SIAM J. OPTIM.
Vol. 23, No. 3, pp. 1575–1606

A NEW CLASS OF FUNCTIONS FOR MEASURING SOLUTION INTEGRALITY IN THE FEASIBILITY PUMP APPROACH*

M. DE SANTIS†, S. LUCIDI†, AND F. RINALDI‡

Abstract. Mixed integer optimization is a powerful tool for modeling many optimization problems arising from real-world applications. Finding a first feasible solution represents the first step for several mixed integer programming (MIP) solvers. The feasibility pump is a heuristic for finding feasible solutions to mixed integer linear programming (MILP) problems which is effective even when dealing with hard MIP instances. In this work, we start by interpreting the feasibility pump as a Frank–Wolfe method applied to a nonsmooth concave merit function. Then we define a general class of functions that can be included in the feasibility pump scheme for measuring solution integrality, and we identify some merit functions belonging to this class. We further extend our approach by dynamically combining two different merit functions. Finally, we define a new version of the feasibility pump algorithm, which includes the original version of the feasibility pump as a special case, and we present computational results on binary MILP problems showing the effectiveness of our approach.

Key words. mixed integer programming, concave penalty functions, Frank–Wolfe algorithm, feasibility problem

AMS subject classifications. 90C06, 90C10, 90C11, 90C30, 90C59

DOI. 10.1137/110855351

1. Introduction.

Many real-world problems can be modeled as mixed integer programming (MIP) problems, namely, as minimization problems where some (or all) of the variables only assume integer values. Finding quickly a first feasible solution is crucial for solving this class of problems. In fact, many local-search approaches for MIP problems such as local branching [19], guided dives, and RINS [14] can be used only if a feasible solution is available.

In the literature, several heuristics methods for finding a first feasible solution for an MIP problem have been proposed (see, e.g., [4, 5, 6, 10, 21, 22, 23, 24, 25, 26, 29]). Recently, Fischetti, Glover, and Lodi [18] proposed a new heuristic, the well-known feasibility pump (FP), which turned out to be very useful in finding a first feasible solution even when dealing with hard MIP instances. The FP heuristic is implemented in various MIP solvers such as BONMIN [11].

The basic idea of the FP is that of generating two sequences of points $\{\bar{x}^k\}$ and $\{\tilde{x}^k\}$ such that $\bar{x}^k$ is LP-feasible, but may not be integer feasible, and $\tilde{x}^k$ is integer, but not necessarily LP-feasible. To be more specific, the algorithm starts with a solution of the LP relaxation $\bar{x}^0$ and sets $\tilde{x}^0$ equal to the rounding of $\bar{x}^0$. Then at each iteration $\bar{x}^{k+1}$ is chosen as the nearest LP-feasible point in ℓ_1-norm to $\tilde{x}^k$, and $\tilde{x}^{k+1}$ is obtained as the rounding of $\bar{x}^{k+1}$. The aim of the algorithm is to reduce at each iteration the distance between the points of the two sequences, until the two points are the same and an integer feasible solution is found. Unfortunately, it can happen that the distance between $\bar{x}^{k+1}$ and $\tilde{x}^k$ is greater than zero and $\tilde{x}^{k+1} = \tilde{x}^k$, and the strategy can stall. In order to overcome this drawback, random perturbations

*Received by the editors November 15, 2011; accepted for publication (in revised form) April 23, 2013; published electronically August 1, 2013.
http://www.siam.org/journals/siopt/23-3/85535.html
†Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza, Università di Roma, 25 - 00185 Rome, Italy (mdesantis@dis.uniroma1.it, stefano.lucidi@dis.uniroma1.it).
‡Dipartimento di Matematica, Università di Padova, 63 - 35121 Padova, Italy (rinaldi@math.unipd.it).

and restart procedures are performed.

As the algorithm has proved to be effective in practice, various papers devoted to its further improvements have been developed. Bertacco, Fischetti, and Lodi [8] extended the ideas on which the FP is based in two different directions: handling MIP problems with both 0-1 and integer variables, and exploiting the FP information to drive a subsequent enumeration phase. In [1], in order to improve the quality of the feasible solution found, Achterberg and Berthold consider an alternative distance function which takes into account the original objective function. In [20], Fischetti and Salvagnin proposed a new rounding heuristic based on a diving-like procedure and constraint propagation. Recently in [3] and [9] new rounding techniques have been proposed. They are both based on the idea of replacing rounding with a procedure that examines rounded solutions along a line segment passing through the LP-feasible solution. The feasibility pump has been further extended to the case of mixed integer nonlinear programming problems in [12, 13].

In [10], Eckstein and Nediak noticed that the FP heuristic may be seen as a form of Frank–Wolfe procedure applied to a nonsmooth merit function which penalizes the violation of the 0-1 constraints. In practice, the FP combines a local algorithm (namely, the Frank–Wolfe algorithm) with a suitably developed perturbing procedure for solving a specific global optimization problem:

$$x^* = \arg\min\{f(x) : x \in P\},$$

where P is the relaxation of the feasible set of the original MIP problem and $f(x)$ is a function penalizing the violation of the integrality constraints. Therefore the FP can be seen as a form of iterated local search or basin hopping algorithm (see, e.g., [7, 28, 30]).

In this paper, we analyze in depth the relationship between the FP and the Frank–Wolfe algorithm. In this context, we define a new class of merit functions that can be included in the basic FP scheme [18]. A reported extended computational experience seems to indicate that the use of these new merit functions improves the FP efficiency.

The paper is organized as follows. In sections 2 and 3, we give a brief review of the FP and the objective feasibility pump heuristics. In section 4, we show the equivalence between the FP heuristic and the Frank–Wolfe algorithm applied to a nonsmooth merit function. In section 5, we define a new class of merit functions for measuring the solution integrality, we introduce new nonsmooth merit functions, and we discuss their properties. We present our algorithm in section 6. In section 7, we extend our approach by dynamically combining two different merit functions. Computational results are shown in section 8, where we give a detailed performance comparison of our algorithm with the FP. Further, we show that using somewhat more than one merit function at a time can improve the efficiency of the algorithm. Some conclusions are drawn in section 9.

In the following, given a concave function $f : R^n \to R$, we denote by $\partial f(x)$ the set of supergradients of f at the point x, namely,

$$\partial f(x) = \{v \in R^n \ : \ f(y) - f(x) \le v^T(y - x) \ \forall \ y \in R^n\}.$$

2. The feasibility pump heuristic. We consider an MIP problem of the form

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \ge b, \\ & x_j \in \{0, 1\} \ \forall j \in I, \end{aligned} \tag{1}$$

where $A \in \mathbf{R}^{m\times n}$ and $I \subset \{1,2,\ldots,n\}$ is the set of indices of 0-1 variables. Let $P = \{x : Ax \geq b,\, 0 \leq x_j \leq 1\, \forall\, j \in I\}$ denote the polyhedron of the LP relaxation of (1). The feasibility pump FP starts from the solution of the LP relaxation problem $\bar{x}^0 := \arg\min\{c^T x : x \in P\}$ and generates two sequences of points $\bar{x}^k$ and $\tilde{x}^k$: $\bar{x}^k$ is LP-feasible, but may be integer infeasible; $\tilde{x}^k$ is integer, but not necessarily LP-feasible. At each iteration $\bar{x}^{k+1} \in P$ is the nearest point in ℓ_1-norm to $\tilde{x}^k$:

$$\bar{x}^{k+1} := \arg\min_{x\in P} \Delta(x, \tilde{x}^k),$$

where

$$\Delta(x, \tilde{x}^k) = \sum_{j\in I} |x_j - \tilde{x}^k_j|.$$

The point $\tilde{x}^{k+1}$ is obtained as the rounding of $\bar{x}^{k+1}$. The procedure stops if at some index l, $\bar{x}^l$ is an integer or, in case of failing, if it reaches a time or iteration limit. In order to avoid stalling issues and loops, the FP performs a perturbation step. Here we report a brief outline of the basic scheme.

The feasibility pump (FP): Basic version
Initialization: Set $k = 0$, let $\bar{x}^0 := \arg\min\{c^T x : x \in P\}$
While (not stopping condition) **do**
  Step 1 If ($\bar{x}^k$ is integer) return $\bar{x}^k$
  Step 2 Compute $\tilde{x}^k = round(\bar{x}^k)$
  Step 3 If (cycle detected) $perturb(\tilde{x}^k)$
  Step 4 Compute $\bar{x}^{k+1} := \arg\min\{\Delta(x, \tilde{x}^k) : x \in P\}$
  Step 5 Update $k = k + 1$
End While

Now we give a better description of the rounding and the perturbing procedures used, respectively, at Step 2 and at Step 3 (see, e.g., [8, 18]).

- *Round:* This function transforms a given point $\bar{x}^k$ into an integer one, $\tilde{x}^k$. The easiest choice is that of rounding each component $\bar{x}^k_j$ with $j \in I$ to the nearest integer, while leaving the continuous components of the solution unchanged. Formally,

$$\tilde{x}^k_j = \begin{cases} [\bar{x}^k_j] & \text{if } j \in I, \\ \bar{x}^k_j & \text{otherwise,} \end{cases} \tag{2}$$

  where $[\cdot]$ represents scalar rounding to the nearest integer.
- *Perturb:* The aim of the perturbation procedure is to avoid cycling, and it consists in two heuristics, given specifically as follows:
  - If $\tilde{x}^k_j = \tilde{x}^{k+1}_j$ for all $j \in I$, a weak perturbation is performed, namely, a random number of integer constrained components, chosen as to minimize the increase in the distance $\Delta(\bar{x}^{k+1}, \tilde{x}^{k+1})$, is flipped.
  - If a cycle is detected by comparing the solutions obtained in the last 3 iterations, or in any case after R iterations, a strong random perturba-

tion is performed. For each $j \in I$ a uniformly random value is generated, $\rho_j \in [-0.3, 0.7]$, and if

$$|\bar{x}_j^{k+1} - \tilde{x}_j^{k+1}| + \max\{\rho_j, 0\} > 0.5,$$

the component $\tilde{x}_j^{k+1}$ is flipped.

Remark 1. The objective function $\Delta(x, \tilde{x}^k)$ discourages the optimal solution of the relaxation from being "too far" from $\tilde{x}^k$. In practice, the method tries to force a large number of variables of $\bar{x}^{k+1}$ to have the same (integer) value as $\tilde{x}^k$ (see [18]).

3. The objective feasibility pump. When using a heuristic like the FP on an MIP problem, one of the targets we have is that of finding a high-quality solution; that is, we would like to find a feasible point with the objective function $c^T x$ as small as possible. In general, since the FP scheme discards the original objective function of the problem after the first iteration, the quality of the feasible solutions found by the algorithm often tends to be poor. In order to overcome this drawback, in [1] a different approach, called the objective feasibility pump (OFP), has been developed. The idea is that of combining the original objective function $c^T x$ of the problem with the FP objective function. At each iteration the algorithm gradually reduces the influence of the objective function and increases the weight of $\Delta(x, \tilde{x})$. In this way the OFP, in its first iterations, concentrates its search on the region of high-quality points. The objective function of the LPs is a convex combination of the original objective function with the distance function $\Delta(x, \tilde{x})$:

$$\Delta_\theta(x, \tilde{x}) = \frac{1-\theta}{\|\Delta\|}\Delta(x, \tilde{x}) + \frac{\theta}{\|c\|}c^T x,$$

where $\|\Delta\| = \sqrt{|I|}$ and $\theta \in [0, 1]$. At each iteration k, the coefficient θ^k is decreased by a factor of $\nu < 1$ (i.e., $\theta^{k+1} = \nu\theta^k$). The introduction of the new function further requires a modification of the cycle detection step. While in the original scheme a cycle is found if the same integer point is visited twice, this is not the case in the modified scheme, because the objective function Δ_θ has changed in the meantime. The algorithm therefore stores, at each iteration k, the pair $(\tilde{x}^k, \theta^k)$ and a cycle is detected if there exist two iterations k_i and k_j, with $k_i < k_j$, such that $\tilde{x}^{k_i} = \tilde{x}^{k_j}$ and $\theta^{k_i} - \theta^{k_j} \leq \delta_\theta$, where $\delta_\theta \in [0, 1]$ is a fixed parameter.

