A new class of functions for measuring solution integrality in the Feasibility Pump approach: Detailed Results

M. De Santis[†], S. Lucidi[†], F. Rinaldi[*]

[†]Dipartimento di Informatica e Sistemistica
Sapienza Università di Roma
Via Ariosto, 25 - 00185 Roma - Italy

[*]Università di Padova
Dipartimento di Matematica
Via Trieste, 63 35121 Padua - Italy

e-mail (De Santis): mdesantis@dis.uniroma1.it
e-mail (Lucidi): stefano.lucidi@dis.uniroma1.it
e-mail (Rinaldi): rinaldi@math.unipd.it

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 MIP solvers. The Feasibility pump is a heuristic for finding feasible solutions to mixed integer linear 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.

Keywords. Mixed integer programming, Concave penalty functions, Frank-Wolfe algorithm, Feasibility Problem.

MSC. 90C06, 90C10, 90C11, 90C30, 90C59

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 [18], 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 a MIP problem have been proposed (see e.g. [4]-[6], [10], [20]-[23], [25], [28]). Recently, Fischetti, Glover and Lodi [17] proposed a new heuristic, the well-known Feasibility Pump (FP), that 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 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. Fischetti, Bertacco 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 [19], 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], J.Eckstein and M.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 Feasibility Pump 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 Feasibility Pump can be seen as a form of Iterated Local Search or Basin Hopping algorithm (see e.g. [7, 27, 29]).
In this paper, we analyze in deep the relationship between the Feasibility Pump 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 [17]. 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 Section 2 and 3, we give a brief review of the Feasibility Pump 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 somehow more than one merit function at 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 a MIP problem of the form:

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

where $A \in \mathbf{R}^{m \times n}$ and $I \subset \{1, 2, \dots, n\}$ is the set of indices of zero-one variables. Let $P = \{x : Ax \ge b,\ 0 \le x_j \le 1,\ \forall\, j \in I\}$ denote the polyhedron of the LP-relaxation of (MIP). The Feasibility Pump 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 integer or, in case of failing, if it reaches a time or iteration limit. In order to avoid stalling issues and loops, the Feasibility Pump 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], [17]):

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 $\bar{x}_j^k$ with $j \in I$ to the nearest integer, while leaving the continuous components of the solution unchanged. Formally,

$$\tilde{x}_j^k = \begin{cases} [\bar{x}_j^k] & \text{if } j \in I \\ \\ \bar{x}_j^k & \text{otherwise} \end{cases} \tag{1}$$

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. To be more specific:

- if $\tilde{x}_j^k = \tilde{x}_j^{k+1}$ 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 perturbation 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 [17]).*

3 The Objective Feasibility Pump

When using a heuristic like the Feasibility Pump on a MIP problem, one of the target 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 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 $\nu < 1$ (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 J.Eckstein and M.Nediak [10] noticed that the feasibility pump 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 non-differentiable functions. Let us consider the problem

$$\begin{array}{l} \min f(x) \\ x \in P \end{array} \tag{2}$$

where $P \subset R^n$ is a non empty polyhedral set that does not contain lines going to infinity in both directions, $f : R^n \to R$ is a concave, non-differentiable 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 linear programming problems, and the following result, proved in [31], 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:

$$(g^\star)^T (x - x^\star) \geq 0, \ \ \forall x \in P \tag{3}$$

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* (2) *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

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

At each iteration, the Feasibility Pump 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

$$\begin{aligned} \bar{x}^{k+1} \in \arg\min\ & \Delta(x, \tilde{x}^k) \\ \text{s.t.}\ & Ax \geq b \\ & 0 \leq x_j \leq 1\ \forall j \in I. \end{aligned} \tag{4}$$

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

$$\begin{aligned} \min\ & \sum_{j \in I: \bar{x}_j^k < \frac{1}{2}} x_j - \sum_{j \in I: \bar{x}_j^k \geq \frac{1}{2}} x_j \\ \text{s.t.}\ & Ax \geq b \\ & 0 \leq x_j \leq 1\ \forall j \in I. \end{aligned} \tag{5}$$

Since the function

$$v(t) = \begin{cases} 1 & \text{if } t < \frac{1}{2} \\ -1 & \text{if } t \geq \frac{1}{2} \end{cases} \tag{6}$$

is such that $v(t) \in \partial \min\{t, 1-t\}$, Problem (5) can be seen as a generic iteration of the Frank Wolfe method with unitary stepsize applied to the following minimization problem

$$\begin{aligned} \min\ & \sum_{i \in I} \min\{x_i, 1 - x_i\} \\ \text{s.t.}\ & Ax \geq b \\ & 0 \leq x_i \leq 1\ \forall i \in I. \end{aligned} \tag{7}$$

5 New nonsmooth merit functions for the FP approach

As we have seen in the previous section, the basic Feasibility Pump is equivalent to minimizing a separable nonsmooth function which penalizes the 0-1 infeasibility, namely

$$f(x) = \sum_{i \in I} \min\{x_i, 1 - x_i\}. \tag{8}$$

When using the Frank Wolfe unitary stepsize algorithm for solving Problem (7), 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

$$f(x^{k+1}) \leq f(x^k) + (g^k)^T (x^{k+1} - x^k) < f(x^k), \tag{9}$$

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*

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

Let f be the function defined in (8). 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:

$$\psi(x) = card\{\, x_i : \, i \in I, \; x_i \notin \{0,1\} \,\}. \tag{10}$$

The function (10) can be rewritten as:

$$\psi(x) = \sum_{i \in I} s(\min\{x_i, 1 - x_i\}) \tag{11}$$

where $s : \mathbf{R} \rightarrow \mathbf{R}^+$ 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 (11) over a polyhedral set is a very hard problem. In the following we prove a general result to define approximations of function (11) 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

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

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

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

$$\phi(x) = \sum_{i \in I} \varphi(x_i). \tag{12}$$

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

1)

$$\varphi(0) = \varphi(1); \tag{13}$$

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

$$\varphi(\bar{\alpha}) - \varphi(\tilde{\alpha}) \leq -M; \tag{14}$$

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

$$|\varphi(\bar{\alpha}) - \varphi(\tilde{\alpha})| \leq \frac{M}{n}. \tag{15}$$

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 non-integer components of x and y:

$$U = \{i \in \{1, \dots, n\} \mid i \in I,\ x_i \notin \{0,1\}\},$$
$$W = \{j \in \{1, \dots, n\} \mid j \in I,\ y_j \notin \{0,1\}\}.$$

Then we can write

$$\begin{aligned}\phi(x) - \phi(y) &= \sum_{i \in I} \varphi(x_i) - \sum_{j \in I} \varphi(y_j) = \\ &= \sum_{i \in U} \varphi(x_i) + \sum_{i \in I \setminus U} \varphi(x_i) - \sum_{j \in W} \varphi(y_j) - \sum_{j \in I \setminus W} \varphi(y_j).\end{aligned} \tag{16}$$

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 $\bar{\jmath}$ such that

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

Then we can write

$$\begin{aligned}\phi(x) - \phi(y) &= \varphi(x_{\bar{\jmath}}) - \varphi(y_{\bar{\jmath}}) + \sum_{j \in U} \varphi(x_j) + \sum_{\substack{j \in I \setminus U \\ j \neq \bar{\jmath}}} \varphi(x_j) - \sum_{\substack{j \in W \\ j \neq \bar{\jmath}}} \varphi(y_j) - \sum_{j \in I \setminus W} \varphi(y_j) = \\ &= \varphi(x_{\bar{\jmath}}) - \varphi(y_{\bar{\jmath}}) + \sum_{\substack{j \in W \\ j \neq \bar{\jmath}}} \varphi(x_j) + \sum_{\substack{j \in I \setminus U \\ j \neq \bar{\jmath}}} \varphi(x_j) - \sum_{\substack{j \in W \\ j \neq \bar{\jmath}}} \varphi(y_j) - \sum_{\substack{j \in I \setminus U \\ j \neq \bar{\jmath}}} \varphi(y_j) = \\ &= \varphi(x_{\bar{\jmath}}) - \varphi(y_{\bar{\jmath}}) + \sum_{\substack{j \in I \setminus U \\ j \neq \bar{\jmath}}} (\varphi(x_j) - \varphi(y_j)) + \sum_{\substack{j \in W \\ j \neq \bar{\jmath}}} (\varphi(x_j) - \varphi(y_j)) \leq \\ &\leq \varphi(x_{\bar{\jmath}}) - \varphi(y_{\bar{\jmath}}) + \sum_{\substack{j \in I \setminus U \\ j \neq \bar{\jmath}}} (\varphi(x_j) - \varphi(y_j)) + \sum_{\substack{j \in W \\ j \neq \bar{\jmath}}} |\varphi(x_j) - \varphi(y_j)|\end{aligned} \tag{17}$$

By using (13) we obtain

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

Now we notice that $x_{\bar{\jmath}} \in \{0,1\}$, $y_{\bar{\jmath}} \in [\alpha_l, \alpha_u]$ and $x_j, y_j \in [\alpha_l, \alpha_u]$ for all $j \in W \setminus \{\bar{\jmath}\}$. Then, by using (14) and (15), we have

$$\phi(x) - \phi(y) \le \varphi(x_{\bar{\jmath}}) - \varphi(y_{\bar{\jmath}}) + \sum_{\substack{j\in W\\ j\neq\bar{\jmath}}} |\varphi(x_j) - \varphi(y_j)| \le -M + (|I|-1)\frac{M}{n} < 0. \tag{19}$$

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

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

By using the same arguments used before we obtain

$$\phi(x) - \phi(y) \le \sum_{j\in\bar{J}} (\varphi(x_j) - \varphi(y_j)) + \sum_{j\in W\setminus\bar{J}} |\varphi(x_j) - \varphi(y_j)|. \tag{20}$$

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 (14) and (15), we have

$$\begin{aligned}\phi(x) - \phi(y) &\le \sum_{j\in\bar{J}} (\varphi(x_j) - \varphi(y_j)) + \sum_{j\in W\setminus\bar{J}} |\varphi(x_j) - \varphi(y_j)| \le \\ &\le -M|\bar{J}| + (|I| - |\bar{J}|)\frac{M}{n} < 0. \end{aligned}\tag{21}$$

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

$$\psi(x) > \psi(y). \tag{22}$$

By (22), 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 (7), the following problem

$$\begin{aligned} \min\ & \phi(x) = \sum_{i \in I} \varphi(x_i) \\ \text{s.t.}\ & Ax \geq b \\ & 0 \leq x_i \leq 1\ \forall i \in I. \end{aligned} \tag{23}$$

As the method we use for solving the minimization problem stated above is the Frank-Wolfe algorithm, which at each step moves from a 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 [30, 34], we consider the following $\varphi(\cdot)$ terms to be used in the objective function of problem (23):

Logarithmic function

$$\varphi(t) = \min\big\{ \ln(t+\varepsilon), \ln[(1-t)+\varepsilon]\big\} \tag{24}$$

Hyperbolic function

$$\varphi(t) = \min\big\{ -(t+\varepsilon)^{-p}, -[(1-t)+\varepsilon]^{-p}\big\} \tag{25}$$

Exponential function

$$\varphi(t) = \min\big\{1 - \exp(-\alpha t), 1 - \exp(-\alpha(1-t))\big\} \tag{26}$$

Logistic function

$$\varphi(t) = \min\big\{[1+\exp(-\alpha t)]^{-1}, [1+\exp(-\alpha(1-t))]^{-1}\big\} \tag{27}$$

where $\varepsilon, \alpha, p > 0$. In Fig. 1, we compare the φ term related to the FP heuristic with those given by (24)-(27).

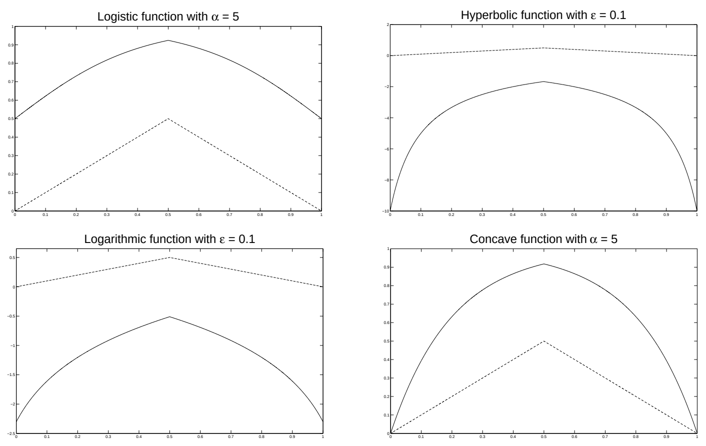

Figure 1: Comparison between the original FP term (dashed line) and the new terms (solid line).

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

Proposition 3 *For the term (24), 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

$$\alpha_l = \min\{\alpha_l, 1 - \alpha_u\} \tag{28}$$

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

$$\ln \varepsilon - \ln(\alpha_l + \varepsilon) + n(\ln(1/2 + \varepsilon) - \ln(\alpha_l + \varepsilon)) \leq 0. \tag{29}$$

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

$$M = \varphi(\alpha_l) - \varphi(0) = \ln(\alpha_l + \varepsilon) - \ln \varepsilon, \tag{30}$$

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 for any choice of $\bar{\alpha}, \tilde{\alpha} \in [\alpha_l, \alpha_u]$:

$$|\varphi(\bar{\alpha}) - \varphi(\tilde{\alpha})| \leq \varphi(1/2) - \varphi(\alpha_l). \tag{31}$$

Since $ii)$ in Assumption 2) needs to be verified for any choice of $\bar{\alpha}, \tilde{\alpha} \in [\alpha_l, \alpha_u]$, by (31) it is sufficient to show that

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

By using (30) and (29), we can easily verify that for any $\varepsilon \in (0, \bar{\varepsilon}]$, the following inequality holds:

$$\varphi(0) - \varphi(\alpha_l) + n(\varphi(1/2) - \varphi(\alpha_l)) = \tag{32}$$

$$= \ln \varepsilon - \ln(\alpha_l + \varepsilon) + n(\ln(1/2 + \varepsilon) - \ln(\alpha_l + \varepsilon)) \leq 0.$$

Then (ii) in Assumption 2) of Proposition 2 is satisfied. □

The result proved in Proposition 3 for the term (24) can also be proved for the terms (25)-(27) repeating the same arguments, thus all the merit functions (24)-(27) are suitable to penalize the number of variables that violate the integrality constraints.

We remark that functions (24)-(27) have also another interesting theoretical property: they can be used in an exact penalty approach like that proposed in [30]. In fact, it is possible to prove that terms (24)-(27) can be used to transform a 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 (24)-(27) a value $\bar{\varepsilon} > 0$ exists such that, for any $\varepsilon \in]0, \bar{\varepsilon}]$, problem

$$\min f(x), \quad s.t. \quad x \in P, \ \ x_i \in \{0,1\}, \ \ \forall i \in I \tag{33}$$

and problem

$$\min f(x) + \tilde{\phi}(x, \varepsilon), \quad s.t. \quad x \in P, \ \ 0 \le x_i \le 1, \ \ \forall i \in I \tag{34}$$

where

$$\tilde{\phi}(x, \varepsilon) = \begin{cases} \phi(x) & \text{if } \varphi \text{ is given by (24)-(25)} \\ \\ \dfrac{1}{\varepsilon}\,\phi(x) & \text{if } \varphi \text{ is given by (26)-(27)} \end{cases}$$

have the same minimum points.

Proof. the proof follows the same arguments as in [30]. See Appendix A for further details. □

This result suggests that these new merit functions can be used to define new Feasibility Pump 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 (34) with the penalty term (8). Furthermore, the restarting rules used in the Feasibility Pump 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 a future work.
In the next Section we will focus on finding a first feasible solution to a MIP problem. In particular, we tackle problem (23) by a modified Feasibility Pump approach based on the concave functions described above.

6 A reweighted version of the Feasibility Pump heuristic

The use of the merit functions (24)-(27) 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

$$\Delta_W(x, \tilde{x}) = \sum_{j \in I} w_j |x_j - \tilde{x}_j| = \|W(x - \tilde{x})\|_1, \tag{35}$$

where

$$W = diag(w_1, \ldots, w_n)$$

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, $j = 1, \dots, n$ is the following:

$$w_j = |g_j|,\; j = 1, \dots, n,$$

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 [19]. Other possibilities can be inspired by the procedures recently proposed in [3, 9] examining rounded solutions along suitable line segments.
Following the same reasoning 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

$$\begin{aligned} \bar{x}^{k+1} \in \arg\min\ & \Delta_{W^k}(x, \tilde{x}^k) \\ \text{s.t. }\ & Ax \geq b \\ & 0 \leq x_j \leq 1\ \forall j \in I. \end{aligned} \tag{36}$$

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

$$\begin{aligned} \min\ & \sum_{j \in I : \bar{x}_j^k < \frac{1}{2}} w_j^k x_j - \sum_{j \in I : \bar{x}_j^k \geq \frac{1}{2}} w_j^k x_j \\ \text{s.t. }\ & Ax \geq b \\ & 0 \leq x_j \leq 1\ \forall j \in I. \end{aligned} \tag{37}$$

By setting

$$w_j^k = |g_j^k|$$

with $g^k \in \partial\phi(\bar{x}^k)$, Problem (37), as we have already said, can be seen as the iteration of the Frank Wolfe method with unitary stepsize applied to the minimization problem (23).

