Noname manuscript No.
(will be inserted by the editor)

An Empirical Evaluation of Walk-and-Round Heuristics for Mixed Integer Linear Programs

Kuo-Ling Huang · Sanjay Mehrotra

Received: date / Accepted: date

Abstract Feasibility pump is a general purpose technique for finding feasible solutions of mixed integer programs. In this paper we report our computational experience on using geometric random walks and a random ray approach to provide good points for the feasibility pump. Computational results on MIPLIB2003 and COR@L test libraries show that the walk-and-round approach improves the upper bounds of a large number of test problems when compared to running the feasibility pump either at the optimal solution or the analytic center of the continuous relaxation. In our experiments the hit-and-run walk (a specific type of random walk strategy) started from near the analytic center is generally better than other random search approaches, when short walks are used. The performance may be improved by expanding the feasible region before walking. Although the upper bound produced in the geometric random walk approach are generally inferior than the best available upper bounds for the test problems, we managed to prove optimality of three test problems which were considered unsolved in the COR@L benchmark library (though the COR@L bounds available to us seem to be out of date).

Keywords Mixed Integer Programs · Heuristics · Hit-and-Run Random Walk · Dikin Random Walk · Geometric Random Walk

1 Introduction

Geometric random walks (henceforth called random walks) have been developed for over 25 years for sampling nearly uniformly from a polyhedra (See Vempala [36] for a survey). Random walks have been proposed to approximately compute the volume

The research of both authors was partially supported by grant N00014-09-10518 and NSF-CMMI-05227650.

S. Mehrotra
Dept. of Industrial Engineering and Management Sciences
Northwestern University, Evanston, IL, 60208
E-mail: mehrotra@iems.northwestern.edu

of a convex body [27], for counting the number of integer solutions in a polyhedra [24], and solving optimization problems [15, 23, 40]. Despite considerable theoretical development, very little is known about their practical performance. This paper empirically studies the use of random walks combined with a feasibility pump (FP) [18] heuristic towards generating feasible, possibly optimum, solutions of mixed integer linear programs (MILPs) arising in practice.

A random walk is a Markov chain. Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a full dimensional polyhedra defined by m inequalities. A random walk starts at some point $x^0 \in \mathcal{P}$, and according to a random procedure moves to a new point $x^1 \in \mathcal{P}$. This random move depends on the current point only. The procedure is repeated at the new point. Several methods for generating random walks are known. Among these are the grid walk, the ball walk, the hit-and-run walk, and the Dikin walk (see [23, 27, 36]). The hit-and-run walk was introduced by Smith [34]. In these random walks the point generated after a sufficient number of steps is shown to follow a nearly uniform stationary distribution on the polyhedra. The number of walk steps required to achieve this stationary distribution is called the mixing time of the walk. For theoretical result on polynomial time mixing of various random walks see Vempala [36], Lovász and Vempala [28], and references therein. More recently, Kannan and Narayanan [23] developed a new walk using Dikin ellipsoids. Kannan and Narayanan [23] showed that when started from the analytic center, the Dikin walk mixes in strongly polynomial time, i.e., the number of Dikin walk steps required to draw a nearly uniform sample from $\mathcal{P}$ is strongly polynomially bounded.

Zabinsky *et al.* [40] gave an improving hit-and-run random search algorithm for optimization problems that uses the hit-and-run walk while making sure that only points with improved objective value are accepted. They showed that for positive definite unconstrained quadratic programs the expected number of function evaluations needed to approximate the optimal solution is at most $O(n^{5/2})$ where n is the dimension of the quadratic function. Polynomial time methods for solving constrained convex programming and linear programming problems are given in [15, 23]. Kannan and Narayanan [23] gave an algorithm using a single Dikin walk to solve linear programs in polynomial time with high probability. This was extended to convex optimization problems in Narayanan [31]. Kannan and Vempala [24] showed that if $\mathcal{P}$ contains a ball of radius $\Omega(m\sqrt{\ln(n)})$, then there is a constant c such that $c\,|\mathcal{P} \cap \mathbb{Z}^n| \leq vol(\mathcal{P}) \leq \frac{1}{c}\,|\mathcal{P} \cap \mathbb{Z}^n|$, where $|\cdot|$ denotes the cardinality of a set, and $vol(\cdot)$ represents the volume of a set. This result is based on uniformly sampling on $\mathcal{P}$ and subsequently using a randomized rounding procedure. Solis and Wets [35] showed global convergence to a neighborhood of the optimum solution of random search methods for global optimization problems. Belisle [13] gave convergence results for simulated annealing based random search methods for global optimization under the assumption that a neighborhood of an optimum solution with positive Lebesgue measure exists. For a survey on the use of random search methods in optimization see Zabinsky [39].

This paper intends to test the usefulness of random walks in the context of generating starting points for the feasibility pump (FP) heuristic to find solutions of MILPs arising in practice. FP is a general heuristic that can be adapted to find feasible solutions of mixed integer linear as well as nonlinear problems. This heuristic was first

proposed by Fischetti et al. [18]. Their computational results show that the FP is remarkably powerful in generating feasible solutions for mixed binary linear problems (0-1 MILPs). This lead to several subsequent papers. Bertacco et al. [14] extended FP to handle general MILPs by using the FP information to perform a subsequent enumeration phase. Achterberg and Berthold [5] slightly modified FP in order to find better feasible solutions. Fischetti and Salvagnin implemented the FP with more clever rounding heuristics [20]. Baena and Castro [7] extended the FP such that the integer point is obtained by rounding a point on the (feasible) segment between the computed feasible point and the analytic center for the relaxed LP.

Practical MILPs have inequality as well as equality constraints while all the random walk methods mentioned above are described on full dimensional sets described using inequality constraints. Generating feasible solutions for a continuous relaxation with equality constraints requires some modifications to random walks described on full dimensional feasible sets. Furthermore, it is not practical to run the random walk for a large number of steps, particularly the number of steps suggested by the analysis for achieving a stationary uniform distribution. With this in mind, we take an empirical approach to test the practical utility of using random walks combined with FP as a heuristic for generating feasible, and possibly optimum solutions of MILPs.

In our empirical study we generate sample points for our test problem with equality constraints by computing appropriate projections. Having sampled a continuous point, we attempt to round this point to an integer solution using the FP heuristic. To understand the value of a random walk, we also implement a simple "random ray" procedure for generating points randomly in a polyhedra and combine it with the FP. Once a feasible solution is found, we add an objective cutoff constraint to cut away this feasible solution before restarting the walk-and-round procedure.

We find that for a large number of test problems an improvement in the upper bound is possible when compared to running the FP either at the optimal solution, or the analytic center of the continuous relaxation. In our experience the upper bound produced from the walk-and-round procedure are generally inferior than the best available upper bounds. However, we managed to prove optimality of three test problems in `COR@L` library whose status is reported unknown. We also find that the performance of the hit-and-run walk started from near the analytic center is generally better than other random search approaches, when short walks are used.

The rest of this paper is organized as follows. In Section 2 we outline the hit-and-run walk and the Dikin walk for generating random points in a polytope. In Section 3 we outline the walk-and-round methods for practical MILP and the feasibility pump heuristic used in our experiments. In Section 4 we provide additional implementation details. Computational results are presented and discussed in Section 5. Here the random walk performances are compared on a set of general MILPs taken from the `MIPLIB2003` [4] test set and from the `COR@L` [2] test set. Finally we conclude in the last section.

2 Geometric Random Walk

In this section we describe the hit-and-run walk and the Dikin walk on a full dimensional polytope.

2.1 Hit-and-Run Walk

Let $\mathcal{P}_I := \{x \mid x \geq 0, Ax \leq b\} \subseteq \mathbb{R}^n$ be a full dimensional polytope, and x^0 be an initial interior point of $\mathcal{P}_I$. The hit-and-run walk is illustrated in Algorithm 1 [34, 36].

Algorithm 1 *Hit-and-Run Walk*

Input: A starting point $x^0 \in \mathcal{P}_I$ and the maximum number of walk steps $k_{\max}$.
Output: Random points $x^k \in \mathcal{P}_I, k = 1, \ldots, k_{\max}$.

1: Set $k := 0$.
2: **while** $k < k_{\max}$ **do**
3: Generate a random direction p uniformly distributed over a direction set in $\mathbb{R}^n$.
4: Find the line segment $\ell := \left\{x \mid x^k + \lambda p, \lambda \in \mathbb{R}\right\} \bigcap \mathcal{P}_I$.
5: Uniformly pick a random point x^{k+1} from the line segment ℓ.
6: Set $k := k + 1$.
7: **end while**

The direction p in Step 3 of Algorithm 1 may be generated by taking each element of p from the standard normal distribution. The vector $p/\|p\|$ is uniformly distributed over the unit sphere [25]. For Step 4, find an interval $[\lambda_{\min}, \lambda_{\max}]$ so that $x + \lambda p \in \mathcal{P}_I$ for $\lambda \in [\lambda_{\min}, \lambda_{\max}]$. Step 5 generates a random λ uniformly distributed on $[\lambda_{\min}, \lambda_{\max}]$ and take $x^{k+1} = x^k + \lambda p$. Note that the hit-and-run walk produces an approximately uniformly distributed sample point in polynomial time [34, 36].

2.2 The Dikin Walk

Recently Kannan and Narayanan [23] proposed a new walk using Dikin ellipsoids. We call this walk a Dikin walk. The mixing time for the Dikin walk is strongly polynomial (see Theorem 1 in Kannan and Narayanan [23]), when started from a "central" point, such as the log-barrier analytic center. The log-barrier analytic center of $\mathcal{P}_I$ is given by the solution of

$$\min B_I(x) := -\sum_{i=1}^{m} \ln (b_i - a_i^T x) - \sum_{j=1}^{n} \ln x_j.$$

The log-barrier analytic center is well-defined for $\mathcal{P}_I$. The Hessian of the log-barrier at a point x is given by

$$\nabla^2 B_I(x) = A^T (B - AX)^{-2} A + X^{-2},$$

where $(B - AX)^{-2} = diag\left\{(b_1 - a_1^T x)^{-2}, (b_2 - a_2^T x)^{-2}, \ldots, (b_m - a_m^T x)^{-2}\right\}$ and $X^{-2} = diag\left\{x_1^{-2}, x_2^{-2}, \ldots, x_n^{-2}\right\}$. The Dikin ellipsoid of radius r centered at $x^c \in \mathcal{P}_I$ is defined as

$$\mathcal{E}_I(x^c, \nabla^2 B(x^c); r) := \left\{x \mid \|x - x^c\|_{\nabla^2 B(x^c)} \leq r\right\}.$$

Let x^0 be the log-barrier analytic center for $\mathcal{P}_I$. The Dikin walk is illustrated in Algorithm 2.

Algorithm 2 *Dikin Walk*

Input: A starting point $x^0 \in \mathcal{P}_I$ and the maximum number of walk steps $k_{\max}$.
Output: Random points $x^k \in \mathcal{P}_I, k = 1, \ldots, k_{\max}$.

Set $k := 0$.
while $k < k_{\max}$ **do**
  Pick a random point z in $\mathcal{E}_I(x^k, \nabla^2 B_I(x^k); 3/40)$.
  if $x^k \notin \mathcal{E}_I(z, \nabla^2 B_I(z); 3/40)$ **then**
    Set $x^{k+1} := x^k$.
  else
    Uniformly pick a random number u from $(0, 1]$.
    if $u \leq \min\left(1, \sqrt{\frac{\det(\nabla^2 B_I(z))}{\det(\nabla^2 B_I(x^k))}}\right)$ **then**
      Set $x^{k+1} := z$.
    else
      Set $x^{k+1} := x^k$ and $k := k + 1$.
    end if
  end if
end while

Kannan and Narayanan [23] showed that if Dikin walk is started at the log-barrier analytic center, it mixes in strongly polynomial time, i.e., the number of Dikin walk steps required to draw a nearly uniform sample from $\mathcal{P}_I$ is strongly polynomially bounded. It is worth noting that warm starting from a central point is also desirable for the hit-and-run walk [26, 36].