4. The FP heuristic as a Frank–Wolfe algorithm for minimizing a nonsmooth merit function. In a recent work Eckstein and Nediak [10] noticed that the FP heuristic may be seen as a Frank–Wolfe procedure applied to a nonsmooth merit function. In order to better understand this equivalence we recall the unitary stepsize Frank–Wolfe method for concave nondifferentiable functions. Let us consider the problem

$$\begin{aligned} &\min f(x), \\ &x \in P, \end{aligned} \tag{3}$$

where $P \subset R^n$ is a nonempty polyhedral set that does not contain lines going to infinity in both directions, and $f : R^n \to R$ is a concave, nondifferentiable function, bounded below on P.

The Frank–Wolfe algorithm with unitary stepsize can be described as follows.

Frank–Wolfe unitary stepsize (FW1) algorithm
Initialization: Set $k = 0$, let $x^0 \in R^n$ be the starting point, compute $g^0 \in \partial f(x^0)$
While $x^k \notin \arg\min_{x \in P} (g^k)^T x$

Step 1 Compute a vertex solution x^{k+1} of

$$\min_{x \in P} (g^k)^T x$$

Step 2 Compute $g^{k+1} \in \partial f(x^{k+1})$, update $k = k + 1$

End While

The algorithm involves only the solution of LP problems, and the following result, proved in [32], shows that the algorithm generates a finite sequence and that it terminates at a stationary point $x^\star$, namely, a point satisfying the following condition:

$$(g^\star)^T(x - x^\star) \geq 0 \ \ \forall x \in P \tag{4}$$

with $g^\star \in \partial f(x^\star)$.

PROPOSITION 1. *The Frank–Wolfe algorithm with unitary stepsize converges to a vertex stationary point of problem* (3) *in a finite number of iterations.*

Now we consider the basic FP heuristic without any perturbation (i.e., without Step 3), and we show that it can be interpreted as the Frank–Wolfe algorithm with unitary stepsize applied to a concave, nondifferentiable merit function.

First of all, we can easily see that

$$\Delta(x, \tilde{x}^k) = \sum_{j \in I : \tilde{x}^k_j = 0} x_j + \sum_{j \in I : \tilde{x}^k_j = 1} (1 - x_j).$$

At each iteration, the FP for mixed 0-1 problems computes, at Step 2, the rounding of the solution $\bar{x}^k$, thus giving $\tilde{x}^k$. Then, at Step 4, it computes the solution of the LP problem

$$\begin{aligned} \bar{x}^{k+1} \in \arg\min \quad & \Delta(x, \tilde{x}^k) \\ \text{s.t.} \quad & Ax \geq b, \\ & 0 \leq x_j \leq 1 \ \forall j \in I. \end{aligned} \tag{5}$$

These two operations can be included in the unique step of calculating the solution of the following LP problem:

$$\begin{aligned} \min \quad & \sum_{j \in I : \bar{x}^k_j < \frac{1}{2}} x_j - \sum_{j \in I : \bar{x}^k_j \geq \frac{1}{2}} x_j \\ \text{s.t.} \quad & Ax \geq b, \\ & 0 \leq x_j \leq 1 \ \forall j \in I. \end{aligned} \tag{6}$$

Since the function

$$v(t) = \begin{cases} 1 & \text{if } t < \frac{1}{2}, \\ -1 & \text{if } t \geq \frac{1}{2} \end{cases} \tag{7}$$

is such that $v(t) \in \partial \min\{t, 1-t\}$, problem (6) can be seen as a generic iteration of the Frank–Wolfe method with unitary stepsize applied to the following minimization problem:

$$\begin{aligned} \min \quad & \sum_{i \in I} \min\{x_i, 1 - x_i\} \\ \text{s.t.} \quad & Ax \geq b, \\ & 0 \leq x_i \leq 1 \ \forall i \in I. \end{aligned} \tag{8}$$

5. New nonsmooth merit functions for the feasibility pump approach. As we have seen in the previous section, the basic FP is equivalent to minimizing a separable nonsmooth function which penalizes the 0-1 infeasibility, namely,

$$f(x) = \sum_{i \in I} \min\{x_i, 1 - x_i\}. \tag{9}$$

When using the Frank–Wolfe unitary stepsize algorithm for solving problem (8), at each iteration, if x^k is not a stationary point, we get a new point x^{k+1} such that

$$(g^k)^T (x^{k+1} - x^k) < 0,$$

with $g^k \in \partial f(x^k)$. Then from the concavity of the objective function we have

$$f(x^{k+1}) \leq f(x^k) + (g^k)^T (x^{k+1} - x^k) < f(x^k), \tag{10}$$

which means that at each iteration a reduction of the merit function is obtained. Anyway, this might not correspond to a reduction in the number of variables that violate integrality.

Example 1. Let us consider the following two points:

$$x = \left(0, \frac{1}{2}, 0, 0\right)^T; \qquad y = \left(0, \frac{1}{6}, \frac{1}{6}, 0\right)^T.$$

Let f be the function defined in (9). It is easy to notice that

$$f(y) < f(x),$$

but the number of noninteger components of y is greater than the number of noninteger components of x. As the main goal is finding an integer feasible solution, it would be better to use a function having the following features:

(i) it decreases whenever the number of integer variables increases;

(ii) if it decreases, then the number of noninteger variables does not increase.

A function satisfying these features is the following:

$$\psi(x) = card\{\, x_i : \, i \in I, \ x_i \notin \{0, 1\} \,\}. \tag{11}$$

The function (11) can be rewritten as:

$$\psi(x) = \sum_{i \in I} s(\min\{x_i, 1 - x_i\}), \tag{12}$$

where $s : \mathbf{R} \to \mathbf{R}^+$ is the *step function*:

$$s(t) = \begin{cases} 1 & \text{if } t > 0, \\ \\ 0 & \text{otherwise.} \end{cases}$$

Since the step function is a nonconvex and discontinuous function, minimizing (12) over a polyhedral set is a very hard problem. In the following we prove a general result to define approximations of function (12) that are easier to handle from a computational point of view and guarantee satisfaction of (i) and (ii) when evaluated on the vertices of a polyhedron.

PROPOSITION 2. *Let $V \subset [0,1]^n$ be the set of vertices of a polytope $P = \{ x : Ax \geq b,\ x \in [0,1] \}$. Let α_l and α_u be the following values:*

$$\alpha_l = \min_{x \in V} l(x),$$

$$\alpha_u = \min_{x \in V} u(x),$$

where

$$l(x) = \begin{cases} \min\{ x_i : i = 1, \ldots, n;\ x_i \neq 0 \} & \text{if } x \neq 0, \\ 1 & \text{if } x = 0; \end{cases}$$

$$u(x) = \begin{cases} \max\{ x_i : i = 1, \ldots, n;\ x_i \neq 1 \} & \text{if } x \neq e, \\ 1 & \text{if } x = e. \end{cases}$$

Let $\phi : [0,1]^n \to \mathbf{R}$ be a separable function

$$\phi(x) = \sum_{i \in I} \varphi(x_i). \tag{13}$$

We assume that $\varphi : [0,1] \to R$ satisfies the following:

(1)

$$\varphi(0) = \varphi(1); \tag{14}$$

(2) *there exists an $M > 0$ such that*

(i) *for $\bar{\alpha} \in \{0,1\}$ and $\tilde{\alpha} \in [\alpha_l, \alpha_u]$ we have*

$$\varphi(\bar{\alpha}) - \varphi(\tilde{\alpha}) \leq -M; \tag{15}$$

(ii) *for $\bar{\alpha}, \tilde{\alpha} \in [\alpha_l, \alpha_u]$ we have*

$$|\varphi(\bar{\alpha}) - \varphi(\tilde{\alpha})| \leq \frac{M}{n}. \tag{16}$$

Then, for $x, y \in V$,

(a) *$\psi(x) < \psi(y)$ implies $\phi(x) < \phi(y)$;*

(b) *$\phi(x) < \phi(y)$ implies $\psi(x) \leq \psi(y)$.*

Proof.

(a) We consider two points $x, y \in V$ such that $\psi(x) < \psi(y)$. We can define two sets of indices related to the noninteger components of x and y:

$$U = \{ i \in \{1, \ldots, n\} \mid i \in I,\ x_i \notin \{0,1\} \},$$

$$W = \{ j \in \{1, \ldots, n\} \mid j \in I,\ y_j \notin \{0,1\} \}.$$

Then we can write

$$\begin{aligned}\phi(x) - \phi(y) &= \sum_{i\in I}\varphi(x_i) - \sum_{j\in I}\varphi(y_j)\\ &= \sum_{i\in U}\varphi(x_i) + \sum_{i\in I\setminus U}\varphi(x_i) - \sum_{j\in W}\varphi(y_j) - \sum_{j\in I\setminus W}\varphi(y_j).\end{aligned} \tag{17}$$

Since $\psi(x) < \psi(y)$, we have that

$$|U| < |W|$$

and

$$|I \setminus U| > |I \setminus W|.$$

Let us first consider the case

$$|W| - |U| = 1.$$

We can assume that there exists an index $\bar{j}$ such that

$$W \setminus \{\bar{j}\} = U,$$

$$(I \setminus U) \setminus \{\bar{j}\} = I \setminus W.$$

Then we can write

$$\begin{aligned}&\phi(x) - \phi(y)\\ &= \varphi(x_{\bar{j}}) - \varphi(y_{\bar{j}}) + \sum_{j\in U}\varphi(x_j) + \sum_{\substack{j\in I\setminus U\\ j\neq\bar{j}}}\varphi(x_j) - \sum_{\substack{j\in W\\ j\neq\bar{j}}}\varphi(y_j) - \sum_{j\in I\setminus W}\varphi(y_j)\\ &= \varphi(x_{\bar{j}}) - \varphi(y_{\bar{j}}) + \sum_{\substack{j\in W\\ j\neq\bar{j}}}\varphi(x_j) + \sum_{\substack{j\in I\setminus U\\ j\neq\bar{j}}}\varphi(x_j) - \sum_{\substack{j\in W\\ j\neq\bar{j}}}\varphi(y_j) - \sum_{\substack{j\in I\setminus U\\ j\neq\bar{j}}}\varphi(y_j)\\ &= \varphi(x_{\bar{j}}) - \varphi(y_{\bar{j}}) + \sum_{\substack{j\in I\setminus U\\ j\neq\bar{j}}}(\varphi(x_j) - \varphi(y_j)) + \sum_{\substack{j\in W\\ j\neq\bar{j}}}(\varphi(x_j) - \varphi(y_j))\\ &\le \varphi(x_{\bar{j}}) - \varphi(y_{\bar{j}}) + \sum_{\substack{j\in I\setminus U\\ j\neq\bar{j}}}(\varphi(x_j) - \varphi(y_j)) + \sum_{\substack{j\in W\\ j\neq\bar{j}}}|\varphi(x_j) - \varphi(y_j)|.\end{aligned}$$

By using (14) we obtain

$$\phi(x) - \phi(y) \le \varphi(x_{\bar{j}}) - \varphi(y_{\bar{j}}) + \sum_{\substack{j\in W\\ j\neq\bar{j}}}|\varphi(x_j) - \varphi(y_j)|. \tag{18}$$

Now we notice that $x_{\bar{j}} \in \{0,1\}$, $y_{\bar{j}} \in [\alpha_l, \alpha_u]$, and $x_j, y_j \in [\alpha_l, \alpha_u]$ for all $j \in W \setminus \{\bar{j}\}$. Then, by using (15) and (16), we have

$$\begin{aligned}\phi(x) - \phi(y) &\le \varphi(x_{\bar{j}}) - \varphi(y_{\bar{j}}) + \sum_{\substack{j\in W\\ j\neq\bar{j}}}|\varphi(x_j) - \varphi(y_j)|\\ &\le -M + (|I| - 1)\frac{M}{n} < 0.\end{aligned}$$