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 MILP problem:*

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & x \in P \\ & x \in \{0,1\}^3 \end{aligned} \tag{38}$$

where $P \subset [0,1]^3$ is the polyhedron in Fig. 2. Let $x^L = \left(\frac{9}{20}, \frac{1}{8}, \frac{1}{8}\right)$ be the solution of the linear relaxation of (38) and $x^I = (0,0,0)$ be its rounding. The minimization of $\Delta(x, x^I) = \|x - x^I\|_1$

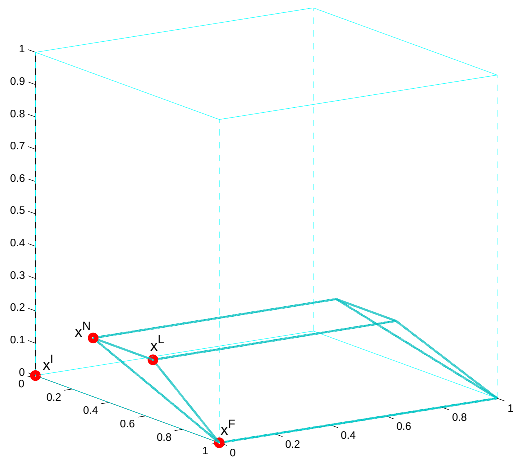

Figure 2: Feasible set of Problem 38.

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 Feasibility Pump Algorithm is a special case of the Reweighted Feasibility Pump obtained by setting $W^k = I$.

We can further use the merit functions (24)-(27) 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:

$$\Delta_{W,\theta}(x, \tilde{x}) = \frac{1-\theta}{\|\Delta\|} \Delta_W(x, \tilde{x}) + \frac{\theta}{\|c\|} c^T x, \tag{39}$$

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) any time 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:

$$\phi(x) = \lambda\phi_1(x) + (1-\lambda)\phi_2(x) \tag{40}$$

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

1) a matrix W^k with the following terms:

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

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 behaviour of a function obtained by combining the exponential and the 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 [17] and the Objective Feasibility Pump described in [1]. The test set used in our numerical experience consists of 153 instances of 0-1 problems from MIPLIB2003 [2] and COR@L libraries. All the algorithms were implemented in C and we have used ILOG Cplex [26] as solver of the linear programming 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 (8), (24)-(27), we solved the corresponding penalty formulation (23) 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.

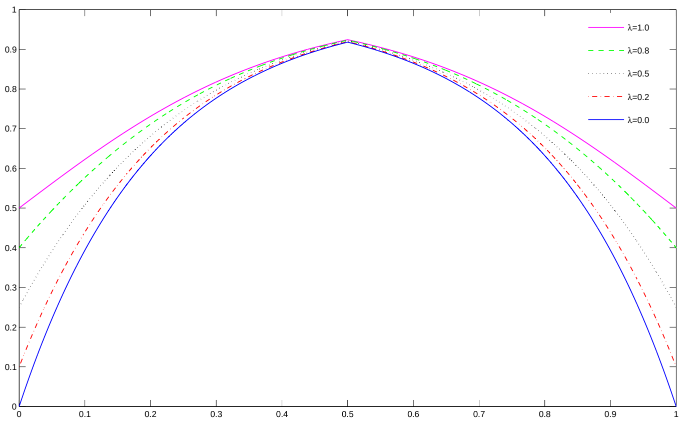

Figure 3: Behaviour of the function obtained combining exponential and logistic function

2 **FP vs RFP**: in order to evaluate the effectiveness of the new penalty functions, we compared the Feasibility Pump algorithm with the Reweighted Feasibility Pump, where the distance $\Delta_W(x, \tilde{x})$ is defined using the terms (24)-(27).

3 **FP vs Combined RFP**: we made a comparison between the Feasibility Pump algorithm and the Reweighted Feasibility Pump 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 Objective Feasibility Pump and the Objective Reweighted Feasibility Pump. 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 Objective Feasibility Pump and the Combined Objective Reweighted Feasibility Pump. 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 Objective Feasibility Pump.

The choice of the merit function parameters is critical for the efficiency of the algorithm. From 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 be solved. We performed our experiments using:

- Penalty term (8) denoted by **FP**;
- Penalty term (24) denoted by **Log**, with $\varepsilon = 0.1$;
- Penalty term (25) denoted by **Hyp**, with $\varepsilon = 0.1$;
- Penalty term (26) denoted by **Exp**, with $\alpha = 0.5$;
- Penalty term (27) denoted by **Logis**, with $\alpha = 0.1$.

On the basis of our numerical experience, 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).

8.1 Computational results for randomly generated starting points

In this first experiment, we applied the Frank-Wolfe algorithm to solve problem (23) with the objective functions (8), (24)-(27). 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 is each function in 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 the 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.

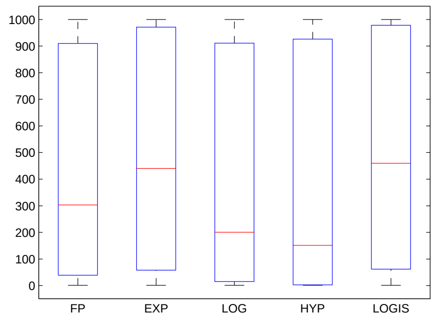

Figure 4: Comparison between the original FP term and the new terms for randomly generated starting points.

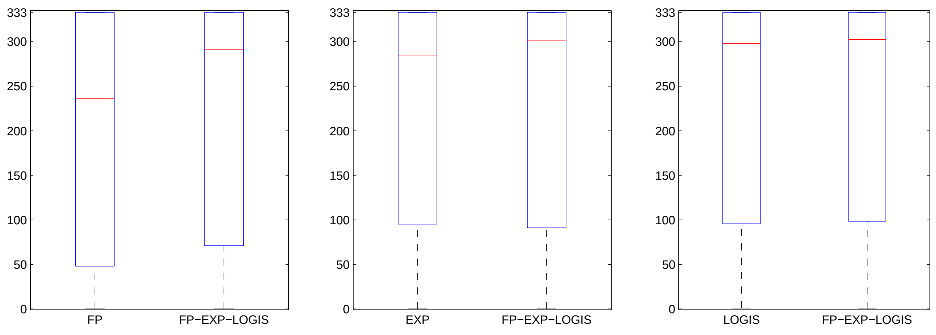

Figure 5: Number of integer feasible solutions found in the parallel experiment.

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 we 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 one with a different starting point.

We discarded the 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 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, into a multistart strategy, the use of different merit functions can help diversifying 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 less than ten times (Found at least once);
- The number of problems for which a feasible solution has been found for all the 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 they 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 the ten runs by all the algorithms and, in Table 2, we report for each penalty term:

- Number of problems for which the best average o.f. value (average over ten runs) is obtained (Best Average o.f.);
- Number of problems for which the best o.f. 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.

	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 1: Comparison between FP and RFP (Feasible solutions).

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

Table 2: Comparison between FP and RFP (Objective function value)

The detailed results of the comparison between the Feasibility Pump algorithm and the reweighted version obtained using the penalty terms (24)-(27) are shown in Tables 17 - 21. The results related to the problems for which an integer feasible solution is found in all the ten runs are reported in Tables 17 - 19. The results related to the problems for which an integer feasible solution is found in less than ten runs are reported in Tables 20 - 21. By taking a look at the 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 FP algorithm.

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

Table 3: Comparison between FP and RFP (Geometric Means)

In order to better assess the differences between the FP algorithm and the Reweighted FP algorithm, we considered the 123 problems for which an integer feasible solution is found in all the ten runs by 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 different algorithms, we ran three algorithms and we chose the solution with the lowest CPU time among the three. We report in Figure 7 the results obtained using:

- 3 runs of the FP algorithm;

- one different algorithm (FP, Exp RFP and Logis RFP) for each run.

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

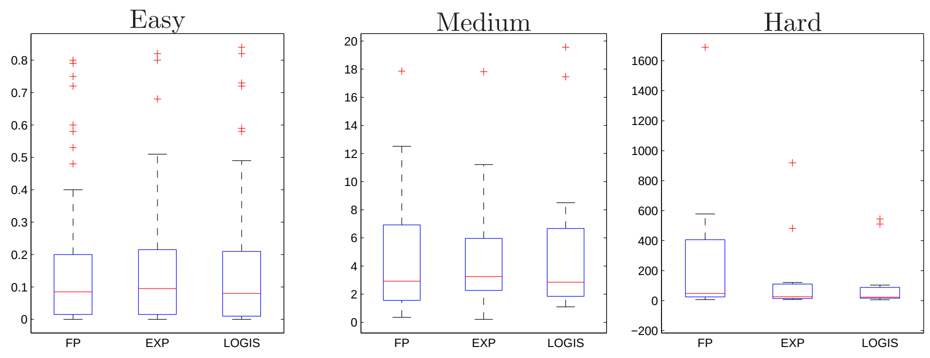

Figure 6: Results in terms of CPU time for the three classes of problems.

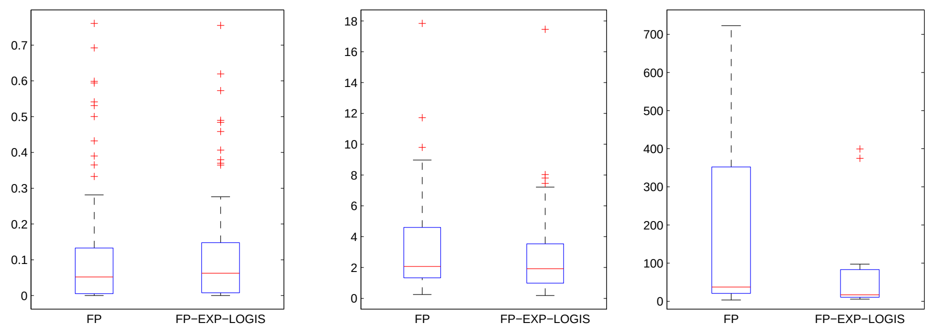

Figure 7: Results in terms of CPU time for the parallel experiment.

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 (40)). We start with $\lambda^0 = 1$ and we 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, so moving towards the function belonging to the second class.
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 less than ten times (Found at least once);
- The number of problems for which a feasible solution has been found for all the 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 the ten runs by all the algorithms and, in Table 5, we report for each penalty term:

- Number of problems for which the best o.f. value (average over ten runs) is obtained (Best Average o.f.);
- Number of problems for which the best o.f. 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, Exp+Log combination gives the best performance.

The detailed results of the comparison between the Feasibility Pump algorithm and the reweighted version obtained using the combined penalty terms are shown in Tables 22 - 26. The results related to the problems for which an integer feasible solution is found in all the ten runs are reported in Tables 22 - 24. The results related to the problems for which an integer feasible

	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 4: Comparison between FP and Combined RFP (Feasible solutions)

	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

Table 5: Comparison between FP and Combined RFP (Objective function value)

solution is found in less than ten runs are reported in Tables 25 - 26. By taking a look at the 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 the ten runs). In the calculations of the geometric means individual values smaller than 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.

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

Table 6: Comparison between FP and Combined RFP (Geometric Means)

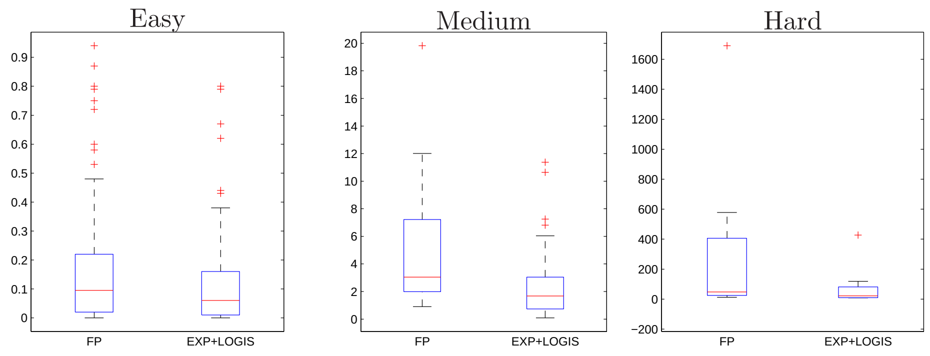

Figure 8: Results in terms of CPU time for the three classes of problems.

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 the 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 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 the 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 Objective Feasibility Pump (OFP) [1] and the Objective Reweighted Feasibility Pump 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 less than ten times (Found at least once);
- The number of problems for which a feasible solution has been found for all the ten runs (Found 10 times);
- The average number of feasible solutions found (Average number of f.s. found).

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 the ten runs.

	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 7: Comparison between OFP and ORFP (Feasible solutions)

The detailed results of the comparison between the Objective Feasibility Pump algorithm and the Objective Reweighted Feasibility Pump are shown in Tables 27 - 31. The results related to the problems for which an integer feasible solution is found in all the ten runs are reported in Tables 27 - 29. The results related to the problems for which an integer feasible solution is found in less than ten runs are reported in Tables 30 - 31. The OFP fails to find a feasible solution in all the 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

	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

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

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 10 instances, while the Exp ORFP for 12 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 a number of 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 calculated 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 9 indicate that both the Exp 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.

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

Table 9: Comparison between OFP and ORFP (Geometric Means)

In order to better assess the differences between the OFP and the ORFP, we considered the problems for which an integer feasible solution is found in all the 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 the 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 on the Easy Class the three algorithms have the same behavior. On 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.

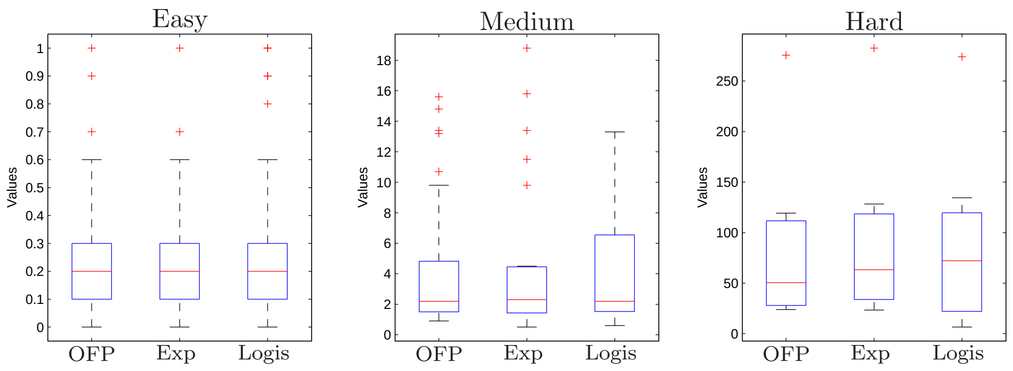

Figure 9: Box plots of the CPU time - OFP vs ORFP.

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

Table 10: Detailed results for the Hard Class - OFP vs ORFP

In order to analyse 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 or equal than 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 is 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 finally want to remark that there are four instances where at least one version of the ORFP closes the gap and OFP do not, while the opposite never happens.

	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

Table 11: Examples of instances where ORFP improves the Gap.

