HYBRIDIZATIONS OF GRASP WITH PATH-RELINKING 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 paper, several hybrid heuristics are described and tested. Experiments on a large set of randomly generated test instances indicate that these randomized hybrid heuristics are both effective and efficient.

1. The Far From Most String Problem (FFMSP)

The Far From Most String Problem (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 sense, all the given sequences. For the FFMSP, the objective is to find a sequence that is far from as many sequences as possible in a given set of sequences all having 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: January 26, 2014.

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

AT&T Labs Research Technical Report.

- 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 sequences as possible in the input set Ω. This can be formalized by stating 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 [13] and in 1999 by Sim and Park [27]. Among sequence consensus problems, the FFMSP is one of the hardest from a computational point of view, as proved in 2003 by Lanctot et al. [19], 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 are recommended. The first attempt in this direction was done in 2005 by Meneses et al. [20], who proposed a heuristic algorithm consisting of a simple greedy construction followed by an iterative improvement phase. Later, in 2007 Festa [7] designed a simple GRASP, recently improved in 2012 by Mousavi et al. [21]. Mousavi et al. noticed that the search landscape of the FFMSP is characterized by many solutions having the same objective value. Consequently, local search is likely to visit many sub-optimal local maxima. To efficiently escape from these local maxima, Mousavi et al. devised a new hybrid heuristic evaluation function and used it in conjunction with the objective function when evaluating neighbor solutions during the local search phase in the GRASP framework.

In 2013 Ferone et al. [6] designed the following pure and hybrid multistart iterative heuristics:

- ◇ a pure GRASP, inspired by [7];
- ◇ a GRASP that uses Forward path-relinking for intensification;
- ◇ a pure VNS;
- ◇ a VNS that uses Forward 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 Forward path-relinking for intensification.

The algorithms were tested on several random instances and the results showed that the hybrid GRASP with VNS and Forward path-relinking always found much better quality solutions compared with the other algorithms, but clearly with higher running times as compared to the pure GRASP and the hybrid GRASP with Forward path-relinking.

The best objective function values found by GRASP and its hybrids were when the construction phase was more greedy than random. For this reason, 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 Forward path-relinking as an intensification procedure in the pure metaheuristics was beneficial in terms of solution quality. A further investigation conducted studying the empirical distributions of the random variable *time-to-target-solution-value* revealed that, given any fixed amount of computing time, GRASP with Forward path-relinking has an empirically higher probability than all competitors of finding a target solution.

Given the results of the analysis conducted involving the empirical distributions of the random variable *time-to-target-solution-value*, we opted to design, implement, and test several different hybridizations of GRASP with path-relinking for the FFMSP. In this paper we report on the implementation details of the different resulting algorithms as well as the results of the experiments we conducted to compare them. Our objective was to better investigate the benefits with respect to computation time and solution quality of the integration of the different path-relinking strategies in GRASP.

The remainder of this article is organized as follows. In Section 2, we describe the GRASP for the FFMSP proposed in [6] as well as the hybrid heuristic evaluation function proposed by Mousavi et al. [21] that we integrate with the GRASP local search. We also outline a self-contained experimental analysis whose objective is to investigate whether the integration of function of Mousavi et al. in GRASP is beneficial. In Section 3, we propose several path-relinking strategies that can be integrated in GRASP. The resulting hybrid GRASP with path-relinking algorithms are described in Section 4. In Section 5 we report on computational experiments with these algorithms on both a set of randomly-generated and a set of real-world problem instances. Concluding remarks and insights about further improvements of the proposed techniques are given in the last section.

2. GRASP for the FFMSP

Each GRASP iteration consists of a construction phase [4, 5], 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 [10, 11, 12] for annotated bibliographies of GRASP.

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 candidates is called the *restricted candidate list* (RCL). A GRASP was proposed in [6] to find optimal and near-optimal solutions for the FFMSP. Figure 1 depicts its

```
algorithm GRASP(t, m, Σ, f_t(·), {V_j(c)}_{j∈{1,...,m}}^{c∈Σ}, Seed)
1  s_best:=∅; f_t(s_best):=−∞;
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    [s, {RCL_j}_{j=1}^m]:=GrRand(m, Σ, {V_j(c)}_{j∈{1,...,m}}^{c∈Σ}, V_j^min, V_j^max, Seed);
7    s:=LocalSearch(t, m, s, f_t(·), {RCL_j}_{j=1}^m);
8    if (f_t(s) > f_t(s_best)) then s_best:=s;
9  endwhile
10 return(s_best);
end GRASP
```

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

```
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 RCL_j:=RCL_j∪{c};
6    endfor
7    s_j:=Random(RCL_j, Seed);
8  endfor
9  return(s, {RCL_j}_{j=1}^m);
end GrRand
```

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

pseudo-code, where $f_t : \Sigma^m \mapsto \mathbb{N}$ denotes the objective function to be maximized according to (3) and (4).

Figure 2 shows a 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 was done in [7], 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 of all characters whose greedy function value is less than or equal to μ (line 5). A character is then selected, uniformly at random, from the RCL (line 7).