Hence we have

$$\phi(x) < \phi(y).$$

Let us now consider the case

$$|W| - |U| > 1.$$

We can assume that there exists a set $\bar{J}$ such that

$$W \setminus \bar{J} = U,$$

$$(I \setminus U) \setminus \bar{J} = I \setminus W.$$

Then we can write

$$\begin{aligned}\phi(x) - \phi(y) = &\sum_{j\in\bar{J}}(\varphi(x_j) - \varphi(y_j)) \\ &+ \sum_{j\in U}\varphi(x_j) + \sum_{\substack{j\in I\setminus U \\ j\notin\bar{J}}}\varphi(x_j) - \sum_{\substack{j\in W \\ j\notin\bar{J}}}\varphi(y_j) - \sum_{j\in I\setminus W}\varphi(y_j).\end{aligned}$$

By using the same arguments used before we obtain

$$\phi(x) - \phi(y) \le \sum_{j\in\bar{J}}(\varphi(x_j) - \varphi(y_j)) + \sum_{j\in W\setminus\bar{J}}|\varphi(x_j) - \varphi(y_j)|. \tag{19}$$

Now we notice that $x_j \in \{0,1\}$, $y_j \in [\alpha_l, \alpha_u]$ for all $j \in \bar{J}$ and $x_j, y_j \in [\alpha_l, \alpha_u]$ for all $j \in W \setminus \bar{J}$. Then, by using (15) and (16), we have

$$\begin{aligned}\phi(x) - \phi(y) &\le \sum_{j\in\bar{J}}(\varphi(x_j) - \varphi(y_j)) + \sum_{j\in W\setminus\bar{J}}|\varphi(x_j) - \varphi(y_j)| \\ &\le -M|\bar{J}| + (|I| - |\bar{J}|)\frac{M}{n} < 0.\end{aligned}$$

Once again we have

$$\phi(x) < \phi(y).$$

(b) We assume by contradiction that there exist two points $x, y \in V$ such that $\phi(x) < \phi(y)$ and

$$\psi(x) > \psi(y). \tag{20}$$

By (20), recalling the first part of the proof, we have that $\phi(x) > \phi(y)$, which contradicts our initial assumption. □

Summarizing, if an approximation $\phi(x)$ satisfying the assumptions of Proposition 2 is available, we can solve, in place of the original FP problem (8), the following problem:

$$\begin{aligned}\min \quad & \phi(x) = \sum_{i\in I}\varphi(x_i) \\ \text{s.t.} \quad & Ax \ge b, \\ & 0 \le x_i \le 1 \ \forall i \in I.\end{aligned}$$

As the method we use for solving the minimization problem stated above is the Frank–Wolfe algorithm, which at each step moves from one vertex to another, guaranteeing the reduction of the chosen approximation, we have (by point (b) of Proposition 2) that, at each iteration of the algorithm, the number of the noninteger components of the current solution does not increase.

Taking into account Proposition 2 and the ideas developed in [31, 34, 35], we consider the following $\varphi(\cdot)$ terms to be used in the objective function of problem (21).

Logarithmic function:

$$\varphi(t) = \min\big\{\ln(t+\varepsilon), \ln[(1-t)+\varepsilon]\big\}. \tag{21}$$

Hyperbolic function:

$$\varphi(t) = \min\big\{-(t+\varepsilon)^{-p}, -[(1-t)+\varepsilon]^{-p}\big\}. \tag{22}$$

Exponential function:

$$\varphi(t) = \min\big\{1-\exp(-\alpha t), 1-\exp(-\alpha(1-t))\big\}. \tag{23}$$

Logistic function:

$$\varphi(t) = \min\big\{[1+\exp(-\alpha t)]^{-1}, [1+\exp(-\alpha(1-t))]^{-1}\big\}. \tag{24}$$

In (21)–(24), $\varepsilon, \alpha, p > 0$. In Figure 1, we compare the φ term related to the FP heuristic with those given by (21)–(24).

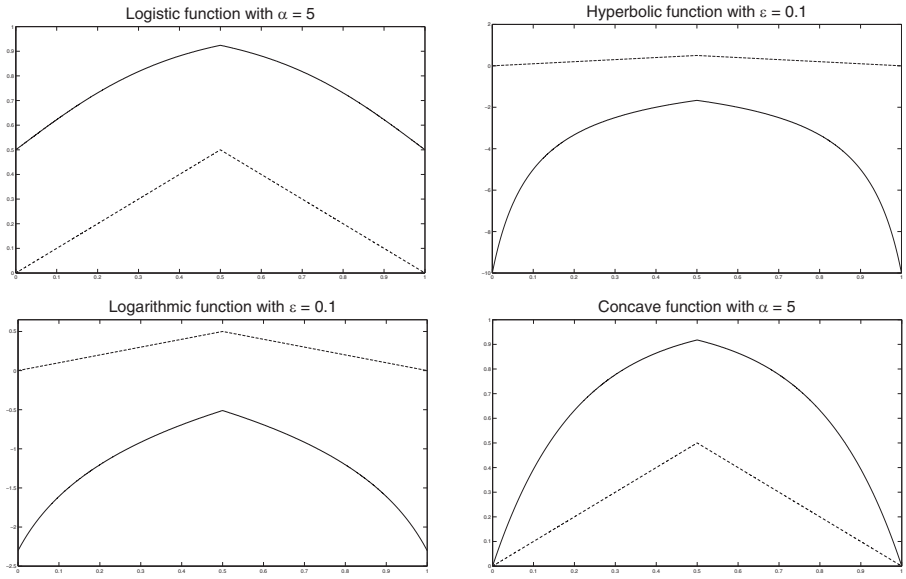

FIG. 1. *Comparison between the original FP term (dotted line) and the new terms (solid line).*

Now we prove that, for a particular choice of the φ term, the assumptions of Proposition 2 are satisfied.

PROPOSITION 3. *For the term* (21)*, there exists a value* $\bar{\varepsilon} > 0$ *such that for any* $\varepsilon \in (0, \bar{\varepsilon}]$ *assumptions* (1) *and* (2) *of Proposition* 2 *are satisfied.*

Proof. It can be easily noticed that when $x \in \{0,1\}$ we have

$$\varphi(x) = \ln \varepsilon;$$

then assumption (1) of Proposition 2 is satisfied.

Now, without any loss of generality, we suppose

$$\alpha_l = \min\{\alpha_l, 1 - \alpha_u\} \tag{25}$$

and notice that there exists a value $\bar{\varepsilon} > 0$ such that for any $\varepsilon \in (0, \bar{\varepsilon}]$ the following inequality holds:

$$\ln \varepsilon - \ln(\alpha_l + \varepsilon) + n(\ln(1/2 + \varepsilon) - \ln(\alpha_l + \varepsilon)) \leq 0. \tag{26}$$

As the function $\varphi(t)$ is strictly increasing in $[0, \frac{1}{2}]$ and strictly decreasing in $(\frac{1}{2}, 1]$, and it is symmetric with respect to the point $t = \frac{1}{2}$, we have for $\bar{\alpha} \in \{0,1\}$ and $\tilde{\alpha} \in [\alpha_l, \alpha_u]$

$$\varphi(\bar{\alpha}) - \varphi(\tilde{\alpha}) \leq \varphi(0) - \varphi(\alpha_l).$$

Then we set

$$M = \varphi(\alpha_l) - \varphi(0) = \ln(\alpha_l + \varepsilon) - \ln \varepsilon, \tag{27}$$

and (i) in assumption (2) of Proposition 2 is satisfied.

As the maximum of $\varphi(t)$ is attained at $t = \frac{1}{2}$ and due to the structure of function $\varphi(t)$, we have the following for any choice of $\bar{\alpha}, \tilde{\alpha} \in [\alpha_l, \alpha_u]$:

$$|\varphi(\bar{\alpha}) - \varphi(\tilde{\alpha})| \leq \varphi(1/2) - \varphi(\alpha_l). \tag{28}$$

Since (ii) in assumption (2) needs to be verified for any choice of $\bar{\alpha}, \tilde{\alpha} \in [\alpha_l, \alpha_u]$, by (28) it is sufficient to show that

$$\varphi(1/2) - \varphi(\alpha_l) \leq \frac{M}{n}.$$

By using (27) and (26), we can easily verify that for any $\varepsilon \in (0, \bar{\varepsilon}]$, the following inequality holds:

$$\begin{aligned} &\varphi(0) - \varphi(\alpha_l) + n(\varphi(1/2) - \varphi(\alpha_l)) \\ &= \ln \varepsilon - \ln(\alpha_l + \varepsilon) + n(\ln(1/2 + \varepsilon) - \ln(\alpha_l + \varepsilon)) \leq 0. \end{aligned} \tag{29}$$

Then (ii) in assumption (2) of Proposition 2 is satisfied. ∎

The result proved in Proposition 3 for the term (21) can also be proved for the terms (22)–(24) repeating the same arguments, and thus all the merit functions (21)–(24) are suitable to penalize the number of variables that violate the integrality constraints.

We remark that functions (21)–(24) have also another interesting theoretical property: they can be used in an exact penalty approach like that proposed in [31]. In fact, it is possible to prove that terms (21)–(24) can be used to transform an MIP problem into an equivalent continuous problem.

PROPOSITION 4. *Let f be a Lipschitz continuous function bounded on P. For every penalty term*

$$\phi(x) = \sum_{i \in I} \varphi(x_i)$$

with φ as in (21)–(24) *a value $\bar{\varepsilon} > 0$ exists such that, for any $\varepsilon \in]0, \bar{\varepsilon}]$, problem*

$$\min f(x) \quad s.t. \quad x \in P, \quad x_i \in \{0,1\} \ \forall i \in I, \tag{30}$$

and problem

$$\min f(x) + \tilde{\phi}(x,\varepsilon) \quad s.t. \quad x \in P, \quad 0 \le x_i \le 1 \ \forall i \in I, \tag{31}$$

where

$$\tilde{\phi}(x,\varepsilon) = \begin{cases} \phi(x) & \text{if } \varphi \text{ is given by (21)–(22)}, \\ \dfrac{1}{\varepsilon}\ \phi(x) & \text{if } \varphi \text{ is given by (23)–(24)}, \end{cases}$$

have the same minimum points.

Proof. The proof follows the same arguments as in [31]. See [16] for further details. □

This result suggests that these new merit functions can be used to define new FP heuristics that improve the quality of the solution in terms of objective function value like those proposed in [1] and [10]. In fact, the heuristic proposed in [1] can be seen as a Frank–Wolfe algorithm applied to problem (31) with the penalty term (9). Furthermore, the restarting rules used in the FP algorithm can be reinterpreted as techniques for escaping from noninteger stationary points.

We can also include these functions into an algorithmic framework to determine the minimizer of a nonlinear programming problem with integer variables (see, e.g., [33]). Anyway, the use of the continuous reformulation of the original mixed integer problem is beyond the scope of this paper and will be the subject of future work.

In the next section we will focus on finding a first feasible solution to an MIP problem. In particular, we tackle problem (21) by a modified FP approach based on the concave functions described above.

6. A reweighted version of the feasibility pump heuristic. The use of the merit functions (21)–(24) defined in the previous section leads to a new FP scheme where the ℓ_1-norm used for calculating the next LP-feasible point is replaced with a "weighted" ℓ_1-norm of the form

$$\Delta_W(x,\tilde{x}) = \sum_{j \in I} w_j |x_j - \tilde{x}_j| = \|W(x - \tilde{x})\|_1, \tag{32}$$

where

$$W = \operatorname{diag}(w_1, \ldots, w_n)$$

and w_j, $j = 1, \ldots, n$, are positive weights depending on the merit function ϕ chosen. The main feature of the method is the use of an infeasibility measure that

- tries to discourage the optimal solution of the relaxation from being far from $\tilde{x}$ (similarly to the original FP algorithm);
- takes into account, in some way, the information carried by the LP-feasible points obtained at the previous iterations of the algorithm for speeding up the convergence to 0-1 feasible points.

A possible choice for the weights w_j, $j = 1, \ldots, n$, is the following:

$$w_j = |g_j|, \quad j = 1, \ldots, n,$$

where $g \in \partial\phi(\bar{x})$ and $\bar{x}$ is the LP-feasible point obtained at the previous iteration of the algorithm.

Here we report an outline of the algorithm.