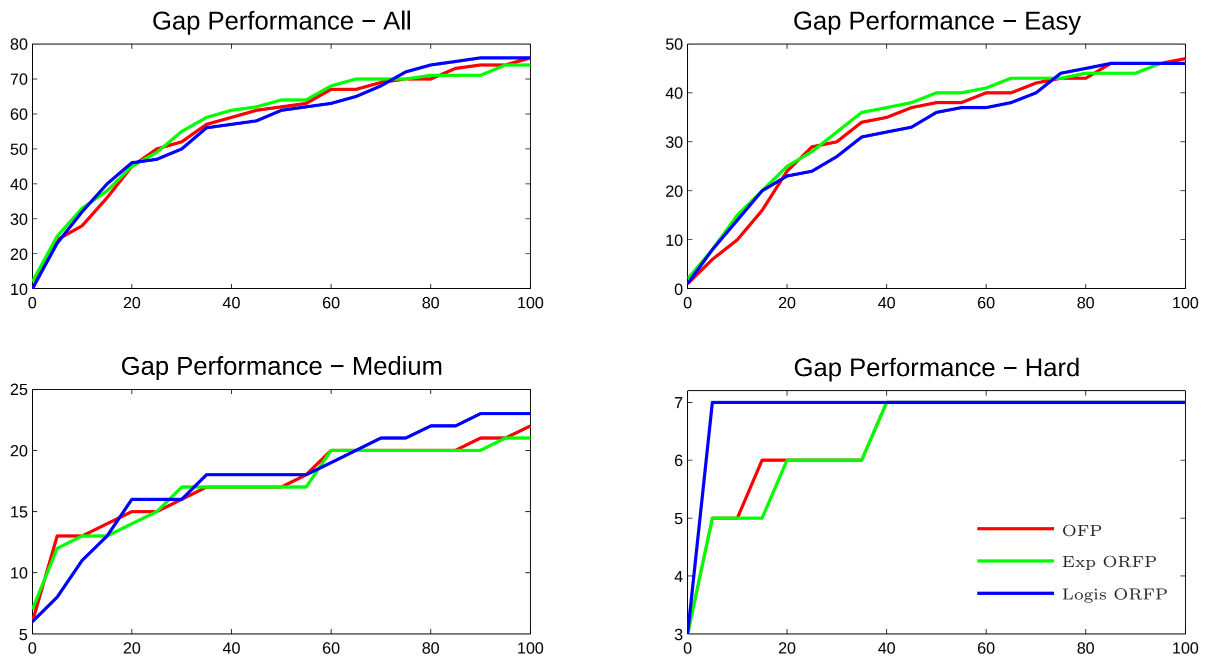

Figure 10: Profiles of the Gap% - OFP vs ORFP.

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 Objective Feasibility Pump (OFP) [1] and the Combined Objective Reweighted Feasibility Pump where we consider 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 less than ten times (Found at least once);
- The number of problems for which a feasible solution has been found for all the 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 the 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 Objective Feasibility Pump algorithm and the Objective Reweighted Feasibility Pump are shown in Tables 27 - 31. The results related to

	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 12: Comparison between OFP and Exp+Logis ORFP (Feasible solutions)

the problems for which an integer feasible solution is found in all the ten runs are reported in Tables 27 - 29. The results related to the problems for which an integer feasible solution is found in less than ten runs are reported in Tables 30 - 31. The OFP fails to find a feasible solution in all the 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.

	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

Table 13: Improvement in the quality of the solution by the introduction of the objective function in the FP and in the RFP Combined.

Overall, both the OFP and the Exp+Logis ORFP found the optimal solution for 10 instances. Also in this case we compare ORFP and Comb 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 the geometric means on those problems for which a feasible solution is calculated 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 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.

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

Table 14: Comparison between OFP and Exp+Logis ORFP (Geometric Means)

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 the 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 the 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.

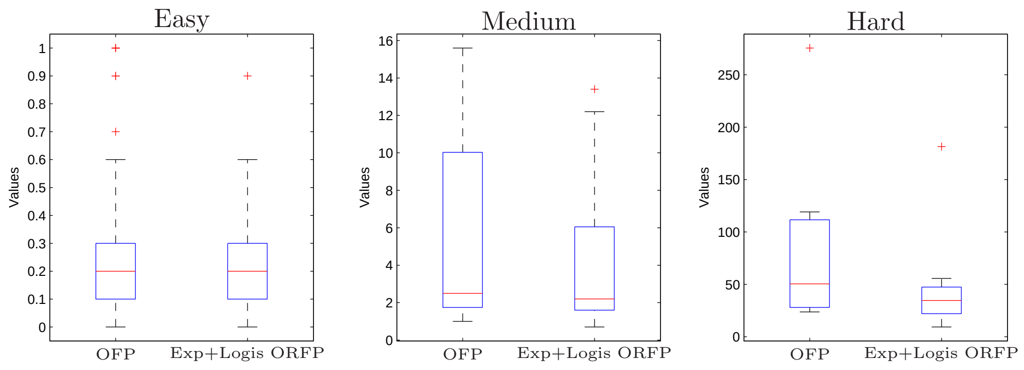

Figure 11: Box plots of the CPU time - OFP vs ORFP Combined.

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

Table 15: Detailed results for the Hard Class - OFP vs Exp+Logis ORFP

In order to analyse 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 algorithm 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.

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

Table 16: Examples of instances where Exp+Logis ORFP improves the 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.

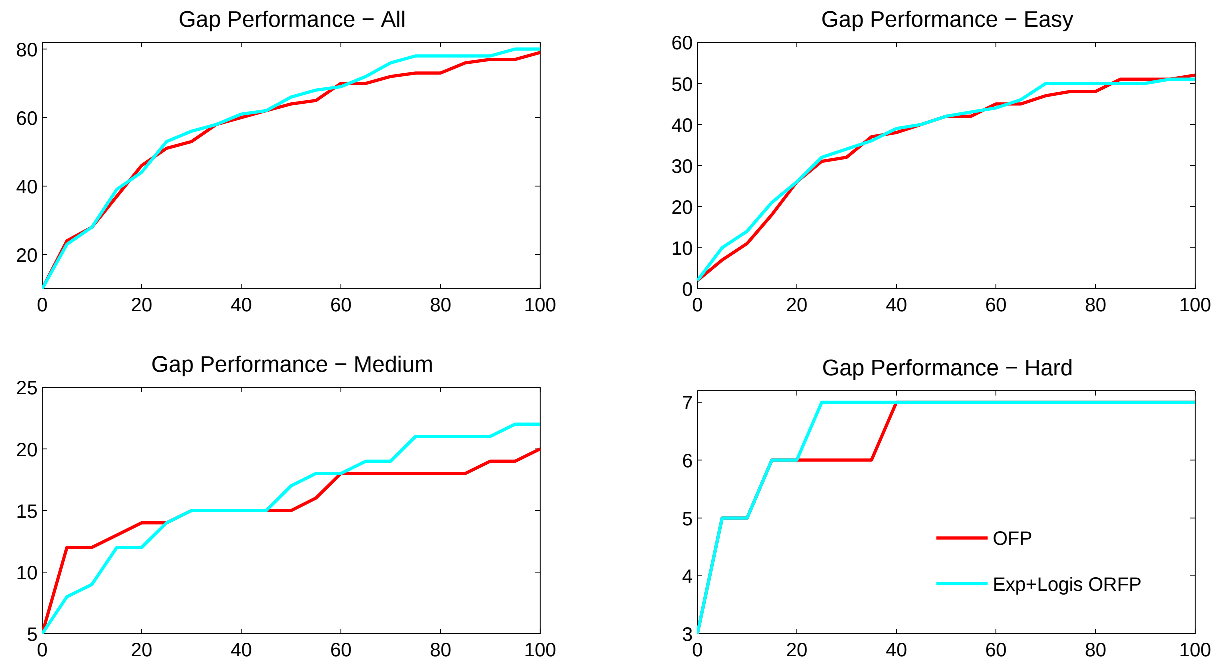

Figure 12: Profiles of the Gap% - OFP vs Exp+Logis ORFP.

8.6 Benchmarking Algorithms via Performance Profiles

In order to give a better interpretation of the results generated by the various algorithms we decided to use performance profiles [16]. 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 following value

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

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 the algorithm a was the most effective, $\rho_a(2)$ gives the fraction of problems for which the 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 both in terms of 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, that are very similar. On the contrary, the Logis ORFP profile related to the CPU 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 both in terms of number of iterations and CPU time.

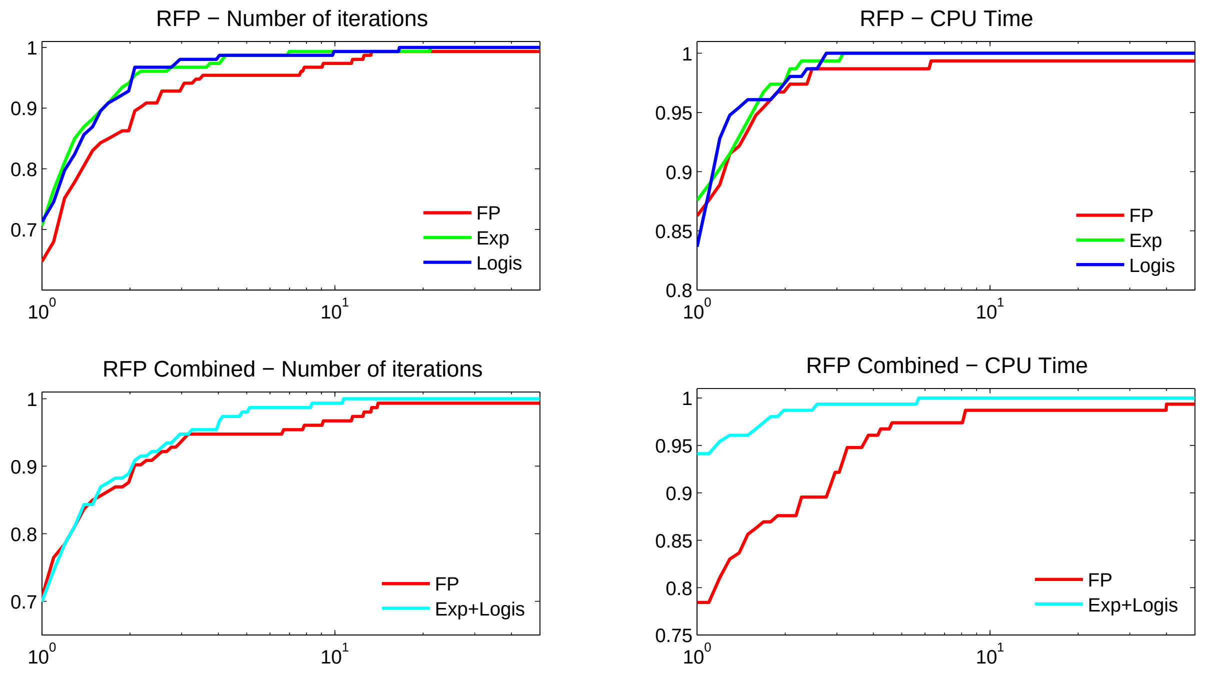

Figure 13: Performance profiles: FP vs RFP (upper figures); FP vs RFP Combined (lower figures).

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 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.

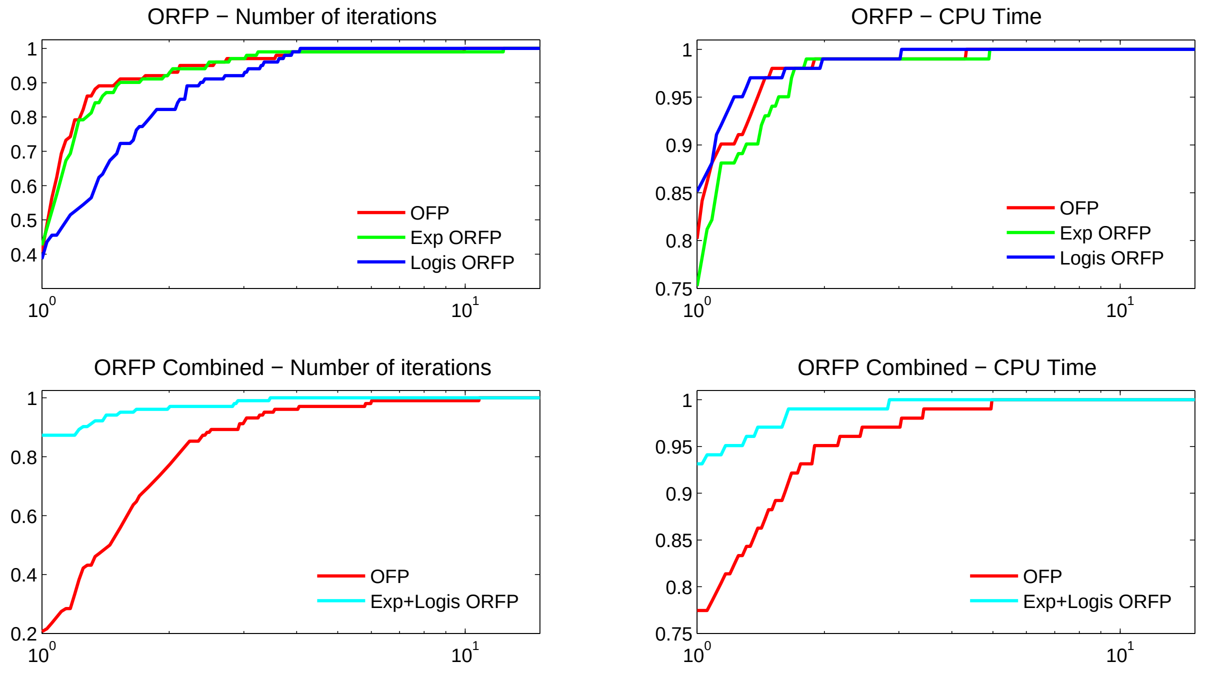

Figure 14: Performance profiles: OFP vs ORFP (upper figures); OFP vs ORFP Combined (lower figures).

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 obtained by combining two of the proposed functions (namely Exp and Logis) compares favorably with the Feasibility Pump 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 propagation like in [19] or with a procedure that examines rounded solutions along a given line segment as in [3, 9]. In particular the proposed merit functions could be also 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.

10 Appendix A

For convenience of the reader we report the proof of Proposition 4. We recall a general result concerning the equivalence between an unspecified optimization problem and a parameterized family of problems.

Consider the problems

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in W \end{array} \tag{41}$$

$$\begin{array}{ll} \min & f(x) + \psi(x,\varepsilon) \\ \text{s.t.} & x \in X \end{array} \tag{42}$$

We state the following

Theorem 1 *Let W and X be compact sets. Let $\|\cdot\|$ be a suitably chosen norm. We make the following assumptions.*

A1) *The function f is bounded on X and there exists an open set $A \supset W$ and a real number $L > 0$, such that, $\forall\ x, y \in A$, f satisfies the following condition:*

$$|f(x) - f(y)| \leq L\|x - y\|. \tag{43}$$

The function ψ satisfies the following conditions:

A2) $\forall\ x, y \in W$ *and* $\forall\ \varepsilon \in \mathbb{R}_+$,

$$\psi(x,\varepsilon) = \psi(y,\varepsilon).$$

A3) *There exist a value $\hat{\varepsilon}$ and, $\forall\ z \in W$, there exists a neighborhood $S(z)$ such that, $\forall\ x \in S(z) \cap (X \setminus W)$, and $\varepsilon \in]0, \hat{\varepsilon}]$, we have*

$$\psi(x,\varepsilon) - \psi(z,\varepsilon) \geq \hat{L}\|x - z\|, \tag{44}$$

where $\hat{L} > L$ is chosen as in (43). Furthermore, let $S = \bigcup_{z \in W} S(z)$, $\exists\ \bar{x} \notin S$ such that

$$\lim_{\varepsilon \to 0} [\psi(\bar{x},\varepsilon) - \psi(z,\varepsilon)] = +\infty, \quad \forall\ z \in W, \tag{45}$$

$$\psi(x,\varepsilon) \geq \psi(\bar{x},\varepsilon), \quad \forall\ x \in X \setminus S,\ \forall\ \varepsilon > 0. \tag{46}$$

Then, $\exists\ \tilde{\varepsilon} \in \mathbb{R}$ such that, $\forall\ \varepsilon \in]0, \tilde{\varepsilon}]$, Problems (41) and (42) have the same minimum points.

Proof. See [30].

Now we give the proof of the Proposition 4, with

$$W = \Big\{x \in P : x_i \in \{0,1\},\ \forall i \in I\Big\}, \quad X = \Big\{x \in P : 0 \leq x_i \leq 1,\ \forall i \in I\Big\}.$$

Proof of Proposition 4. As we assume that the function f satisfies assumption $A1$) of Theorem 1, the proof can be derived by showing that every penalty term (24)-(27) satisfies assumption $A2$) and $A3$) of Theorem 1.