The basic step of the local search described in Figure 3 is slightly different from the one implemented in [7]. In our GRASP, it consists in investigating all positions $j \in \{1, \ldots, m\}$ (loop in lines 4–13) and changing the character in position j in

```
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.

the sequence s to another character in RCL_j. In [7] 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 locally optimal solution (line 15).

2.1. **The hybrid heuristic evaluation function.** In [21], Mousavi et al. observed that the objective function of the FFMSP is characterized by large plateaus, since there are $|\Sigma|^m$ points in the search space with only n different objective values. To efficiently handle the numerous local maxima, they devised a new function to be used in the GRASP framework in conjunction with the objective function when evaluating neighbor solutions during the local search phase. They first define a function $\widetilde{GpC}$, called *Gain-per-Cost*, to evaluate candidate solutions based on their likelihood to lead to better solutions. Then, they define a heuristic function $h_{f_t,\widetilde{GpC}}(\cdot)$ which takes into account both the objective function $f_t(\cdot)$ and the estimated Gain-per-Cost function $\widetilde{GpC}$.

Given two candidate solutions $\overline{s}, \hat{s} \in \Sigma^m$, $\overline{s} \neq \hat{s}$, function $h_{f_t,\widetilde{GpC}}(\cdot)$ must combine $f_t(\cdot)$ and $\widetilde{GpC}$ in such a way that

$$h_{f_t,\widetilde{GpC}}(\overline{s}) > h_{f_t,\widetilde{GpC}}(\hat{s}),$$

if and only if either $f_t(\overline{s}) > f_t(\hat{s})$, or $f_t(\overline{s}) = f_t(\hat{s})$ and $\widetilde{GpC}(\overline{s}) > \widetilde{GpC}(\hat{s})$. In other words, according to function $h_{f_t,\widetilde{GpC}}(\cdot)$, the candidate solution $\overline{s}$ is considered better than $\hat{s}$ if it corresponds to a better objective function value or the objective function assumes the same value when evaluated in both solutions, but $\overline{s}$ has a higher probability to lead to better solutions.

Looking at the results of a few experiments conducted in [21], it emerges that using the hybrid heuristic evaluation function $h_{f_t,\widetilde{GpC}}(\cdot)$ in place of the objective function $f_t(\cdot)$ in the GRASP local search proposed by Festa [7] results in a reduction in the the number of local maxima. Consequently the algorithm's climb toward better solutions is sped up.

2.2. **Comparing GRASP with and without hybrid heuristic evaluation function.** We next present some experimental analysis to help determine whether the integration of the hybrid heuristic evaluation function $h_{f_t,\widetilde{GpC}}(\cdot)$ of Mousavi et al. in a GRASP is beneficial. To this end, we implemented two GRASP heuristics. One is the original algorithm as proposed in [6] with the objective evaluation function $f_t(\cdot)$ and the other is the same GRASP but using in the local search phase the heuristic evaluation function $h_{f_t,\widetilde{GpC}}(\cdot)$. In what follows, we refer to these two algorithms as `grasp` and `grasp-h-ev`, respectively.

The two algorithms were implemented in the *C language*, compiled with "*cc (GCC) 4.1.3 20070929 (prerelease) (Ubuntu 4.1.2-16ubuntu2)*", and run on an "*Intel Core i7 Quad core @ 2.67 GHz RAM 6GB*" with *Linux* (*Ubuntu 11.10*) operating system.

The following two sets of problem instances were used.

Set $\mathcal{A}$. This is the set of benchmark instances introduced in [6], consisting of 600 random instances of different size. More specifically, the number n of input sequences in Ω is in $\{100, 200\}$, and the length m of each of the input sequences is in $\{300, 600, 800\}$. In all cases, the alphabet size is four, i.e., $|\Sigma| = 4$. For each combination of n and m, the set $\mathcal{A}$ consists of 100 random instances. Finally, as in [6], the two algorithms were applied on all instances for different settings of the threshold parameter t varying from $.75m$ to $.85m$.

Set $\mathcal{B}$. This set of problem instances was used in [21] and was kindly provided to us by the authors. In this set, some instances were randomly generated, while the remaining are real-world instances from biology (Hyaloperonospora parasitica V 6.0 sequence data). In both the randomly generated and the real-world instances, the alphabet size is four, i.e., $|\Sigma| = 4$, the number n of input sequences in Ω is in $\{100, 200, 300\}$, and the length m of each of the input sequences ranges from 100 to 1200. Finally, the threshold parameter t varies from $.75m$ to $.95m$.

We performed a first experiment on the instances in set $\mathcal{A}$. For each problem size, all the variants were run on 100 random instances and average solution value computed. The results obtained are summarized in Table 1, where for each problem type, the instance size is reported in the first column. The remaining columns report the average running times (in seconds) and the average objective function values (z) obtained by each algorithm.