Reweighted feasibility pump (RFP): Basic version
Initialization: Set $k = 0$, let $\bar{x}^0 := \arg\min\{c^T x : x \in P\}$
While (not stopping condition) **do**
Step 1 If ($\bar{x}^k$ is integer) return $\bar{x}^k$
Step 2 Compute $\tilde{x}^k = round(\bar{x}^k)$
Step 3 If (cycle detected) $perturb(\tilde{x}^k)$
Step 4 Compute $\bar{x}^{k+1} := \arg\min\{\|W^k(x - \tilde{x}^k)\|_1 : x \in P\}$
Step 5 Update $k = k + 1$
End While

We assume that the *round* and *perturb* procedures are the same as those described in section 2 for the original version of the FP heuristic. Anyway, different rounding and perturbing procedures can be suitably developed.

In particular, the rounding procedure could be replaced with a scheme based on constraint propagation like that one proposed in [20]. Other possibilities can be inspired by the procedures recently proposed in [3, 9] examining rounded solutions along suitable line segments.

Following the same reasoning as that of section 4, we can reinterpret the reweighted FP heuristic without perturbation as the unitary stepsize Frank–Wolfe algorithm applied to the merit function ϕ. Let us now consider a generic iteration k of the reweighted FP. At Step 2, the algorithm rounds the solution $\bar{x}^k$, thus giving $\tilde{x}^k$. Then, at Step 4, it computes the solution of the LP problem

$$\begin{aligned} \bar{x}^{k+1} \in \arg\min \quad & \Delta_{W^k}(x, \tilde{x}^k) \\ \text{s.t.} \quad & Ax \geq b, \\ & 0 \leq x_j \leq 1 \ \forall j \in I. \end{aligned}$$

Similarly to the FP algorithm, these two operations can be included in the unique step of calculating the solution of the following LP problem:

$$\begin{aligned} \min \quad & \sum_{j \in I : \bar{x}^k_j < \frac{1}{2}} w^k_j x_j - \sum_{j \in I : \bar{x}^k_j \geq \frac{1}{2}} w^k_j x_j \\ \text{s.t.} \quad & Ax \geq b, \\ & 0 \leq x_j \leq 1 \ \forall j \in I. \end{aligned} \tag{33}$$

By setting

$$w^k_j = |g^k_j|$$

with $g^k \in \partial\phi(\bar{x}^k)$, problem (33), as we have already said, can be seen as the iteration of the Frank–Wolfe method with unitary stepsize applied to the minimization problem (21).

In order to highlight the differences between the ℓ_1-norm and the weighted ℓ_1-norm we report the following example.

Example 2. Consider the following mixed integer linear programming problem:

$$\begin{aligned} &\min\ c^T x \\ &\text{s.t. } x \in P, \\ &\quad x \in \{0,1\}^3, \end{aligned} \tag{34}$$

where $P \subset [0,1]^3$ is the polyhedron in Figure 2. Let $x^L = \left(\frac{9}{20}, \frac{1}{8}, \frac{1}{8}\right)$ be the solution of the linear relaxation of (34) and let $x^I = (0,0,0)$ be its rounding.

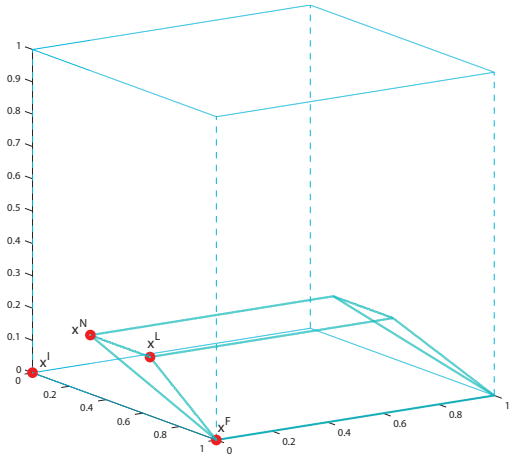

FIG. 2. *Feasible set of problem* 34.

The minimization of $\Delta(x, x^I) = \|x - x^I\|_1$ over P leads to $x^N = \left(\frac{1}{8}, \frac{1}{8}, \frac{1}{8}\right)$, since $\|x^N - x^I\|_1 < \|x - x^I\|_1$ for all $x \in P$.

Consider now the weighted ℓ_1-norm obtained using the logarithmic merit function

$$\phi(x) = \sum_{i \in I} \min\left\{ \ln(x_i + \varepsilon), \ln[(1 - x_i) + \varepsilon] \right\},$$

where ε is a small positive value. By minimizing the weighted distance between x and x^I over P, we obtain the point $x^F = (1,0,0)$. In fact, we have

$$\Delta_W(x^F, x^I) < \Delta_W(x, x^I)$$

for all $x \in P$. Thus the ℓ_1-norm finds a solution which does not satisfy the integrality constraints, while the reweighted ℓ_1-norm gets an integer feasible solution.

We want to remark that the original FP algorithm is a special case of the RFP obtained by setting $W^k = I$.

We can further use the merit functions (21)–(24) in the OFP approach recalled in section 3 to obtain a reweighted version of the algorithm, the objective reweighted feasibility pump (ORFP). The new objective function of the LPs becomes the following:

$$\Delta_{W,\theta}(x, \tilde{x}) = \frac{1-\theta}{\|\Delta\|} \Delta_W(x, \tilde{x}) + \frac{\theta}{\|c\|} c^T x, \tag{35}$$

where $\|\Delta\| = \sqrt{|I|}$ and $\theta \in [0,1]$. As in the standard OFP, at each iteration k, the coefficient θ^k is decreased by a factor $\nu < 1$ (i.e., $\theta^{k+1} = \nu\theta^k$).

Anyway, this choice follows exactly the approach proposed in [1] and does not take into account the fact that the proposed merit functions and the original FP merit function have different behaviors. Hence, new approaches could be developed to combine those merit functions with the original objective function (e.g., a convex combination with different coefficients and updating rules).

7. Combining two merit functions. As we have already said, the main drawback of the FP heuristic is its tendency to stall (i.e., to get stuck in a point that is not an integer feasible solution). For this reason, a random perturbation (or a restart) is performed. A good idea might be that of modifying the objective function (in addition to the random perturbation/restart usually adopted) anytime the algorithm stalls. This modification may help escaping from the last stationary point obtained and speed up the convergence to an integer feasible solution. A possibility might be that of considering a convex combination of two different merit functions,

$$\phi(x) = \lambda\phi_1(x) + (1-\lambda)\phi_2(x), \tag{36}$$

with $\lambda \in [0,1]$, and modifying the λ parameter as soon as the algorithm stalls. This is equivalent to using, in the RFP algorithm, the following:

(1) a matrix W^k with the terms

$$w_j^k = \lambda^k |g_j^k| + (1-\lambda^k)|h_j^k|, \quad j = 1, \ldots, n,$$

where $g_j^k \in \partial\phi_1(\bar{x}^k)$ and $h_j^k \in \partial\phi_2(\bar{x}^k)$;

(2) an updating rule for the λ parameter that slightly (significantly) changes the penalty term anytime a perturbation (restart) is performed.

In Figure 3 we can see the behavior of a function obtained by combining the exponential and the logistic function.

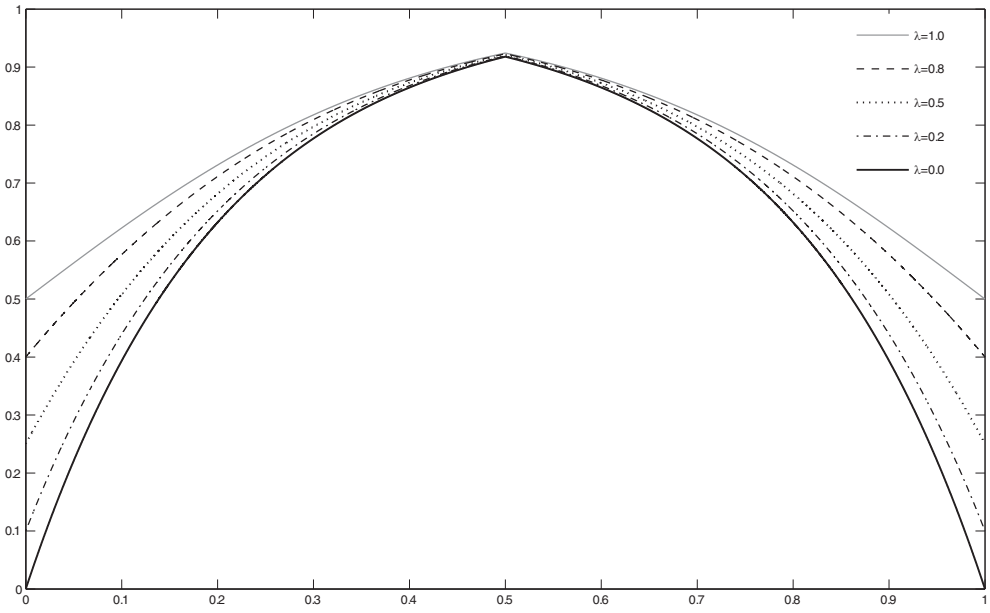

FIG. 3. *Behavior of the function obtained combining exponential and logistic function.*

8. Numerical results. In this section we report computational results to compare our version of the FP algorithm with the original FP described in [18] and the OFP described in [1]. The test set used in our numerical experience consists of 153 instances of 0-1 problems from the MIPLIB2003 [2] and COR@L libraries. All the algorithms were implemented in C, and we have used ILOG Cplex [27] as the solver of the LP problems. All tests have been run on an Intel Core2 E8500 system (3.16GHz) with 3.25GB of RAM.

We compare the FP with the reweighted version in different scenarios:

1. *Randomly generated starting points.* For the terms (9), (21)–(24), we solved the corresponding penalty formulation (21) by means of the Frank–Wolfe algorithm using 1000 randomly generated starting points. The aim of the experiment was to highlight the ability of each penalty formulation to find an integer feasible solution.
2. *FP vs. RFP.* In order to evaluate the effectiveness of the new penalty functions, we compared the FP algorithm with the RFP algorithm, where the distance $\Delta_W(x, \tilde{x})$ is defined using the terms (21)–(24).
3. *FP vs. combined RFP.* We made a comparison between the FP algorithm and the RFP algorithm where the distance $\Delta_W(x, \tilde{x})$ is the combination of two different penalty terms. The aim of the experiment was to show that the combination of two different functions can somehow improve the RFP algorithm performance.
4. *OFP vs. ORFP.* We made a comparison between the OFP and the ORFP. In this experiment, the distance $\Delta_{W,\theta}(x, \tilde{x})$ is the combination of the original objective function of the problem considered and the exponential and logistic penalty terms.
5. *OFP vs. combined ORFP.* We made a comparison between the OFP and the combined ORFP. In this experiment, the distance $\Delta_{W,\theta}(x, \tilde{x})$ is the combination of the original objective function of the problem considered and a term given by the combination of the exponential and logistic penalty terms. The aim of the experiment was to show that the combination of the two merit functions proposed is beneficial also for the OFP.

The choice of the merit function parameters is critical for the efficiency of the algorithm. On one hand, by following Proposition 2, it would be better setting the parameter of a chosen merit function to a sufficiently small value. On the other hand, when the parameter is very small, the slope of the graph related to the function φ gets very large close to 0 or 1, thus making the problem, in some cases, hard to solve. We performed our experiments using

- penalty term (9) denoted by FP;
- penalty term (21) denoted by Log, with $\varepsilon = 0.1$;
- penalty term (22) denoted by Hyp, with $\varepsilon = 0.1$;
- penalty term (23) denoted by Exp, with $\alpha = 0.5$;
- penalty term (24) denoted by Logis, with $\alpha = 0.1$.

On the basis of our numerical experiments, the values of the parameters reported above represent a good compromise between theory and practice.

In scenarios 2, 3, 4, and 5, we stop the algorithms if an integer solution is found or if the limit of 1500 iterations is reached. Due to the random effects introduced by perturbations and major restarts, each problem is tested on a particular penalty function on 10 runs (with different random seeds).