Consider the penalty term (24).
Let c be the cardinality of I, for any $x \in W$ we have

$$\psi(x,\varepsilon) = c \cdot \log(\varepsilon)$$

and $A2$) is satisfied.
We now study the behavior of the function $\phi(x_i)$, $i \in I$, in a neighborhood of a point $z_i \in \{0,1\}$. We distinguish three different cases:

1. $z_i = 0$ and $0 < x_i < \rho$ with $\rho < \frac{1}{2}$: We have that $\phi(x_i) = \ln(x_i + \varepsilon)$ which is continuous and differentiable for $0 < x_i < \rho$, so we can use mean value Theorem obtaining that
$$\phi(x_i) - \phi(z_i) = \left(\frac{1}{\tilde{x}_i + \varepsilon}\right)|x_i - z_i|, \tag{47}$$
with $\tilde{x}_i \in (0, x_i)$. Since $\tilde{x}_i < \rho$, we have
$$\phi(x_i) - \phi(z_i) \geq \left(\frac{1}{\rho + \varepsilon}\right)|x_i - z_i|. \tag{48}$$
Choosing ρ and ε such that
$$\rho + \varepsilon \leq \frac{1}{\hat{L}}, \tag{49}$$
we obtain
$$\phi(x_i) - \phi(z_i) \geq \hat{L}|x_i - z_i|. \tag{50}$$

2. $z_i = 1$ and $1 - \rho < x_i < 1$ with $\rho < \frac{1}{2}$: We have that $\phi(x_i) = \ln(1 - x_i + \varepsilon)$ which is continuous and differentiable for $1 - \rho < x_i < 1$, so we can use mean value Theorem obtaining that
$$\phi(x_i) - \phi(z_i) = \left(-\frac{1}{1 - \tilde{x}_i + \varepsilon}\right)(x_i - z_i) = \left(\frac{1}{1 - \tilde{x}_i + \varepsilon}\right)|x_i - z_i|, \tag{51}$$
with $\tilde{x}_i \in (x_i, 1)$. Since $\rho < \frac{1}{2}$ and $\tilde{x}_i > 1 - \rho$ we have $\frac{1}{1-\tilde{x}_i} > \frac{1}{\rho}$ then
$$\phi(x_i) - \phi(z_i) \geq \left(\frac{1}{\rho + \varepsilon}\right)|x_i - z_i|. \tag{52}$$
We have again that (50) holds when ρ and ε satisfy (49).

3. $z_i = x_i = 0$ or $z_i = x_i = 1$: We have $\phi(x_i) - \phi(z_i) = 0$.

We can conclude that, when ρ and ε satisfy (49),
$$\psi(x, \varepsilon) - \psi(z, \varepsilon) \geq \hat{L} \sum_{i \in I} |x_i - z_i| \geq \hat{L} \sup_{i \in I} |x_i - z_i| \tag{53}$$
for all $z \in W$ and all x such that $\sup_{i \in I} |x_i - z_i| < \rho$.
Now we define $S(z) = \{x \in \mathbf{R}^n : \sup_{i \in I} |x_i - z_i| < \rho\}$ and $S = \bigcup_{i=1}^{N} S(z_i)$ where N is the number of points $z \in W$.
Let $\bar{x} \notin S$ be such that $\exists j \in I :\ \bar{x}_j = \rho$ $(\bar{x}_j = 1 - \rho)$ and $\bar{x}_i \in \{0, 1\}$ for all $i \neq j$, $i \in I$.
Let $\{\varepsilon^k\}$ be a sequence such that $\varepsilon^k \to 0$ for $k \to \infty$, we can write for each $z \in W$:
$$\begin{aligned}\lim_{k\to\infty} [\psi(\bar{x}, \varepsilon^k) - \psi(z, \varepsilon^k)] &= \lim_{k\to\infty} \Big([\ln(\rho + \varepsilon^k) + (c - 1)\ln(\varepsilon^k)] - c\ln(\varepsilon^k)\Big) = \\ & \lim_{k\to\infty} \Big(\ln(\rho + \varepsilon^k) - \ln(\varepsilon^k)\Big) = +\infty\end{aligned}$$
and (45) holds.
Then $\forall x \in X \backslash S$, and $\forall \varepsilon > 0$ we have for the monotonicity of the logarithm:
$$\begin{aligned}\psi(x, \varepsilon) - \psi(\bar{x}, \varepsilon) &= \sum_{i \neq j} \min\{\ln(x_i + \varepsilon), \ln(1 - x_i + \varepsilon)\} - (c - 1)\ln(\varepsilon) \\ &+ \min\{\ln(x_{\bar{j}} + \varepsilon), \ln(1 - x_{\bar{j}} + \varepsilon)\} - \ln(\rho + \varepsilon) \geq 0,\end{aligned}$$
where $\rho \leq x_{\bar{j}} \leq 1 - \rho$. Then (46) holds, and Assumption $A3$) is satisfied.
The proofs of the equivalence between (33) and (34) using the other penalty terms follow by repeating the same arguments used here. □

11 Tables

Here we report, in Tables 17 - 31, the detailed results related to our computational experience. On the vertical axis of the tables related to the problems for which an integer feasible solution is found in all the ten runs (Tables 17-19, 22-24 and 27-29) , we have

- the average number of iterations needed to find a solution (Iter),
- the average objective function value of the first integer feasible solution found (Obj),
- the average percentage gap with respect to the best known solution (Gap %),
- the average CPU time (Time).

We report "-" for the percentage gap when there is no best solution available. On the vertical axis of the tables related to the problems for which an integer feasible solution is found in less than ten runs (Tables 20, 21, 25, 26, 30, 31), we have

- the number of times an integer feasible solution is found (F.s. found),
- the average number of iterations needed to find a solution (Iter),
- the average CPU time (Time).

In case of failure, we report "-" for both Iter and Time.

References

[1] T. Achterberg, T. Berthold. *Improving the feasibility pump.* Discrete Optimization, 4, pp 77–86, 2007.

[2] T. Achterberg, T. Koch, A. Martin. *MIPLIB 2003.* Operations Research Letters, 34, pp 361–372, 2006. Problems available at http://miplib.zib.de.

[3] D. Baena, J. Castro. *Using the analytic center in the feasibility pump.* Operations Research Letters, 39, pp 310-317, 2011.

[4] E. Balas, S. Ceria, M. Dawande, F. Margot, G. Pataki. *OCTANE: A new heuristic for pure 0-1 programs.* Operations Research, 49(2), pp 207-225, 2001.

[5] E. Balas, C.H. Martin. *Pivot-and-complement: A heuristic for 0-1 programming.* Management Science, 26(1), pp 86-96, 1980.

[6] E. Balas, S. Schmieta, C. Wallace.*Pivot and shifta mixed integer programming heuristic.* Discrete Optimization, 1(1), pp 3-12, 2004.

[7] J. Baxter. , *Local optima avoidance in depot location.* Journal of the Operational Research Society, 32, pp. 815-819, 1981.

[8] L. Bertacco, M. Fischetti, A. Lodi. *A feasibility pump heuristic for general mixed-integer problems.* Discrete Optimization, 4, pp 63–76, 2007.

[9] N.L. Boland, A.C. Eberhard, F.G. Engineer, M. Fischetti, M.W.P. Savelsbergh, A. Tsoukalas *Boosting the Feasibility Pump.* Report C-OPT 2012-03, The University of Newcastle, Callaghan, NSW, 2308, Australia, 2012.

[10] J. Eckstein, M. Nediak. *Pivot, cut, and dive: a heuristic for 0-1 mixed integer programming.* Journal of Heuristics, 13, pp 471–503, 2007.

[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, A. Waechter. *An Algorithmic Framework for Convex Mixed Integer Nonlinear Programs.* Discrete Optimization, 5(2), pp 186–204, 2008.

[12] P. Bonami, G. Cornuejols, A. Lodi, F. Margot. *A feasibility pump for mixed integer nonlinear programs.* Mathematical Programming, 119, pp 331–352, 2009.

[13] C. D'Ambrosio, A. Frangioni, L. Liberti, A. Lodi. *A Storm of Feasibility Pumps for Nonconvex MINLP.* Mathematical Programming, 136(2), pp 375–402, 2012.

[14] E. Danna, E. Rothberg, C. Le Pape. *Exploring relation induced neighborhoods to improve MIP solution.* Mathematical Programming 102, 1, pp 71–90, 2005.

[15] M. De Santis, S. Lucidi, F. Rinaldi. *Feasibility Pump-Like Heuristics for Mixed Integer Problems.* DIS Technical Report n. 15, 2010.

[16] E. D. Dolan, J. J. Moré. *Benchmarking optimization software with performance profile.* Mathematical Programming 91, pp 201–213, 2002.

[17] M. Fischetti, F. Glover, A. Lodi. *The Feasibility Pump.* Mathematical Programming, 104, pp 91–104, 2005.

[18] M. Fischetti, A. Lodi. *Local Branching.* Mathematical Programming, 98(1-3), pp 23–47, 2003.

[19] M. Fischetti, D. Salvagnin. *Feasibility pump 2.0.* Mathematical Programming Computation, 1, pp 201–222, 2009.

[20] F. Glover, M. Laguna. *General purpose heuristics for integer programming part I.* Journal of Heuristics, 3, 1997.

[21] F. Glover, M. Laguna. *General purpose heuristics for integer programming part II.* Journal of Heuristics, 3, 1997.

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

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

[24] F.S. Hillier, *Efficient heuristic procedures for integer linear programming with an interior.* Operations Research, 17, pp 600-637, 1969.

[25] F.S. Hillier, R.M. Saltzman., *A heuristic ceiling point algorithm for general integer linear programming.* Management Science, 38(2), pp 263-283, 1992.

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

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

[28] A. Løkketangen, F. Glover. , *Solving zero/one mixed integer programming problems using tabu search.* European Journal of Operations Research, 106, pp 624-658, 1998.

[29] H.R. Lourenço, O. C. Martin, T. Stülze. , *Iterated local search.* Handbook of metaheuristics - Eds F. W. Glover and G. A. Kochenberger - Kluwer Academic Publishers, Boston, Dordrecht, London, pp. 321-353, 2003.

[30] S. Lucidi, F. Rinaldi. *Exact penalty functions for nonlinear integer programming problems.* Journal of Optimization Theory and Applications, 145, pp 479–488, 2010.

[31] O. L. Mangasarian. *Solutions of General Linear Complementarity Problems via Nondifferentiable Concave Minimization.* Acta Mathematica Vietnamica, 22(1), pp 199–205, 1997.

[32] O. L. Mangasarian. *Machine learning via polyhedral concave minimization.* in: Fischer, H., Riedmueller, B., Schaeffler S. Applied mathematics and parallel computing- Festschrift for Klaus Ritter, pp 175–188, Physica, Heidelberg, 1996.

[33] W. Murray, K.M. Ng *An Algorithm for Nonlinear Optimization Problems with Binary Variables*, Computational Optimization and Applications, Vol. 47, No. 2, pp 257–288, 2010.

[34] F. Rinaldi *New results on the equivalence between zero-one programming and continuous concave programming*, Optimization Letters, Vol. 3, No. 3, 377–386, 2009.

[35] F. Rinaldi, F. Schoen, M. Sciandrone. *Concave programming for minimizing the zero-norm over polyhedral sets.*, Computational Optimization and Applications, vol. 46, pp. 467–486, 2010.

Problem	**FP**				**Exp**, $\alpha = 0.5$				**Log**, $\varepsilon = 0.1$				**Hyp**, $\varepsilon = 0.1$				**Logis**,$\alpha = 0.1$			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap%	Time	Iter	Obj	Gap%	Time
a1c1s1	25.6	21615.71	87.91	2.96	22.7	20365.53	77.04	2.37	15.9	20648.03	79.50	1.67	26.2	20248.53	76.02	2.1	14.7	19736.22	71.57	2.56
aflow30a	19.9	5691.7	391.51	0.07	12	4685.3	304.60	0.05	13.4	4049.8	249.72	0.04	11.3	3431.1	196.30	0.03	14.8	5224.8	351.19	0.05
aflow40b	8.5	5711.9	389.03	0.12	12.8	5897.2	404.90	0.16	11.7	6230.5	433.43	0.14	23.3	5934.3	408.07	0.21	6.7	4911.1	320.47	0.09
cap6000	18.1	-1.734E6	29.23	1.2	21.1	-1.478E6	39.65	1.5	19.2	-1.725E6	29.60	1.01	23.4	-1.608E6	34.37	0.94	15.7	-1.887E6	22.97	1.22
dano3mip	3	1000	-	19.82	1	1000	-	18.92	1	1000	-	15.51	1	1000	-	15.63	2	1000	-	26.64
danoint	113.3	87.15	32.72	2.78	122.4	87.05	32.56	3.69	45.4	85.8	30.66	1.09	151.2	88.85	35.30	2.84	97.7	87.63	33.45	3.35
fast0507	3	179	2.87	97.38	2	185	6.32	98.55	1	190	9.20	85.09	1	192	10.34	86.99	3	185	6.32	103.97
fiber	7.6	1.495E7	3583.32	0.02	7.9	1.509E7	3618.95	0.03	9.2	1.570E7	3767.89	0.03	7.6	1.260E7	3005.05	0.03	7.6	1.547E7	3711.51	0.02
fixnet6	11.4	11727.7	194.44	0.02	114	31150.4	682.08	0.25	78.7	24590.6	517.39	0.17	64.6	23190.4	482.23	0.09	5.4	14731.3	269.85	0.01
glass4	25	1.153E10	860.48	0.05	105.6	1.011E10	742.47	0.22	73.6	1.071E10	792.68	0.15	258.5	1.234E10	928.72	0.26	100.7	9.849E9	720.75	0.21
harp2	188.8	-4.796E7	35.10	1.52	398	-4.827E7	34.68	3.31	431.4	-4.176E7	43.49	3.65	245	-4.635E7	37.28	2.18	360.2	-4.486E7	39.30	3.03
liu	1	8398	-	0.09	1	8398	-	0.09	1	8398	-	0.1	1	8398	-	0.09	1	8398	-	0.1
markshare1	1	292	29100.00	0	1	292	29100.00	0	1	292	29100.00	0	1	292	29100.00	0	1	292	29100.00	0
markshare2	1	160	15900.00	0	1	160	15900.00	0	1	160	15900.00	0	1	160	15900.00	0	1	160	15900.00	0
mas74	1	19197.47	62.67	0	1	19197.47	62.67	0	1	19197.47	62.67	0	1	19197.47	62.67	0	1	19197.47	62.67	0
mas76	1	44877.42	12.18	0	1	44877.42	12.18	0	1	44877.42	12.18	0	1	44877.42	12.18	0	1	44877.42	12.18	0
mkc	3.6	-271.65	51.82	0.1	3.8	-271.85	51.79	0.11	3.7	-271.85	51.79	0.11	3.5	-271.85	51.79	0.09	3.3	-271.65	51.82	0.15
mod011	1	0	100.00	0.07	1	0	100.00	0.07	1	0	100.00	0.1	1.9	3683598.35	106.75	0.14	1	0	100.00	0.07
modglob	1	6.027E8	2811.72	0	1	5.628E8	2619.20	0	1	6.677E8	3125.47	0	1	6.637E8	3106.09	0.01	1	5.987E8	2792.16	0
net12	42	337	57.48	6.79	153.8	337	57.48	21.06	142.2	337	57.48	14.97	113.9	337	57.48	18.53	117.1	337	57.48	18.4
nsrand-ipx	3.6	346416	576.59	0.22	3.2	402048	685.25	0.27	4.1	355872	595.06	0.25	5.1	304112	493.97	0.25	3.1	401408	684.00	0.27
opt1217	1	0	100.00	0.01	1	0	100.00	0	1	0	100.00	0	1	0	100.00	0.01	1	-12	25.00	0.01
pk1	1	36	227.27	0	1	36	227.27	0	1	36	227.27	0	1	36	227.27	0	1	36	227.27	0
pp08aCUTS	3.4	12982	76.63	0.01	4	13104	78.29	0.01	3.6	11770	60.14	0.01	3.4	12051	63.96	0.01	3.9	12581	71.17	0.01
pp08a	3.1	12810	74.29	0	3.4	13152	78.94	0.01	3	13189	79.44	0	3	13615	85.24	0	5.2	13226	79.95	0.01
qiu	5.6	1539.38	1258.53	0.19	4.8	1524.65	1247.45	0.21	5	1387.35	1144.12	0.19	4.4	669.84	604.12	0.22	4.3	1687.76	1370.21	0.28
set1ch	4.2	104900.2	92.34	0.01	3.9	101702.8	86.48	0.01	33.6	96175.68	76.35	0.03	31.1	92687.93	69.95	0.03	4.6	105014.45	92.55	0.01
seymour	4	471	11.35	2.5	3	480	13.48	2.41	3	482	13.95	1.8	2	495	17.02	1.57	3	471	11.35	2.52
sp97ar	5.2	1.468E9	122.15	5.39	4.5	1.722E9	160.61	6.59	4	8.939E8	35.24	5.65	4	1.766E10	2573.06	4.47	4.7	1.479E9	123.87	6.86
swath	84.8	36527.08	7714.83	7.11	61.3	28614.67	6022.00	5.37	61.9	35450.07	7484.41	5.23	566.8	48160.31	10203.72	36.22	34.3	21903.21	4586.11	3.51
tr12-30	83.7	243560.8	86.50	0.22	154.2	260762.2	99.67	0.45	62.5	260330.9	99.34	0.2	114.1	247401.3	89.44	0.31	114.3	245892.5	88.28	0.3
vpm2	5.8	23.88	73.67	0	4	20.93	52.22	0.01	5	20.65	50.18	0	3.5	19.25	40.00	0	5.3	20.25	47.27	0