We then performed a second experiment on the instances in set $\mathcal{B}$. For each problem size, this set contains 10 randomly generated instances and three real-world instances. The average solution values obtained on the random instances are summarized in Table 2. Table 3 reports the average solution values obtained on the real instances.

We make the following observations about the experiments described above:

TABLE 1. Comparison between `grasp` and `grasp-h-ev` on instances in set $\mathcal{A}$.

	`grasp`		`grasp-h-ev`	
Instance size	Time (s)	z	Time (s)	z
$n = 100, m = 300, t = 225$	0.01	**100.00**	0.02	**100.00**
$n = 100, m = 300, t = 240$	2.06	67.53	4.18	**72.70**
$n = 100, m = 300, t = 255$	1.54	6.56	7.71	**27.80**
$n = 100, m = 600, t = 450$	0.00	**100.00**	0.06	**100.00**
$n = 100, m = 600, t = 480$	3.33	65.37	16.78	**75.50**
$n = 100, m = 600, t = 510$	3.10	1.67	35.10	**27.42**
$n = 100, m = 800, t = 600$	0.00	**100.00**	0.09	**100.00**
$n = 100, m = 800, t = 640$	4.34	64.75	29.97	**77.21**
$n = 100, m = 800, t = 680$	4.16	0.95	59.79	**26.17**
$n = 200, m = 300, t = 225$	4.77	197.36	0.13	**200.00**
$n = 200, m = 300, t = 240$	5.36	81.08	7.95	**87.55**
$n = 200, m = 300, t = 255$	3.03	4.49	17.24	**30.48**
$n = 200, m = 600, t = 450$	0.49	199.98	0.38	**200.00**
$n = 200, m = 600, t = 480$	6.85	61.18	39.00	**81.23**
$n = 200, m = 600, t = 510$	6.04	0.96	74.59	**26.07**
$n = 200, m = 800, t = 600$	0.03	**200.00**	0.40	**200.00**
$n = 200, m = 800, t = 640$	8.53	51.16	78.31	**85.27**
$n = 200, m = 800, t = 680$	8.09	0.16	149.65	**24.36**

- ☐ The stopping criterion for `grasp` was set `MaxIterations` = 5000 or an objective function value $z = n$ (i.e., an optimal solution);
- ☐ Since the computation time of a single iteration of `grasp-h-ev` is higher than that of `grasp`, for `grasp-h-ev` the stopping criterion was set to `MaxIterations` = 150 or an objective function value $z = n$ (i.e., an optimal solution);
- ☐ At the expense of increased running times, the integration of Mousavi et al.'s hybrid heuristic evaluation function in `grasp` was beneficial in terms of solution quality.

As further investigation, given the random component of the algorithms and the great variety in their running times per iteration, we plot in Figures 4 the empirical distributions of the random variable *time-to-target-solution-value* considering the following random instances:

(1) $n = 100$, $m = 600$, $t = 480$, and target value $\hat{z} = 0.68 \times n$ (Figure 4(a));
(2) $n = 200$, $m = 300$, $t = 255$, and target value $\hat{z} = 0.025 \times n$ (Figure 4(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

TABLE 2. Comparison between `grasp` and `grasp-h-ev` on the *random* problem instances in set $\mathcal{B}$.

	`grasp`		`grasp-h-ev`	
Instance size	Time (s)	z	Time (s)	z
$n = 100, m = 100, t = 75$	0.77	99.10	0.01	**100.00**
$n = 100, m = 100, t = 85$	0.79	25.10	0.74	**30.20**
$n = 100, m = 100, t = 95$	0.53	1.00	0.87	**6.80**
$n = 100, m = 200, t = 150$	0.02	**100.00**	0.02	**100.00**
$n = 100, m = 200, t = 170$	1.08	12.60	3.06	**28.00**
$n = 100, m = 200, t = 190$	1.05	0.00	3.14	**5.10**
$n = 100, m = 400, t = 300$	0.00	**100.00**	0.03	**100.00**
$n = 100, m = 400, t = 340$	2.06	3.90	13.62	**27.70**
$n = 100, m = 400, t = 380$	2.10	0.00	10.46	**4.20**
$n = 200, m = 200, t = 150$	4.23	195.10	0.61	**199.90**
$n = 200, m = 200, t = 170$	2.05	10.70	6.01	**30.60**
$n = 200, m = 200, t = 190$	2.02	0.00	6.28	**5.00**
$n = 200, m = 400, t = 300$	4.45	198.90	0.21	**200.00**
$n = 200, m = 400, t = 340$	4.06	2.50	32.26	**29.30**
$n = 200, m = 400, t = 380$	4.14	0.00	21.76	**3.70**
$n = 200, m = 800, t = 600$	0.02	**200.00**	0.36	**200.00**
$n = 200, m = 800, t = 680$	8.09	0.20	130.24	**24.40**
$n = 200, m = 800, t = 760$	8.27	0.00	87.15	**3.00**
$n = 300, m = 300, t = 225$	11.64	286.90	5.52	**295.10**
$n = 300, m = 300, t = 255$	4.83	3.60	28.98	**32.60**
$n = 300, m = 300, t = 285$	4.92	0.00	22.47	**3.80**
$n = 300, m = 600, t = 450$	13.02	296.70	0.65	**300.00**
$n = 300, m = 600, t = 510$	9.48	0.60	127.36	**24.90**
$n = 300, m = 600, t = 570$	9.70	0.00	83.43	**2.50**
$n = 300, m = 1200, t = 900$	0.12	**300.00**	1.96	**300.00**
$n = 300, m = 1200, t = 1020$	19.18	0.00	538.66	**21.80**
$n = 300, m = 1200, t = 1140$	19.67	0.00	252.41	**1.50**

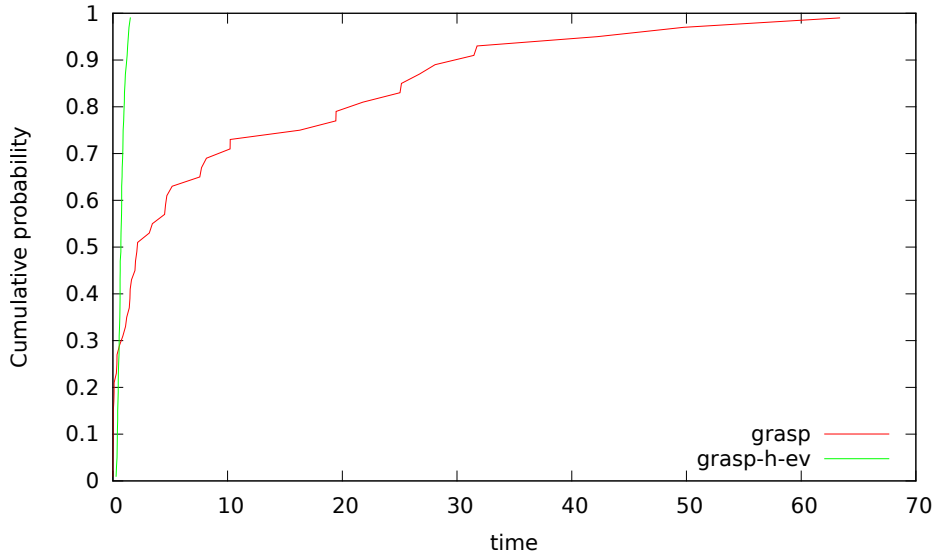
good as the target value $\hat{z}$. As in [1], to plot the empirical distribution we associate with the i^{th} sorted running time (t_i) a probability $p_i = \frac{i-1/2}{100}$, and plot the points $z_i = (t_i, p_i)$, for $i = 1, \ldots, 100$. We observe in Figure 4 that the relative position of the curves implies that, given any fixed amount of running time, `grasp-h-ev` has

TABLE 3. Comparison between `grasp` and `grasp-h-ev` on the *real-world* problem instances in set $\mathcal{B}$.

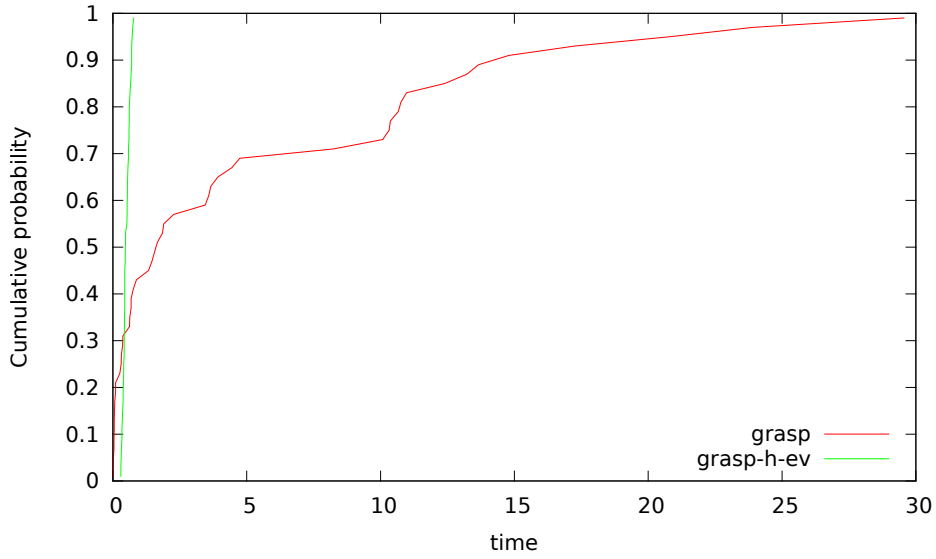
	`grasp`		`grasp-h-ev`	
Instance size	Time (s)	z	Time (s)	z
$n = 100, m = 100, t = 75$	0.00	**100.00**	0.00	**100.00**