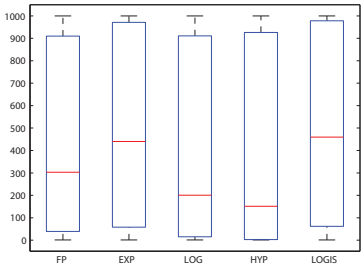

FIG. 4. *Comparison between the original FP term and the new terms for randomly generated starting points.*

8.1. Computational results for randomly generated starting points. In this first experiment, we applied the Frank–Wolfe algorithm to solve problem (21) with the objective functions (9), (21)–(24). The algorithm stops when it finds a stationary point (which is not necessarily integer feasible). The goal of the experiment was to understand how good each function is at finding an integer feasible solution. In order to obtain reliable statistics we used 1000 randomly generated starting points. The results obtained on the MIP problems when using randomly generated starting points are shown in Figure 4, where we report the box plots related to the distribution of the number of integer feasible solutions found by each function (we discarded problems where no function found an integer feasible solution). On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually.

We can observe that the results obtained by means of the Exp and the Logis functions, in terms of number of integer feasible solutions found, are slightly better than those obtained using the FP. FP, in turn, gives better results than Log and Hyp penalty functions.

This preliminary computational experience seems to show that the functions have a different behavior in forcing the integrality of the solution. These diversities could be somehow exploited into a multistart strategy. In particular, we could develop a new framework where the minimization of different functions is carried out in parallel. In order to investigate the effect of the parallel use of different functions, we applied the Frank–Wolfe algorithm to three merit functions (using three different randomly generated starting points) and chose the solution with the highest number of integer components among the three. We compared this strategy with the one where we use the same merit function on three different starting points. In Figure 5, we report the results obtained on 333 repetitions of the parallel experiment, when using for each repetition

- the same merit function with three different starting points;
- three different merit functions (FP, Exp, and Logis), each with a different starting point.

We discarded problems where in both cases no integer feasible solution over the 333 repetitions was found. We can see from Figure 5 that the use of three different merit functions in parallel outperforms the use of only one merit function in the case of FP and Exp. The difference in the performances between the use of three different merit functions in parallel and the use of the Logis merit function is less evident; however, the results obtained by the Logis function have a median of 298.0

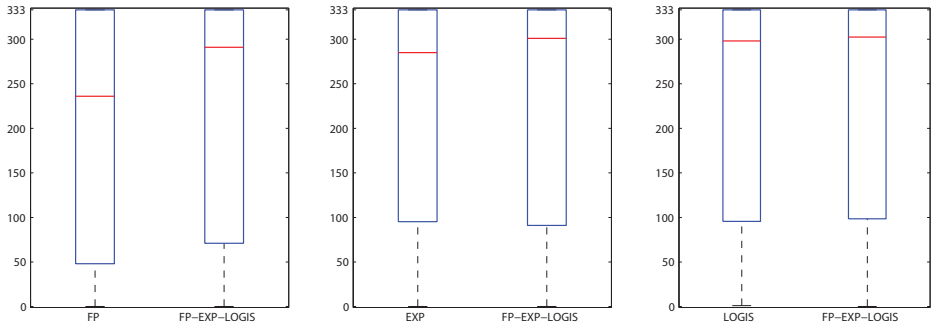

FIG. 5. *Number of integer feasible solutions found in the parallel experiment.*

and a 25th percentile of 95.5, while the results obtained by using three different merit functions have a median of 302.5 and a 25th percentile of 98.5. The results obtained in the parallel experiment suggest that, in a multistart strategy, the use of different merit functions can help diversify the local minima computed by the algorithm, thus increasing the number of integer feasible solutions found.

8.2. Comparison between FP and RFP. In order to evaluate the ability of finding a first feasible solution, we report in Table 1, for each penalty term,

- the number of problems for which no feasible solution has been found (Not found);
- the number of problems for which a feasible solution has been found at least once, but fewer than ten times (Found at least once);
- the number of problems for which a feasible solution has been found for all ten runs (Found 10 times);
- the average number of feasible solutions found (Average number of f.s. found).

As we can see from Table 1, FP, Exp, and Logis terms have a similar behavior and are slightly better than Hyp and Log terms.

In order to show the efficiency in terms of objective function value, we consider the 108 problems for which an integer feasible solution is found in all ten runs by all of the algorithms and, in Table 2, we report for each penalty term

- the number of problems for which the best average objective function value (average over ten runs) is obtained (Best average o.f.);
- the number of problems for which the best objective function value (minimum over ten runs) is obtained (Best min o.f.).

As we can see by taking a look at Table 2, the Log and Hyp terms give the best performance in terms of objective function value. Furthermore, Exp and Logis terms are comparable and perform better than FP term.

TABLE 1
Comparison between FP and RFP (feasible solutions).

	Not found	Found at least once	Found 10 times	Average number of f.s. found
FP	16	9	128	8.61
Exp	15	11	127	8.75
Log	18	15	120	8.28
Hyp	27	15	111	7.65
Logis	16	11	126	8.71

TABLE 2
Comparison between FP and RFP (objective function value).

	Best average o.f.	Best min o.f.
FP	24	24
Exp	28	27
Log	30	26
Hyp	32	28
Logis	27	25

TABLE 3
Comparison between FP and RFP (geometric means).

FP		Exp, $\alpha = 0.5$		Log, $\varepsilon = 0.1$		Hyp, $\varepsilon = 0.1$		Logis, $\alpha = 0.1$	
Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
5.774	1.793	4.851	1.683	5.684	1.657	7.193	1.757	4.869	1.678

The detailed results of the comparison between the FP algorithm and the reweighted version obtained using the penalty terms (21)–(24) are reported in Tables 17–21 of [16]. In particular, both the results related to the problems for which an integer feasible solution is found in all ten runs (see Tables 17–19 of [16]) and the results related to the problems for which an integer feasible solution is found in fewer than ten runs (see Tables 20–21 of [16]) are reported. By taking a look at those tables, we can notice that the RFP algorithm obtained using the Exp merit function (Exp RFP algorithm) and the one obtained using the Logis merit function (Logis RFP algorithm) are competitive with the FP in terms of both number of iterations and CPU time. They are also better than the RFP algorithm with the Log merit function (Log RFP algorithm) and the one with the Hyp merit function (Hyp RFP algorithm) that, in addition, have a larger number of failures. Despite these facts, Log RFP and Hyp RFP algorithms generally give good results in terms of objective function value. In order to better assess the differences in terms of iterations and CPU time between FP and the various versions of the RFP algorithm, we report in Table 3 the geometric means for all the algorithms calculated over 108 instances (those problems for which a feasible solution is found in all the ten runs). In the calculations of the geometric means, individual values smaller than 1 are replaced by 1. The results in Table 3 seem to confirm that Exp and Logis RFP algorithms are competitive with the FP algorithm.

In order to better assess the differences between the FP algorithm and the RFP algorithm, we considered the 123 problems for which an integer feasible solution is found in all ten runs by the FP, Exp RFP, and Logis RFP algorithms. We divided the problems into three different classes depending on the CPU time t (seconds) needed by the algorithms to find a feasible solution:

- *Easy.* Problems for which a feasible solution has been found by all the algorithms in a time $t \leq 1$ (76 problems).
- *Hard.* Problems for which a feasible solution has been found by any algorithm in a time $t > 20$ (12 problems).
- *Medium.* All the problems that are neither Easy nor Hard (35 problems).

We report in Figure 6 the results, in terms of CPU time, obtained by the FP, Exp RFP, and Logis RFP algorithms on the three classes of problems. Exp RFP and Logis RFP are comparable with FP on the Easy and Medium classes, while they outperform it on the Hard class. Once again, we could develop a new framework where different algorithms are used in parallel. In order to investigate the effect of the parallel use of

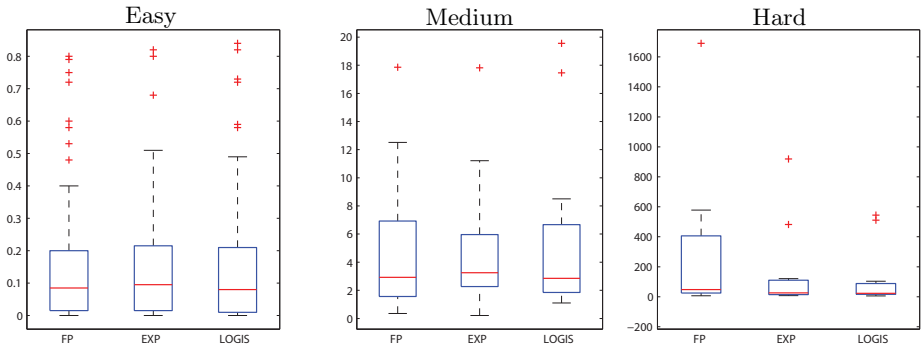

FIG. 6. *Results in terms of CPU time for the three classes of problems.*

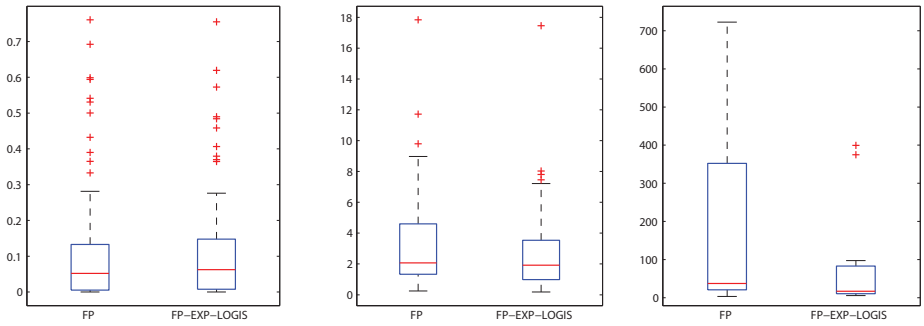

FIG. 7. *Results in terms of CPU time for the parallel experiment.*

different algorithms, we ran three algorithms and chose the solution with the lowest CPU time among the three. We report in Figure 7 the results obtained using

- three runs of the FP algorithm;
- one different algorithm (FP, Exp RFP, and Logis RFP) for each run.

By taking a look at those results, we can see that the use of different functions improves the performance in the Medium and Hard classes, while giving comparable results on the Easy class.

8.3. Comparison between FP and combined RFP. In this subsection, we show the effects of combining two different functions. We report the results obtained combining the following functions:

- FP term and Log term, denoted by FP+Log;
- Exp term and Log term, denoted by Exp+Log;
- Logis term and Log term, denoted by Logis+Log;
- Exp term and Logis term, denoted by Exp+Logis.

We set $\phi_1(x)$ equal to the merit function obtained using the first term and $\phi_2(x)$ equal to the merit function obtained using the second term (see (36)). We start with $\lambda^0 = 1$ and reduce it every time a perturbation occurs. More precisely, we can have two different cases:

- *Weak perturbation update:* $\lambda^{k+1} = 0.5\lambda^k$.
- *Strong perturbation update:* $\lambda^{k+1} = 0.1\lambda^k$.

When a strong perturbation occurs, it means that the algorithm is stuck in a cycle. Then the updating rule significantly changes the penalty term, thus moving towards the function belonging to the second class.

TABLE 4
Comparison between FP and combined RFP (feasible solutions).

	Not found	Found at least once	Found 10 times	Average number of f.s. found
FP	16	9	128	8.61
FP+Log	17	11	125	8.61
Exp+Log	19	6	128	8.55
Logis+Log	17	9	127	8.58
Exp+Logis	16	10	127	8.59

TABLE 5
Comparison between FP and combined RFP (objective function value).

	Best average o.f.	Best min o.f.
FP	19	19
FP+Log	31	30
Exp+Log	35	33
Logis+Log	32	30
Exp+Logis	32	30

In order to evaluate the ability of finding a first feasible solution, we report in Table 4, for each penalty term,

- the number of problems for which no feasible solution has been found (Not found);
- the number of problems for which a feasible solution has been found at least once, but fewer than ten times (Found at least once);
- the number of problems for which a feasible solution has been found for all ten runs (Found 10 times);
- the average number of feasible solutions found (Average number of f.s. found).

As we can see from Table 4, all terms have a similar behavior.