Table 17: Comparison on MIPLIB problems (integer feasible solution found in all the ten runs). FP vs RFP

Problem	**FP**				**Exp**, $\alpha = 0.5$				**Log**, $\varepsilon = 0.1$				**Hyp**, $\varepsilon = 0.1$				**Logis**,$\alpha = 0.1$			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time
22433	8.5	21527.4	0.23	0.06	12.2	21527.5	0.24	0.07	13.2	21509	0.15	0.09	50.8	21550.3	0.34	0.14	7.8	21550.8	0.34	0.07
23588	51.6	8310.4	2.72	0.11	46.8	8314.9	2.78	0.12	392.3	8316.4	2.80	0.8	380.9	8293.7	2.52	0.77	69.4	8325	2.90	0.17
bc1	2.2	12.9	286.42	0.58	2.4	15.34	359.51	0.51	2.9	13.36	300.20	0.46	2.7	15.59	367.00	0.44	2.4	16.24	386.47	0.59
bienst1	11.4	89.92	92.34	0.09	1	68.25	45.99	0.06	1	68.25	45.99	0.07	1	68.25	45.99	0.07	1.3	75.93	62.42	0.1
bienst2	13.3	127.1	132.78	0.12	1	68.25	25.00	0.07	1	68.25	25.00	0.06	1	68.25	25.00	0.08	1	102.22	87.22	0.1
binkar10-1	27.2	609256	8936.46	0.15	26.7	909412	13388.36	0.15	31.4	608918	8931.45	0.17	74.7	1.510E6	22291.32	0.4	29.4	1.010E6	14873.36	0.17
dano3-3	12.5	1000	73.51	31.74	1	1000	73.51	13.29	1	996.08	72.83	16.48	1	758.11	31.54	11.58	1.2	997.24	73.03	17.73
dano3-4	7.8	1000	73.48	23.95	1	1000	73.48	13.61	1	1000	73.48	13.25	1	1000	73.48	13.17	1	974.74	69.10	15.88
dano3-5	9.1	997.67	72.93	26.46	1	1000	73.33	14.59	1	1000	73.33	14.88	1	1000	73.33	14.83	1	1000	73.33	16.52
mcf2	146.7	82.97	26.35	3.67	85.2	85.7	30.51	2.62	100.3	86.5	31.73	2.38	183.3	86.85	32.26	3.65	173.7	82.7	25.94	6.06
mkc1	1	-460.93	24.08	0.12	1	-146.86	75.81	0.08	1	-311.19	48.75	0.15	1	-289.23	52.36	0.07	1	-525.33	13.48	0.12
neos5	1	21	40.00	0	1	21	40.00	0	1	22	46.67	0.01	1	21	40.00	0	1	22	46.67	0.01
neos6	11.8	141.6	70.60	3.5	20	146.8	76.87	4.8	191.3	157.4	89.64	20.42	540.5	158.4	90.84	49.85	34.6	142.2	71.33	6.97
neos13	1	-28.43	70.22	1.29	1	0	100.00	0.72	1	0	100.00	0.64	1	-37.43	60.80	0.75	1	-13.14	86.24	1.27
neos14	5.5	2.157E8	290112	0.03	6.4	2.371E8	318830	0.03	6.4	2.536E8	341091	0.03	5	2.759E8	371095	0.02	5	2.473E8	332615	0.03
neos17	2.6	0.68	353.32	0.04	2.6	0.66	339.99	0.04	2.6	0.61	306.66	0.04	2.6	0.61	306.66	0.04	2.6	0.75	399.99	0.04
neos18	1	36	125.00	0.13	2	34	112.50	0.14	20.6	37.8	136.25	0.7	39	40.5	153.13	1.12	2	34	112.50	0.13
neos-430149	137.7	497.95	779.77	0.79	177.1	499.6	782.69	0.82	423.1	498.66	781.02	1.76	356.3	539.58	853.32	1.42	118.4	516.19	812.00	0.72
neos-476283	3	1056.42	159.97	444.74	1	729.57	79.54	121.23	1	681.38	67.68	116.94	1	630.09	55.06	152.1	1	680.77	67.53	71.75
neos-480878	3	590.7	19.94	0.1	3	624.72	26.84	0.1	3	546.81	11.03	0.08	3	556.93	13.08	0.09	3	610.31	23.92	0.11
neos-494568	2	29	128.71	1.48	1	-74	26.73	2.96	2	-83	17.82	1.45	4.9	238.7	336.34	1.88	1	26	125.74	1.67
neos-504674	85.8	30961.35	751.55	0.25	45.9	29748.56	718.20	0.14	56	29114.83	700.77	0.17	83.8	30126	728.58	0.25	11.3	29898.73	722.33	0.05
neos-504815	82.4	13912.75	505.90	0.2	118.3	15388.38	570.16	0.29	82.1	13813.72	501.59	0.2	158	15177.66	560.98	0.38	164.8	14854.18	546.90	0.4
neos-512201	191.2	5373.11	946.23	0.53	171.8	5165.76	905.85	0.5	210	5248.57	921.98	0.62	160.2	5270.02	926.15	0.48	198.8	5287.04	929.47	0.58
neos-522351	6.4	103262.07	477.17	0.48	4.9	38323.98	114.21	0.34	4.7	32313.14	80.61	0.3	6.4	31111	73.89	0.26	5.9	86648.7	384.31	0.58
neos-525149	1	61	0.00	12.01	1	63	3.28	11.22	1	63	3.28	9.88	1	66	8.20	7.89	1	63	3.28	7.34
neos-538867	60.4	6425	5166.39	0.33	70.6	6072.5	4877.46	0.43	53.8	8814	7124.59	0.23	124.9	9108.5	7365.98	0.5	58.3	6242	5016.39	0.34
neos-538916	38.2	5650	4116.42	0.2	24.9	5955.8	4344.63	0.16	102.4	7938.1	5823.96	0.46	160.6	7922.1	5812.01	0.7	35.8	6139.8	4481.94	0.2
neos-547911	18.4	15.3	17.69	7.81	5.2	15.6	20.00	3.25	12.8	15.8	21.54	3.41	15.6	15.6	20.00	2.39	9.7	14.7	13.08	6.1
neos-555694	9	55.9	203.80	0.35	4	24.8	34.78	0.21	8.5	90.39	391.25	0.33	28.1	106.77	480.27	0.57	66.3	108.52	489.78	1.1
neos-555771	56	130.84	603.44	1.1	17.1	91.99	394.57	0.45	45.8	123.1	561.83	0.87	16.1	95.79	415.00	0.38	169	110.21	492.53	2.64
neos-565815	1	14	0.00	9.12	2	14	0.00	10.13	55	14.7	5.00	24.38	62.2	14.8	5.71	23.44	1	14	0.00	7.32
neos-570431	4.7	27	200.00	0.27	5.4	37.7	318.89	0.32	4.2	14.3	58.89	0.23	5	19.3	114.44	0.17	5	29.7	230.00	0.29
neos-584851	4	-4	63.64	0.04	2.8	-3.9	64.55	0.04	39.9	-3.3	70.00	0.14	87.1	-2.5	77.27	0.31	3.8	-4.8	56.36	0.05
neos-603073	8	47327.85	181.88	0.08	10.4	46853.05	179.05	0.13	5	46611.7	177.61	0.06	5	46550.72	177.25	0.06	9.3	46704.76	178.17	0.11
neos-611838	4	4.849E6	174.90	2.18	3	4.342E6	146.20	1.91	3.4	4.102E6	132.59	2.93	3.5	4.309E6	144.31	2.45	3.4	5043085.61	185.89	2.16
neos-612125	3	4.792E6	159.85	2.81	4.8	4.232E6	129.46	3.64	4.6	4.378E6	137.42	4.26	4.1	4.364E6	136.63	3.17	3	4793407.04	159.89	1.83
neos-612143	3	4.805E6	167.56	2.92	5.2	4.140E6	130.51	2.44	4.1	4.167E6	131.99	1.8	4.2	4.408E6	145.45	3.14	3	4598667.63	156.05	1.9

Table 18: Comparison on COR@L problems (integer feasible solution found in all the ten runs). FP vs RFP - Part I

Problem	**FP**				**Exp**, $\alpha = 0.5$				**Log**, $\varepsilon = 0.1$				**Hyp**, $\varepsilon = 0.1$				**Logis**,$\alpha = 0.1$			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time
neos-612162	3.4	4.827E6	172.28	2.93	3.4	4.436E6	150.22	3.16	4.9	4.252E6	139.88	2.71	4.5	4.311E6	143.16	3.46	3	5.167E6	191.42	1.99
neos-655508	0	6.302E7	0.00	0.04	0	6.302E7	0.00	0.04	0	6.302E7	0.00	0.04	0	6.302E7	0.00	0.04	0	6.302E7	0.00	0.04
neos-775946	124.1	764.3	4768.15	3.25	126.4	857.61	5362.48	3.26	41.9	714.14	4448.66	1.87	80.8	749.25	4672.29	1.98	98.8	794.52	4960.64	2.47
neos-780889	2	1.082E7	216.28	48.19	2.4	1.104E7	222.67	52.38	2	1.097E7	220.54	50.68	2.8	1.126E7	229.05	63.65	2	1.091E7	218.75	50.17
neos-801834	2	64502	28.02	0.8	1	54872	8.90	0.4	2	61289	21.64	0.36	2	60964	20.99	0.38	2	62990	25.01	0.84
neos-824695	3.7	77	148.39	0.75	3.7	77	148.39	0.8	3.9	77	148.39	0.85	3.7	77	148.39	0.82	4.1	77	148.39	0.82
neos-825075	4	218	180.15	0.06	8	544	300.00	0.1	3	8	102.94	0.06	198.3	903	431.99	0.92	3	218	180.15	0.06
neos-826250	3.1	63	125.00	0.4	3.3	63	125.00	0.44	3.3	63	125.00	0.42	3.3	63	125.00	0.42	3.1	63	125.00	0.38
neos-826812	2.7	83.01	43.10	0.72	2.8	83.01	43.10	0.68	2.7	83.01	43.10	0.66	2.8	83.01	43.10	0.69	2.7	83.01	43.10	0.73
neos-827175	2	121	8.04	1.8	2	121	8.04	2.24	2	121	8.04	2.23	2	121	8.04	2.24	2	121	8.04	1.81
neos-839859	1	9.425E7	860.77	0.2	1	1.317E8	1242.13	0.21	1	5.856E7	496.93	0.2	1	5.856E7	496.93	0.21	1	1.317E8	1242.13	0.21
neos-860300	14.3	7685.3	140.09	3.13	13.7	8203.3	156.27	2.45	25.6	7092.9	121.58	2.03	144.9	9005.7	181.34	4.52	10.7	6677.8	108.62	2.85
neos-886822	2	138398	381.30	0.26	1	178597.5	521.10	0.2	1	28820.5	0.23	0.17	1	28820.5	0.23	0.16	2	178597.5	521.10	0.25
neos-892255	3.6	18.7	33.57	0.15	3.7	18.9	35.00	0.14	3.8	18.8	34.29	0.13	10.8	48.4	245.71	0.33	3.7	18.9	35.00	0.14
neos-906865	2	9105.2	186.78	0.05	2	9910.6	212.14	0.05	2	9910.2	212.13	0.05	2	10714.8	237.47	0.04	2	10712.4	237.40	0.05
neos-955215	2.2	9037.66	1924.11	0.01	3	967.6	116.71	0.01	3	911.58	104.16	0.01	3	897.42	100.99	0.01	3.4	928.7	108.00	0.01
neos-1058477	2.8	3.58	550.91	0.02	2.4	3.76	583.64	0.02	2.8	2.78	405.45	0.02	2.4	3.74	580.00	0.03	3.8	5.4	881.82	0.03
neos-1171448	1	0	100.00	0.6	1	0	100.00	0.5	1	0	100.00	0.53	1	0	100.00	0.45	1	0	100.00	0.49
neos-1200887	1	-38	48.65	0.02	1	-52	29.73	0.02	1	-42	43.24	0.02	1	-38	48.65	0.02	1	-44	40.54	0.02
neos-1211578	1	-51	33.77	0	1	-48	37.66	0.01	1	-44	42.86	0	1	-52	32.47	0	1	-48	37.66	0
neos-1225589	27.2	2.36E10	1815.07	0.05	10.6	2.42E10	1873.41	0.02	29.3	2.30E10	1773.28	0.06	26.2	2.38E10	1832.65	0.05	16.6	2.24E10	1723.08	0.03
neos-1228986	1	-92	25.20	0	1	-80	34.96	0	1	-72	41.46	0.01	1	-70	43.09	0.01	1	-75	39.02	0
neos-1337489	1	-51	33.77	0	1	-48	37.66	0.01	1	-44	42.86	0	1	-52	32.47	0	1	-48	37.66	0
neos-1413153	2	119.12	13.32	0.37	1	119.12	13.32	0.39	1	119.12	13.32	0.38	1	119.12	13.32	0.4	1	119.12	13.32	0.37
neos-1415183	1	425.6	302.53	0.53	1	128.61	21.64	0.46	1	128.61	21.64	0.48	1	128.61	21.64	0.47	1	425.6	302.53	0.58
neos-1437164	23.6	25.9	223.75	0.14	63.8	23.3	191.25	0.35	22.5	22.7	183.75	0.13	42	21	162.50	0.25	9.2	23.5	193.75	0.06
neos-1440447	1	-52	48.00	0.01	1	-56	44.00	0.01	1	-60	40.00	0.01	1	-46	54.00	0.01	1	-60	40.00	0.01
neos-1460265	35.7	15925	18.84	0.18	17.4	15820	18.06	0.11	25.9	15910	18.73	0.15	40.8	15840	18.21	0.21	28.8	15905	18.69	0.16
neos-1480121	2	89.33	107.74	0	2	95.8	122.79	0	2	95.8	122.79	0	2	95.8	122.79	0	2	89.33	107.74	0
neos-1489999	5.8	476.9	34.72	0.05	6.8	483	36.44	0.06	5.1	498.3	40.76	0.05	4.7	487.4	37.68	0.05	6.2	481.5	36.02	0.05
neos-1516309	9	54363.5	51.20	0.13	12.7	53987	50.16	0.17	11.7	53707	49.38	0.15	9.8	54282	50.98	0.13	11.8	53105	47.70	0.15
neos-1595230	3.5	20.4	126.67	0.1	4.1	21.3	136.67	0.1	4.5	20.5	127.78	0.11	4.9	21.8	142.22	0.1	4	20.3	125.56	0.1
neos-1597104	4.6	-7.1	76.33	8.08	4.3	-7.5	75.00	11.1	8.7	-2	93.33	8.98	28.5	-7.5	75.00	30.9	4	-6.5	78.33	8.5
neos-1599274	3	36277.6	13.10	0.17	3	37547.6	17.06	0.17	6	37347.6	16.44	0.14	17.2	52419.12	63.42	0.38	5.3	53258.16	66.04	0.17
neos-1620807	8.8	9.5	58.33	0.02	7.2	9.5	58.33	0.02	10.5	9.8	63.33	0.02	7.8	9.8	63.33	0.02	7	9.1	51.67	0.02
prod1	1	0	100.00	0	1	0	100.00	0	1	0	100.00	0	1	0	100.00	0	1	0	100.00	0
qap10	516.8	502.4	47.76	1690.54	1	406	19.41	7.45	1	406	19.41	7.91	2	406	19.41	11.95	1	406	19.41	8.74
roy	38.3	5810.25	81.06	0.03	30.3	5887.4	83.47	0.02	17.2	5761.75	79.55	0.02	30.7	5806.61	80.95	0.03	37.2	5590.45	74.21	0.03

Table 19: Comparison on COR@L problems (integer feasible solution found in all the ten runs). FP vs RFP - Part II

Problem	**FP**			**Exp**, $\alpha = 0.5$			**Log**, $\varepsilon = 0.1$			**Hyp**, $\varepsilon = 0.1$			**Logis**,$\alpha = 0.1$		
	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time
10teams	10	122.40	7.34	10	107.00	6.24	1	-	-	0	-	-	10	92.30	5.86
air04	10	11.20	12.52	10	4.60	8.13	5	-	-	0	-	-	10	21.20	19.56
air05	10	2.00	2.42	10	3.00	2.88	10	7.00	4.58	1	-	-	10	5.00	3.51
misc07	10	39.60	0.13	10	62.90	0.20	10	490.80	0.96	8	-	-	10	46.50	0.16
momentum1	10	474.20	577.99	10	382.40	482.49	5	-	-	0	-	-	10	450.70	544.05
nw04	10	1.00	0.94	10	1.00	1.79	8	-	-	7	-	-	10	1.00	1.48
p2756	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
protfold	10	360.20	107.67	9	-	-	0	-	-	0	-	-	10	553.50	162.36
t1717	10	18.00	366.80	10	56.40	918.95	5	-	-	1	-	-	10	24.10	511.38