3 Walk-and-Round Methods for Practical MILPs

We now present random rounding methods that we implemented for generating feasible solutions of general MILPs in our computational testing. First a modification of random walks is given for problems with equality constraints. Next it is proposed to round the randomly sampled points using the FP heuristic. The first modification is needed because transforming problems with equality constraints to one with inequality constraints only (e.g., through reduced space reformulations) is expensive, and it destroys the sparsity of the original problem data. The second modification is motivated from the fact that in the presence of equality constraints, simple rounding is ineffective.

3.1 Projected Hit-and-Run Walk

Let $\mathcal{P} \coloneqq \{x \mid \mathcal{P}_I \bigcap \mathcal{P}_E\}$, where $\mathcal{P}_I \coloneqq \{x \mid A_I x \leq b_I, x \geq 0\}$ and $\mathcal{P}_E \coloneqq \{x \mid A_E x = b_E\}$. We will assume that A_E has full row rank. In the following version of the hit-and-run walk we first compute a random direction d in $\mathbb{R}^n$ as in Step 3 of Algorithm 1. Next we orthogonally project d onto the affine space given by the equality constraints, yielding direction p. Finally, we compute a random point along $\{x + \lambda p\} \bigcap \mathcal{P}$. This scheme is illustrated in Algorithm 3.

Algorithm 3 *Projected Hit-and-Run Walk*

Input: A starting point $x^0 \in \mathcal{P}$ and the maximum number of walk steps $k_{\max}$.
Output: Random points $x^k \in \mathcal{P}, k = 1, \ldots, k_{\max}$.

1: Set $k \coloneqq 0$.
2: **while** $k < k_{\max}$ **do**
3: Pick a uniform random direction d in $\mathcal{P}_I$.
4: **if** $\mathcal{P}_E \neq \{\phi\}$ **then**
5: Compute p by solving $\min_{A_E p = 0} \|p - d\|^2$.
6: **else**
7: Set $p \coloneqq d$.
8: **end if**
9: Find the line $\ell = \{x \mid x + \lambda p, \lambda \in \mathbb{R}\} \bigcap \mathcal{P}$.
10: Uniformly pick a random point x^{k+1} from the line segment ℓ.
11: Set $k \coloneqq k + 1$.
12: **end while**

The computations in Steps 3, 4, and 9 of Algorithm 3 are performed as Steps 3–5 in Algorithm 1. The direction p in Step 5 is computed from $p = [I - A_E^T (A_E A_E^T)^{-1} A_E] d$.

3.2 Modified Dikin Walk

We modify Dikin walk by ignoring its lazy (Metropolis) steps. This modification is made because the computation of determinants as required in Step 4 of Algorithm 2 is not practical when equality constraints are present. Furthermore, it is convenient (and potentially computationally efficient) to compute the analytic center and Dikin ellipsoids by converting the entire problem to one with equality constraints only. Hence, for the Dikin walk we consider the polytope in the form $\mathcal{P} \coloneqq \{x \mid A_I x + w = b_I, A_E x = b_E, (x, w) \geq 0\} \subseteq \mathbb{R}^n$ where w represents the vector of slacks for the constraint $A_I x \leq b_I$. To simplify the presentation, we use $\bar{x} \coloneqq \begin{pmatrix} x \\ w \end{pmatrix}$, $b \coloneqq \begin{pmatrix} b_I \\ b_E \end{pmatrix}$, and $A \coloneqq \begin{bmatrix} A_I & I \\ A_E & 0 \end{bmatrix}$. Hence $\mathcal{P} = \{x \mid A\bar{x} = b, \bar{x} \geq 0\}$, where $\bar{x} \in \mathbb{R}^{\bar{n}}$ and $\bar{n} = n + m$.

The log-barrier analytic center of $\mathcal{P}$ is defined as a solution of

$$\min \left\{ B(x) := -\sum_{j=1}^{\bar{n}} \ln \bar{x}_j \mid A\bar{x} = b, \bar{x} \geq 0 \right\}.$$

The log-barrier analytic center is well-defined for $\mathcal{P}$ if it is bounded with a non-empty relative interior. Let $\bar{X} = [diag(\bar{x}_j)_{j=1,\ldots,m+n}]$ be a diagonal matrix whose diagonal elements are $\bar{x}_j$, and $e = (1, 1, \ldots, 1)^T$. The gradient and Hessian of the log-barrier at a point x are given by

$$\nabla B(x) = -\bar{X}^{-1}e, \ \nabla^2 B(x) = \bar{X}^{-2}, \text{ where } w = b_I - A_I x.$$

The Dikin ellipsoid of radius r centered at $x^c \in \mathcal{P}$ is the ellipsoid containing all points x such that

$$\mathcal{E}(x^c, \nabla^2 B(x^c); r) := \left\{ x \mid \left\| p^T \nabla^2 B(x^c) p \right\| \leq r, A\bar{x} = b, \bar{x} \geq 0 \right\},$$

where $p := \begin{pmatrix} x - x^c \\ w - w^c \end{pmatrix}$ and $w^c := b_I - A_I x^c$. The modified Dikin walk is illustrated in Algorithm 4.

Algorithm 4 *Modified Dikin Walk*

Input: A starting point $x^0 \in \mathcal{P}$ and the maximum number of walk steps $k_{\max}$.
Output: Random points $x^k \in \mathcal{P}, k = 1, \ldots, k_{\max}$.

1: Set $k := 0$.
2: **while** $k < k_{\max}$ **do**
3: Generate a random direction d uniformly distributed over a direction set in $\mathbb{R}^{n+m}$.
4: Compute p by solving

$$\min \ p^T d \tag{1}$$
$$\text{s.t. } Ap = 0, \tag{2}$$
$$\|p^T \nabla^2 B(x^k) p\| \leq r. \tag{3}$$

5: Set $x_j^{k+1} := x_j^k + \lambda p_j, j = 1, \ldots, n$, where λ is specified by one of the following strategies.
6: **Strategy I** (Constant Radius Dikin Ellipsoid): Take r to be a constant (we used $r = 0.95$) and λ uniformly in $(0, 0.95]$.
7: **Strategy II** (Random Step Strategy): Take any $r \neq 0$, find the line $\ell = \{x \mid x + \lambda p, \lambda \in \mathbb{R}\} \bigcap \mathcal{P}$ and generate x^{k+1} uniformly over ℓ.
8: Set $k := k + 1$.
9: **end while**

Let y be the Lagrange multiplier vector for the equality constraint $Ap = 0$. The direction p in Step 4 of Algorithm 4 satisfies

$$A^T y + d = (\bar{X}^k)^{-2} p, \tag{4}$$

in addition to the constraints in (3). The linear equations (2) and (4) can be solved either using an augmented system approach or using a Cholesky decomposition of the Normal equations obtained after block eliminating variables y by pre-multiplying (4) by $(\bar{X}^k)^2$ and subsequently using (2) to block eliminate y [21].

In particular, an explicit solution of the projected direction p is given by

$$p = \frac{(\nabla^2 B(x^k))^{-1/2} p(x)}{r\|p(x)\|}, \tag{5}$$

where

$$p(x) = \left[I - A^T (A(\nabla^2 B(x^k))^{-1}) A^T\right] A(\nabla^2 B(x^k))^{-1/2} d.$$

3.3 Simple Random Ray Sampling

Since Dikin ellipsoid centered at the log-barrier analytic center provides a reasonably good approximation for $\mathcal{P}$ [32], another alternative for generating random points is to simply shoot random rays from the analytic center uniformly distributed on the ellipsoid. In other words, we just perform the steps of Algorithm 4 (Strategy I) repeatedly without updating the solution, i.e., in Step 5 of Algorithm 4, we set $x_j^{k+1} := x_j^0 + \lambda p_j, j = 1, \ldots, n$, where λ is specified by **Strategy I**. We test this random ray sampling approach to evaluate the relevance of random walks in our context.

3.4 Combining Walks with a Rounding Heuristic

For a given feasible set $\mathcal{P}$ as defined in Section 3.1, our goal is to find a feasible solution of an MILP of the form

$$\min \left\{ c^T x \mid x \in \mathcal{P}, x_i \in \mathbb{Z}, \forall i \in \mathcal{I} \right\}.$$

Note that $|\mathcal{I}| \leq n$.

The basic scheme of FP can be sketched as follows. The method starts from a given point $x \in \mathcal{P}$ and searches for a point x^* that is heuristically as close as possible to a rounded integer solution $\tilde{x}$ of x by solving an l_1-norm distance minimization problem. For a given x^k let $\hat{x}^k$ be such that $\hat{x}_i^k := \lfloor x_i^k \rceil$ for $i \in \mathcal{I}$, and $\hat{x}_i^k := x_i^k \; i \notin \mathcal{I}$. Also let

$$f^k(x) := \sum_{i \in \mathcal{I}} \left| x_i - \hat{x}_i^k \right|.$$

Now consider the problem

$$\min_{x \in \mathcal{P}} f^k(x). \tag{6}$$

Let $\hat{x}^{k*}$ be the optimum solution of (6). If $f^k(\hat{x}^{k*}) = 0$, we have a feasible solution of the MILP, else take $x^{k+1} = \hat{x}^{k*}$, and $k = k + 1$. The algorithm iterates until a maximum iteration or some other set criterion is reached. Also, some safeguards are implemented to avoid stalling and cycling problems.

FP is quite effective in finding a feasible solution of 0-1 MILP; however, for general MILP the performance of the basic heuristic is not as good. Bertacco et al. [14] developed a modified FP scheme, which uses the FP information to perform a subsequent enumeration phase. The quality of the generated solution is usually poor since FP does not consider the objective function. To overcome this drawback, an artificial objective function is constructed while solving the l_1-norm distance minimization problem in [5] with the hope that FP not only converges to a feasible solution but concentrates the search on the region of high-quality points.

4 Implementation

We now present additional details about the implemented walk-and-round heuristics. These heuristics are implemented in a software package referred to as `iOptimize`, which is developed in C++. Details of continuous optimization algorithmic implementation are in Mehrotra and Huang [30]. Here we provide relevant information for completeness.

4.1 LP Optimization

For solving the linear relaxation using simplex method, `COIN-OR` library `Clp` [1] was used. To find the interior optimum solution, we used `iOptimize` linear solver with Mehrotra predictor-corrector [29] and two Gondzio higher-order correctors [22] variant of the homogeneous self-dual (HSD) IPM [30, 37]. The implementation is equipped with a refined potential function that ensures global convergence [30]. We terminate the method when the following termination criterion [6] is satisfied:

$$\frac{|c^T x - b^T y|}{\tau + |b^T y|} \leq 10^{-8}, \tag{7}$$

$$\frac{\|Ax - b\tau\|}{\tau + \|x\|} \leq 10^{-8}, \tag{8}$$

$$\frac{\|A^T y + s - c\tau\|}{\tau + \|s\|} \leq 10^{-8}, \tag{9}$$

where τ represents an homogeneous variable used in the HSD reformulation [30, 37], y represents the Lagrange multiplier vector for the constraint $Ax = b$, and s is the vector of slacks for the dual constraints $A^T y \leq c$. Equation (7) is the current duality gap and equations (8) and (9) respectively represents the current primal and dual infeasibilities.