In order to show the efficiency in terms of objective function value, we consider the 123 problems for which an integer feasible solution is found in all ten runs by all of the algorithms and, in Table 5, we report for each penalty term

- the number of problems for which the best objective function value (average over ten runs) is obtained (Best average o.f.);
- the number of problems for which the best objective function value (minimum over ten runs) is obtained (Best min o.f.).

As we can see by taking a look at Table 5, the combined terms give better performance in terms of objective function value than the FP term. Furthermore, the Exp+Log combination gives the best performance.

The detailed results of the comparison between the FP algorithm and the reweighted version obtained using the combined penalty terms are shown in Tables 22–26 of [16]. In particular, both the results related to the problems for which an integer feasible solution is found in all the ten runs (see Tables 22–24 of [16]) and the results related to the problems for which an integer feasible solution is found in fewer than ten runs (see Tables 25–26 of [16]) are reported. By taking a look at those tables, we can notice that the combined RFP algorithm obtained using the Exp and the Logis merit functions (Exp+Logis RFP algorithm) gives the best performance. Furthermore, all the versions of the combined RFP algorithm are competitive with the standard FP algorithm. We report in Table 6 the geometric means for all the algorithms calculated over 123 instances (those problems for which a feasible solution is found in all ten runs). In the calculations of the geometric means individual values smaller than

TABLE 6
Comparison between FP and combined RFP (geometric means).

FP		FP+Log		Exp+Log		Logis+Log		Exp+Logis	
Iter	Time	Iter	Time	Iter	Time	Iter	Time	Iter	Time
6.252	2.034	6.474	1.630	6.438	1.650	6.388	1.663	5.765	1.617

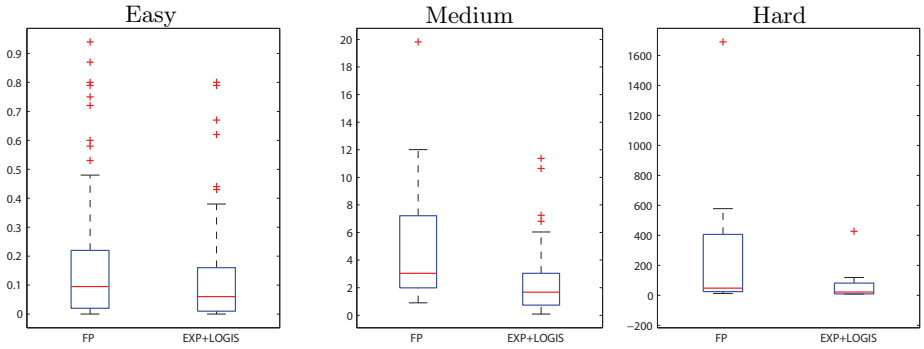

FIG. 8. *Results in terms of CPU time for the three classes of problems.*

1 are replaced by 1. The results in Table 6 seem to confirm that the Exp+Logis RFP algorithm is the best among the combined versions of the RFP algorithm and that all the combined RFP algorithms behave favorably when compared to the original FP algorithm in terms of CPU time.

In order to better assess the differences between the FP algorithm and the Exp+Logis RFP algorithm, we considered the 124 problems for which an integer feasible solution is found in all ten runs by the two algorithms. We divided the problems into three different classes depending on the CPU time t (seconds) needed by the algorithms to find a feasible solution:

- *Easy.* Problems for which a feasible solution has been found by all of the algorithms in a time $t \leq 1$ (80 problems).
- *Hard.* Problems for which a feasible solution has been found by any algorithm in a time $t > 20$ (12 problems).
- *Medium.* All problems that are neither Easy nor Hard (32 problems).

We report in Figure 8 the results, in terms of CPU time, obtained by the FP and Exp+Logis RFP algorithms on the three classes of problems. As we can see, Exp+Logis RFP improves the performance in all the classes.

8.4. Comparison between OFP and ORFP. In the following we report a comparison between the OFP [1] and the ORFP with the Exp (Exp ORFP) and Logis (Logis ORFP) terms. In order to evaluate the ability of finding a first feasible solution, we report in Table 7, for each penalty term,

- the number of problems for which no feasible solution has been found (Not found);
- the number of problems for which a feasible solution has been found at least once, but fewer than ten times (Found at least once);
- the number of problems for which a feasible solution has been found for all ten runs (Found 10 times);
- the average number of feasible solutions found (Average number of f.s. found).

TABLE 7
Comparison between OFP and ORFP (feasible solutions).

	Not found	Found at least once	Found 10 times	Average number of f.s. found
OFP	17	29	107	7.99
Exp ORFP	17	32	104	7.94
Logis ORFP	21	21	111	7.92

TABLE 8
Improvement in the quality of the solution by the introduction of the objective function in the FP and in the RFP.

	FP		OFP		Exp RFP		Exp ORFP		Logis RFP		Logis ORFP	
	Gap %	Time	Gap %	Time	Gap %	Time	Gap %	Time	Gap %	Time	Gap %	Time
binkar10-1	8936	0.2	18	0.1	13388	0.2	18	0.1	14873	0.2	18	0.1
dano3-3	73	31.7	0	53.2	73	13.3	0	79.3	73	17.7	0	104.8
dano3-4	73	23.9	0	119.2	73	13.6	0	108.5	69	15.9	0	96.4
dano3-5	72	26.5	0	104.1	73	14.5	0	128.3	73	16.5	0	134.5
neos-476283	160	444.7	1	47.7	80	121.2	1	47.2	68	71.8	2	48.0
neos-780889	216	48.2	0	13.4	223	52.4	0	13.4	219	50.2	0	13.3
qap10	48	1690.5	14	27.8	19	7.5	20	36.0	19	8.7	3	21.2

As we can see from Table 7, OFP, Exp ORFP, and Logis ORFP terms have a similar average number of feasible solutions found. Logis ORFP has the highest number of failures in terms of number of problems for which no feasible solution has been found, but also the highest number of problems for which a feasible solution has been found for all ten runs.

The detailed results of the comparison between the FP algorithm and the reweighted version obtained using the combined penalty terms are shown in Tables 27–31 of [16]. In particular, both the results related to problems for which an integer feasible solution is found in all ten runs (see 27–29 of [16]) and the results related to problems for which an integer feasible solution is found in fewer than ten runs (see Tables 30–31 of [16]) are reported. The OFP fails to find a feasible solution in all ten runs for 46 instances, the Exp ORFP for 49, the Logis ORFP for 42.

The introduction of the objective function generally improves the quality of the feasible solution found, and in some cases we notice a relevant improvement in the percentage gap with respect to the best known solution. This improvement can sometimes correspond to an improvement in the computational time, too. We report in Table 8 the CPU time and the gap with respect to the optimal solution for some instances where the introduction of the objective function improves the quality of the solution.

Overall, the OFP and the Logis ORFP found the optimal solution for ten instances, while the Exp ORFP for twelve instances. Since in this case we are interested in finding the algorithm with best performance in terms of both CPU time and gap value, we compare Exp OFP and Logis OFP with OFP in terms of wins (minimum CPU time and minimum gap):

- *OFP vs. Exp ORFP:* The OFP has 39 wins against 46 wins of the Exp ORFP.
- *OFP vs. Logis ORFP:* Both the OFP and the Logis ORFP have 38 wins.

Let us now analyze the behavior of the various algorithms in terms of number of iterations and computational time. We report in Table 9 the geometric means for all the algorithms calculated over those problems for which a feasible solution is obtained in all ten runs. In the calculations of the geometric means individual values smaller than 1 are replaced by 1. The results in Table 9 indicate that both the Exp ORFP

TABLE 9
Comparison between OFP and ORFP (geometric means).

OFP		Exp ORFP		Logis ORFP	
Iter	Time	Iter	Time	Iter	Time
16.7396	1.8725	16.3445	1.8694	18.6573	1.8123

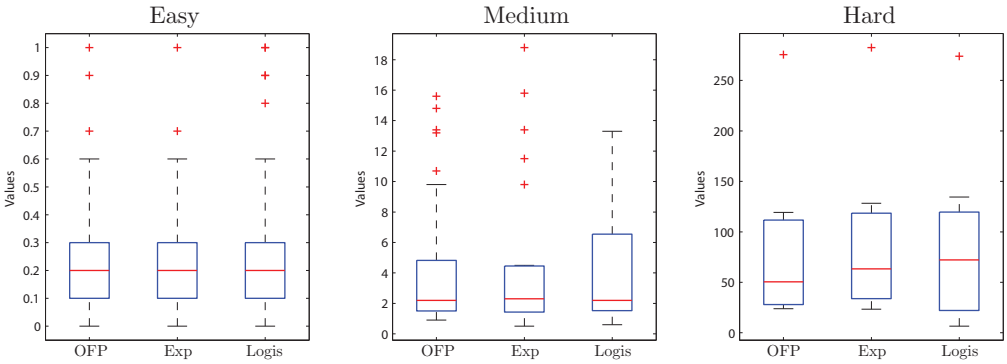

FIG. 9. *Box plots of the CPU time for OFP vs. ORFP.*

and the Logis ORFP have a geometric mean in terms of CPU time slightly lower than the geometric mean of the OFP, while the Logis ORFP has a geometric mean in terms of number of iterations higher than the other two.

In order to better assess the differences between the OFP and the ORFP, we considered problems for which an integer feasible solution is found in all ten runs by the algorithms in comparison. We divided the problems into three different classes depending on the CPU time t (seconds) needed by the algorithms to find a feasible solution:

- *Easy.* Problems for which a feasible solution has been found by all the algorithms in a time $t \leq 1$ (66 problems).
- *Hard.* Problems for which a feasible solution has been found by any algorithm in a time $t > 20$ (8 problems).
- *Medium.* All problems that are neither Easy nor Hard (27 problems).

We report in Figure 9 the results in terms of CPU time, obtained by the OFP, the Exp ORFP, and the Logis ORFP algorithms for the three classes of problems. We further report the CPU time and gap percentage for the instances in the Hard class in Table 10. We can notice that in the Easy class the three algorithms have the same behavior. In the medium class they are comparable, while in the Hard class the Logis ORFP has the highest median. However, looking at the results in Table 10, it can be seen that the Logis ORFP has the lowest CPU time on 5 instances over 8.

In order to analyze the behavior of the algorithms in terms of solution quality, we consider in Figure 10 the data profiles for the gap percentage obtained by the various algorithms for the various classes of problems. The plots in Figure 10 give on the y-axis the number of problems whose gap is smaller than or equal to the value given on the x-axis. We can notice that the profiles of the three algorithms are comparable. For the Easy class the Exp ORFP profiles are slightly better than the others, while for the Hard class the Logis ORFP is the best of the three. We further report in Table 11 some instances where the use of ORFP is beneficial in terms of gap. We want to point out that there are four instances where at least one version of the ORFP closes

TABLE 10
Detailed results for the Hard class for OFP vs. ORFP.

Problem	OFP		EXP ORFP		LOGIS ORFP	
	Time	Gap%	Time	Gap%	Time	Gap%
air04	23.8	4	23.3	4	23	4
dano3mip	275.6	-	282.6	-	273.9	-
dano3-3	53.2	0	79.3	0	104.8	0
dano3-4	119.2	0	108.5	0	96.4	0
dano3-5	104.1	0	128.3	0	134.5	0
neos12	28.1	37	31.8	36	6.5	0
neos476283	47.7	1	47.2	1	48	2
qap10	27.8	14	36	20	21.2	3

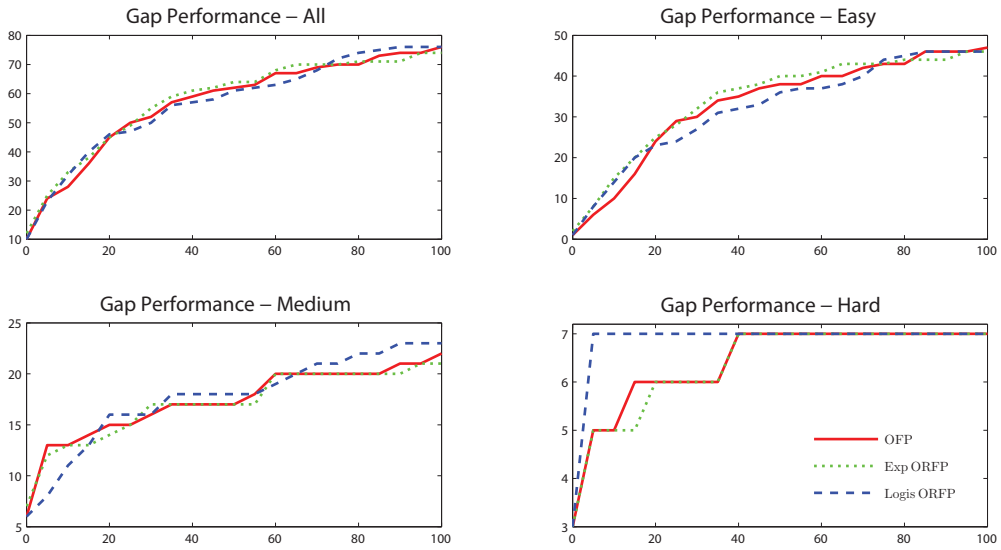

FIG. 10. *Profiles of the gap percentage for OFP vs. ORFP.*

TABLE 11
Examples of instances where ORFP improves the gap.