$n = 100, m = 100, t = 85$	1.16	59.67	0.66	**61.00**
$n = 100, m = 100, t = 95$	0.49	3.00	0.81	**9.67**
$n = 100, m = 200, t = 150$	0.00	**100.00**	0.00	**100.00**
$n = 100, m = 200, t = 170$	1.29	48.00	2.63	**52.67**
$n = 100, m = 200, t = 190$	1.00	1.00	3.29	**7.67**
$n = 100, m = 400, t = 300$	0.00	**100.00**	0.02	**100.00**
$n = 100, m = 400, t = 340$	2.10	47.00	9.66	**57.33**
$n = 100, m = 400, t = 380$	2.03	0.00	11.16	**7.33**
$n = 200, m = 200, t = 150$	0.05	**200.00**	0.01	**200.00**
$n = 200, m = 200, t = 170$	3.26	83.33	5.00	**86.00**
$n = 200, m = 200, t = 190$	1.94	0.33	6.56	**9.67**
$n = 200, m = 400, t = 300$	0.01	**200.00**	0.03	**200.00**
$n = 200, m = 400, t = 340$	4.32	60.67	22.65	**73.67**
$n = 200, m = 400, t = 380$	3.99	0.00	23.44	**7.00**
$n = 200, m = 800, t = 600$	0.00	**200.00**	0.09	**200.00**
$n = 200, m = 800, t = 680$	8.09	52.67	92.90	**79.67**
$n = 200, m = 800, t = 760$	8.10	0.00	101.74	**4.33**
$n = 300, m = 300, t = 225$	0.14	**300.00**	0.04	**300.00**
$n = 300, m = 300, t = 255$	6.37	96.00	20.78	**101.00**
$n = 300, m = 300, t = 285$	4.71	0.00	22.96	**6.67**
$n = 300, m = 600, t = 450$	0.20	**300.00**	0.12	**300.00**
$n = 300, m = 600, t = 510$	13.10	108.33	121.87	**125.67**
$n = 300, m = 600, t = 570$	12.50	0.00	128.38	**4.00**
$n = 300, m = 1200, t = 900$	0.03	**300.00**	0.25	**300.00**
$n = 300, m = 1200, t = 1020$	18.79	63.00	328.62	**98.67**
$n = 300, m = 1200, t = 1140$	19.01	0.00	365.46	**1.67**

a higher probability of finding a solution whose objective function value is at least as good as the target objective function value than does `grasp`.

To confirm these results, we have used a tool proposed by Ribeiro et al. in [26] to compare algorithms based on stochastic local search. Ribeiro et al. developed a



(a) Random instances with $n = 100$, $m = 600$, $t = 480$, and target value $\hat{z} = 0.68 \times n$.



(b) Random instance with $n = 200$, $m = 300$, $t = 255$, and target value $\hat{z} = 0.025 \times n$.

FIGURE 4. Time to target distributions (in seconds) comparing `grasp` and `grasp-h-ev`.

closed form index that compares two algorithms $\texttt{Alg}_1$ and $\texttt{Alg}_2$ and gives the probability that algorithm $\texttt{Alg}_1$ finds a target solution value in a smaller computation time than algorithm $\texttt{Alg}_2$. We denote this probability by $P(\texttt{Alg}_1 \leq \texttt{Alg}_2)$. We obtained the following results:

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

FIGURE 5. Pseudo-code of path-relinking for the FFMSP.

- Figure 4(a): $P(\texttt{grasp-h-ev} \leq \texttt{grasp}) = 0.72$;
- Figure 4(b): $P(\texttt{grasp-h-ev} \leq \texttt{grasp}) = 0.67$.

We conclude that the integration of Mousavi et al.'s hybrid heuristic evaluation function $h_{f_t,\widetilde{GpC}}(\cdot)$ in the local search of `grasp` beneficial. The resulting hybrid GRASP on average finds better solutions in smaller running times. For this reason, in the remainder of this paper, we will restrict ourselves to the hybrid GRASP `grasp-h-ev`.

3. Path-relinking

Path-relinking is a heuristic proposed in 1996 by Glover [14] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [15, 16, 17, 24].

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 path-relinking for the FFMSP. It is applied to a pair of sequences $(s', \hat{s})$. Their common elements are kept constant, and the space of solutions spanned by this pair of solutions is searched with the objective of finding a better solution. This search is done by exploring a path in the solution space linking solution s' to solution $\hat{s}$. s' is called the *initial solution* and $\hat{s}$ the *guiding solution*.

The procedure then computes (line 2) 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 11).

The path of solutions is computed in the loop in lines 3 through 10. 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 4), 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 6). The set of available moves is updated (line 5). If necessary, the best solution s^* is updated (lines 7–9). Clearly, the algorithm stops when $\Delta(s', \hat{s}) = \emptyset$.

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

FIGURE 6. Pseudo-code of mixed path-relinking [25] for the FFMSP.

We adopted several strategies of implementing path-relinking for the FFSMP. Given two solutions s' and $\hat{s}$, we implemented

- **Forward PR:** The path emanates from s' which is the worst solution between s' and $\hat{s}$ (Figure 5);
- **Backward PR:** The path emanates from s' which is the best solution between s' and $\hat{s}$ (Figure 5);
- **Mixed PR:** Two paths are generated, one emanating from s' and the other emanating from $\hat{s}$: the process stops as soon as an intermediate common solution is met (Figure 6);
- **Greedy Randomized Adaptive Forward PR:** The path emanates from the worst solution between s' and $\hat{s}$, and at each iteration the step is moved towards an intermediate solution selected at random among a subset of top ranked solutions (Figure 7);

