

As further investigation, given the random component of each proposed algorithm and the great variety in their running times per iteration, we plot in Figures 7–8 the empirical distributions of the random variable *time-to-target-solution-value* considering the following four random instances:

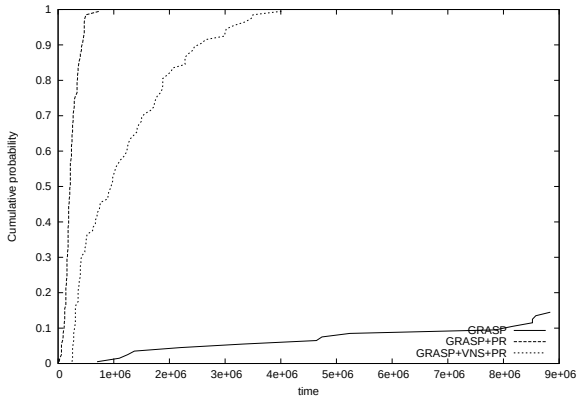
(1) $n = 100$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.70 \times n$ (Figure 7(a));
(2) $n = 100$, $m = 300$, $t = 252$, and target value $\hat{z} = 0.12 \times n$ (Figure 7(b));
(3) $n = 200$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.40 \times n$ (Figure 8(a));
(4) $n = 300$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.28 \times n$ (Figure 8(b)).

We performed 100 independent runs of each heuristic using 100 different random number generator seeds and recorded the time taken to find a solution at least as good as the target value $\hat{z}$. As in [1], to plot the empirical distribution we associate with the i^{th} sorted running time (t_i) a probability $p_i = \frac{i-1/2}{100}$, and plot the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, 100$. About these further experiments, looking

at Figures 7 and 8, we observe that the relative position of the curves implies that, given any 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.



(a) Random instance with $n = 100$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.70 \times n$.

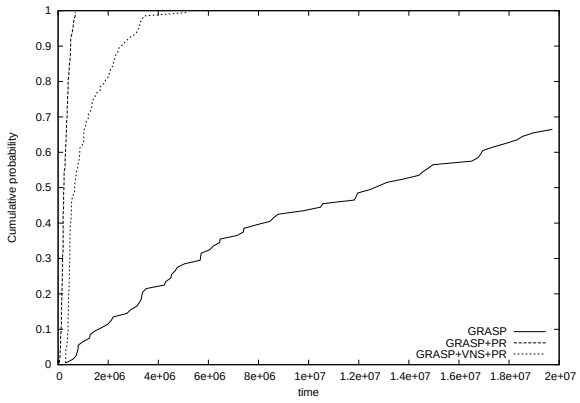


(b) Random instance with $n = 100$, $m = 300$, $t = 252$, and target value $\hat{z} = 0.12 \times n$.

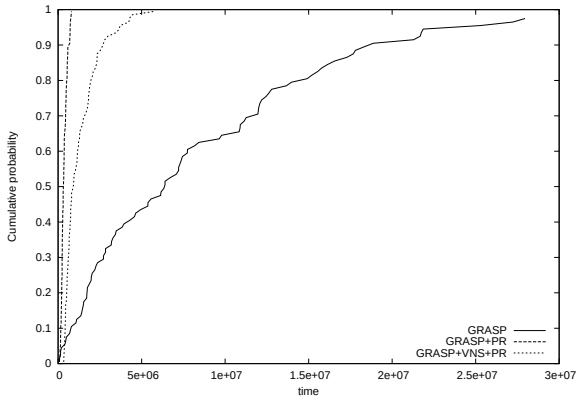
FIGURE 7. Time to target distributions comparing `GRASP`, `GRASP+PR`, and `GRASP+VNS+PR`.

4. Concluding remarks and future work

Given the computational intractability of one of the consensus problems known as the Far From Most String Problem, we have designed several hybrid metaheuristics that guarantee good quality solutions within realistic and acceptable amount of time. The algorithms were tested on several random instances and the results show that the hybrid GRASP with VNS and Path-relinking always finds much better quality solutions compared with the other competitor algorithms, but clearly with higher running times as compared to the pure GRASP and the hybrid GRASP



(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 $\hat{z} = 0.28 \times n$.

FIGURE 8. Time to target distributions comparing `GRASP`, `GRASP+PR`, and `GRASP+VNS+PR`.

with Path-relinking. In the following, we summarize our observations about our computational experience.

- The processing time for the pure GRASP was the smallest but the objective function values found by the algorithm were worse than those found by its hybridizations;
- Best objective function values found by GRASP and its hybrids were when the construction phase was more greedy than random.
- Overall, the hybrid GRASP with VNS and path-relinking found the best solutions, followed by GRASP with Path-relinking and the pure GRASP;
- Overall, the objective function values found by the pure VNS were the worst. This bad behavior is not surprising, given the totally random criterion adopted in the VNS construction.
- The integration of Path-relinking as intensification procedure in the pure metaheuristics was beneficial in terms of solution quality.