	OFP	Exp ORFP	Logis ORFP
opt1217	20	0	17
sp97ar	717	597	66
neos-12	37	36	0
neos-826812	1	1	0
neos-932816	1	0	1
neos-1200887	20	14	5
neos-1228986	18	7	4
qap10	14	20	3

the gap and OFP does not, while the opposite never happens.

As a concluding remark, we would like to point out the fact that Exp ORFP has a larger number of wins than ORFP and comparable performance in terms of gap, while Logis ORFP has the same number of wins and good performance in terms of gap (see results for the Hard class).

8.5. Comparison between OFP and combined ORFP. In the following we report a comparison between the OFP [1] and the combined ORFP where we consider

TABLE 12
Comparison between OFP and Exp+Logis ORFP (feasible solutions).

	Not found	Found at least once	Found 10 times	Average number of f.s. found
OFP	17	29	107	7.99
Exp+Logis ORFP	23	17	113	8.06

TABLE 13
Improvement in the quality of the solution by the introduction of the objective function in the FP and in the RFP combined.

	FP		OFP		Exp+Logis RFP		Exp+Logis ORFP	
	Gap %	Time	Gap %	Time	Gap %	Time	Gap %	Time
binkar10-1	8936	0.15	18	0.10	8930	0.06	18	0.10
dano3-3	73	31.74	0	53.20	13	8.67	0	24.60
dano3-4	73	23.95	0	119.20	15	8.65	0	34.80
dano3-5	72	26.46	0	104.10	16	8.62	0	55.70
neos-476283	160	444.74	1	47.70	33	11.13	3	34.40
neos-780889	216	48.19	0	13.40	193	83.28	0	13.40
qap10	48	1690.54	14	27.80	19	10.64	21	19.40

the combination of the Exp and Logis terms (Exp+Logis ORFP). In order to evaluate the ability of finding a first feasible solution, we report in Table 12, for each penalty term,

- the number of problems for which no feasible solution has been found (Not found);
- the number of problems for which a feasible solution has been found at least once, but fewer than ten times (Found at least once);
- the number of problems for which a feasible solution has been found for all ten runs (Found 10 times);
- the average number of feasible solutions found (Average number of f.s. found).

As we can see from Table 12 the Exp+Logis ORFP was not able to find a feasible solution in six instances more than the OFP. On the other hand it found a feasible solution for all ten runs in six instances more than the OFP. The average number of feasible solutions found is similar for the two algorithms.

The detailed results of the comparison between the FP algorithm and the reweighted version obtained using the combined penalty terms are shown in Tables 27–31 of [16]. In particular, both the results related to problems for which an integer feasible solution is found in all the ten runs (see Tables 27–29 of [16]) and the results related to problems for which an integer feasible solution is found in fewer than ten runs (see Tables 30–31 of [16]) are reported. The OFP fails to find a feasible solution in all ten runs for 46 instances and the Exp+Logis ORFP for 40 instances.

We report in Table 13 the CPU time and the gap with respect to the optimal solution for some instances where the introduction of the objective function improves the quality of the solution.

Overall, both the OFP and the Exp+Logis ORFP found the optimal solution for ten instances. Also in this case we compare ORFP and combined ORFP in terms of number of wins, and we have that the OFP has 28 wins, while the Exp+Logis ORFP has 46 wins.

Let us now analyze the behavior of the two algorithms in terms of number of iterations and computational time by computing the geometric means on those problems for which a feasible solution is calculated in all ten runs. In the calculations of the geometric means individual values smaller than 1 are replaced by 1. The results

TABLE 14
Comparison between OFP and Exp+Logis ORFP (geometric means).

OFP		Exp+Logis ORFP	
Iter	Time	Iter	Time
16.7396	1.8725	11.5181	1.7134

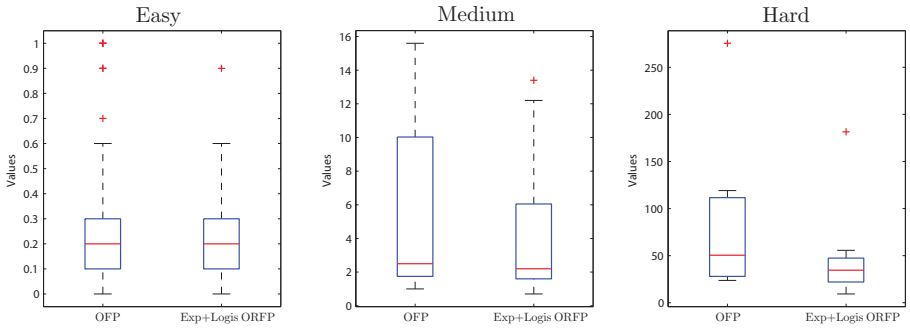

FIG. 11. *Box plots of the CPU time for OFP vs. ORFP combined.*

in Table 14 indicate that the Exp+Logis ORFP has a lower geometric mean both in terms of number of iterations and in terms of CPU time.

In order to better assess the differences between the OFP and the Exp+Logis ORFP, we considered the problems for which an integer feasible solution is found in all ten runs by the algorithms in comparison. We divided the problems into three different classes depending on the CPU time t (seconds) needed by the algorithms to find a feasible solution:

- *Easy.* Problems for which a feasible solution has been found by the two algorithms in a time $t \leq 1$ (69 problems).
- *Hard.* Problems for which a feasible solution has been found by any algorithm in a time $t > 20$ (8 problems).
- *Medium.* All problems that are neither Easy nor Hard (25 problems).

We report in Figure 11 the results in terms of CPU time, obtained by the OFP and the Exp+Logis ORFP on the three classes of problems. We further report the CPU time and gap percentage for the instances in the Hard class in Table 15. We can notice that in the Easy class the two algorithms have a very similar behavior, while both in the Medium and in the Hard classes the Exp+Logis ORFP improves the performance.

In order to analyze the behavior of the algorithms in terms of solution quality, we again consider in Figure 12 the data profiles for the gap percentage obtained by the two algorithms for the various classes of problems. Each plot gives the number of instances where a solution was obtained by a given algorithm within a certain gap percentage. We can notice that the two algorithms are comparable in all the classes. We further report in Table 16 some instances where the use of Exp+Logis ORFP is beneficial in terms of gap.

As a concluding remark, we would like to point out the fact that Exp+Logis ORFP has a quite larger number of wins than ORFP, better performance in terms of CPU time, and comparable performance in terms of gap.

8.6. Benchmarking algorithms via performance profiles. In order to give a better interpretation of the results generated by the various algorithms, we decided

TABLE 15
Detailed results for the Hard class for OFP vs. Exp+Logis ORFP.

Problem	OFP		Exp+Logis ORFP	
	Time	Gap%	Time	Gap%
air04	23.8	4	39.1	4
dano3mip	275.6	-	181.5	-
dano3-3	53.2	0	24.6	0
dano3-4	119.2	0	34.8	0
dano3-5	104.1	0	55.7	0
neos12	28.1	37	9.3	13
neos476283	47.7	1	34.4	3
qap10	27.8	14	19.4	21

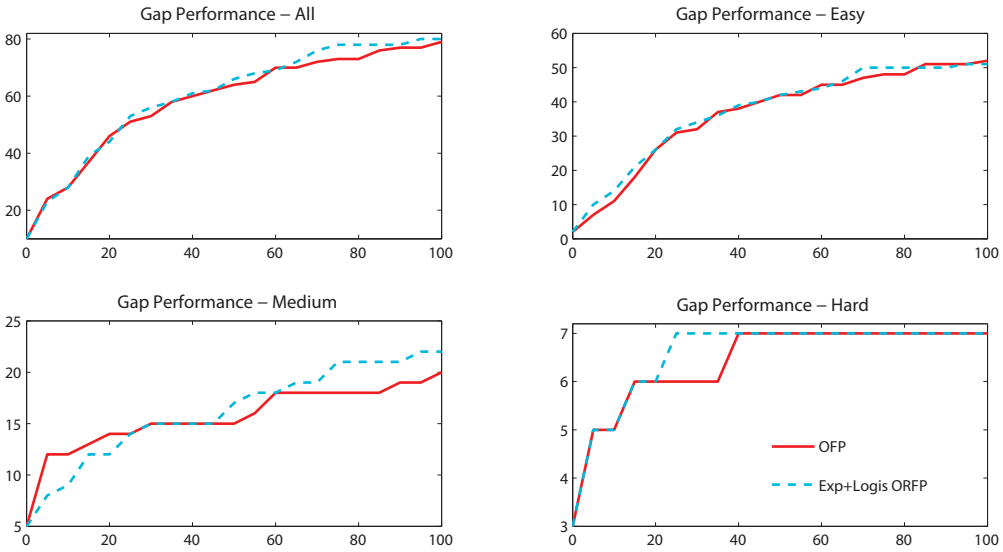

FIG. 12. *Profiles of the gap percentage for OFP vs. Exp+Logis ORFP.*

to use performance profiles [17]. We consider a set A of n_a algorithms, a set P of n_p problems, and a performance measure $m_{p,a}$ (e.g., in our case, average number of iterations, average CPU time). We compare the performance on problem p by algorithm a with the best performance by any algorithm on this problem using the following *performance ratio*:

$$r_{p,a} = \frac{m_{p,a}}{\min\{m_{p,a} \ : \ a \in A\}}.$$

Then we obtain an overall assessment of the performance of the algorithm by defining the value

$$\rho_a(\tau) = \frac{1}{n_p}\text{size}\{p \in P \ : \ r_{p,a} \leq \tau\},$$

which represents the probability for algorithm $a \in A$ that the performance ratio $r_{p,a}$ is within a factor $\tau \in R$ of the best possible ratio. The function ρ_a represents the distribution function for the performance ratio. Thus $\rho_a(1)$ gives the fraction of problems for which algorithm a was the most effective, $\rho_a(2)$ gives the fraction of

TABLE 16
Examples of instances where Exp+Logis ORFP improves the gap.

	OFP	Exp+Logis ORFP
bc1	304	3
neos-522351	28	4
neos-584851	56	19
neos-829552	1038	140

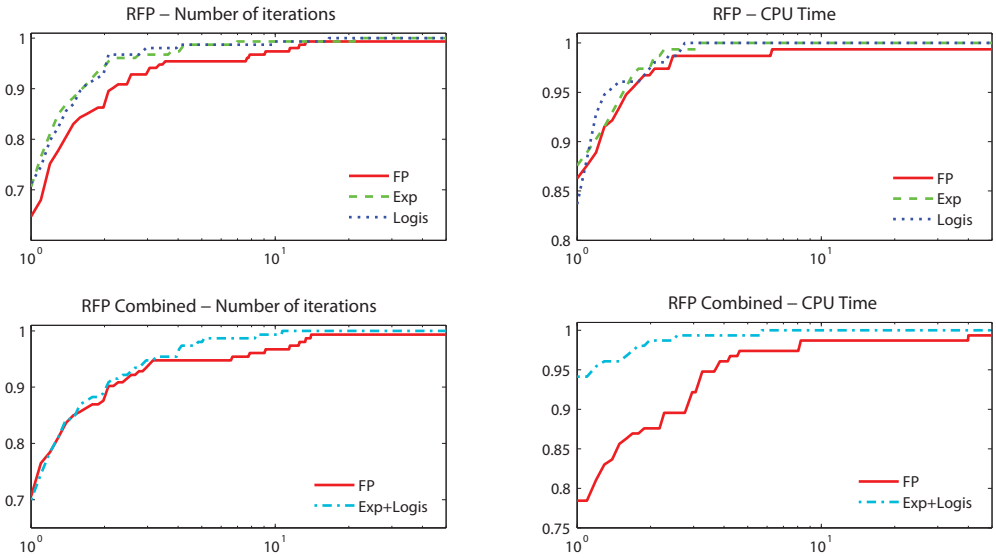

FIG. 13. *Performance profiles: FP vs. RFP (upper figures); FP vs. RFP combined (lower figures).*

problems for which algorithm a is within a factor of 2 of the best algorithm, and so on.