```
algorithm grapr(t, m, f_t(·), s', ŝ, Seed, α)
1  f* := max{f_t(s'), f_t(ŝ)}; s* := arg max{f_t(s'), f_t(ŝ)};
2  Δ(s', ŝ):={i = 1, ..., m | s'_i ≠ ŝ_i};
3  while (Δ(s', ŝ) ≠ ∅) →
4        i_min := arg min{f_t(s' ⊕ i) | i ∈ Δ(s', ŝ)};
5        i_max := arg max{f_t(s' ⊕ i) | i ∈ Δ(s', ŝ)};
6        μ := i_min + α(i_max − i_min);
7        RCL := {i ∈ Δ(s', ŝ) | f_t(s' ⊕ i) ≥ μ};
8        i* := Random(RCL, Seed);
9        Δ(s' ⊕ i*, ŝ) := Δ(s', ŝ) \ {i*};
10       s' := s' ⊕ i*;
11       if (f_t(s') > f*) then
12             f* := f_t(s'); s* := s';
13       endif;
14 endwhile;
15 return (s*);
end grapr
```

FIGURE 7. Pseudo-code of Greedy Randomized Adaptive Path-relinking [3] for FFMSP.

```
algorithm Evolution(t, m, f_t(·), E, Seed)
1  Ê := ∅; f* := −∞
2  forall s' ∈ E →
3        forall s'' ∈ E →
4              if (s' ≠ s'') then
5                    s_min := arg min(f(s'), f(s''));
6                    s_max := arg max(f(s'), f(s''));
7                    α := Random([0, 1], Seed);
8                    s := grapr(t, m, f_t(·), s_min, s_max, Seed, α);
9                    if (f(s) > f*) then s* := s; f* = f(s*);
10                   AddToElite(Ê, s);
11             endif;
12       endfor;
13 endfor;
14 return (Ê, s*);
end Evolution
```

FIGURE 8. Pseudo-code of evolutionary path-relinking [22, 23] for FFMSP.

Evolutionary PR: A Greedy Randomized Adaptive Forward PR is performed, where at each `EvIterations` iterations the algorithm `Evolution` is invoked (see Figure 8) that performs a path-relinking between each pair of current elite set solutions with the aim of eventually improve the current elite set.

4. Hybridizations of 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 [18]. It was followed by several extensions, improvements, and successful applications [2, 8, 9, 24].

We integrated a path-relinking intensification procedure in the hybrid GRASP `grasp-h-ev` described in Section 2. Path-relinking is applied at each GRASP iteration to pairs $(s, \hat{s})$ of solutions, where s is the locally optimal solution obtained by the GRASP local search and $\hat{s}$ is randomly chosen from a pool with at most `MaxElite` high quality solutions found along the search. The pseudo-code for the proposed hybrid GRASP with path-relinking is shown in Figure 9.

algorithm `GRASP+PR`(t, m, Σ, $f_t(\cdot)$, $\{V_j(c)\}_{j\in\{1,\ldots,m\}}^{c\in\Sigma}$, `Seed`, `MaxElite`, `c`, `EvIterations`)
1 s_{best}:=$\emptyset$; $f_t(s_{best})$:=$-\infty$; $\mathcal{E}$:= $\emptyset$; $iter$:=0;
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 $iter$:=$iter + 1$;
7 $[s, \{\mathrm{RCL}_j\}_{j=1}^m]$:=`GrRand`($m$, Σ, $\{V_j(c)\}_{j\in\{1,\ldots,m\}}^{c\in\Sigma}$, $V_j^{\min}$, $V_j^{\max}$, `Seed`);
8 s:=`LocalSearch`(t, m, s, $f_t(\cdot)$, $\{\mathrm{RCL}_j\}_{j=1}^m$);
9 **if** ($iter \leq$ `MaxElite`) **then**
10 $\mathcal{E} := \mathcal{E} \cup \{s\}$;
11 **if** ($f_t(s) > f_t(s_{best})$) **then** s_{best}:=s;
12 **else**
13 $s' := \mathrm{Random}(\mathcal{E}, \mathtt{Seed})$; $\alpha = \mathrm{Random}([0, 1], \mathtt{Seed})$;
14 $(\mathcal{E}, \overline{s})$:= `Choose-PR-Strategy`(`c`, s', s, t, m, $f_t(\cdot)$, $\mathcal{E}$, `Seed`, α, `EvIterations`);
15 `AddToElite`($\mathcal{E}, \overline{s}$);
16 **if** ($f_t(\overline{s}) > f_t(s_{best})$) **then** s_{best}:=$\overline{s}$;
17 **endif**
18 **endwhile**
19 **for** ($s' \in \mathcal{E}$) $\rightarrow$
20 **for** ($s'' \in \mathcal{E}$) $\rightarrow$
21 **if** ($s' \neq s''$) **then**
22 $\alpha := \mathrm{Random}([0, 1], \mathtt{Seed})$;
23 $(\mathcal{E}, \overline{s})$:= `Choose-PR-Strategy`(`c`, s', s, t, m, $f_t(\cdot)$, $\mathcal{E}$, `Seed`, α, `EvIterations`);
24 **if** ($f_t(\overline{s}) > f_t(s_{best})$) **then** s_{best}:=$\overline{s}$;
25 **endif**
26 **endfor**
27 **endfor**
28 **return**(s_{best});
end `GRASP+PR`

Figure 9. Pseudo-code of a hybrid GRASP with path-relinking for the FFMSP.