4.2 Analytic Center Computation

The results reported here were obtained by using a two-phase approach in `iOptimize` for computing the analytic center. Phase-I problem was solved as an LP except with a null-objective function and artificial upper bounds for the primal variables that ensure boundedness of the feasible region. The solution from Phase-I was preprocessed to remove the redundant variables by forcing variables to their bounds, if the slack to the bound is less than 10^{-8}. For phase-II analytic center computation problem, a feasible primal-dual potential-reduction method [38] was used. Newton iterations were used on the perturbed-KKT conditions of the log-barrier problem. The method is equipped with a potential function and is iterated with a controllable step-size chosen by golden-section line-search method or by the theoretical step-size [30]. Phase-II uses the termination criterion $\|Xs - e\|_\infty \leq 10^{-1}$. It is also ensured that the primal and dual solutions satisfy the infeasibility criterion in (8)–(9).

We note that both phases are solved by an interior point method equipped with a potential function, as a result the global polynomial convergence of this two-phase approach is also ensured.

4.3 Objective Cutoff Constraint

Once a walk-and-round procedure yields a feasible solution, we take a sliding objective approach to find a better solution. That is, we add an artificial objective cutoff constraint

$$c^T x \leq c^T \hat{x}^* - \epsilon,$$

where $\hat{x}^*$ is the best feasible integer solution found thus far. We take $\epsilon = 1$ for the pure integer problems (assuming c are all integers); otherwise we take $\epsilon = 10^{-1}$.

We note that Phase-I of the analytic center computation is executed only once in the beginning. A new interior feasible point can be obtained easily on the line connecting the LP-relaxation optimum solution with the current analytic center as follows. Let x^* be the LP relaxation optimum solution, and let x_a be the analytic center of $\mathcal{P}$. The starting interior solution for Phase-II for the set $\left\{x \mid \mathcal{P} \cap \left\{x \mid c^T x \leq c^T \hat{x}^* - \epsilon\right\}\right\}$ is chosen as $x_a + \lambda(x^* - x_a)$, where $\lambda = 1 - \frac{c^T \hat{x}^* - c^T x^*}{2(c^T \hat{x}_a - c^T x^*)}$ is used in all our computations.

4.4 Feasibility Pump Implementation

The feasibility pump implementation in `COIN-OR Branch-and-Cut (CBC)` version 2.3 [1] is used for all feasibility pump computations. This is referred as `CBC-FP` (or `FP`) in the future. In these computations we disabled the `preprocess` and `initialSolve` routines in `CBC` to enhance the computation speed.

We used all `CBC-FP` default settings except we set the iteration limit (the number of times l_1-norm minimization problem is solved) to 500 for LP optimum solution, and 75 for all random walk points, including the analytic center. By default `CBC-FP` uses `maximumPasses` = 100 (maximum number of LPs solved) to terminate

CBC−FP. Once a feasible integer solution is found in CBC−FP, the feasibility pump can be restarted for the new region using the sliding objective approach. CBC−FP has such a provision, with some additional heuristics, and it can be invoked by setting the parameter `maximumRetries`. We set `maximumRetries = 1` when performing random walk runs. However, results for other choices of this parameter value are discussed when comparing results at the LP optimum solutions and the analytic center.

4.5 Computational Environment

All computations were performed on a 3.2 GHz Intel Core 2 Duo CPU with 4GB RAM. No parallelization is done in our current implementation. Our problem test bed is made up of 28 small and medium size instances taken from MIPLIB2003 library [4] and 74 small and medium size instances taken from COR@L library [2]. These problems were chosen because of the need to solve multiple instances of the random walks to perform statistical analysis.

In our computational runs (where all problems are included) we perform at most 50 random walk steps in a particular walk. The maximum number of walk steps $k_{\max}$ (over all the sliding objective runs) is 500. These limits are arbitrary, but allowed us to perform computations in a reasonable amount of time. We will, however, discuss results obtained from using long-walks on a few problems.

To provide a clearer summary of the results, we use performance profiles [17]. Consider a set A of n_a algorithms, a set P of n_p problems, and a performance measure $m_{a,p}$, e.g., computation time or a objective value. 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

$$r_{p,a} = \frac{m_{p,a}}{\min\left\{m_{p,a} \mid a \in A\right\}}.$$

We therefore obtain an overall assessment of the performance of the algorithm by defining the following value

$$\rho_a(\tau) = \frac{1}{n_p} cardinality\left\{p \in P \mid r_{p,a} \leq \tau\right\}.$$

This represents the probability for algorithm a that the performance ratio $r_{p,a}$ is within a factor τ of the best possible ratio. The function $\rho_a(\cdot)$ represents the distribution function for the performance ratio.

5 Computational Results

We first compare the performance of FP at the LP optimum (vertex and interior point), and the analytic center. We then compare the performance of the hit-and-run walk (HR), Dikin Walk with Strategies I and II (DW1, DW2), and Random Ray (RR) methods. All methods are equipped with the same FP. The default random seed (1) is

used for problems where the results are reported based on a single run of the walk-and-round heuristic. Multiple different random seeds (1 – 30) are used for the results where performance over multiple (30) runs for the same problem are reported. On a few occasions the runs crashed because of the numerical instability in the computations. In these cases the problems were rerun using a different starting seed (30+ the old seed).

Table 1 reports the table number (table #) and its goal in the experiment (Goal), the parameters (Parameter) and the setting values (Value). Table 2 summarizes the FP performances. Tables 3–7 summarize the average performances of walk-and-round methods. Detailed results are given in Tables 8, 11–17. Tables 9–17 are only available in the online supplement.

Table 9 gives the instance names, corresponding objective value (Obj), the number of rows (Rows), columns (Cols), non-zeros, continuous variables (Conti), binary variables (Bin), and integer variables (Int) for `MIPLIB2003` test set. Problem type (Type) is also reported in this table. Out of the 28 `MIPLIB2003` problems four are pure binary, 18 are mixed binary, and six are mixed integer problems. Table 10 gives the profiles of the selected MILP instances from the `COR@L` test set. Out of the 74 `COR@L` problems 11 are pure binary, 62 are mixed binary, and only one instance is a general mixed integer problem. Note that instances `neos-582605`, `neos-848150`, `neos-881765`, `neos-905856`, `neos-1056905`, `neos-1121679`, and `neos-1616732` in `COR@L` are reported to have undetermined lower or upper bound.

The best available upper bound on the objective value is reported in (Column "Obj"). The total computational times are reported in Column "Time" in seconds. The optimality "Gap" is calculated as

$$\text{Gap} := 100\% \times \frac{Obj - Obj^*}{|Obj^*|},$$

where Obj^* is the optimum or best known solution value of the instance. If the problem is unsolved, we take the best lower bound as Obj^*. An unknown bound is indicated by "?".

5.1 Feasibility Pump at LP-Optimum and Analytic Center Solutions

The optimum solutions of LP relaxation are often good candidates for generating an integer solution, so this is an obvious starting choice for FP. In the following we compare FP performance at a vertex versus an interior optimum solution. We choose the following parameter setting: `maximumPasses` = 10000 and `maximumRetries` = 30, where parameter `maximumPasses` means the maximum number of LPs solved in FP, and parameter `maximumRetries` means the maximum number of LP objectives found by FP. These numbers are chosen to allow us to compare FP at optimum solutions only with walk-and-round approaches.

Table 2 summarizes the FP performance. Computational results provided in this table are the optimality gap (Column "Gap"), the arithmetic computational times (Column "Time") in seconds, and the number of problems for which a method gives

the best upper bound (Column "# Best"). Note that the count given in Column "# Best" does not include the cases when all methods give the same upper bound. The arithmetical average CPU times are used to make sure that the FP and the walk-and-round methods performed with similar computational efforts so that we can make a fair comparison regarding their best objective values. Note that the instances for which one of the method can not find a feasible solution are excluded in the arithmetic mean calculation.

Tables 11 and 12 report FP performance at the LP-optimum vertex solutions (LPS-FP), at the LP-optimum interior solutions (LPI-FP), and at the analytic center solutions (AC-FP) on the `MIPLIB2003` and `COR@L` test problems, respectively. Computational results provided in these tables are the best available upper bound on the objective value (Column "Obj"), the optimality gap (Column "Gap"), and the total computational times (Column "Time") in seconds. The average performance of each method is also given in these tables. "Avg" in the last row of Table 11 provides the average computational performances.

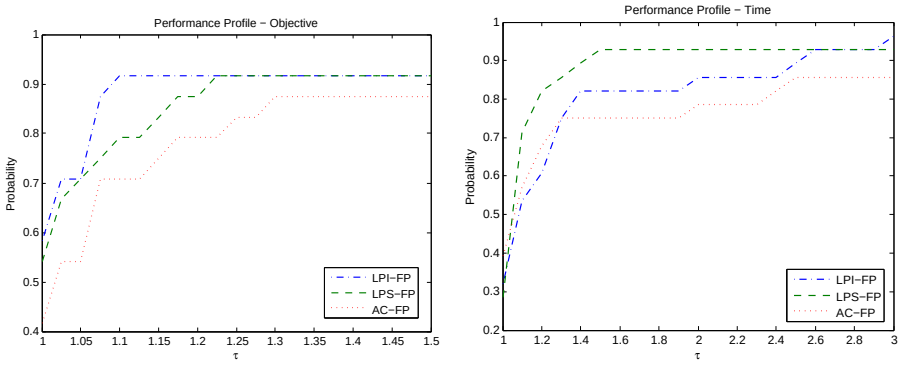
First, we compare LPI-FP, LPS-FP, and AC-FP on the `MIPLIB2003` test set. Considering Avg in these experiments, the mean-gap of LPS-FP (Avg = 38.44%) and LPI-FP (Avg = 35.4%) are comparable to each other, and both are better than that for AC-FP (Avg = 48.41%). In these experiments LPS-FP ranked first in 11 cases, LPI-FP ranked first in 10 cases, and and AC-FP ranked first in only six cases. In six cases no method is able to find a feasible solution. In one case all three methods provide the same upper bound.

LPI-FP, LPS-FP, and AC-FP could obtain at least one feasible solution in 20 out of the 22 binary `MIPLIB2003` instances. No feasible solution was found for instances `manna81` and `p2576`. A feasible solution of four out of the six mixed integer instances was not found by LPS-FP, LPI-FP, and AC-FP. This is worse than the results reported in FP [18], where integer solutions for all test instances are reported. This is because `CBC-FP` does not fully implement the FP algorithm described in [14, 18]. For example, in the enumeration stage of FP, `CBC-FP` uses a small branch-and-bound tree to resolve the sub-MILPs whereas the results reported in [14] uses commercial solvers like `Cplex` [3].

Figure 1 shows the performance profiles for `MIPLIB2003` test set comparing the best objective value and the CPU time performances of LPS-FP, LPI-FP, and AC-FP. One can observe that AC-FP performs the worst in terms of its CPU time performance.

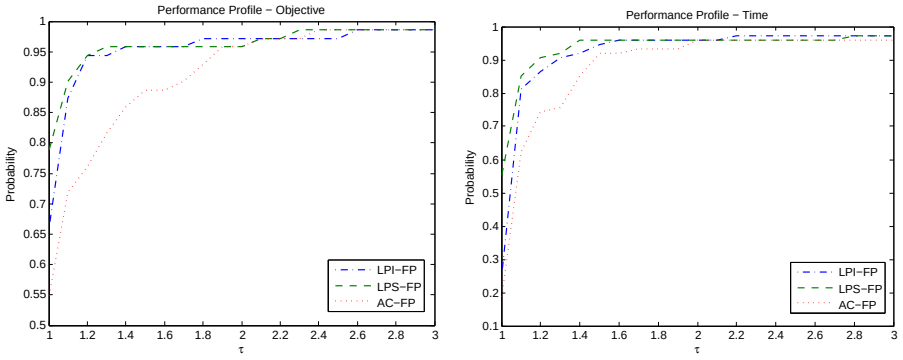
We note that in terms of the computation times, AC-FP (Avg = 111.27 sec.) uses less average computation times than LPI-FP (Avg = 119.66 sec.) and LPS-FP (Avg = 117.91 sec.). Recall that the computation of analytic center uses a two-phase approach, which may require additional interior point iterations (results not reported here). Hence, one may expect AC-FP to have worse CPU time performance than LPI-FP and LPS-FP. We think this is because AC-FP performs fewer FP iterations, but unfortunately we are unable to verify this because the number of FP iterations is not available to us in the `CBC` callable library.