Table 20: Comparison on MIPLIB problems (feasible solution found in less than ten runs). FP vs RFP

Problem	FP			Exp, $\alpha = 0.5$			Log, $\varepsilon = 0.1$			Hyp, $\varepsilon = 0.1$			Logis,$\alpha = 0.1$		
	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time
aligninq	10	380.10	6.01	10	621.70	9.54	2	-	-	0	-	-	8	-	-
lrn	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos2	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos3	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos11	10	5.30	0.90	10	14.40	1.81	10	111.80	4.56	8	-	-	10	14.70	1.76
neos12	10	5.00	7.80	10	5.00	8.02	6	724.00	154.67	0	-	-	10	6.00	8.28
neos-583731	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-593853	1	-	-	10	69.70	0.93	7	-	-	0	-	-	6	-	-
neos-598183	10	91.70	0.87	9	-	-	10	71.40	0.65	6	-	-	10	83.30	0.78
neos-631694	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-709469	4	-	-	3	-	-	0	-	-	0	-	-	3	-	-
neos-777800	10	13.70	5.19	10	16.90	6.52	10	54.70	12.67	2	-	-	10	4.00	1.98
neos-791021	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-799711	0	-	-	10	83.80	659.80	10	1.30	194.87	10	26.70	198.20	9	-	-
neos-799716	0	-	-	9	-	-	9	-	-	7	-	-	4	-	-
neos-803219	0	-	-	0	-	-	2	-	-	5	-	-	0	-	-
neos-803220	5	-	-	9	-	-	10	253.00	1.55	10	183.30	1.15	9	-	-
neos-806323	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-807639	2	-	-	2	-	-	1	-	-	1	-	-	0	-	-
neos-807705	0	-	-	0	-	-	0	-	-	2	-	-	0	-	-
neos-810286	10	139.10	46.72	10	90.30	29.90	3	-	-	0	-	-	10	10.00	5.77
neos-810326	10	668.10	76.05	6	-	-	0	-	-	0	-	-	9	-	-
neos-820879	10	5.00	1.68	10	19.10	4.79	10	47.30	8.69	6	-	-	10	11.00	3.86
neos-829552	10	1.00	17.86	10	2.00	17.82	10	33.50	52.66	1	-	-	10	1.00	17.46
neos-862348	9	-	-	9	-	-	10	415.30	4.34	10	259.40	2.95	8	-	-
neos-880324	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-912015	6	-	-	5	-	-	0	-	-	0	-	-	6	-	-
neos-932816	2	-	-	2	-	-	1	-	-	0	-	-	0	-	-
neos-941698	10	29.80	0.80	10	48.80	1.11	10	435.00	5.78	0	-	-	10	47.40	1.10
neos-948268	10	5.00	6.36	10	6.00	6.16	10	9.00	9.07	2	-	-	10	7.00	7.93
neos-957270	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-957389	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-1215259	7	-	-	5	-	-	0	-	-	0	-	-	8	-	-
neos-1281048	10	131.80	1.79	10	338.60	4.60	7	-	-	0	-	-	10	173.00	2.65
neos-1396125	2	-	-	0	-	-	5	-	-	4	-	-	2	-	-
neos-1441553	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-

Table 21: Comparison on COR@L problems (feasible solution found in less than ten runs). FP vs RFP

Problem	FP				FP+Log				Exp+Log				Logis+Log				Exp+Logis			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time
10teams	122.4	994.4	7.62	7.34	231.9	975.8	5.61	13.09	197.6	1008.6	9.16	9.48	192.4	998	8.01	10.84	201	1007.2	9.00	11.37
a1c1s1	25.6	21615.71	87.91	2.96	30.1	22509.27	95.67	2.2	38	22635.76	96.77	2.74	24.9	23436.21	103.73	2.12	20.8	22357.3	94.35	2.05
aflow30a	19.9	5691.7	391.51	0.07	10.1	3636.3	214.02	0.01	7.5	3309.3	185.78	0.01	8.4	3747.8	223.64	0.02	8.2	4176.3	260.65	0.01
aflow40b	8.5	5711.9	389.03	0.12	6.7	4085.4	249.78	0.04	9.2	4663.1	299.24	0.05	8	4581.2	292.23	0.05	7.2	4962.5	324.87	0.04
air04	11.2	61461.9	9.49	12.52	28.6	69078.2	23.05	65.59	19.3	70144.9	24.95	45.23	9.8	59614.3	6.19	23.25	28.2	68491.4	22.01	70.62
air05	2	32368	22.73	2.42	13.8	36682.5	39.09	16.54	21	36708.8	39.19	24.2	40.7	44456.9	68.56	27.54	3	29948	13.55	6.04
cap6000	18.1	-1.734E6	29.23	1.2	28.8	-1.799E6	26.57	1.01	17.6	-1.756E6	28.34	0.67	8.2	-1.735E6	29.17	0.35	6.2	-2.008E6	18.05	0.29
dano3mip	3	1000.00	-	19.82	1	1000.00	-	10.83	1	1000.00	-	10.17	1	1000.00	-	10.44	1	1000.00	-	10.64
danoint	113.3	87.15	32.72	2.78	173.8	82	24.87	3.4	80.1	83.28	26.82	1.76	180.2	84	27.92	3.6	111	82.1	25.03	2.46
fast0507	3	179	2.87	97.38	1	186	6.90	7.37	1	186	6.90	6.63	1	186	6.90	9.94	3	195	12.07	23.47
fiber	7.6	1.495E7	3583.32	0.02	6.2	9.512E6	2243.29	0.02	6.2	9.513E6	2243.41	0.02	8	1.420E7	3397.05	0.02	7	1.412E7	3379.30	0.02
fixnet6	11.4	11727.7	194.44	0.02	113.7	27806.5	598.13	0.12	143.7	31703.8	695.98	0.15	114.5	29716.3	646.08	0.12	121.9	27972	602.28	0.13
glass4	25	1.153E10	860.48	0.05	76.4	8.157E9	579.76	0.05	107	7.479E9	523.28	0.07	89.4	7.453E9	521.11	0.06	102.7	7.864E9	555.31	0.07
liu	1	8398.00	-	0.09	1	4720.00	-	0.06	1	4720.00	-	0.06	1	4720.00	-	0.06	1	4720.00	-	0.07
markshare1	1	292	29100	0	1	292	29100	0	1	292	29100	0	1	292	29100	0	1	292	29100	0
markshare2	1	160	15900	0	1	160	15900	0	1	160	15900	0	1	160	15900	0	1	160	15900	0
mas74	1	1917.47	83.75	0	1	19197.47	62.67	0	1	19197.47	62.67	0	1	19197.47	62.67	0	1	19197.47	62.67	0
mas76	1	44877.42	12.18	0	1	44877.42	12.18	0	1	44877.42	12.18	0	1	44877.42	12.18	0	1	44877.42	12.18	0
misc07	39.6	4236.5	50.77	0.13	75.1	4388.5	56.17	0.1	75.2	4442.5	58.10	0.11	58.9	4251.5	51.30	0.09	67.2	4237	50.78	0.1
mkc	3.6	-271.65	51.82	0.1	3.5	-271.85	51.79	0.1	3.4	-271.85	51.79	0.08	3.5	-271.85	51.79	0.1	3.4	-271.85	51.79	0.1
mod011	1	0	100.00	0.07	1	0	100.00	0.04	1	0	100.00	0.04	1	0	100.00	0.04	1	0	100.00	0.03
modglob	1	6.027E8	2811.72	0	1	6.258E8	2923.21	0	1	5.620E8	2615.18	0	1	5.600E8	2605.54	0	1	6.258E8	2923.21	0
net12	42	337	57.48	6.79	133.2	337	57.48	4.2	109.3	337	57.48	3.32	197.1	337	57.48	5.56	168.8	337	57.48	4.88
nsrand-ipx	3.6	346416	576.59	0.22	4	345600	575.00	0.28	3.2	378336	638.94	0.3	4.2	347168	578.06	0.31	3.2	393680	668.91	0.29
nw04	1	19882	17.91	0.94	5	19657	16.58	25.29	12.8	29484.6	74.86	64.31	11.2	42121.8	149.80	50.59	1	19882	17.91	0.43
opt1217	1	0	100.00	0.01	1	-12	25.00	0.01	1	-12	25.00	0.01	1	-12	25.00	0	1	-12	25.00	0.01
pk1	1	36	227.27	0	1	36	227.27	0	1	36	227.27	0	1	36	227.27	0	1	36	227.27	0
pp08aCUTS	3.4	12982	76.63	0.01	3.7	12223	66.30	0.01	3.7	12030	63.67	0.01	3.7	12208	66.10	0.01	4.6	12624	71.76	0.01
pp08a	3.1	12810	74.29	0	3	12505	70.14	0	3	12453	69.43	0	3	12439	69.24	0	3.8	12949	76.18	0
qiu	5.6	1539.38	1258.53	0.19	3.8	1390.67	1146.62	0.13	4.3	1491.29	1222.34	0.13	5.1	954.68	818.49	0.13	4.8	1741.11	1410.36	0.13
set1ch	4.2	104900.2	92.34	0.01	6	83983	53.99	0.01	6	83983	53.99	0.01	8.9	83247.7	52.64	0.01	4	84122.05	54.25	0.01
seymour	4	471	11.35	2.5	3	482	13.95	1.05	3	482	13.95	1.07	2	481	13.71	1.06	3	477	12.77	1.12
sp97ar	5.2	1.468E9	122.15	5.39	6	8.768E8	32.65	1.39	4	9.985E8	51.06	1.36	4	9.344E8	41.37	1.4	7	1.606E9	142.99	1.66
swath	84.8	36527.08	7714.83	7.11	69.6	36824.53	7778.47	3.12	69.2	33118.44	6985.57	3.32	64.1	27368.93	5755.48	3.13	61.9	29633.73	6240.03	3.17
t1717	18	201829.70	-	366.8	69.7	512750.40	-	373.76	65.5	525868.20	-	386	59.2	363772.90	-	256.13	52	417769.40	-	427.11
tr12-30	83.7	243560.8	86.50	0.22	55.8	262726.4	101.17	0.09	83.6	265720.4	103.47	0.14	113.2	263093.1	101.46	0.17	98.9	271277.7	107.72	0.15
vpm2	5.8	23.88	73.67	0	6.2	20.38	48.22	0	6.6	19.75	43.64	0	6.2	20.23	47.13	0	6.4	22.1	60.73	0

Table 22: Comparison on MIPLIB problems (integer feasible solution found in all the ten runs). FP vs Combined RFP

Problem	**FP**				**FP+Log**				**Exp+Log**				**Logis+Log**				**Exp+Logis**			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time
22433	8.5	21527.4	0.23	0.06	16.4	21540.3	0.29	0.05	12.7	21548.3	0.33	0.04	12.2	21517	0.19	0.04	11.4	21541.8	0.30	0.04
23588	51.6	8310.4	2.72	0.11	95.5	8322.6	2.88	0.14	39.8	8301.8	2.62	0.06	61.9	8287.7	2.44	0.09	35.1	8325	2.90	0.05
bc1	2.2	12.9	286.42	0.58	2.1	9.44	182.77	0.17	2.4	10.28	207.94	0.17	2	10.66	219.32	0.17	2.1	9.93	197.45	0.17
bienst1	11.4	89.92	92.34	0.09	1	83.92	79.51	0.05	1	68.25	45.99	0.06	1	68.25	45.99	0.06	1	68.25	45.99	0.06
bienst2	13.3	127.1	132.78	0.12	1	76.03	39.25	0.05	1	78	42.86	0.05	1	74	35.53	0.06	1	72.42	32.64	0.05
binkar10-1	27.2	609256.29	8936.46	0.15	20.9	408583.99	5960.10	0.05	20.7	508570.6	7443.10	0.05	35.9	508968.41	7449.00	0.08	25.6	608858.49	8930.56	0.06
dano3-3	12.5	1000	73.51	31.74	1	641.91	11.38	8.47	1	623.32	8.15	8.46	1	627.64	8.90	8.48	1	653.48	13.38	8.67
dano3-4	7.8	1000	73.48	23.95	1	651.9	13.09	8.63	1	674.14	16.95	8.51	1	668.04	15.89	8.53	1	665.13	15.39	8.65
dano3-5	9.1	997.67	72.93	26.46	1	691.88	19.93	8.62	1	709.29	22.94	8.72	1	706.27	22.42	8.74	1	670.24	16.17	8.62
mcf2	146.7	82.97	26.35	3.67	118.8	85.7	30.51	2.39	136.3	85.8	30.66	3.03	103.8	83.85	27.69	2.12	115.2	84.45	28.60	2.58
mkc1	1	-460.93	24.08	0.12	1	-566.15	6.75	0.03	1	-566.15	6.75	0.03	1	-566.15	6.75	0.04	1	-566.15	6.75	0.03
neos5	1	21	40.00	0	2	18	20.00	0	1	17	13.33	0	2	17.5	16.67	0	2	18	20.00	0
neos6	11.8	141.6	70.60	3.5	5.3	129	55.42	0.87	23.4	131.8	58.80	2.33	17.5	149.4	80.00	2.06	31.4	142.9	72.17	2.92
neos11	5.3	10	11.11	0.9	6.9	9.1	1.11	0.84	7.8	9	0.00	0.74	8.7	9	0.00	0.77	14.8	10.6	17.78	1.93
neos12	5	20	53.85	7.8	39	19.5	50.00	35.79	20.2	16.2	24.62	10.34	41.7	20.6	58.46	20	4	19	46.15	7.25
neos13	1	-28.43	70.22	1.29	1	-14.95	84.34	0.86	1	-15.04	84.25	0.86	1	-16.47	82.75	0.78	1	-46.34	51.46	0.48
neos14	5.5	2.157E8	290112	0.03	4.7	2.436E8	327631	0.01	5	2.819E8	379148	0.01	5	2.914E8	391898	0.01	5.2	2.655E8	357016	0.01
neos17	2.6	0.68	353.32	0.04	2.6	0.58	286.66	0.03	2.6	0.58	286.66	0.03	2.3	0.54	259.99	0.04	2.8	0.58	286.66	0.03
neos18	1	36	125.00	0.13	7.2	32.9	105.63	0.17	8.3	29.1	81.88	0.17	10	33.3	108.13	0.22	2	34	112.50	0.07
neos-430149	137.7	497.95	779.77	0.79	132.3	438.86	675.37	0.27	219	465.04	721.63	0.39	162.2	522.29	822.77	0.29	174.4	533.29	842.21	0.3
neos-476283	3	1056.42	159.97	444.74	1	523.17	28.75	8.48	1	511.24	25.81	8.37	1	514.88	26.71	8.48	1	541.27	33.20	11.13
neos-480878	3	590.7	19.94	0.1	3.6	542.04	10.06	0.04	3.5	540.16	9.67	0.03	3.6	553.33	12.35	0.04	3.3	562.9	14.29	0.03
neos-494568	2	29	128.71	1.48	2	-82	18.81	0.22	2	-82	18.81	0.22	2	-81	19.80	0.22	1	-72	28.71	0.23
neos-504674	85.8	30961.35	751.55	0.25	124.6	30946.73	751.15	0.15	35	31121.71	755.96	0.06	80.8	31777.31	773.99	0.12	31.1	29473.78	710.64	0.05
neos-504815	82.4	13912.75	505.90	0.2	96.1	13720.27	497.52	0.11	32.9	13982.71	508.94	0.05	103.4	15708.03	584.08	0.13	54.3	13976.53	508.68	0.07
neos-512201	191.2	5373.11	946.23	0.53	157.6	5557.65	982.16	0.19	165.3	5458.2	962.80	0.29	134.1	5407.88	953.00	0.23	193.2	5524.32	975.67	0.24
neos-522351	6.4	103262.07	477.17	0.48	5.3	40010.8	123.64	0.07	5.8	46605.3	160.49	0.08	4.7	30080.06	68.13	0.07	4.7	49141.5	174.67	0.08
neos-525149	1	61	0.00	12.01	1	65	6.56	1.6	1	65	6.56	1.61	1	65	6.56	1.47	1	63	3.28	1.46
neos-538867	60.4	6425	5166.39	0.33	82.2	6830	5498.36	0.23	70.3	5419.5	4342.21	0.19	50.2	5645	4527.05	0.12	69.7	6989.5	5629.10	0.19
neos-538916	38.2	5650	4116.42	0.2	30.7	6109.2	4459.10	0.08	49.3	6398.7	4675.15	0.12	31.7	5846.6	4263.13	0.08	36.3	6430.4	4698.81	0.09
neos-547911	18.4	15.3	17.69	7.81	12.6	15	15.38	1.15	15.7	15	15.38	2.2	11.5	15.4	18.46	2.6	7.3	15.3	17.69	1.7
neos-555694	9	55.9	203.80	0.35	16.2	78.56	326.96	0.18	17.7	87.49	375.49	0.21	17	61.18	232.50	0.21	4	25	35.87	0.09
neos-555771	56	130.84	603.44	1.1	17.6	86.74	366.34	0.2	11.6	104.41	461.34	0.15	16.4	90.83	388.33	0.19	4	43.6	134.41	0.09
neos-565815	1	14	0.00	9.12	8.3	14.5	3.57	2.36	5.2	14.8	5.71	2.22	9.3	15.4	10.00	3.07	5.1	14.2	1.43	2.41
neos-570431	4.7	27	200.00	0.27	4.3	16	77.78	0.12	3.7	15.1	67.78	0.11	3.7	14.8	64.44	0.12	5.5	23.8	164.44	0.16
neos-584851	4	-4	63.64	0.04	9.5	-5.5	50.00	0.04	10.6	-5.2	52.73	0.04	12.3	-6.3	42.73	0.04	2.5	-4.1	62.73	0.03
neos-598183	91.7	48288.78	162.01	0.87	18.2	47013.6	155.09	0.06	218.3	47841.88	159.59	0.46	16.4	47547.14	157.99	0.06	135.1	49824.98	170.35	0.29
neos-603073	8	47327.85	181.88	0.08	5.7	46725.08	178.29	0.02	5.8	46760.97	178.50	0.02	5.5	46171.88	174.99	0.02	38.3	49371.71	194.05	0.09
neos-611838	4	4.849E6	174.90	2.18	6.2	3.730E6	111.48	0.75	5.2	3.646E6	106.68	0.66	5.8	3.811E6	116.07	0.82	3	3.577E6	102.81	0.69
neos-612125	3	4.793E6	159.85	2.81	5.7	4.098E6	122.18	0.97	4.3	3.999E6	116.81	0.82	4.2	3.929E6	113.02	0.7	3.7	4.068E6	120.57	0.93
neos-612143	3	4.805E6	167.56	2.92	5.9	3.838E6	113.72	0.76	5.9	3.849E6	114.31	0.69	3.8	3.911E6	117.78	0.63	4	3.667E6	104.17	0.74
neos-612162	3.4	4.827E6	172.28	2.93	5.8	3.681E6	107.63	0.73	5.9	3.928E6	121.54	0.73	4.2	3.834E6	116.27	0.61	3.1	3.534E6	99.33	0.52
neos-655508	0	6.302E7	0.00	0.04	0	6.302E7	0.00	0.03	0	6.302E7	0.00	0.02	0	6.302E7	0.00	0.02	0	6.302E7	0.00	0.03