```
function Choose-PR-Strategy(choice, s', s'', t, m, f_t(·), E, Seed, α, EvIterations)
1   s_min := arg min(f(s'), f(s''));
2   s_max := arg max(f(s'), f(s''));
3   if (choice = FORWARD) then
4         s* := path-relinking(t, m, f_t(·), s_min, s_max, Seed);
5   elseif (choice = BACKWARD) then
6         s* := path-relinking(t, m, f_t(·), s_max, s_min, Seed);
7   elseif (choice = MIXED) then
8         s* := path-relinking-mixed(t, m, f_t(·), s_min, s_max, Seed);
9   elseif (choice = GRAPR) then
10        s* := grapr(t, m, f_t(·), s_min, s_max, Seed, α);
11  elseif (choice = EVOLUTIONARY) then
12        s* := grapr(t, m, f_t(·), s_min, s_max, Seed, α);
13        if (iteration mod EvIterations = 0) then
14              (E, s*) := Evolution(t, m, f_t(·), E, Seed);
15        endif;
16  endif;
17  return (E, s*)
end Choose-PR-Strategy
```

FIGURE 10. Function for selection of path-relinking strategy.

The pool of elite solutions $\mathcal{E}$ is originally empty (line 1) and, until $\mathcal{E}$ is not full, the current GRASP locally optimal solution s is inserted in $\mathcal{E}$. As soon as the pool becomes full ($|\mathcal{E}| =$ `MaxElite`), through procedure `Choose-PR-Strategy`, the desired strategy for implementing path-relinking is chosen. It involves s and solution $\hat{s}$ randomly chosen from $\mathcal{E}$. The best solution $\overline{s}$ found along the relinking trajectory is returned and considered as a candidate to be inserted into the pool. Procedure `AddToElite` evaluates its insertion into $\mathcal{E}$. 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 5) from all elite solutions.

Path-relinking is applied also as post-optimization phase (lines 19–27): at the end of all GRASP iterations, for each different pair of solutions $s', s'' \in \mathcal{E}$, if s' and s'' are *sufficiently different*, path-relinking is performed between s' and s'' by using the same strategy used in the intensification phase.

5. Experimental results

In this section, we present results of the computational experiments with the following heuristics proposed in this paper:

- ◇ `grasp-h-ev`, the hybrid GRASP that integrates Mousavi et al.'s evaluation function into the local search;
- ◇ `grasp-h-ev_f`, the hybrid GRASP that adds forward path-relinking to `grasp-h-ev`;
- ◇ `grasp-h-ev_b`, the hybrid GRASP that adds backward path-relinking to `grasp-h-ev`;
- ◇ `grasp-h-ev_m`, the hybrid GRASP that adds mixed path-relinking to `grasp-h-ev`;

- ◇ `grasp-h-ev_grapr`, the hybrid GRASP that adds greedy randomized adaptive forward path-relinking to `grasp-h-ev`;
- ◇ `grasp-h-ev_ev_pr`, the hybrid GRASP that adds evolutionary path-relinking to `grasp-h-ev`.

The computer environment and problem instances used to experimentally evaluate the different algorithms are those described in Subsection 2.2, and, as in Subsection 2.2, we run the algorithms first on the instances in set $\mathcal{A}$ and then on those in set $\mathcal{B}$.

For each problem size in set $\mathcal{A}$, all the variants were run on 100 random instances and average solution value and average running times were computed. The results obtained are summarized in Table 4, where for each problem type, in the first column the instance size (n, m, and t) is reported. The remaining columns report the average running times (in seconds) and the average objective function values (z) obtained by each algorithm.

We make the following observations:

- ☐ The stopping criterion for all algorithms was set to `MaxIterations` $= 150$ or objective function value $z = n$ (i.e., an optimal solution);
- ☐ The maximum number `MaxElite` of elite solutions was set to 50;
- ☐ A candidate solution $\overline{s}$ is inserted in the elite set $\mathcal{E}$ if $\overline{s}$ is better than the best elite set solution or if it is better than the worst elite set solution but not better than the best and its Hamming distance to at least half of the solutions in the elite set is at least $.75m$. Solution $\overline{s}$ replaces the worst solution in the elite set;
- ☐ In the post-optimization phase, path-relinking is performed between s' and s'' if their Hamming distance is at least $.75m$;
- ☐ At the expense of increased running times, the integration of path-relinking in the hybrid GRASP that integrates Mousavi et al.'s evaluation function into the local search improves slightly the algorithm in terms of solution quality;
- ☐ Overall, the hybrid GRASP that integrates Mousavi et al.'s evaluation function into the local search with evolutionary path-relinking found better quality solutions as compared to the other algorithms.

The same observations can be made looking at the results of the experiments conducted on both the randomly generated and the real-world instances in the set $\mathcal{B}$. The results obtained on the random instances are summarized in Table 5, while Table 6 summarizes the results obtained on the real instances.

In Figures 11, we plot the empirical distributions of the random variable *time-to-target-solution-value*, involving algorithms `grasp-h-ev`, `grasp-h-ev_b`, and `grasp-h-ev_ev_pr`. Although Tables 4–6 show that `grasp-h-ev_ev_pr` outperforms the other variants when running for the same number of iterations, Figures 4(b) show that, given any fixed amount of computing time, `grasp-h-ev_b` has a higher probability than the other algorithms of finding a good quality target solution.

Using the tool proposed by Ribeiro et al. in [21] to compare algorithms based on stochastic local search, we have obtained that:

- Figure 11(a): $P(\texttt{grasp-h-ev\_b} \leq \texttt{grasp-h-ev}) = 0.54$;
- Figure 11(a): $P(\texttt{grasp-h-ev\_b} \leq \texttt{grasp-h-ev\_ev\_pr}) = 0.60$;
- Figure 11(b): $P(\texttt{grasp-h-ev\_b} \leq \texttt{grasp-h-ev}) = 0.61$;
- Figure 11(b): $P(\texttt{grasp-h-ev\_b} \leq \texttt{grasp-h-ev\_ev\_pr}) = 0.73$.

TABLE 4. Average running times (in seconds) and average objective function values obtained by each algorithm on the instances in the set $\mathcal{A}$.

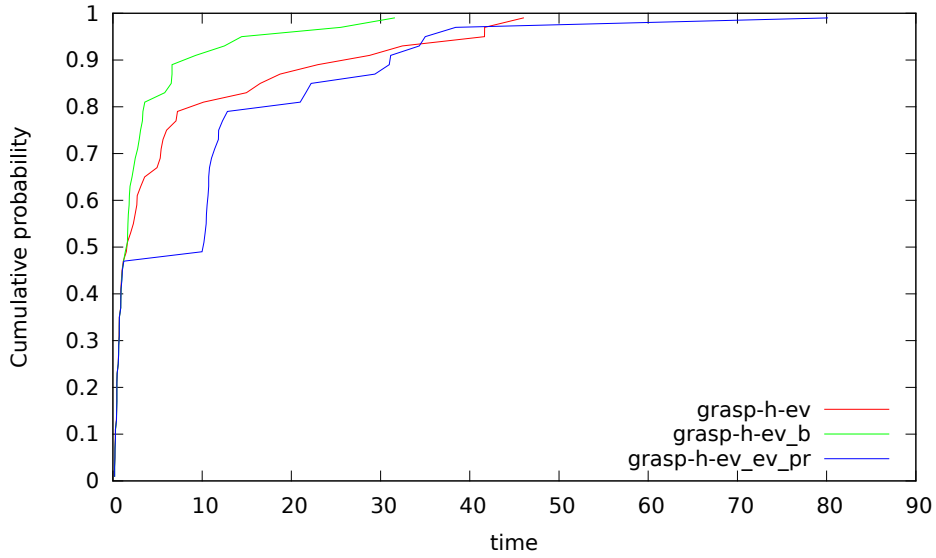
	grasp-h-ev		grasp-h-ev_f		grasp-h-ev_b		grasp-h-ev_m		grasp-h-ev_ev_pr		grasp-h-ev_grapr	
Instance size	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z
$n = 100, m = 300, t = 225$	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**
$n = 100, m = 300, t = 240$	4.18	72.70	5.00	72.88	4.99	73.99	5.01	73.85	22.70	**74.48**	5.04	72.98
$n = 100, m = 300, t = 255$	7.71	27.80	8.46	27.80	8.48	27.81	8.48	27.81	22.75	**27.82**	8.46	27.80
$n = 100, m = 600, t = 450$	0.06	**100.00**	0.07	**100.00**	0.07	**100.00**	0.07	**100.00**	0.07	**100.00**	0.07	**100.00**