In Figure 13, we report the performance profiles related to the comparison among FP, Exp RFP, and Logis RFP in terms of number of iterations (upper left) and CPU time (upper right). It is clear that Exp RFP and Logis RFP functions have a higher number of wins in terms of number of iterations and Exp RFP has the highest number of wins in terms of computational time. Furthermore, the two RFP algorithms are better in terms of robustness.

We further report, in Figure 13, the performance profiles related to the comparison between FP and the combined version of the RFP obtained using Exp and Logis functions in terms of number of iterations (lower left) and CPU time (lower right). If we take a look at the profiles related to the iterations, we can notice that the FP is slightly better in the number of wins, but the combined RFP is better in terms of robustness. The performance profiles related to the CPU time clearly show that the combined RFP outperforms the FP in terms of both number of wins and robustness.

In Figure 14, we report the performance profiles related to the comparison among OFP, Exp ORFP, and Logis ORFP in terms of number of iterations (upper left) and CPU time (upper right). The performance profiles related to the number of iterations shows that the Logis ORFP profile is below the OFP and the Exp ORFP profiles, which are very similar. On the contrary, the Logis ORFP profile related to the CPU

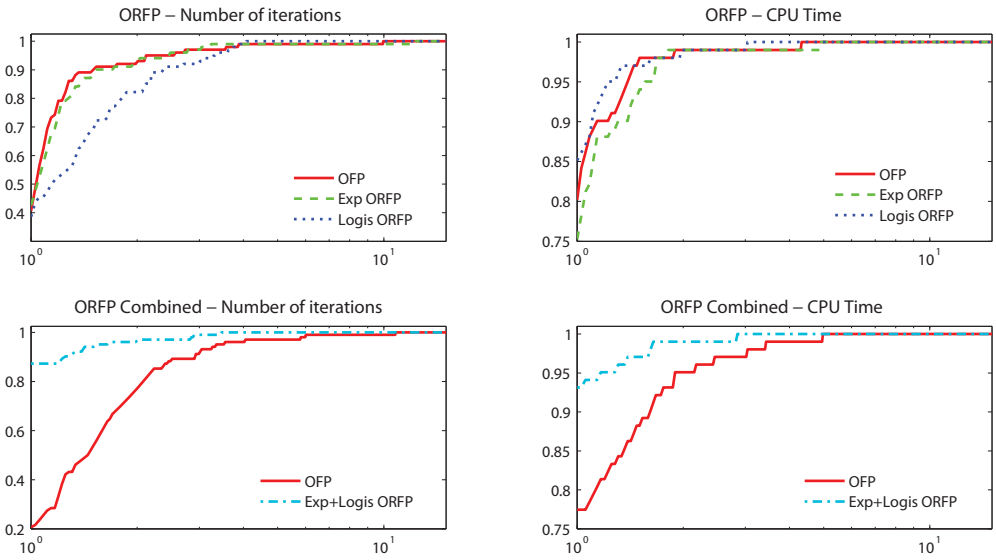

FIG. 14. *Performance profiles: OFP vs. ORFP (upper figures); OFP vs. ORFP Combined (lower figures).*

time has the highest number of wins and is slightly better than the other two in terms of robustness.

We further report, in Figure 14, the performance profiles related to the comparison between OFP and Exp+Logis ORFP in terms of number of iterations (lower left) and CPU time (lower right). We can notice that the Exp+Logis ORFP profile outperforms the OFP profile in terms of both number of iterations and CPU time.

9. Conclusions. In this paper, we focused on the problem of finding a first feasible solution for a 0-1 MIP problem. We started by interpreting the feasibility pump (FP) heuristic as a Frank–Wolfe method applied to a nonsmooth concave merit function. Then we noticed that the reduction of the merit function used in the FP scheme can correspond to an increase in the number of noninteger variables of the solution. For this reason, we proposed new concave merit functions that can be included in the FP scheme having two important properties: they decrease whenever the number of integer variables increases; if they decrease, then the number of noninteger variables does not increase. Due to these properties, the functions proposed should speed up the convergence towards integer feasible points. We reported computational results on a set of 153 0-1 MIP problems. This numerical experience shows that the version of the reweighted feasibility pump (RFP) obtained by combining two of the proposed functions (namely, Exp and Logis) compares favorably with the FP both in its original version and in the enhanced version with the introduction of the objective function [1]. Furthermore, it highlights that the use of more than one merit function at time (i.e., parallel framework, combination of functions) can significantly improve the efficiency of the algorithm.

In [15], we reinterpret the FP for general MIP problems as a Frank–Wolfe method applied to a suitably chosen function, and we extend our approach to this class of problems. Possible improvements of our approach could be accomplished along different lines, for example by replacing the rounding with a scheme based on constraint prop-

agation like in [20] or with a procedure that examines rounded solutions along a given line segment as in [3, 9]. In particular the proposed merit functions could also be used in order to drive the choice of the new rounded point.

Finally, we want to remark that a wider availability of functions for measuring integrality is important since it can ease the search of feasible solutions for different classes of MIP problems.

Acknowledgments. The authors would like to thank the Associate Editor and the anonymous reviewers for their kind and constructive comments, which helped to improve this paper. The authors are also grateful to Giovanni Rinaldi for his useful suggestions.

REFERENCES

[1] T. Achterberg and T. Berthold, *Improving the feasibility pump*, Discrete Optim., 4 (2007), pp. 77–86.
[2] T. Achterberg, T. Koch, and A. Martin, *MIPLIB* 2003, Oper. Res. Lett., 34 (2006), pp. 361–372. Problems available at http://miplib.zib.de.
[3] D. Baena and J. Castro, *Using the analytic center in the feasibility pump*, Oper. Res. Lett., 39 (2011), pp. 310–317.
[4] E. Balas, S. Ceria, M. Dawande, F. Margot, and G. Pataki, *OCTANE: A new heuristic for pure* 0-1 *programs*, Oper. Res., 49 (2001), pp. 207–225.
[5] E. Balas and C. H. Martin, *Pivot-and-complement: A heuristic for* 0-1 *programming*, Management Sci., 26 (1980), pp. 86–96.
[6] E. Balas, S. Schmieta, and C. Wallace, *Pivot and shift—a mixed integer programming heuristic*, Discrete Optim., 1 (2004), pp. 3–12.
[7] J. Baxter, *Local optima avoidance in depot location*, J. Oper. Res. Soc., 32 (1981), pp. 815–819.
[8] L. Bertacco, M. Fischetti, and A. Lodi, *A feasibility pump heuristic for general mixed-integer problems*, Discrete Optim., 4 (2007), pp. 63–76.
[9] N. L. Boland, A. C. Eberhard, F. G. Engineer, M. Fischetti, M. W. P. Savelsbergh, and A. Tsoukalas, *Boosting the Feasibility Pump*, Report C-OPT 2012-03, The University of Newcastle, Callaghan, NSW, Australia, 2012.
[10] J. Eckstein and M. Nediak, *Pivot, cut, and dive: A heuristic for* 0-1 *mixed integer programming*, J. Heuristics, 13 (2007), pp. 471–503.
[11] P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuejols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Waechter, *An algorithmic framework for convex mixed integer nonlinear programs*, Discrete Optim., 5 (2008), pp. 186–204.
[12] P. Bonami, G. Cornuejols, A. Lodi, and F. Margot, *A feasibility pump for mixed integer nonlinear programs*, Math. Program., 119 (2009), pp. 331–352.
[13] C. D'Ambrosio, A. Frangioni, L. Liberti, and A. Lodi, *A storm of feasibility pumps for nonconvex MINLP*, Math. Program., 136 (2012), pp. 375–402.
[14] E. Danna, E. Rothberg, and C. Le Pape, *Exploring relation induced neighborhoods to improve MIP solution*, Math. Program. 102 (2005), pp. 71–90.
[15] M. De Santis, S. Lucidi, and F. Rinaldi, *Feasibility Pump-Like Heuristics for Mixed Integer Problems*, DIS Technical Report 15, Università di Roma, Rome, Italy, 2010.
[16] M. De Santis, S. Lucidi, and F. Rinaldi, *A New Class of Functions for Measuring Solution Integrality in the Feasibility Pump Approach: Complete Results*, DIS Technical Report 2, Università di Roma, Rome, Italy, 2013.
[17] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profile*, Math. Program. 91 (2002), pp. 201–213.
[18] M. Fischetti, F. Glover, and A. Lodi, *The feasibility pump*, Math. Program., 104 (2005), pp 91–104.
[19] M. Fischetti and A. Lodi, *Local branching*, Math. Program., 98 (2003), pp. 23–47.
[20] M. Fischetti and D. Salvagnin, *Feasibility pump* 2.0, Math. Program. Comput., 1 (2009), pp. 201–222.
[21] F. Glover and M. Laguna, *General purpose heuristics for integer programming, part* I, J. Heuristics, 2 (1997), pp. 343–358.
[22] F. Glover and M. Laguna, *General purpose heuristics for integer programming, part* II, J. Heuristics, 3 (1997), pp. 161–179.

[23] F. Glover and M. Laguna, *Tabu Search*, Kluwer Academic, Boston, Dordrecht, London, 1997.

[24] F. Glover, A. Løkketangen, and D. L. Woodruff, *Scatter search to generate diverse MIP solutions*, in OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, M. Laguna and J. Gonzàlez-Velarde, eds., Kluwer Academic, Boston, Dordrecht, London, 2000, pp. 299–317.

[25] F. S. Hillier, *Efficient heuristic procedures for integer linear programming with an interior*, Oper. Res., 17 (1969), pp. 600–637.

[26] F. S. Hillier and R. M. Saltzman, *A heuristic ceiling point algorithm for general integer linear programming*, Management Sci., 38 (1992), pp. 263–283.

[27] ILOG, *Cplex*, http://www.ilog.com/products/cplex.

[28] R. H. Leary, *Global optimization on funneling landscapes*, J. Global Optim., 18 (2000), pp. 367–383.

[29] A. Løkketangen and F. Glover, *Solving zero/one mixed integer programming problems using tabu search*, European J. Oper. Res., 106 (1998), pp. 624–658.

[30] H. R. Lourenço, O. C. Martin, and T. Stülze, *Iterated Local Search*, in Handbook of Metaheuristics, F. W. Glover and G. A. Kochenberger, eds., Kluwer Academic, Boston, Dordrecht, London, 2003, pp. 321–353.

[31] S. Lucidi and F. Rinaldi, *Exact penalty functions for nonlinear integer programming problems*, J. Optim. Theory Appl., 145 (2010), pp. 479–488.

[32] O. L. Mangasarian, *Solutions of general linear complementarity problems via nondifferentiable concave minimization*, Acta Math. Vietnam, 22 (1997), pp. 199–205.

[33] W. Murray and K. M. Ng, *An algorithm for nonlinear optimization problems with binary variables*, Comput. Optim. Appl., 47 (2010), pp. 257–288.

[34] F. Rinaldi, *New results on the equivalence between zero-one programming and continuous concave programming*, Optim. Lett., 3 (2009), pp. 377–386.

[35] F. Rinaldi, F. Schoen, and M. Sciandrone, *Concave programming for minimizing the zero-norm over polyhedral sets*, Comput. Optim. Appl., 46 (2010), pp. 467–486.