Table 23: Comparison on COR@L problems (integer feasible solution found in all the ten runs). FP vs Combined RFP - Part I

Problem	**FP**				**FP+Log**				**Exp+Log**				**Logis+Log**				**Exp+Logis**			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time
neos-775946	124.1	764.3	4768.15	3.25	14.6	391.98	2396.69	0.5	22	530.29	3277.64	0.61	4.5	531.1	3282.80	0.39	18.8	496.09	3059.81	0.57
neos-777800	13.7	-80	0.00	5.19	4	-80	0.00	1.37	11.3	-80	0.00	5.14	19.9	-80	0.00	9.23	10.2	-80	0.00	6.81
neos-780889	2	1.082E7	216.28	48.19	2.1	1.003E7	193.22	100.65	3.3	1.026E7	199.88	96.59	2	1.018E7	197.47	93.2	2.1	1.003E7	193.29	83.28
neos-801834	2	64502	28.02	0.8	2	55577	10.30	0.37	2	60875	20.82	0.37	2	61233	21.53	0.37	1	54051	7.27	0.38
neos-810286	139.1	3431.9	19.29	46.72	81.3	3435.4	19.41	44.06	74.2	3377.3	17.39	44.79	83.3	3316.1	15.26	42.2	114.1	3485.8	21.16	80.34
neos-820879	5	34433.7	35.20	1.68	10.7	38492.4	51.14	1.32	12.4	37749.1	48.22	1.4	13.5	37945.1	48.99	1.61	6.5	37208.2	46.10	0.98
neos-824695	3.7	77	148.39	0.75	3.8	77	148.39	0.64	3.7	77	148.39	0.63	3.9	77	148.39	0.65	3.9	77	148.39	0.67
neos-825075	4	218	180.15	0.06	9	465	270.96	0.06	4	108	139.71	0.04	3	8	102.94	0.04	6.2	395	245.22	0.04
neos-826250	3.1	63	125.00	0.4	3.2	63	125.00	0.35	3.3	63	125.00	0.38	3.4	63	125.00	0.37	3.4	63	125.00	0.38
neos-826812	2.7	83.01	43.10	0.72	2.7	83.01	43.10	0.59	2.4	83.01	43.10	0.56	2.8	83.01	43.10	0.62	2.7	83.01	43.10	0.62
neos-827175	2	121	8.04	1.8	2	121	8.04	1.12	2	121	8.04	1.12	2	121	8.04	1.13	2	121	8.04	1.14
neos-829552	1	26.69	1050.43	17.86	7	2.91	25.43	24.91	5.2	2.92	25.86	24.53	11.6	42.4	1727.59	32.62	2	6.67	187.50	20.52
neos-839859	1	9.425E7	860.77	0.2	1	5.856E7	496.93	0.18	1	5.856E7	496.93	0.17	1	5.856E7	496.93	0.17	1	1.317E8	1242.13	0.18
neos-860300	14.3	7685.3	140.09	3.13	15.9	7321.6	128.73	0.87	21.6	7044.4	120.07	1.16	17.1	6285.7	96.37	0.93	9.7	8861.7	176.84	0.73
neos-886822	2	138398	381.30	0.26	1	28820.5	0.23	0.17	1	28820.5	0.23	0.16	1	28820.5	0.23	0.16	1	178597.5	521.10	0.27
neos-892255	3.6	18.7	33.57	0.15	3.9	18.9	35.00	0.09	8	45.6	225.71	0.18	3.9	20.6	47.14	0.1	11.5	46.3	230.71	0.25
neos-906865	2	9105.2	186.78	0.05	2	10823.9	240.91	0.03	2	10819.7	240.78	0.03	2	11060.3	248.36	0.03	2	9744.1	206.90	0.03
neos-941698	29.8	22.3	1015.00	0.8	48.4	10	400.00	0.55	98.2	10.4	420.00	1.02	64.5	8.3	315.00	0.73	62.3	10.2	410.00	0.79
neos-948268	5	60	0.00	6.36	13.7	60	0.00	12.52	6	60	0.00	6.28	7	60	0.00	6.52	3	60	0.00	5.44
neos-955215	2.2	9037.66	1924.11	0.01	3	809.42	81.28	0.01	3	809.35	81.27	0.01	3	808.92	81.17	0.01	3.4	1029.15	130.49	0.01
neos-1058477	2.8	3.58	550.91	0.02	2	1.47	167.27	0.01	2.8	1.46	165.45	0.01	3.2	11	1900.00	0.02	4.4	31.25	5581.82	0.02
neos-1171448	1	0	100.00	0.6	1	0	100.00	0.26	1	0	100.00	0.26	1	0	100.00	0.26	1	0	100.00	0.28
neos-1200887	1	-38	48.65	0.02	1	-52	29.73	0.02	1	-52	29.73	0.01	1	-52	29.73	0.01	1	-52	29.73	0.02
neos-1211578	1	-51	33.77	0	1	-69	10.39	0	1	-69	10.39	0	1	-69	10.39	0	1	-69	10.39	0
neos-1225589	27.2	2.356E10	1815.07	0.05	43.6	2.524E10	1952.17	0.08	30	2.348E10	1809.29	0.06	51.3	2.716E10	2108.25	0.1	31.4	2.383E10	1837.17	0.06
neos-1228986	1	-92	25.20	0	1	-104	15.45	0	1	-104	15.45	0	1	-104	15.45	0	1	-104	15.45	0
neos-1281048	131.8	173712.9	28803	1.79	243	174774.2	28980	1.9	285.6	183703.9	30466	2.08	167.7	175805.5	29152	1.31	308.4	180675.6	29962	2.25
neos-1337489	1	-51	33.77	0	1	-69	10.39	0	1	-69	10.39	0	1	-69	10.39	0	1	-69	10.39	0
neos-1413153	2	119.12	13.32	0.37	1	119.12	13.32	0.35	1	119.12	13.32	0.35	1	119.12	13.32	0.35	1	119.12	13.32	0.36
neos-1415183	1	425.6	302.53	0.53	1	128.61	21.64	0.43	1	128.61	21.64	0.43	1	128.61	21.64	0.43	1	128.61	21.64	0.44
neos-1437164	23.6	25.9	223.75	0.14	37.3	17.6	120.00	0.19	21.1	18.9	136.25	0.11	19.7	19	137.50	0.1	28.7	19.4	142.50	0.15
neos-1440447	1	-52	48.00	0.01	1	-77	23.00	0.01	1	-79	21.00	0	1	-78	22.00	0	1	-78	22.00	0
neos-1460265	35.7	15925	18.84	0.18	175.1	15410	15.00	1.03	91.7	15490	15.60	0.51	108.2	15520	15.82	0.61	143.2	15520	15.82	0.8
neos-1480121	2	89.33	107.74	0	2	95.8	122.79	0	2	95.8	122.79	0	2	95.8	122.79	0	2	96.6	124.65	0
neos-1489999	5.8	476.9	34.72	0.05	6.9	484.3	36.81	0.05	6.3	481.6	36.05	0.05	6.2	488.3	37.94	0.05	6.2	483.5	36.58	0.05
neos-1516309	9	54363.5	51.20	0.13	11.9	54069	50.38	0.12	12.4	52941	47.25	0.13	10.8	52827	46.93	0.12	17.7	53687.5	49.32	0.16
neos-1595230	3.5	20.4	126.67	0.1	3.8	20.5	127.78	0.07	5	21	133.33	0.07	4.7	22.1	145.56	0.07	3.7	20.5	127.78	0.07
neos-1597104	4.6	-7.1	76.33	8.08	8.2	-2.6	91.33	1.07	8.2	-2.6	91.33	1.05	6	-3.4	88.67	1.11	4.6	-6.9	77.00	0.98
neos-1599274	3	36277.6	13.10	0.17	8.6	52367.76	63.26	0.13	9.2	51694.48	61.16	0.14	9.4	51652.8	61.03	0.14	3	37687.6	17.50	0.07
neos-1620807	8.8	9.5	58.33	0.02	10.6	9.7	61.67	0.02	7	9.1	51.67	0.02	9.2	9.7	61.67	0.02	6.9	9.7	61.67	0.01
prod1	1	0	100.00	0	1	0	100.00	0	1	0	100.00	0	1	0	100.00	0	1	0	100.00	0
qap10	516.8	502.4	47.76	1690.54	1	406	19.41	7.33	1	406	19.41	10.21	1	406	19.41	7.37	1	406	19.41	10.64
roy	38.3	5810.25	81.06	0.03	212	5788.85	80.40	0.08	98.1	5393.15	68.07	0.04	264	5622.95	75.23	0.1	319.8	5878.4	83.19	0.12

Table 24: Comparison on COR@L problems (integer feasible solution found in all the ten runs). FP vs Combined RFP - Part II

Problem	FP			FP+Log			Exp+Log			Logis+Log			Exp+Logis		
	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time
harp2	10	188.80	1.52	10	525.90	4.28	9	-	-	9	-	-	10	257.40	2.19
momentum1	10	474.20	577.99	9	-	-	10	215.80	56.10	9	-	-	10	578.20	118.96
p2756	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
protfold	10	360.20	107.67	9	-	-	10	524.90	89.36	10	361.60	83.31	9	-	-

Table 25: Comparison on MIPLIB problems (integer feasible solution found in less than ten runs). FP vs Combined RFP

Problem	FP			FP+Log			Exp+Log			Logis+Log			Exp+Logis		
	F.s found	Iter	Time	F.s found	Iter	Time	F.s found	Iter	Time	F.s found	Iter	Time	F.s found	Iter	Time
aligninq	10	380.10	6.01	10	623.90	3.68	9	-	-	8	-	-	7	-	-
lrn	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos2	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos3	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-583731	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-593853	1	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-631694	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-709469	4	-	-	3	-	-	2	-	-	4	-	-	7	-	-
neos-791021	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-799711	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-799716	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-803219	0	-	-	0	-	-	0	-	-	1	-	-	1	-	-
neos-803220	5	-	-	8	-	-	10	258.00	0.53	10	273.70	0.60	10	275.60	0.55
neos-806323	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-807639	2	-	-	2	-	-	2	-	-	2	-	-	1	-	-
neos-807705	0	-	-	0	-	-	0	-	-	0	-	-	2	-	-
neos-810326	10	668.10	76.05	8	-	-	10	773.30	109.01	10	366.50	59.75	9	-	-
neos-862348	9	-	-	9	-	-	10	65.50	0.86	10	180.40	2.15	10	62.80	0.81
neos-880324	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-912015	6	-	-	7	-	-	5	-	-	2	-	-	2	-	-
neos-932816	2	-	-	4	-	-	0	-	-	2	-	-	2	-	-
neos-957270	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-957389	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
neos-1215259	7	-	-	8	-	-	1	-	-	5	-	-	5	-	-
neos-1396125	2	-	-	1	-	-	0	-	-	0	-	-	0	-	-
neos-1441553	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-

Table 26: Comparison on COR@L problems (feasible solution found in less than ten runs). FP vs Combined RFP

Problem	OFP				Exp ORFP				Logis ORFP				Exp+Logis ORFP			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time
a1c1s1	119.4	19198.34	67	0.5	33.5	18859.35	64	0.4	50.8	19535.79	70	0.3	50.5	19522.04	70	0.2
aflow30a	0	983.17	15	0.1	0	983.17	15	0	0	983.17	15	0	0	983.17	15	0.1
aflow40b	25.2	3095.8	165	0.3	62.2	4430	279	0.4	75.4	4243.1	263	0.6	35.1	5396.8	362	0.4
air04	21.5	58273.9	4	23.8	20.8	58230.1	4	23.3	18.3	58527.4	4	23	32.6	58250	4	39.1
air05	5	27384	4	2.3	5	27384	4	2.3	11	30127.1	14	7	17.3	29612.4	12	6.5
dano3mip	80.6	821.42	–	275.6	82	836.28	–	282.6	83	777.14	–	273.9	52.6	801.13	–	181.5
fast0507	4	181	4	15.6	4	181	4	15.8	4	199	14	11.3	4	194	11	12
fiber	18	6.339E6	1462	0.2	16.2	4.113E6	913	0.1	17.6	3.989E6	883	0.1	6	1.639E6	304	0
fixnet6	16.2	22614.6	468	0.1	13.8	17547.4	341	0	19.4	18611.2	367	0	11.2	17956.3	351	0.2
markshare1	69	521.1	–	0.1	68.4	438.4	–	0.1	76.4	752.4	–	0	58	561	–	0
markshare2	69.4	1190.4	–	0	71	1203.5	–	0.1	78.5	812.5	–	0.1	57.4	572.6	–	0.1
mas74	112.7	35399.07	200	0.2	110	24219.37	105	0.1	105.6	21041.66	78	0.2	93.8	19578.36	66	0.2
mas76	109	45068.61	13	0.1	110.8	52902.53	32	0.2	103	46238.12	16	0.1	90.2	46604.28	16	0.1
mkc	109.7	-231.1	59	1	111.6	-223.12	60	1.1	40.3	-245.51	56	0.6	46.2	-208.91	63	0.3
mod011	17.5	-4.511E7	17	0.4	19.3	-4.599E7	16	0.4	57.2	-1.547E7	72	0.4	14.5	-4.254E7	22	0.2
modglob	66	2.134E7	3	0.2	62.4	2.138E7	3	0	59.6	2.111E7	2	0.1	41.2	2.132E7	3	0.1
nsrand-ipx	7.3	245888	380	0.6	8	263792	415	0.5	6.8	261920	412	0.5	3.4	242768	374	0.6
nw04	21.6	19283.2	14	4.9	17	21180.2	26	4.3	5.6	19702.4	17	2.6	2	19124	13	2
opt1217	42.9	-12.8	20	0.2	37	-16	0	0.2	53.1	-13.8	14	0.2	23.6	-14	12	0
pk1	57.6	91.2	729	0.1	57.3	126.2	1047	0.1	73	149	1255	0.1	49.2	129	1073	0
pp08aCUTS	10.6	12270	67	0	10.7	12025	64	0.1	22	11050	50	0	8	8879	21	0.1
pp08a	10.8	11428	55	0	10	11020	50	0	23.5	12480	70	0.1	9	9880	34	0
qiu	7.5	167.37	226	0	8	432.57	426	0	18.2	608.86	558	0.2	6.8	160.56	221	0.1
set1ch	25.1	94116.18	73	0.2	27	86727.85	59	0	24.5	89406	64	0.1	11.8	81987.4	50	0
seymour	8	446	5	1.5	8	446	5	1.5	8	454	7	1.4	4	460	9	1.2
sp97ar	7.4	5.399E9	717	2.5	6.9	4.603E9	597	2.6	6	1.099E9	66	2.3	6	1.151E9	74	2.2
tr12-30	29.8	175086.5	34	0	30.6	173209.4	33	0	23.5	195109	49	0.3	15.6	160418.8	23	0.1
vpm2	10	20.5	49	0	12	24.38	77	0	21.9	23.77	73	0.1	10	23	67	0