Next, we compare results using `COR@L` test set. In these experiments, LPS-FP ranked first (Avg = 61.71%), LPI-FP ranked second (Avg = 70.24%), and AC-FP ranked last (Avg = 93.81%). Comparing the performance LPI-FP ranked first in 25



(a) Objective performance (b) CPU time performance

Fig. 1: Performance profiles of FP with different starting solutions on `MIPLIB2003` test set



(a) Objective performance (b) CPU time performance

Fig. 2: Performance profiles of FP with different starting solutions on `COR@L` test set

cases, LPS-FP ranked first in 34 cases, and AC-FP ranked first in nine cases. In 13 cases all three methods provided the same upper bound. No method could find a feasible solution of problem `neos-905856`. Although LP-AC generates the worst average performance, it is the only method that could find a feasible solution of problem `neos-1420205`.

Figure 2 shows the performance profiles for `COR@L` test set comparing the best objective value and the CPU time performances of LPS-FP, LPI-FP, and AC-FP. Figure 2(b) suggests that AC-FP performs the worst in terms of its CPU time performance, when similar computational efforts (LPI-FP = 95.4 sec. LPS-FP = 94.67, and AC-FP = 96.17 sec.) are used to solve the problems.

5.2 Computational Comparison of Random Walks on MIPLIB2003 and COR@L Problems

Since the performance of random walk methods may depend on the initial seed of the random number generator, we now discuss the obtained results in two settings: (i) using the "default" seed as an initial seed to generate a walk; (ii) average performance over 30 runs using different initial seeds. The computational results for the first setting are discussed in Sections 5.2.1 and 5.2.3; the results of the second setting are discussed in Sections 5.2.2 and 5.2.4. We also note that because of the inherent randomness in the generated "short" walk, the conclusions must be validated with multiple walks. However, in practice only a short walk will be performed.

In these experiments we choose the following parameter setting: maximumPasses = 75 and maximumRetries = 1. Moreover, the total number of walk steps allowed is 500, and the number of walk steps allowed in a particular walk (the maximum number of steps where no update is found) is 50.

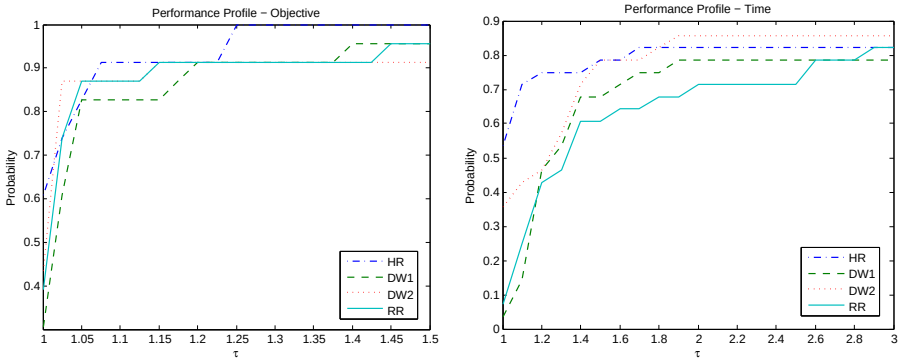
5.2.1 Performance of Random Walks on MIPLIB2003 *Problems*

The results for MIPLIB2003 test set are summarized in Table 3. Detailed results for MIPLIB2003 test set are given in Table 13. These tables report the best objective value (Columns "Obj") obtained using the hit-and-run (HR), Dikin walk with two strategies (DW1 = Constant Radius Dikin Ellipsoid, DW2 = Random Step Strategy), and Random Ray (RR), all of which are combined with FP as discussed in Section 3. Computational time (in seconds) is given under Column "Time", and Column "Step" shows the number of random walk steps to find the solution. The optimality "Gap" is calculated as indicated before. Tables 3 further reports the average time in seconds per walk step ("$\frac{\text{Time}}{\text{Step}}$"), and the average proportional effort in computing the LP optimal ("PropLP") and the analytic center solution ("PropAC"), of the FP computation ("PropFP"), and that of the rest of the computation ("PropRest"). The statistics "$\frac{\text{Time}}{\text{Step}}$" is calculated by averaging the ratio of "Time" and "Step" for each problem instance. Note that all the initial LP relaxation problems are solved by iOptimize interior point solver.

The "Avg" mean-gap in these experiments are 27.72%, 30.95%, 30.48% and 29.97% for HR, DW1, DW2, and RR, respectively. In this case HR ranked first whereas DW1 ranked last. The overall performance of these methods is better than the performance of LPS-FP discussed in Section 5.1.

Figure 3 shows the performance profiles for MIPLIB2003 test set comparing the best objective value and the CPU time performances of HR, DW1, DW2, and RR.

It is also worth noting that HR is able to find some feasible solutions of instance gesa2, while other walks could not. Moreover, in terms of the objective value HR ranked first in nine instances, DW2 and RR ranked first in seven cases, and DW1 ranked first in two instances. In two instances all these methods generated a solution with the same objective value. This result is similar to the "Avg" mean-gap for each method. Overall, HR generates better quality solutions in this experiment. This may be surprising at first because DW1 (with lazy step) mixes in strongly polynomial time



(a) Objective performance

(b) CPU time performance

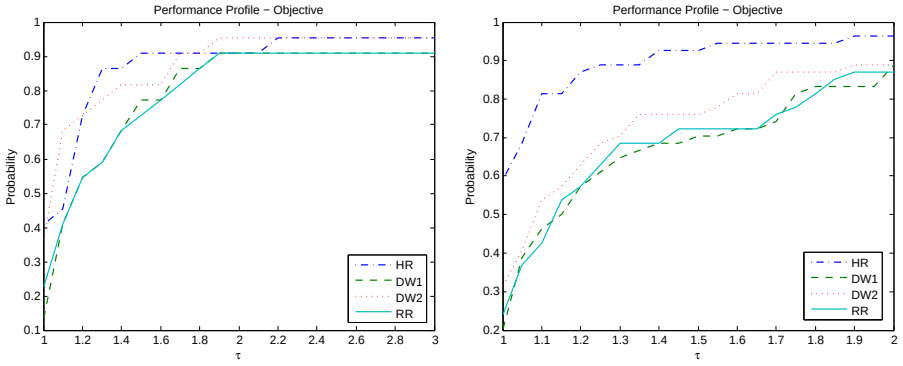
Fig. 3: Performance profiles of different walk-and-round methods on MIPLIB2003 test set

from a central point [23], while HR mixes in polynomial time; hence one would have expected better performance from DW1. One possible reason is as follows. The total number of walk steps in the experiment is relatively small. HR chooses a random point on the chord in the feasible region, and consequently when combined with FP they may be able to explore "more" region than DW1. Overall, the four search methods generate better solution quality than the pure FP at LP or at the analytic center. Note that in the reported results the average times of these walks are slightly larger (around 10% more) than that for an equivalent run of FP at the optimum solution.

Now we focus on the average computation times for the various steps of the walk-and-round methods. Observe that FP takes about 80%, the analytic center computation takes about 11%, and the LP computation takes less than 5% time of average. As one can expect, the walk computation in HR ($< 0.2\%$) is much less than the computation time in DW1, DW2, and RR (1.5% – 2.5%). In fact, for each method FP takes more than 85% of computation time in most test problems (detailed results are not reported here). This suggests that when compared to the FP heuristic, the additional computation efforts in a walk-and-round method may not be substantial. Finally we note that the Dikin walks need to compute a new matrix factorization in each step, whereas HR reuses the same factorization.

5.2.2 Performance of Random Walks over Multiple Runs on MIPLIB2003 *Problems*

The walk-and-round method may behave differently with different starting seeds to generate random numbers. Hence, to draw conclusions on their average performance we test all random walks with 30 different initial random seeds. The summary of the average performance is reported in Table 4. Table 14 reports the average computational behavior for each problem in the test set. Column "MinGap" shows the minimum gap of a solution among the 30 runs whereas column "MaxGap" shows the maximum gap among a solution of the 30 runs, column "AvgGap" and "StDevGap"



(a) MIPLIB test set (b) COR@L test set

Fig. 4: Average objective value performance of walk-and-round methods over multiple runs on MIPLIB2003 test set and COR@L test set

shows the arithmetic mean-gap and the corresponding standard deviation of the 30 runs. Column "AvgStep" reports the average number of iterations of the 30 runs.

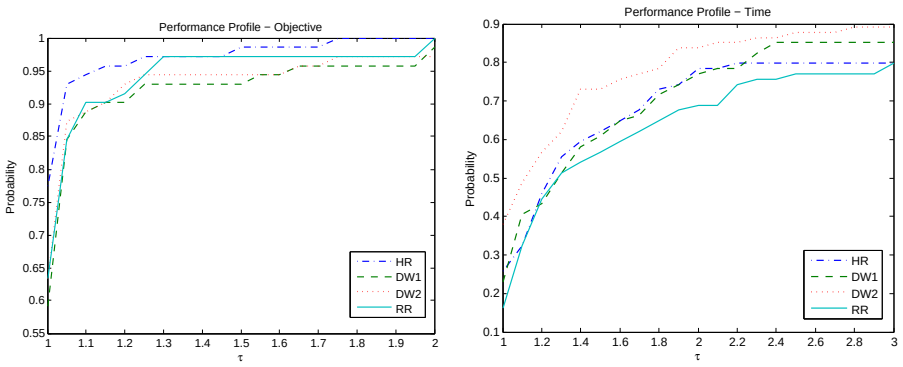
In terms of the average objective generated among the 30 runs, DW2 ranked first in nine cases and HR ranked first in eight cases. RR ranked first in six cases and DW1 ranked first in four cases. In two cases all methods generate the same objective value. By looking at the average of the "AvgGap", HR generates the best average result (26.64%), and DW2 generates the second best average result (28.85%). DW1 has an average result (31.47%) and RR ranked last with average gap (31.92%).

This result is somehow different from the single run computation presented before, where RR ranked second. We note that the average mean-gap (29.91%) for RR in the single run computation is better than its "AvgGap" (31.92%) presented here. In fact, the standard deviation of the mean-gap for RR (3.23) are also slightly larger than that for the other three methods. The standard deviations of HR (2.35) is generally less than those of DW1 (3.06), DW2 (2.63), and RR (3.23). However, comparisons using pairwise t-test between the mean-gaps of the methods show no significant difference among these four methods.

Figure 4(a) shows the performance profiles for MIPLIB2003 test set comparing the average objective value performances of HR, DW1, DW2, and RR over multiple runs.

5.2.3 Performance of Random Walks on COR@L *Problems*

We now report results from our computational experiments on COR@L test set. A summary of average performances is given in Table 5. Detailed results are given in Table 15. Comparing the performance, we found that HR ranked first in 26 cases, DW1 ranked first in seven cases, DW2 ranked first in 15 cases, and RR ranked first in 11 cases. This result is similar to that for the MIPLIB2003. In 27 cases all methods generate the same objective value. No method is able to find a feasible solution for neos-905856. HR generates the best average mean-gap (50.19%); DW2



(a) Objective performance (b) CPU time performance

Fig. 5: Performance profiles of different walk-and-round methods on COR@L test set

generates the second best mean-gap (56.75%). RR has slightly worse average mean-gap (58.72%) and DW1 generates the worst gap (65.38%). It seems that the average mean-gap of HR is significantly better than others. HR, DW2, and RR generate better solution quality than the pure FP at LP or at the analytic center solutions, and the average computation times of these walks are slightly shorter (except RR).