$n = 100, m = 600, t = 480$	16.78	75.50	21.07	75.61	21.44	76.25	21.39	76.14	81.62	**76.61**	21.14	75.67
$n = 100, m = 600, t = 510$	35.10	**27.42**	39.16	**27.42**	39.23	**27.42**	39.74	**27.42**	89.12	**27.42**	39.63	**27.42**
$n = 100, m = 800, t = 600$	0.09	**100.00**	0.11	**100.00**	0.10	**100.00**	0.11	**100.00**	0.10	**100.00**	0.11	**100.00**
$n = 100, m = 800, t = 640$	29.97	77.21	35.52	77.21	35.63	77.60	37.47	77.59	137.57	**78.03**	37.06	77.26
$n = 100, m = 800, t = 680$	59.79	**26.17**	65.40	**26.17**	65.49	**26.17**	66.06	**26.17**	156.05	**26.17**	66.03	**26.17**
$n = 200, m = 300, t = 225$	0.13	**200.00**	0.14	**200.00**	0.14	**200.00**	0.14	**200.00**	0.14	**200.00**	0.14	**200.00**
$n = 200, m = 300, t = 240$	7.95	87.55	9.69	88.24	9.68	90.04	9.71	89.97	50.88	**91.48**	9.77	88.19
$n = 200, m = 300, t = 255$	17.24	30.48	18.83	30.48	18.83	30.48	18.95	30.48	53.02	**30.55**	18.92	30.48
$n = 200, m = 600, t = 450$	0.38	**200.00**	0.42	**200.00**	0.41	**200.00**	0.42	**200.00**	0.42	**200.00**	0.42	**200.00**
$n = 200, m = 600, t = 480$	39.00	81.23	47.59	81.28	47.46	81.85	48.00	81.67	190.77	**82.19**	47.92	81.32
$n = 200, m = 600, t = 510$	74.59	**26.07**	84.43	**26.07**	84.45	**26.07**	84.56	**26.07**	202.97	**26.07**	84.45	**26.07**
$n = 200, m = 800, t = 600$	0.40	**200.00**	0.44	**200.00**	0.44	**200.00**	0.44	**200.00**	0.44	**200.00**	0.44	**200.00**
$n = 200, m = 800, t = 640$	78.31	85.27	87.94	85.27	87.94	85.38	89.75	85.40	331.68	**85.54**	89.68	85.30
$n = 200, m = 800, t = 680$	149.65	**24.36**	153.22	**24.36**	153.40	**24.36**	153.96	**24.36**	368.75	**24.36**	153.80	**24.36**

TABLE 5. Average running times (in seconds) and average objective function values obtained by each algorithm on the instances in the set $\mathcal{B}$.

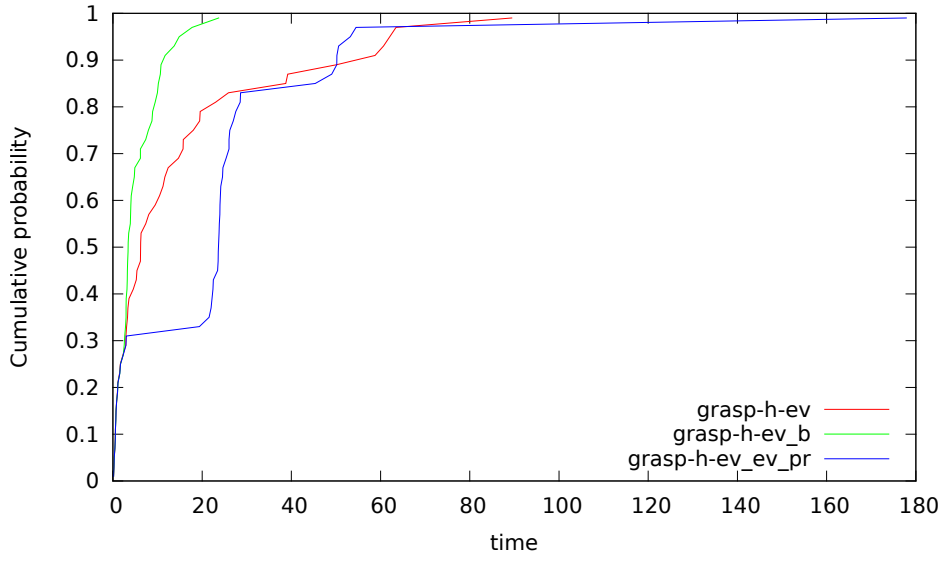
Instance size	grasp-h-ev		grasp-h-ev_f		grasp-h-ev_b		grasp-h-ev_m		grasp-h-ev_ev_pr		grasp-h-ev_grapr	
	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z
$n = 100, m = 100, t = 75$	0.01	**100.00**	0.01	**100.00**	0.01	**100.00**	0.01	**100.00**	0.01	**100.00**	0.01	**100.00**
$n = 100, m = 100, t = 85$	0.74	30.20	0.83	30.20	0.83	**30.40**	0.83	**30.40**	3.04	**30.40**	0.83	30.20
$n = 100, m = 100, t = 95$	0.87	**6.80**	0.97	**6.80**	0.97	**6.80**	0.97	**6.80**	3.17	**6.80**	0.97	**6.80**
$n = 100, m = 200, t = 150$	0.02	**100.00**	0.01	**100.00**	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**
$n = 100, m = 200, t = 170$	3.06	28.00	3.38	28.00	3.42	28.10	3.39	28.10	10.28	**28.30**	3.39	28.00
$n = 100, m = 200, t = 190$	3.14	**5.10**	3.41	**5.10**	3.45	**5.10**	3.42	**5.10**	9.01	**5.10**	3.42	**5.10**
$n = 100, m = 400, t = 300$	0.03	**100.00**	0.04	**100.00**	0.04	**100.00**	0.03	**100.00**	0.03	**100.00**	0.04	**100.00**
$n = 100, m = 400, t = 340$	13.62	**27.70**	15.30	**27.70**	15.41	**27.70**	15.29	**27.70**	37.32	**27.70**	15.32	**27.70**
$n = 100, m = 400, t = 380$	10.46	**4.20**	11.71	**4.20**	11.76	**4.20**	11.71	**4.20**	29.81	**4.20**	11.75	**4.20**
$n = 200, m = 200, t = 150$	0.61	199.90	0.77	199.90	0.62	**200.00**	0.77	199.90	4.21	**200.00**	0.74	**200.00**
$n = 200, m = 200, t = 170$	6.01	30.60	6.77	30.60	6.75	30.70	6.80	30.70	23.18	**30.80**	6.78	30.60
$n = 200, m = 200, t = 190$	6.28	**5.00**	6.85	**5.00**	6.83	**5.00**	6.86	**5.00**	18.91	**5.00**	6.87	**5.00**
$n = 200, m = 400, t = 300$	0.21	**200.00**	0.22	**200.00**	0.22	**200.00**	0.22	**200.00**	0.22	**200.00**	0.22	**200.00**
$n = 200, m = 400, t = 340$	32.26	**29.30**	35.58	**29.30**	35.49	**29.30**	35.62	**29.30**	89.44	**29.30**	35.61	**29.30**
$n = 200, m = 400, t = 380$	21.76	**3.70**	23.69	**3.70**	23.63	**3.70**	23.71	**3.70**	54.39	**3.70**	23.73	**3.70**
$n = 200, m = 800, t = 600$	0.36	**200.00**	0.41	**200.00**	0.41	**200.00**	0.41	**200.00**	0.44	**200.00**	0.41	**200.00**
$n = 200, m = 800, t = 680$	130.24	**24.40**	141.70	**24.40**	141.67	**24.40**	142.54	**24.40**	372.99	**24.40**	142.09	**24.40**
$n = 200, m = 800, t = 760$	87.15	**3.00**	87.05	**3.00**	87.20	**3.00**	87.26	**3.00**	168.07	**3.00**	87.24	**3.00**
$n = 300, m = 300, t = 225$	5.52	295.10	8.71	296.10	8.62	297.00	8.95	296.60	92.60	**297.60**	9.05	295.40
$n = 300, m = 300, t = 255$	28.98	**32.60**	32.17	**32.60**	32.02	**32.60**	32.37	**32.60**	102.28	**32.60**	32.36	**32.60**
$n = 300, m = 300, t = 285$	22.47	**3.80**	24.33	**3.80**	24.26	**3.80**	24.47	**3.80**	64.69	**3.80**	24.45	**3.80**
$n = 300, m = 600, t = 450$	0.65	**300.00**	0.69	**300.00**	0.69	**300.00**	0.69	**300.00**	0.69	**300.00**	0.69	**300.00**
$n = 300, m = 600, t = 510$	127.36	**24.90**	141.03	**24.90**	140.68	**24.90**	141.74	**24.90**	359.20	**24.90**	141.54	**24.90**
$n = 300, m = 600, t = 570$	83.43	**2.50**	85.59	**2.50**	85.37	**2.50**	85.83	**2.50**	147.40	**2.50**	85.84	**2.50**
$n = 300, m = 1200, t = 900$	1.96	**300.00**	2.14	**300.00**	2.14	**300.00**	2.14	**300.00**	2.14	**300.00**	2.14	**300.00**
$n = 300, m = 1200, t = 1020$	538.66	**21.80**	586.78	**21.80**	577.16	**21.80**	607.03	**21.80**	1272.24	**21.80**	577.29	**21.80**
$n = 300, m = 1200, t = 1140$	252.41	**1.50**	259.36	**1.50**	259.93	**1.50**	259.82	**1.50**	334.73	**1.50**	259.87	**1.50**

TABLE 6. Average running times (in seconds) and average objective function values obtained by each algorithm on the *real-world* instances in the set $\mathcal{B}$.

	grasp-h-ev		grasp-h-ev_f		grasp-h-ev_b		grasp-h-ev_m		grasp-h-ev_ev_pr		grasp-h-ev_grapr	
Instance size	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z	Time (s)	z
$n = 100, m = 100, t = 75$	0.00	**100.00**	0.00	**100.00**	0.00	**100.00**	0.00	**100.00**	0.00	**100.00**	0.00	**100.00**
$n = 100, m = 100, t = 85$	0.66	61.00	0.72	61.33	0.71	61.33	0.72	61.33	2.08	**62.33**	0.72	61.00
$n = 100, m = 100, t = 95$	0.81	**9.67**	0.89	**9.67**	0.88	**9.67**	0.89	**9.67**	2.49	**9.67**	0.88	**9.67**
$n = 100, m = 200, t = 150$	0.00	**100.00**	0.00	**100.00**	0.00	**100.00**	0.00	**100.00**	0.00	**100.00**	0.00	**100.00**
$n = 100, m = 200, t = 170$	2.63	52.67	2.87	52.67	2.88	53.67	2.88	53.33	7.64	**54.67**	2.90	52.67
$n = 100, m = 200, t = 190$	3.29	**7.67**	3.49	**7.67**	3.50	**7.67**	3.49	**7.67**	6.89	**7.67**	3.50	**7.67**
$n = 100, m = 400, t = 300$	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**	0.02	**100.00**
$n = 100, m = 400, t = 340$	9.66	**57.33**	10.93	**57.33**	11.10	**57.33**	10.94	**57.33**	25.05	**57.33**	10.90	**57.33**
$n = 100, m = 400, t = 380$	11.16	**7.33**	12.18	**7.33**	12.17	**7.33**	12.19	**7.33**	22.33	**7.33**	12.20	**7.33**