- We plot in Figures 7–8 the empirical distributions of the random variable *time-to-target-solution-value* considering four different random instances. Our conclusion after this further investigation is that, given any fixed amount of computing time, GRASP with Path-relinking has a higher probability than all competitors of finding a target solution.

As future work, we plan to better investigate the practical behavior of the proposed algorithms, by introducing the recently published tool designed by Ribeiro et al. [21] 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.

We plan also to validate the numerical results on computational experiments with the heuristics proposed in this article, by applying them on a larger dataset of instances, both randomly generated and taken from real–world applications of the problem.

Furthermore, it would be also interesting to design some variants of the approaches proposed in this paper. Three natural extensions would be 1) to perform at the end of computation a post-optimization phase, for example by invoking Path-relinking among pairs of elite solutions; 2) to implement alternative linking strategies in Path-relinking, such as backward, mixed, and randomized Path-relinking; 3) to integrate in the local search of the algorithms the new function devised by Mousavi et al. [20] and to be used in alternative to the objective function when evaluating neighbor solutions.

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] S.A. Canuto, M.G.C. Resende, and C.C. Ribeiro. Local search with perturbations for the prize-collecting Steiner tree problem in graphs. *Networks*, 38:50–58, 2001.

[3] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Oper. Res. Lett.*, 8:67–71, 1989.

[4] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *J. Global Optim.*, 6:109–133, 1995.

[5] P. Festa. On some optimization problems in mulecolar biology. *Mathematical Bioscience*, 207(2):219–234, 2007.

[6] P. Festa, P.M. Pardalos, L.S. Pitsoulis, and M.G.C. Resende. GRASP with path-relinking for the weighted MAXSAT problem. *ACM J. of Experimental Algorithmics*, 11:1–16, 2006.

[7] P. Festa, P.M. Pardalos, M.G.C. Resende, and C.C. Ribeiro. Randomized heuristics for the MAX-CUT problem. *Optimization Methods and Software*, 7:1033–1058, 2002.

[8] P. Festa and M.G.C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys on Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[9] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP – Part I: Algorithms. *International Transactions in Operational Research*, 16(1):1–24, 2009.

[10] P. Festa and M.G.C. Resende. An annotated bibliography of GRASP – Part II: Applications. *International Transactions in Operational Research*, 16(2):131–172, 2009.

[11] M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.

[12] F. Glover. Tabu search and adaptive memory programing – Advances, applications and challenges. In R.S. Barr, R.V. Helgason, and J.L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer, 1996.

[13] F. Glover. Multi-start and strategic oscillation methods – Principles to exploit adaptive memory. In M. Laguna and J.L. Gonzáles-Velarde, editors, *Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, pages 1–24. Kluwer, 2000.

[14] F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.

[15] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.

[16] P. Hansen and N. Mladenović. Developments of variable neighborhood search. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 415–439. Kluwer Academic Publishers, 2002.

[17] M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, 11:44–52, 1999.

[18] J. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. *Information and Computation*, 185(1):41–55, 2003.

[19] C.N. Meneses, C.A.S. Oliveira, and P.M. Pardalos. Optimization techniques for string selection and comparison problems in genomics. *IEEE Engineering in Medicine and Biology Magazine*, 24(3):81–87, 2005.

[20] S.R. Mousavi, M. Babaie, and M. Montazerian. An improved heuristic for the far from most strings problem. *Journal of Heuristics*, 18:239–262, 2012.

[21] C.C. Ribeiro, I. Rosseti, and R. Vallejos. Exploiting run time distributions to compare sequential and parallel stochastic local search algorithms. *Journal of Global Optimization*, 54:405–429, 2012.

[22] J.S. Sim and K. Park. The consensus string problem for a metric is NP-complete. In *Proceedings of the Annual Australiasian Workshop on Combinatorial Algorithms (AWOCA)*, pages 107–113, 1999.

(D. Ferone) Department of Mathematics and Applications "R. Caccioppoli", University of Napoli FEDERICO II, Compl. MSA, Via Cintia, 80126 Napoli, Italy.
E-mail address, D. Ferone: danieleferone@gmail.com

(P. Festa) Department of Mathematics and Applications "R. Caccioppoli", University of Napoli FEDERICO II, Compl. MSA, Via Cintia, 80126 Napoli, Italy.
E-mail address, P. Festa: paola.festa@unina.it

(M.G.C. Resende) Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.
E-mail address, M.G.C. Resende: mgcr@research.att.com