The FP computation is the most time-consuming step. It takes around 85% of the computation time. This value is slightly greater when compared to MIPLIB2003 test set, since LP and analytic center computations take less time for these problems. The average fraction of effort for solving the LP relaxation problem is generally smaller (about 3%). The average proportion of effort of analytic center computation is larger (about 8.5%). This value however is less than that in MIPLIB2003 (about 11%). This is due to the fact that in MIPLIB2003, instance opt1217 can be optimally solved in the first walk step, and hence the fraction of effort for solving the LP relaxation problem and of the analytic center computation becomes significant. Moreover, instances liu, manna81 and seymour, which are relatively large, can not be solved very efficiently in our current interior point code. One reason may be that the preprocessor implemented in the current iOptimize continuous solver being not as efficient as those in the commercial continuous solvers. The other reason may come from the efficiency of the standard callable matrix computation library we used, while the commercial packages have possibly more efficient matrix computation libraries. The average fraction of the walk step computation time in the Dikin type methods (about 2%) is larger than that of the HR (0.14%), which is consistent with our expectation and the results for MIPLIB2003.

Figure 5 shows the performance profiles for COR@L test set comparing the best objective value and the CPU time performances of HR, DW1, DW2, and RR walks.

Finally, we note that these methods are able to find new upper bounds for the following four instances, neos-881765, neos-1056905, neos-503737, and neos-848150, though the pure FP runs are also able to find some of them, and original bounds available to us seem to be out of date. Three of these four instances

are now proved to be solved, since the generated upper bound is same as the lower bound.

5.2.4 Performance of Random Walks over Multiple Runs on COR@L *Problems*

In this section we compare the average performance of all random walks over 30 runs with different initial random seeds on the COR@L. The summary of the average performance is reported in Table 6, and the detailed computational behavior of each problem are reported in Table 16.

In terms of the average objective generated among the 30 runs, five cases could not be evaluated because the gap could not be computed due to the unknown lower bounds. In this experiment HR ranked first in 28 cases and DW2 ranked first in 12 cases. RR ranked first in nine cases and DW1 ranked first in seven cases. In 18 cases all methods generate the same objective value. By looking at the average of the "Avg-Gap", HR generates the best average result (50.13%), and DW2 generates the second best average result (57.55%). DW1 has an average result (62.13%) and RR ranked last with average gap (62.73%). This result is similar to the that of the multiple runs on MIPLIB2003 test set presented before.

The standard deviations of HR (4.74) is generally less than those of DW1 (7.98), DW2 (5.91), and RR (7.14). This implies that HR may be significantly better than the other walks. In fact, comparisons using pairwise t-test between the mean-gaps of the methods also show that HR is better than the other three methods when short walks are taken. Note that this result is different from the conclusion reported before, where no statistical difference is observed.

Figure 4(b) shows the performance profiles for COR@L test set comparing the average objective value performances of HR, DW1, DW2, and RR over multiple runs.

We note that all these methods are able to find at least a feasible solution for problem neos-905856 during one or more runs. Recall no method found a feasible solution in the previous single run computations. This suggests that multiple random walks may have additional value in generating feasible solutions.

5.3 Performance of a Long Random Walk

One may consider using a long walk in the walk-and-round methods in the hope to keep improving the objective value. We present the practical performance of this strategy on a few problem instances. In Table 8 we give results for such long walks using 2000 steps for problems aflow30a, aflow30b, neos-504674, neos-512201, neos-538916, and neos-538867. Computational results provided in the table are the objective and the optimality gap after 500 walk steps (Columns "Start Obj" and "Start Gap") and after 2000 walk steps (Columns "End Obj" and "End Gap"). Figure 6 illustrates the objective gaps of the best feasible solution over iterations for HR, DW1, DW2, and RR walks for these problems. One can see that all methods improve the objective value as the number of steps increase, and it becomes more difficult to generate new solutions as the objective of the incumbent solution gets close to the known optimum objective.

Observe that for problems neos-504674, neos-512201, neos-538916, and neos-538867, the mean-gap of each method has an improvement of more than 35%. For problems aflow30a and aflow30b, the overall improvement for each method is not significant. This is because the "Start Gap" of each method is already small (less than 35%).

For problem aflow30b, all four methods generate similar objective mean-gap. For the remaining five problems, HR performs worse than the other three approaches. We note that HR using short walk is already not the best method in three out of these five problems. However, for the remaining two problems (neos-538916 and neos-538867), HR has the best "Start Gap" but generates worse result than the other three walks. One can observe that for problem neos-538867, HR failed to further improve the objective value after 1029 walk steps.

Figure 6 shows the performances of HR, DW1, DW2, and RR over time on these six problems.

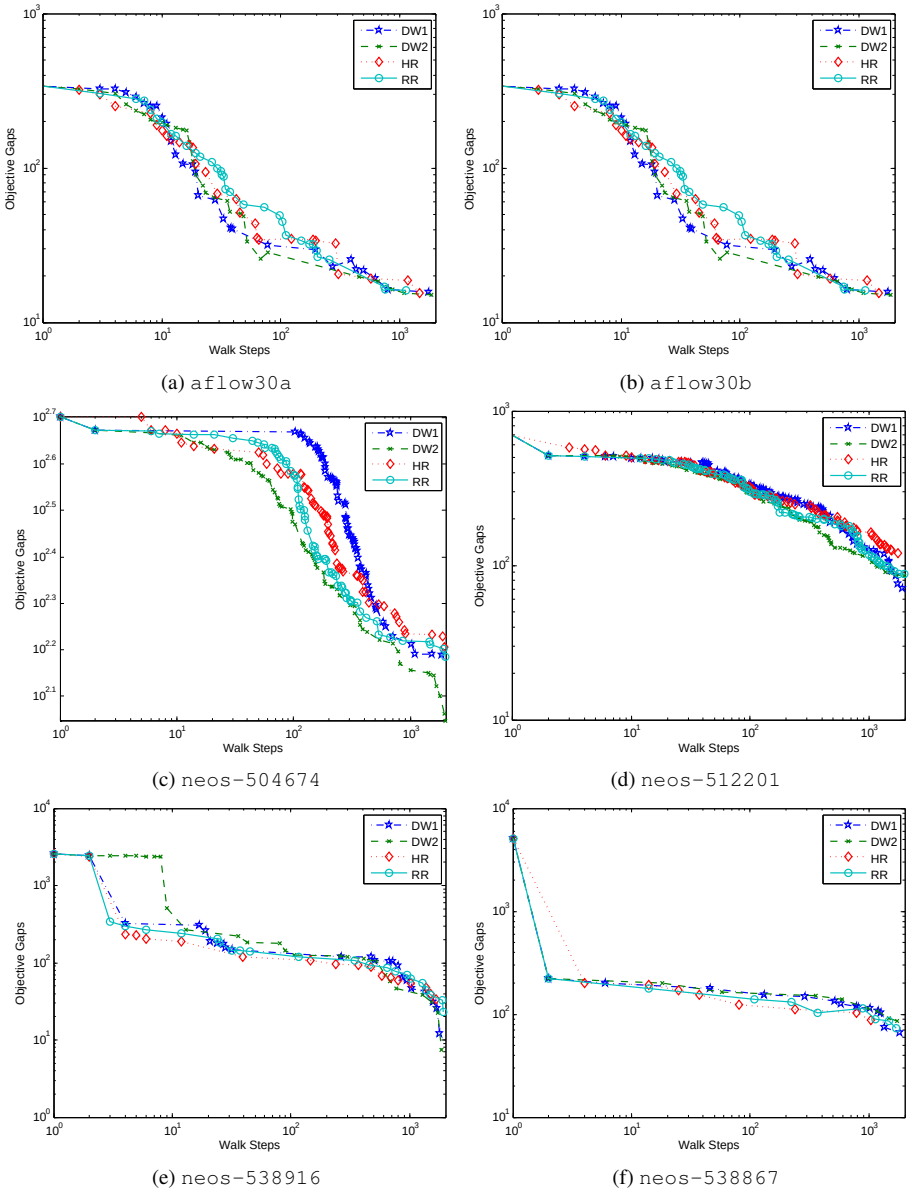
5.4 Performance of Random Walks Using Expanded Feasible Regions over Multiple Runs on COR@L Problems

To prevent the behavior of "strolling around a corner" in random walks, in this section we perform the random walks over an expanded relaxation region to potentially avoid this strolling. This is the reverse of bound tightening usually done in integer programming to generate tighten feasible regions. We test all random walks with 30 different initial random seeds, which are set to be the same as the ones used in Section 5.2.4. The upper bounds of the integer variables are expanded by 0.5. In particular, the following formulation was considered

$$\begin{aligned} &\min\ c^T x && \text{(MILP)}\\ &\text{s.t. } Ax = b,\\ &\quad l_i \le x_i \le u_i + 0.5, x_i \in \mathbb{Z}\ \forall i \in \mathcal{I},\\ &\quad l_i \le x_i \le u_i,\ \forall i \notin \mathcal{I}, \end{aligned}$$

where $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times n}$, l_i and u_i is the lower and upper bounds for integer variable x_i.

We now report results from our computational experiments on COR@L test set using expanded feasible regions. A summary of average performances is given in Table 7. Detailed results are given in Table 17. Computational results show that in 18 cases all methods yield the same objective value. HR ranked first in 23 cases and RR ranked first in 16 cases. DW2 ranked first in 11 cases whereas DW1 ranked first in eight cases. By looking at the average of the "AvgGap", HR still generates the best average result (51.13%), and DW2 generates the second best average result (53.39%). RR has an average result (54.91%) and DW1 ranked last with average gap (55.89%). This result is different from the multiple run computation presented before, where DW1 ranked third and RR ranked last. The standard deviation of the mean-gap for HR (3.86), DW1 (5.97), DW2 (4.52), and RR (5.51) have become smaller. This

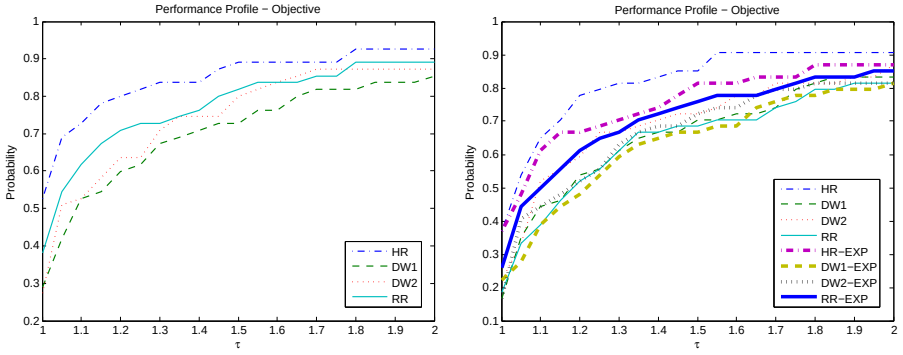


(a) `aflow30a` (b) `aflow30b`

(c) `neos-504674` (d) `neos-512201`

(e) `neos-538916` (f) `neos-538867`

Fig. 6: Incumbent solution over time on selected problems



(a) Average objective performance using expanded feasible regions

(b) Average objective performance with and without using expanded feasible regions

Fig. 7: Performance profiles of walk-and-round methods over multiple runs on `COR@L` test set

means that the performance of random walks using expanded feasible regions is more stable.

Comparisons using pairwise t-test between the mean-gaps of the methods show that HR is better than the other three methods when short walks are taken. This result is the same as that in Section 5.2.4.

When compared to Tables 16 and 17, the mean-gap of DW1 is improved from 62.13% to 55.89%, of DW2 it improved from 57.55% to 53.39%, and of RR improved from 62.73% to 54.91%. The t-tests show that all the improvements are statistically significant. The mean-gap of HR is slightly increased from 50.13% to 51.13%, but the difference is not statistically significant. Thus, the results show that expanding the feasible region may benefit Dikin type walks.