Table 27: Comparison on MIPLIB problems (integer feasible solution found in all the ten runs). OFP vs ORFP

Problem	OFP				Exp ORFP				Logis ORFP				Exp+Logis ORFP			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time
bc1	8.6	13.49	304	1.5	7.9	9.92	197	1.4	12.8	11.81	253	1.5	4	3.44	3	1.4
bienst1	77.9	61.76	32	0.9	59	57.38	23	0.5	82.4	54.79	17	1.1	47.8	63.76	36	0.5
bienst2	68.9	67.29	23	0.7	64.4	66.42	22	0.7	86.5	73.69	35	1	49	85.28	56	0.5
binkar10-1	1	7935.98	18	0.1	1	7935.98	18	0.1	1	7964.89	18	0.1	2	7976.03	18	0.1
dano3-3	21.7	577.12	0	53.2	26.5	577.21	0	79.3	45.8	577.43	0	104.8	12.1	576.52	0	24.6
dano3-4	36.4	578.53	0	119.2	35.1	578.34	0	108.5	41.5	578.77	0	96.4	16.4	577.25	0	34.8
dano3-5	34.4	579.3	0	104.1	41	579.74	0	128.3	50.1	580.15	1	134.5	21.2	579	0	55.7
mkc1	7	-565.92	7	0.3	10.7	-563.76	7	0.1	7.8	-573.01	6	0.3	6.6	-576.81	5	0
neos5	5	17	13	0	5.9	17	13	0.1	16.7	15.65	4	0.1	6	16	7	0
neos11	11.2	9.9	–	1.6	22.8	10.1	–	2.3	12.2	9.8	–	1.8	5.6	9.6	–	0.7
neos12	57.7	17.8	37	28.1	71.4	17.7	36	31.8	5.8	13	0	6.5	9.6	14.7	13	9.3
neos13	18	-41.04	57	1	17.3	-39.42	59	1.3	16.6	-37.06	61	1.2	5.1	-47.83	50	1.6
neos14	7.2	107870.61	45	0.1	8.8	104252.78	40	0.1	12.9	105913.04	42	0	20.5	101501.98	37	0.1
neos18	15	17.8	11	0.2	18.3	20.1	26	0.1	20.2	18.8	17	0.3	7.4	19.6	23	0.2
neos-476283	25	411.48	1	47.7	24.4	411.97	1	47.2	22.5	413.04	2	48	8.6	416.95	3	34.4
neos-480878	25.8	709.57	44	0.2	13	555.71	13	0.2	28.4	649.13	32	0.2	7.9	549.71	12	0.2
neos-504674	40.9	7155	97	0.1	53.7	10852.61	198	0.3	55.2	7420.4	104	0.2	28	11256.83	210	0.1
neos-504815	43.2	4178	82	0.1	36.9	4471.18	95	0.2	55.8	3960.03	72	0.2	29.2	4379.05	91	0.2
neos-512201	42.6	1316.45	156	0.2	45.2	1223.81	138	0.3	55.8	1359.89	165	0.1	29.1	1408.57	174	0.2
neos-522351	46.6	22851.7	28	0.2	46.8	19611.55	10	0.2	84.8	31042.64	74	0.2	27.7	18589.42	4	0.1
neos-525149	7.3	1460.8	2295	2	10	2261.3	3607	2.1	5.8	1461.9	2297	2	5.5	1860.3	2950	2
neos-547911	10.4	15.6	20	1.8	13.9	15.5	19	1.9	9.9	15.6	20	1.9	6.8	16.2	25	1.6
neos-555694	17.1	24.35	32	0.4	25.7	24.82	35	0.3	17.4	25.73	40	0.2	15.7	30.91	68	0.2
neos-555771	4.2	21.02	13	0.3	4	21.63	16	0.1	2	20.7	11	0.2	2	20.7	11	0.3
neos-565815	6.6	14	0	1	7.6	14	0	1	5.8	14	0	1.1	9.2	14	0	0.9
neos-570431	11	14.8	–	0.2	11	16.4	–	0.2	15.3	15.2	–	0.3	8.8	16	–	0.2
neos-584851	26.6	-4.8	56	0.3	26.5	-7.7	30	0.3	33.6	-8	27	0.5	19.6	-8.9	19	0.2
neos-611838	11	1.777E6	1	0.3	11	1.777E6	1	0.3	42.8	1.875E6	6	0.9	14.5	1.794E6	2	0.4
neos-612125	11	1.854E6	1	0.4	10.5	1.854E6	1	0.3	38	1.889E6	2	0.9	11	1.854E6	1	0.3
neos-612143	11	1.809E6	1	0.2	10.7	1.809E6	1	0.4	43.5	1.862E6	4	0.8	10.3	1.809E6	1	0.3
neos-612162	13.4	1.790E6	1	0.3	11.3	1.786E6	1	0.4	42.1	1.849E6	4	0.8	9.7	1.786E6	1	0.2
neos-655508	0	63015042	0	0.5	0	63015042	0	0.4	0	63015042	0	0.4	0	63015042	0	0.5
neos-777800	24.9	-80	0	14.8	31.5	-80	0	18.8	16.3	-80	0	10.4	20.3	-80	0	8.5
neos-780889	0	3421500	0	13.4	0	3421500	0	13.4	0	3421500	0	13.3	0	3421500	0	13.4
neos-801834	3.8	53732.8	7	0.5	3.8	54664.3	8	0.5	3.6	53770.1	7	0.6	3.6	59861.2	19	0.5

Table 28: Comparison on COR@L problems (integer feasible solution found in all the ten runs). OFP vs ORFP - Part I

Problem	OFP				Exp ORFP				Logis ORFP				Exp+Logis ORFP			
	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time	Iter	Obj	Gap %	Time
neos-820879	14.9	32762	29	1.3	12.9	32141.7	26	1.1	8.6	34295.3	35	1	14.9	37091.7	46	1.8
neos-824695	12.8	32.2	4	3.9	14.4	33.4	8	4	13.3	33.9	9	4.6	5.1	39.8	28	3.2
neos-825075	15.5	-222	18	0.1	13.8	-152	44	0.1	22.9	-146	46	0.2	13.2	178	165	0.2
neos-826250	4	28	0	1	4	28	0	1	7	30.5	9	1.6	5	34	21	1.3
neos-826812	7.4	58.72	1	2	7.8	58.52	1	1.9	5.9	58.12	0	2.1	5	59.11	2	2
neos-827175	2	112	0	2.1	2	112	0	2.1	6.1	112.3	0	2.8	5.8	112.5	0	2.2
neos-829552	1	26.41	1038	13.2	1	26.41	1038	13.4	1	26.74	1053	9.5	1	5.57	140	7
neos-839859	29.3	2.249E7	129	0.3	28.6	2.073E7	111	0.3	23.9	1.989E7	103	0.3	37.8	4.019E7	310	0.3
neos-860300	22.6	6272.4	96	2.2	26.8	6676.7	109	2.4	22.8	5724.5	79	2.2	11.4	6233.5	95	1.5
neos-886822	114.9	54237.05	89	2.9	114.7	54922.6	91	2.9	116.5	54100.65	88	2.9	92.1	50257.85	75	2.5
neos-892255	12.9	15.6	11	0.3	11.6	14.8	6	0.3	10.8	15.2	9	0.4	6	18	29	0.2
neos-906865	8	5820	83	0.1	8.7	10553.4	232	0.1	12.2	8387.8	164	0.1	5.1	12393.6	290	0
neos-932816	6.7	15475.2	1	2.7	8.1	15376	0	3	5.9	15478.9	1	1.8	2	15378	0	2
neos-948268	8.4	60	0	10.7	9.8	60	0	11.5	4	60	0	7.4	14	60	0	12.2
neos-955215	9	1145.56	157	0.1	6	548.44	23	0.1	12.4	1565.62	251	0.1	5	720.62	61	0.1
neos-1058477	59.6	0.99	81	0.2	61.6	1.05	92	0.1	73.6	1	83	0.2	50.1	0.83	52	0.2
neos-1171448	16.4	-296.49	4	4.6	14.9	-296.19	4	4.5	17.2	-292.96	5	5.2	5.6	-294.8	5	2.8
neos-1200887	13.8	-59.2	20	0.1	13.1	-63.4	14	0.2	18.9	-70.2	5	0.1	8.5	-69	7	0.1
neos-1211578	15.2	-70.2	9	0	17	-66	14	0.1	25	-68	12	0	5	-75.2	2	0.1
neos-1225589	53.6	4.219E10	3328	0.2	53.3	4.077E10	3212	0.2	48.8	4.180E10	3296	0.1	53	4.431E10	3499	0.2
neos-1228986	19.6	-101.4	18	0	16	-114	7	0	28.7	-118.4	4	0.1	11.4	-110.4	10	0.1
neos-1337489	16.8	-69	10	0.1	15	-75	3	0	24.9	-71.3	7	0.1	9.6	-71.7	7	0
neos-1413153	69	127.78	22	0.9	67.2	115.29	10	0.6	67.5	117.04	11	0.9	39.4	117.23	12	0.5
neos-1415183	64.5	124.25	18	1	70.8	116.35	10	1	64.2	117.79	11	1	38.5	118.43	12	0.5
neos-1437164	21.8	36.2	–	0	21.4	40.2	–	0.1	21	32.4	–	0.2	19.6	38.6	–	0.3
neos-1440447	16	-78	22	0	16	-82.8	17	0.1	27.2	-92.8	7	0	10.6	-84.4	16	0
neos-1480121	85.4	52	21	0.1	84.3	56.06	30	0.2	86	65.8	53	0.1	72.6	61.8	44	0.1
neos-1489999	9.1	474.6	34	0	10.7	472.3	33	0	12	464.5	31	0.2	5	527	49	0.4
neos-1516309	9.1	42993.5	20	0.2	11.6	45554.6	27	0.3	11	48341.4	34	0.2	11	48720.6	36	0.2
neos-1595230	10.6	10	–	0.3	10.3	10.3	–	0.3	15.3	10	–	0.3	6.6	10.6	–	0.2
neos-1597104	15.4	-14.5	52	9.8	15.4	-13.2	56	9.8	14.5	-20.7	31	8.6	9.7	-10.7	64	5.9
neos-1599274	10.6	43456.18	35	0.3	9.2	39650.26	24	0.3	8	41632.14	30	0.2	6.4	43108.88	34	0.2
neos-1620807	10.8	6	–	0.1	14.8	6.1	–	0.2	20	6	–	0.2	10.4	6.7	–	0
prod1	21.7	-42.1	25	0.1	16	-45.9	18	0	21	-44	21	0.1	17	-43	23	0
qap10	5.6	386	14	27.8	7.1	408.6	20	36	2.2	350.4	3	21.2	2.3	410.8	21	19.4
roy	12.7	4427.89	38	0.1	10	4696.38	46	0	27	4180.05	30	0	9.7	3883.22	21	0.1

Table 29: Comparison on COR@L problems (integer feasible solution found in all the ten runs). OFP vs ORFP - Part II

Problem	**OFP**			**Exp ORFP**			**Logis ORFP**			**Exp+Logis ORFP**		
	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time
10teams	8	-	-	7	-	-	5	-	-	4	-	-
cap6000	10	39.9	1	9	-	-	10	28.8	0.5	10	18.5	0.5
danoint	8	-	-	9	-	-	9	-	-	5	-	-
glass4	1	-	-	3	-	-	3	-	-	9	-	-
harp2	3	-	-	2	-	-	2	-	-	0	-	-
liu	7	-	-	5	-	-	3	-	-	6	-	-
misc07	10	92.5	0.2	7	-	-	6	-	-	7	-	-
momentum1	0	-	-	0	-	-	0	-	-	0	-	-
net12	9	-	-	10	193.3	6	8	-	-	7	-	-
p2756	0	-	-	0	-	-	0	-	-	0	-	-
protfold	5	-	-	4	-	-	3	-	-	6	-	-
swath	7	-	-	7	-	-	6	-	-	10	71.1	3.1
t1717	8	-	-	8	-	-	9	-	-	10	61	186.4

Table 30: Comparison on MIPLIB problems (integer feasible solution found in less than ten runs). OFP vs ORFP

Problem	**OFP**			**Exp ORFP**			**Logis ORFP**			**Exp+Logis ORFP**		
	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time	F.s. found	Iter	Time
22433	7	-	-	9	-	-	9	-	-	10	50	0.1
23588	9	-	-	8	-	-	10	89	0.1	10	110.1	0.2
aligninq	4	-	-	7	-	-	2	-	-	6	-	-
lrn	0	-	-	0	-	-	0	-	-	0	-	-
mcf2	8	-	-	6	-	-	9	-	-	4	-	-
neos2	0	-	-	0	-	-	0	-	-	0	-	-
neos3	0	-	-	0	-	-	0	-	-	0	-	-
neos6	10	119.8	14.4	8	-	-	10	91.6	10.6	10	29.7	2.9
neos17	10	54	0.2	10	54	0.3	10	54	0.2	9	-	-
neos-430149	0	-	-	0	-	-	0	-	-	0	-	-
neos-494568	9	-	-	10	30	0.9	10	45	1.1	10	25.1	0.8
neos-538867	9	-	-	8	-	-	10	30.3	0.1	10	57.9	0.1
neos-593853	0	-	-	0	-	-	0	-	-	0	-	-
neos-598183	2	-	-	2	-	-	5	-	-	8	-	-
neos-603073	2	-	-	9	-	-	10	76	0.3	10	68.1	0.2
neos-631694	0	-	-	0	-	-	0	-	-	0	-	-
neos-709469	0	-	-	0	-	-	3	-	-	0	-	-
neos-775946	9	-	-	10	25.5	0.6	10	26.6	0.7	10	15.4	0.6
neos-791021	0	-	-	0	-	-	0	-	-	0	-	-
neos-799711	1	-	-	0	-	-	0	-	-	0	-	-
neos-799716	0	-	-	0	-	-	0	-	-	0	-	-
neos-803219	10	234	0.4	4	-	-	2	-	-	10	40.3	0.1
neos-803220	9	-	-	9	-	-	10	130	0.1	10	26.9	0.1
neos-806323	0	-	-	3	-	-	0	-	-	0	-	-
neos-807639	10	74.1	0.5	4	-	-	8	-	-	3	-	-
neos-807705	0	-	-	2	-	-	0	-	-	6	-	-
neos-810286	5	-	-	6	-	-	0	-	-	5	-	-
neos-810326	1	-	-	2	-	-	3	-	-	1	-	-
neos-862348	10	20.2	0.3	9	-	-	10	38.5	0.5	9	-	-
neos-880324	0	-	-	0	-	-	0	-	-	0	-	-
neos-912015	0	-	-	1	-	-	0	-	-	0	-	-
neos-941698	10	20.8	0.4	10	22.6	0.3	10	45.3	0.6	8	-	-
neos-957270	2	-	-	2	-	-	2	-	-	0	-	-
neos-957389	0	-	-	1	-	-	0	-	-	0	-	-
neos-1215259	1	-	-	1	-	-	0	-	-	0	-	-
neos-1281048	1	-	-	0	-	-	0	-	-	0	-	-
neos-1396125	1	-	-	0	-	-	0	-	-	0	-	-
neos-1441553	2	-	-	0	-	-	3	-	-	8	-	-
neos-1460265	5	-	-	3	-	-	2	-	-	10	22.3	0.3

Table 31: Comparison on COR@L problems (integer feasible solution found in less than ten runs). OFP vs ORFP