$n = 200, m = 200, t = 150$	0.01	**200.00**	0.02	**200.00**	0.02	**200.00**	0.01	**200.00**	0.01	**200.00**	0.02	**200.00**
$n = 200, m = 200, t = 170$	5.00	86.00	5.47	86.67	5.46	**89.33**	5.47	88.67	15.24	89.00	5.49	86.33
$n = 200, m = 200, t = 190$	6.56	**9.67**	6.94	**9.67**	6.94	**9.67**	6.98	**9.67**	14.85	**9.67**	6.94	**9.67**
$n = 200, m = 400, t = 300$	0.03	**200.00**	0.04	**200.00**	0.03	**200.00**	0.04	**200.00**	0.03	**200.00**	0.04	**200.00**
$n = 200, m = 400, t = 340$	22.65	73.67	24.80	74.00	24.78	73.67	24.94	73.67	55.15	**74.33**	28.89	73.67
$n = 200, m = 400, t = 380$	23.44	**7.00**	29.11	**7.00**	24.80	**7.00**	29.14	**7.00**	40.60	**7.00**	29.18	**7.00**
$n = 200, m = 800, t = 600$	0.09	**200.00**	0.11	**200.00**	0.11	**200.00**	0.11	**200.00**	0.10	**200.00**	0.12	**200.00**
$n = 200, m = 800, t = 680$	92.90	79.67	100.77	79.67	98.59	79.67	115.62	79.67	186.31	**80.00**	99.91	79.67
$n = 200, m = 800, t = 760$	101.74	**4.33**	100.51	**4.33**	100.39	**4.33**	100.49	**4.33**	133.60	**4.33**	100.51	**4.33**
$n = 300, m = 300, t = 225$	0.04	**300.00**	0.05	**300.00**	0.04	**300.00**	0.04	**300.00**	0.04	**300.00**	0.04	**300.00**
$n = 300, m = 300, t = 255$	20.78	101.00	22.58	101.00	22.56	**103.33**	22.69	103.00	61.63	103.00	22.76	101.33
$n = 300, m = 300, t = 285$	22.96	**6.67**	24.01	**6.67**	24.00	**6.67**	24.07	**6.67**	41.31	**6.67**	24.06	**6.67**
$n = 300, m = 600, t = 450$	0.12	**300.00**	0.13	**300.00**	0.13	**300.00**	0.13	**300.00**	0.13	**300.00**	0.14	**300.00**
$n = 300, m = 600, t = 510$	121.87	125.67	132.84	125.67	132.38	126.33	133.27	126.33	258.61	**126.67**	133.12	125.67
$n = 300, m = 600, t = 570$	128.38	**4.00**	128.01	**4.00**	127.56	**4.00**	128.20	**4.00**	148.96	**4.00**	128.07	**4.00**
$n = 300, m = 1200, t = 900$	0.25	**300.00**	0.26	**300.00**	0.27	**300.00**	0.27	**300.00**	0.26	**300.00**	0.26	**300.00**
$n = 300, m = 1200, t = 1020$	328.62	98.67	343.11	98.67	343.16	98.67	345.11	98.67	531.82	**99.00**	343.73	98.67
$n = 300, m = 1200, t = 1140$	365.46	**1.67**	369.51	**1.67**	369.32	**1.67**	369.79	**1.67**	392.61	**1.67**	369.22	**1.67**



(a) Random instances with $n = 100$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.73 \times n$.



(b) Random instance with $n = 200$, $m = 300$, $t = 240$, and target value $\hat{z} = 0.445 \times n$.

FIGURE 11. Time to target distributions (in seconds) comparing `grasp-h-ev`, `grasp-h-ev_b` and `grasp-h-ev_ev_pr`.

6. Concluding remarks and future work

Given the computational intractability of the Far From Most String Problem (FFMSP), we designed several hybrid GRASP based heuristics that guarantee good quality solutions for this problem within realistic and acceptable running

times. The algorithms combine a state-of-the-art GRASP heuristic for the FFMSP proposed by Ferone et al. [6] with the evaluation function proposed by Mousavi et al. [21] in the GRASP local search. The resulting hybrid algorithm was then hybridized with path-relinking procedures implemented with different strategies.

The algorithms were tested on several random and real-world instances and the results show that the hybrid GRASP with the evaluation function and evolutionary path-relinking finds better quality solutions compared with the other algorithms, but at the expense of longer running times. In the following, we summarize our observations about our computational experience.

- The integration into the GRASP local search of Mousavi et al.'s hybrid heuristic evaluation function is beneficial, since it improves also the GRASP proposed by Ferone et al. [6], besides the GRASP previously proposed by Festa [7], as shown by Mousavi et al. [21].
- The integration of path-relinking as an intensification and post-optimization procedure in the pure heuristic was beneficial in terms of mean solution quality, but at the expense of increased running times.
- Overall, the objective function values found by the hybrid GRASP that integrates Mousavi et al.'s evaluation function into the local search with evolutionary path-relinking proved to be the best.
- The plots in Figure 11 show the empirical distributions of the random variable *time-to-target-solution-value* considering two random instances. Our conclusion is that, given any fixed amount of computing time, the hybrid GRASP algorithm that integrates Mousavi et al.'s evaluation function into the local search with backward path-relinking has a higher probability than the other algorithms of finding a good quality target solution. This consideration emerges also by computing the probabilities with Ribeiro et al.'s tool.

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] H. Faria Jr., S. Binato, M.G.C. Resende, and D.J. Falc ao. Transmission network design by a greedy randomized adaptive path relinking approach. *IEEE Transactions on Power Systems*, 20(1):43–49, 2005.

[4] 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.

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

[6] D. Ferone, P. Festa, and M.G.C. Resende. Hybrid metaheuristics for the far from most string problem. In *Proceedings of 8th International Workshop on Hybrid Metaheuristics*, volume 7919 of Lecture Notes in Computer Science, pages 174–188, 2013.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.
[12] 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.
[13] M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.
[14] 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.
[15] 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.
[16] F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.
[17] F. Glover, M. Laguna, and R. Martí. Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39:653–684, 2000.
[18] M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, 11:44–52, 1999.
[19] J. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. *Information and Computation*, 185(1):41–55, 2003.
[20] 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.
[21] 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.
[22] M.G.C. Resende, R. Martí, M. Gallego, and A. Duarte. GRASP and path relinking for the max-min diversity problem. *Computers and Operations Research*, 37:498–508, 2010.
[23] M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10:59–88, 2004.
[24] C.C. Ribeiro and M.G.C. Resende. Path-relinking intensification methods for stochastic local search algorithms. *Journal of Heuristics*, 18:193–214, 2012.
[25] C.C. Ribeiro and I. Rosseti. Efficient parallel cooperative implementations of GRASP heuristics. *Parallel Computing*, 33:21–35, 2007.
[26] 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.
[27] 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