Figure 7 shows the performance profiles for `COR@L` test set comparing the average objective value performances of HR, DW1, DW2, and RR with and without using expanded feasible regions over multiple runs.

Finally, we note that only DW2 is not able to find a feasible solution for problem `neos-905856` during the runs. Recall that all methods are able to find at least a feasible solution in the previous multiple run computations.

6 Conclusions

As already shown by Fischetti et al. [18], Bertacco et al. [14], and Achterberg and Berthold [5], FP is a useful heuristic for MILP because it usually finds feasible solutions in a reasonable computational time. However, in their computational experiments they used the FP heuristic at the LP vertex optimum. Computation results in this paper show that FP started at the LP interior optimum is often comparable to that at the LP vertex optimum. We experimented with a walk-and-round framework that uses random walks to sample feasible solutions, and FP to round the generated walk points. Based on this framework, four different types of random search methods are

tested. We found that the Hit-and-Run walk based walk-and-round heuristic performance was better than Dikin-walk based heuristics when short walks are taken from near the analytic center. We were able to find better solutions and update the benchmark records in some cases. Results also suggest that expanding the region provides some improvements in the walk-and-round methods using Dikin type walks. The computational overhead of running the FP at multiple points is relatively large, however, we expect that this will be overcome in a parallel computational environment.

In the context of MILP several alternative heuristics have also been proposed. We provide a brief overview of these heuristics here, as any of these heuristics may be combined with random walks. Balas and Martin [9] presented a heuristic called Pivot-and-Complement for 0-1 MILP and extended it to general MILP later [10]. The underlying idea of the method is to perform simplex method-type pivot operations which drive the integer variables out of the basis and the slacks into the basis. This method has been further improved by Balas et al. [11]. Balas et al. [8] proposed the `OCTANE` heuristic for 0-1 MILP. `OCTANE` finds a feasible solution by enumerating extended facets of the octahedron, the outer polar of the unit hypercube. A local branching heuristic is developed in Fischetti and Lodi [19]. Danna et al. [16] presented a Relaxation Induced Neighborhood Search (`RINS`) heuristic. `RINS` exploits the three fundamental concepts of local search (neighborhood, intensification, and diversification) and transposes them to MILP. Both the local branching and `RINS` heuristics try to improve an incumbent solution of an MILP by solving a sufficiently smaller sub-MILPs. Rothberg [33] uses the idea of evolutionary algorithms to develop another heuristic, which forms the basis of the "solution polishing" in `Cplex` [3]. Combining random walks with these heuristics is a topic of future research.

Recently Baumert *et al.* [12] have proposed a discrete hit-and-run walk to sample over an integer hyper-rectangle. An adaptation and comparison of the discrete-walk in the walk-and-round framework to find solutions of the mixed integer problems remains a topic of future research.

Table 1: The Organization of the Tables

Table #	Goal	Parameter	Value
11–12	FP Performance on MIPLIB2003 Library and COR@L Library	maximumPasses maximumRetries	10000 30
13	HR, DW1, DW2, and RR Performance on MIPLIB2003 Library	maximumPasses	75
14	30 Runs of HR, DW1, DW2, and RR Performance on MIPLIB2003 Library	maximumRetries	1
15	HR, DW1, DW2, and RR Performance on COR@L Library	maximumSteps	500
16	30 Runs of HR, DW1, DW2, and RR Performance on COR@L Library	maximumSingleSteps	50
17	30 Runs of HR, DW1, DW2, and RR Performance on COR@L with Expanded Regions		
8	Numerical Experiments of the Long Random Walks	maximumSteps maximumSingleSteps	2000 2000

Table 2: Summary of the FP Performance on MIPLIB2003 and COR@L Libraries

Method	LPI-FP			LPS-FP			AC-FP		
Test set	Gap	Time	# Best	Gap	Time	# Best	Gap	Time	# Best
MIPLIB2003	35.4	119.66	10	38.44	117.91	11	48.41	111.27	6
COR@L	70.24	95.4	25	61.71	94.64	34	93.81	96.17	9

Table 3: Summary of the Walk-and-Round Performance on MIPLIB2003 Library

Method	Gap	Time	Step	$\frac{\text{Time}}{\text{Step}}$	PropLP	PropAC	PropFP	PropRest	# Best
HR	27.72	128.4	88.64	1	4.65	11.65	80.71	0.18	9
DW1	30.95	144.93	107.86	1.88	4.97	10.68	80.71	1.73	2
DW2	30.48	102.35	89.77	1.69	5.18	11.68	78.91	1.83	7
RR	29.97	144.51	107.64	1.85	4.76	10.32	82.56	1.51	7

Table 4: Summary of the 30 Run Walk-and-Round Performance on MIPLIB2003 Library

Method	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	# Best
HR	23.03	29.12	26.64	2.35	64.93	8
DW1	28.02	35.89	31.47	3.06	76.47	4
DW2	25.59	32.12	28.85	2.63	77.8	9
RR	28.47	36.56	31.92	3.23	75.16	6

Table 5: Summary of the Walk-and-Round Performance on COR@L Library

Method	Gap	Time	Step	$\frac{\text{Time}}{\text{Step}}$	PropLP	PropAC	PropFP	PropRest	# Best
HR	50.19	54.59	76.41	1.38	3.22	8.44	87.26	0.14	26
DW1	65.38	45.51	83.19	0.84	2.96	7.98	86.65	2.09	7
DW2	56.75	48.88	90.68	0.83	3.43	9.72	84.21	2.16	15
RR	58.72	52.83	100.03	0.88	3.49	7.73	86.41	2.02	11

Table 6: Summary of the 30 Run Walk-and-Round Performance on COR@L Library

Method	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	# Best
HR	42.84	61.56	50.13	4.74	79.86	28
DW1	53.07	81.13	62.13	7.98	90.3	7
DW2	49.93	74.82	57.55	5.91	91.48	12
RR	53.01	77.59	62.73	7.14	87.52	9

Table 7: Summary of the 30 Run Walk-and-Round Performance on COR@L with Expanded Regions

Method	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	# Best
HR	44.64	60.29	51.13	3.86	78.89	23
DW1	48.25	70.49	55.89	5.97	87.89	8
DW2	46.02	65.39	53.39	4.52	85.82	11
RR	48.21	70.51	54.91	5.51	88.03	16

Table 8: Numerical Experiments using the Long Random Walks

Problem	Method	Start Obj	Start Gap	End Obj	End Gap
aflow30a	HR	1313	13.38	1284	10.88
	DW1	1314	13.47	1247	7.69
	DW2	1280	10.53	1242	7.25
	RR	1321	14.07	1254	8.29
aflow30b	HR	1571	34.5	1350	15.58
	DW1	1536	31.5	1352	15.75
	DW2	1501	28.51	1344	15.07
	RR	1478	26.54	1356	16.10
neos-504674	HR	12129.1	233.59	9483.4	160.82
	DW1	20779.1	471.5	9244.05	154.24
	DW2	9798.77	169.5	7699.85	111.77
	RR	10365.4	185.08	9192.88	152.83
neos-512201	HR	1790.74	248.68	1130.25	120.07
	DW1	1851.76	260.56	875.90	70.55
	DW2	1217.98	137.16	955.55	86.06
	RR	1573.32	206.34	962.59	87.43
neos-538916	HR	298	122.38	179	33.58
	DW1	333	148.5	150	11.94
	DW2	303	126.11	144	7.46
	RR	324	141.79	165	23.13
neos-538867	HR	272	122.95	229	87.70
	DW1	337	176.22	202	65.57
	DW2	322	163.93	227	86.07
	RR	337	176.22	212	73.77

References

1. COIN-OR: Computational infrastructure for operations research (http://www.coin-or.org/)
2. COR@L: Computational optimization research at lehigh (http://coral.ie.lehigh.edu/mip-instances/)
3. IBM ILog Cplex optimizer (http://www.ibm.com/)
4. MIPLIB2003: Mixed integer library 2003 (http://miplib.zib.de/)
5. Achterberg, T., Berthold, T.: Improving the feasibility pump. Discrete Optimization **4**(1), 77–86 (2007)
6. Andersen, E.D., Gondzio, J., Mészáros, C., Xu, X.: Implementation of interior point methods for large scale linear programming, chap. 6, pp. 189–252. Kluwer Academic Publishers (1996)
7. Baena, D., Castro, J.: Using the analytic center in the feasibility pump. http://www.optimization-online.org/DB_HTML/2010/11/2793.html (2010)
8. Balas, E., Ceria, S., Dawande, M., Margot, F., Pataki, G.: Octane: A new heuristic for pure 0-1 programs. Operations Research **49**(2), 207–225 (2001)
9. Balas, E., Martin, C.H.: Pivot-and-complement: A heuristic for 0-1 programming. Management Science **26**(1), 8696 (1980)
10. Balas, E., Martin., C.H.: Pivot-and-shift: A heuristic for mixed integer programming. Tech. rep., GSIA, Carnegie Mellon University (August 1986)
11. Balas, E., Schmieta, S., Wallace., C.: Pivot and shift - a mixed integer programming heuristic. Discrete Optimization **1**(1), 3–12 (2004)
12. Baumert, S., Ghate, A., Kiatsupaibul, S., Shen, Y., Smith, R.L., Zabinsky, Z.B.: Discrete hit-and-run for sampling points from arbitrary distributions over subsets of integer hyper-rectangles. Operations Research **57**, 727–739 (2009)
13. Belisle, C.J.P.: Convergence theorems for a class of simulated annealing algorithms on $\mathbb{R}^d$. Journal of Applied Probability **29**, 885–895 (1992)
14. Bertacco, L., Fischetti, M., Lodi, A.: A feasibility pump heuristic for general mixed-integer problems. Discrete Optimization **4**(1), 63–76 (2007)
15. Bertsimas, D., Vempala, S.: Solving convex programs by random walks. The Journal of the ACM **51**(4), 540–556 (2004)
16. Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming **102**(1), 71–90 (2005)
17. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201–213 (2002)
18. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Mathematical Programming **104**(1), 91–104 (2005)
19. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming **98**(1), 23–47 (2003)
20. Fischetti, M., Salvagnin, D.: Feasibility pump 2.0. Mathematical Programming Computation **1**, 201–222 (2009)
21. Fourer, R., Mehrotra, S.: Solving symmetrical indefinite systems in an interior-point method for linear-programming. Mathematical Programming **62**(1), 15–39

(1993)
22. Gondzio, J.: Multiple centrality corrections in a primal-dual method for linear programming. Computational Optimization and Applications **6**, 137–156 (1996)
23. Kannan, R., Narayanan, H.: Random walks on polytopes and an affine interior point method for linear programming. In: Proceedings of the 41st annual ACM symposium on Theory of computing, pp. 561–570 (2009)
24. Kannan, R., Vempala, S.: Sampling lattice points. Proceedings of the twenty-ninth annual ACM symposium on Theory of computing pp. 696 – 700 (1997)
25. Knuth, D.E.: The Art of Computer Programming, Vol. 2: Seminumerical Algorithms:. Addison-Wesley (1969)
26. Lovász, L.: Hit-and-run mixes fast. Mathematical Programming **86(3)**, 443–461 (1999)
27. Lovász, L., Vempala, S.: Hit-and-run from a corner. SIAM Journal on Computing **35**(4), 985–1005 (2006)
28. Lovász, L., Vempala, S.: Simulated annealing in convex bodies and an $O^*(n^4)$ volume algorithm. Journal of Computer and System Sciences **72**(2), 392–417 (2006)
29. Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM Journal on Optimization **2**(4), 575–601 (1992)
30. Mehrotra, S., Huang, K.L.: Computational experience with a modified potential reduction algorithm for linear programming. Optimization Method and Software (2011). DOI 10.1080/10556788.2011.634911
31. Narayanan, H.: Randomized interior point methods for sampling and optimization. `http://arxiv.org/abs/arXiv:0911.3950` (2009)
32. Renegar, J.: A Mathematical View Of Interior-point Methods In Convex Optimization. SIAM (2001)
33. Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS Journal on Computing **19**(4), 534–541 (2007)
34. Smith, R.L.: Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. Operations Research **32**(6), 1296 – 1308 (1984)
35. Solis, F.J., Wets, R.J.B.: Minimization by random search techniques. Mathematics of Operations Research **6(1)**, 19–30 (1981)
36. Vempala, S.: Geometric random walks: a survey. Combinatorial and Computational Geometry **52**, 573–612 (2005)
37. Xu, X., Hung, P., Ye, Y.: A simplified homogeneous and self-dual linear programming algorithm and its implementation. Annals of Operations Research **62**(1), 151–171 (1996)
38. Ye, Y.: Interior Point Algorithms: Theory and Analysis. Wiley (1997)
39. Zabinsky, Z.B.: Random search algorithms. Tech. rep., Department of Industrial and Systems Engineering, University of Washington, Seattle, WA (2009)
40. Zabinsky, Z.B., Smith, R.L., McDonald, J.F., Romeijn, H.E., Kaufman, D.E.: Improving hit-and-run for global optimization. Journal of Global Optimization **3**(2), 171–192 (1993)

A appendix

Table 9: Problem Profiles of MIPLIB2003 library

Problem	Objective	Rows	Cols	Nonzeros	Conti	Bin	Int
10teams	924	230	2025	12150	225	1800	0
a1c1s1	11503.44	3312	3648	10178	3456	192	0
aflow30a	1158	479	842	2091	421	421	0
aflow40b	1168	1442	2728	6783	1364	1364	0
cap6000	-2451380	2176	6000	48243	0	6000	0
danoint	65.66	664	521	3232	465	56	0
disctom	-5000	399	10000	30000	0	10000	0
fixnet6	3983	478	878	1756	500	378	0
gesa2	25779900	1392	1224	5064	816	240	168
gesa2-o	25779900	1248	1224	3672	504	384	336
liu	1172	2178	1156	10626	67	1089	0
manna81	-13164	6480	3321	12960	0	18	3303
mas74	11801.2	13	151	1706	1	150	0
mas76	40005.1	12	151	1640	1	150	0
mkc	-563.846	3411	5325	17038	2	5323	0
modglob	20740500	291	422	968	324	98	0
noswot	-41	182	128	735	28	75	25
opt1217	-16	64	769	1542	1	768	0
p2756	3124	755	2756	8937	0	2756	0
pp08aCUTS	7350	246	240	839	176	64	0
pp08a	7350	136	240	480	176	64	0
rout	1077.56	291	556	2431	241	300	15
set1ch	54537.8	492	712	1412	472	240	0
seymour	423	4944	1372	33549	0	1372	0
timtab1	764772	171	397	829	226	64	107
timtab2	1096560	294	675	1482	381	113	181
tr12-30	130596	750	1080	2508	720	360	0
vpm2	13.75	234	378	917	210	168	0

Table 10: Problem Profiles of COR@L library

Problem	Lower Bound	Upper Bound	Rows	Columns	Conti	Bin	Int
neos5	15	15	63	63	10	53	0
neos-584851	-11	-11	661	445	40	405	0
neos-881765	0	+∞	278	712	0	712	0
neos-905856	-8	+∞	403	686	0	686	0
neos-1121679	−∞	16	6	62	12	50	0
neos-1211578	-77	-77	356	260	130	130	0
neos-1228986	-123	-123	356	260	130	130	0
neos-1337489	-77	-77	356	260	130	130	0
neos-1420205	40	40	383	231	0	126	105
neos-1430701	-78	-77	668	312	156	156	0
neos-1440447	-100	-100	561	260	130	130	0
neos-1460246	2325	2606	306	285	19	266	0
neos-1480121	43	43	363	222	152	70	0
ran14x18_1	3667.46	3736	284	504	252	252	0
rlp1	-1	15	68	461	11	450	0
roy	3208.957	3208.957	162	149	99	50	0
neos14	74333.34	74333.34	552	792	656	136	0
neos15	77895.21	81525.14	552	792	632	160	0
neos-1489999	354	354	1046	534	2	532	0
neos-911880	49.85281	54.83	83	888	48	840	0
neos-504815	2296.22	2296.22	1067	674	554	120	0
bienst1	46.75	46.75	576	505	477	28	0
bienst2	54.6	54.6	576	505	470	35	0
mcf2	65.66667	65.66667	664	521	465	56	0
neos-911970	50.572	54.76	107	888	48	840	0
neos-631517	11275806	11503309	351	1090	231	859	0
neos-1439395	-182	-180.33	775	364	182	182	0
22433	21477	21477	198	429	198	231	0
neos-1620807	6	6	1340	231	0	231	0
neos-1346382	-178	-176	796	520	260	260	0
neos-1426635	-178	-176	796	520	260	260	0
23588	8090	8090	137	368	137	231	0
neos-512201	513.57	513.57	1335	838	688	150	0
neos-504674	3635.87	3635.87	1344	844	694	150	0
neos-582605	−∞	1	1240	1265	865	400	0
neos-1225589	1.23E+09	1.23E+09	675	1300	650	650	0
neos-631164	10948328	11315752	406	1282	245	1037	0
neos-955215	446.5	446.5	723	1302	672	630	0
neos-1440460	-180	-179.25	989	468	234	234	0
neos-1056905	−∞	30	900	463	43	420	0
neos-1595230	9	9	1750	490	0	490	0
neos-1429461	-102	-101.25	1096	520	260	260	0
neos-1467067	-103.667	-103	1084	1196	598	598	0
neos-522351	17891.08	17891.08	1705	1524	1284	240	0
neos-538867	122	122	1170	792	0	792	0
neos-1200887	-74	-74	633	234	117	117	0
neos-1616732	−∞	159	1999	200	0	200	0
neos-825075	-272	-272	328	800	0	800	0
neos-933815	759.7285	766	947	1728	888	840	0
neos-538916	134	134	1314	864	0	864	0
neos-906865	3175	3175	1634	1184	784	400	0
neos-863472	9.94541	11.69	523	588	56	532	0
binkar10_1	6742.2	6742.2	1026	2298	2128	170	0
neos11	9	9	2706	1220	320	900	0
neos-503737	50	52	500	2850	350	2500	0
neos-593853	1.17E+09	1.17E+09	1606	2400	1200	1200	0
neos-598183	18429.98	18429.98	992	1696	1260	436	0
neos-603073	16790.24	16790.24	992	1696	1260	436	0
neos-848150	0	+∞	731	949	0	949	0
neos-892255	14	14	2137	1800	0	1800	0
neos-933364	760.4215	766	1006	1728	888	840	0
neos-934184	760.4215	766	1006	1728	888	840	0
neos-942323	15	17	754	732	48	684	0
neos-1311124	-182	-181	1643	1092	546	546	0
neos-1426662	-52	-44	1914	832	416	416	0
neos-1427181	-104	-102	1786	832	416	416	0
neos-1427261	-130	-127	2226	1040	520	520	0
neos-1429185	-78	-76	1346	624	312	312	0
neos-1436709	-129	-128	1417	676	338	338	0
neos-1437164	8	8	187	2256	0	2256	0
neos-1440457	-180	-179	1952	936	468	468	0
neos-1442119	-182	-181	1524	728	364	364	0
neos-1442657	-156	-154.5	1310	624	312	312	0
neos-1603512	0	5	555	730	1	729	0

Table 11: FP Performance on MIPLIB2003 Library

Method	LPI-FP			LPS-FP			AC-FP		
Problem	Obj	Gap	Time	Obj	Gap	Time	Obj	Gap	Time
10teams	928	0.43	617.02	924	0	573.06	924	0	546.5
a1c1s1	14280.28	24.13	109.25	15461.71	34.4	155.01	17838.79	55.07	124.99
aflow30a	1377	18.91	41.67	1377	18.91	42.65	1460	26.07	39.86
aflow40b	1508	29.1	184.4	1628	39.38	177.39	1417	21.31	175.3
cap6000	-2440624	0.43	94.92	-2441371	0.4	98.75	-2444695	0.27	101.21
danoint	70.75	7.75	339.78	69.67	6.1	338.87	67	2.04	336.12
disctom	-5000	0	12.65	-5000	0	9.54	-5000	0	12.33
fixnet6	10503	163.69	20.36	10503	163.69	20.61	16092	304.01	21.08
gesa2	NA	NA	0.3	NA	NA	0.12	NA	NA	0.53
gesa2-o	NA	NA	0.23	NA	NA	0.09	NA	NA	0.34
liu	3952	237.2	107.54	4606	293	79.7	4212	259.38	152.52
manna81	NA	NA	24.31	NA	NA	1.1	NA	NA	64.59
mas74	14372.87	21.79	5.26	13408.76	13.62	5.61	15190.39	28.71	6.05
mas76	40578.62	1.43	5.86	40578.62	1.43	6.44	43057.82	7.63	4.75
mkc	-199.35	64.64	100.57	-262.66	53.41	98.27	-202.64	64.06	82.77
modglob	20786787	0.22	12.21	20786787	0.22	11.66	20869013	0.61	10.53
noswot	-37	9.75	0.04	-40	2.43	0.63	-37	9.75	0.1
opt1217	-16	0	11.97	-16	0	12.92	-10	37.5	15.27
p2756	NA	NA	28.88	NA	NA	29.79	NA	NA	28.77
pp08aCUTS	8800	19.72	17.88	8800	19.72	18.13	8330	13.33	20.78
pp08a	9660	31.42	5.12	8980	22.17	4.74	9100	23.8	4.86
rout	1508.83	40.02	0.17	1824.57	69.32	0.44	1603.76	48.83	0.14
set1ch	65181	19.51	14.01	65418.25	19.95	14.81	83876.25	53.79	14.44
seymour	426	0.7	890.98	429	1.41	880.88	432	2.12	737.39
timtab1	NA	NA	0.06	NA	NA	0.03	NA	NA	0.07
timtab2	NA	NA	0.15	NA	NA	0.05	NA	NA	0.23
tr12-30	183785	40.72	25.88	183785	40.72	26.9	179795	37.67	27.37
vpm2	20.25	47.27	15.02	20	45.45	17	23.25	69.09	13.66
Avg		35.4	119.66		38.44	117.91		48.41	111.27

[1] NA: Statistics not available.

Table 12: FP Performance on COR@L Library

Method	LPI-FP			LPS-FP			AC-FP		
Problem	Obj	Gap	Time	Obj	Gap	Time	Obj	Gap	Time
neos5	15	0	31.05	15	0	32.98	16	6.67	42.87
neos-584851	-7	36.36	68.44	-11	0	73.51	-9	18.18	81.54
neos-881765	0	0	4.3	0	0	5.8	0	0	0.46
neos-905856	NA	NA	247.86	NA	NA	250.99	NA	NA	248.34
neos-1121679	68	?	1.58	59	?	1.49	72	?	1.57
neos-1211578	-77	0	17.06	-77	0	16.63	-77	0	16.25
neos-1228986	-121	1.63	18.55	-123	0	18.31	-121	1.63	18.67
neos-1337489	-77	0	17.07	-77	0	16.26	-77	0	16.12
neos-1420205	NA	NA	0.11	NA	NA	0.05	108	170	0.17
neos-1430701	-76	2.56	37.63	-76	2.56	37.51	-73	6.41	37.84
neos-1440447	-100	0	32.03	-98	2	30.36	-96	4	30.53
neos-1460246	2658	14.32	35.89	2606	12.09	29.33	2658	14.32	33.68
neos-1480121	43	0	12.1	55.5	29.07	11.13	61	41.86	10.84
ran14x18_1	4430	20.79	17.5	4344	18.45	17.57	4576	24.77	17.07
rlp1	22	2300	9.83	21	2200	12.69	21	2200	9.35
roy	3228.96	0.62	5.44	3218.96	0.31	5.35	3218.96	0.31	5.89
neos14	98825.34	32.95	14.28	98825.34	32.95	13.93	92779.1	24.81	13.32
neos15	106126.3	36.24	13.19	106126.3	36.24	13.02	110386.4	41.71	14.33
neos-1489999	377	6.5	78.34	372	5.08	73.22	362	2.26	103.43
neos-911880	230.95	363.26	0.06	91.68	83.9	13.99	128.29	157.34	14.26
neos-504815	4807.2	109.35	41.03	5552.68	141.82	41.31	8083.86	252.05	43.7
bienst1	49.75	6.42	152.88	48	2.67	158.08	50.29	7.56	139.86
bienst2	56	2.56	188.3	56.8	4.03	178	65	19.05	133.59
mcf2	70.75	7.74	344.23	71.4	8.73	338.68	67	2.03	344.4
neos-911970	77.52	53.29	18.76	173.05	242.19	21.41	115.7	128.78	20.34
neos-631517	14102884	25.07	56.07	12754652	13.12	50.62	13735269	21.81	37.2
neos-1439395	-180	1.1	46.64	-180	1.1	44.57	-179	1.65	48.19
22433	21477	0	27.85	21477	0	27.56	21505	0.13	44.22
neos-1620807	6	0	28.37	6	0	25.31	6	0	29.3
neos-1346382	-175	1.69	37.07	-176	1.12	36.49	-175	1.69	36.9
neos-1426635	-175	1.69	36.81	-176	1.12	36.18	-175	1.69	36.45
23588	8099	0.11	18.7	8099	0.11	18.03	8113	0.28	18.04
neos-512201	750.88	46.21	64.08	1504.21	192.89	52.01	963.18	87.55	75.6
neos-504674	10796.29	196.94	40.99	8125.13	123.47	52.12	14443.26	297.24	54.55
neos-582605	1	?	68.12	1	?	71.79	1	?	69.11
neos-1225589	2.66E+09	116.43	57.99	2.62E+09	113.18	57.55	2.32E+09	88.71	60.72
neos-631164	13251437	21.04	66.32	12963731	18.41	47.61	14054883	28.38	51.6
neos-955215	572	28.11	41.89	573.7	28.49	38.88	695.5	55.77	38.26
neos-1440460	-177	1.67	59.3	-177.5	1.39	59.9	-176.67	1.85	60.27
neos-1056905	31	?	44.2	33	?	42.74	36	?	46.93
neos-1595230	9	0	67.62	9	0	67.19	10	11.11	69.04
neos-1429461	-101	0.98	69.53	-100	1.96	68.54	-98.25	3.68	73.09
neos-1467067	-103	0.64	26.31	-103	0.64	26.23	-103	0.64	27.85
neos-522351	20009.06	11.84	91.65	23078.51	28.99	95.07	35644.23	99.23	109.38
neos-538867	392	221.31	49.63	367	200.82	51.89	697	471.31	55.11
neos-1200887	-73	1.35	64.83	-74	0	56.76	-73	1.35	65.21
neos-1616732	170	?	164.69	167	?	155.71	170	?	226.43
neos-825075	-272	0	81.79	-272	0	85.29	-262	3.68	109.25
neos-933815	867.8	14.22	39.8	822	8.2	38.74	6595.9	768.19	76.24
neos-538916	1198	794.03	53.72	668	398.51	53.49	1217	808.21	73.17
neos-906865	3176	0.03	75.21	3176	0.03	72.79	3977	25.26	103.45
neos-863472	12.27	23.39	43.94	11.92	19.84	42.3	12.15	22.15	58.84
binkar10_1	6929.06	2.77	51.03	6929.06	2.77	50.49	7032.77	4.31	70.24
neos11	9	0	1645	9	0	1645.26	9	0	1438.57
neos-503737	50	0	120.88	50	0	84.72	50	0	120.6
neos-593853	1.25E+09	6.57	101.29	1.23E+09	5.34	105.25	1.21E+09	3.3	104.68
neos-598183	21304.26	15.6	58.95	21354.81	15.87	60	50804.7	175.66	62.04
neos-603073	24577.13	46.38	61.39	22411.98	33.48	47.34	46484.17	176.85	64.02
neos-848150	0	0	21.59	0	0	3.95	0	0	15.33
neos-892255	14	0	586.03	14	0	602.98	14	0	597.73
neos-933364	959.8	26.22	59.81	813.3	6.95	58.89	1081.7	42.25	63.04
neos-934184	959.8	26.22	59.97	813.3	6.95	58.58	1081.7	42.25	63.37
neos-942323	17	13.33	130.93	17	13.33	142.66	17	13.33	135.38
neos-1311124	-181	0.55	92.02	-181	0.55	87.59	-181	0.55	85.61
neos-1426662	-42	19.23	131.43	-42	19.23	131.83	-41	21.15	131.08
neos-1427181	-98	5.77	132	-94	9.62	132.38	-102	1.92	166.75
neos-1427261	-116.83	10.13	179.96	-121	6.92	198.44	-113.67	12.56	185.06
neos-1429185	-74	5.13	94.27	-73.67	5.56	92.62	-72.5	7.05	102.2
neos-1436709	-126	2.33	95.39	-123	4.65	92.17	-125	3.1	96.17
neos-1437164	9	12.5	14.09	9	12.5	38.38	9	12.5	27.37
neos-1440457	-170	5.56	154.81	-166.33	7.59	150.33	-168	6.67	154.38
neos-1442119	-177	2.75	107.12	-173	4.95	103.55	-175.33	3.66	107.96
neos-1442657	-152	2.56	89.93	-152	2.56	89.42	-152	2.56	91.73
neos-1603512	5	∞	204.16	5	∞	192.19	5	∞	206.34
Avg		70.24	95.4	25	61.71	94.64	34	93.81	96.17

[1] NA: Statistics not available.
[2] ?: Unknown bound.

Table 13: Walk-and-Round Performance on MIPLIB2003 Library

Method	HR				DW1			
Problem	Obj	Gap	Time	Step	Obj	Gap	Time	Step
10teams	978	5.84	134.452	67	924	0	134.693	36
a1c1s1	15128.96	31.51	100.563	58	15172.38	31.89	116.551	51
aflow30a	1313	13.38	29.379	115	1314	13.47	38.171	130
aflow40b	1571	34.5	125.842	113	1536	31.5	380.596	238
cap6000	-2442801	0.34	49.559	53	-2422109	1.19	766.951	500
danoint	67.66667	3.05	333.137	128	70.2	6.91	308.931	115
disctom	-5000	0	7.921	1	-5000	0	8.059	1
fixnet6	8876	122.84	10.601	55	7236	81.67	9.685	54
gesa2	27681471	7.37	5.437	51	NA	NA	7.444	51
gesa2-o	NA	NA	3.822	51	NA	NA	5.801	51
liu	2954	152.04	650.681	174	4102	250	353.583	161
manna81	NA	NA	140.049	51	NA	NA	151.435	51
mas74	15039.84	27.44	29.921	488	14372.87	21.79	1.916	51
mas76	41726.41	4.3	2.311	59	42569.76	6.41	1.885	53
mkc	-240.39	57.36	57.645	51	-240.65	57.31	78.756	51
modglob	20787801	0.22	12.393	87	20789715	0.23	45.504	245
noswot	-38	7.31	0.494	52	-37	9.75	0.809	52
opt1217	-16	0	0.071	1	-16	0	0.078	1
p2756	NA	NA	12.055	51	NA	NA	16.359	51
pp08aCUTS	8710	18.5	4.941	60	8910	21.22	5.905	58
pp08a	8170	11.15	5.879	83	8350	13.6	6.473	81
rout	1683.13	56.19	0.317	51	1584.4	47.03	0.364	53
set1ch	65959.75	20.94	8.109	62	65959.75	20.94	10.049	74
seymour	431	1.89	1234.287	58	435	2.83	841.544	52
timtab1	NA	NA	1.748	51	NA	NA	0.165	51
timtab2	NA	NA	3.853	51	NA	NA	0.395	51
tr12-30	153350	17.42	18.827	72	153798	17.76	70.466	265
vpm2	17	23.63	7.524	62	20	45.45	7.544	51
Avg		27.72	128.4	88.64		30.95	144.93	107.86

Method	DW2				RR			
Problem	Obj	Gap	Time	Step	Obj	Gap	Time	Step
10teams	924	0	183.159	53	924	0	147.433	41
a1c1s1	15172.38	31.89	99.479	51	15172.38	31.89	113.052	51
aflow30a	1280	10.53	41.547	149	1321	14.07	39.434	136
aflow40b	1501	28.51	173.227	128	1478	26.54	400.532	253
cap6000	-2442812	0.34	349.115	288	-2442812	0.34	766.565	500
danoint	69	5.08	233.677	94	69	5.08	240.834	85
disctom	-5000	0	6.972	1	-5000	0	7.377	1
fixnet6	7130	79.01	9.68	65	7101	78.28	12.405	76
gesa2	NA	NA	7.291	51	NA	NA	7.125	51
gesa2-o	NA	NA	5.158	51	NA	NA	5.925	51
liu	4498	283.78	192.249	80	4240	261.77	342.348	152
manna81	NA	NA	136.119	51	NA	NA	150.669	51
mas74	14372.87	21.79	0.498	51	14372.87	21.79	1.796	51
mas76	42569.76	6.41	0.424	56	42569.76	6.41	1.806	53
mkc	-240.65	57.31	70.791	51	-241.41	57.18	149.031	93
modglob	20789715	0.23	42.578	170	20789715	0.23	35.632	181
noswot	-37	9.75	0.707	52	-37	9.75	0.649	52
opt1217	-16	0	0.075	1	-16	0	0.066	1
p2756	NA	NA	14.65	51	NA	NA	15.798	51
pp08aCUTS	8630	17.41	8.922	75	8910	21.22	5.61	60
pp08a	8040	9.38	6.249	101	8330	13.33	11.367	151
rout	1390.27	29.02	1.969	62	1370.8	27.21	1.367	60
set1ch	65072.5	19.31	14.385	115	65959.75	20.94	8.905	66
seymour	435	2.83	747.864	52	435	2.83	837.971	52
timtab1	NA	NA	0.17	51	NA	NA	0.197	51
timtab2	NA	NA	0.344	51	NA	NA	0.358	51
tr12-30	153943	17.87	58.73	204	154959	18.65	47.668	184
vpm2	19.25	40	9.383	76	19.5	41.81	7.318	69
Avg		30.48	102.35	89.77		29.97	144.51	107.64

[1] NA: Statistics not available.

Table 14: 30 Runs of Walk-and-Round Performance on `MIPLIB2003` Library

Method	HR					DW1				
Problem	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	MinGap	MaxGap	AvgGap	StDevGap	AvgStep
10teams	0	9.95	5.95	3.28	51.28	0	0.21	0.04	0.09	55.33
a1c1s1	24.14	33.21	29.94	3.15	53.47	31.48	33.75	32.47	0.85	47.88
aflow30a	11.13	18.39	14.5	2.81	77.6	11.74	16.4	13.4	1.82	87.29
aflow40b	29.7	39.12	33.76	3.79	127.85	31.5	49.74	39.74	7.47	138.97
cap6000	0.34	0.34	0.34	0	44.22	0.34	2.21	1.65	0.75	264.37
danoint	3.05	7.52	5.59	1.95	79.42	2.04	8.13	5	2.54	86.17
disctom	0	0	0	0	0.83	0	0	0	0	0.83
fixnet6	79.63	122.84	115.64	17.64	66.31	78.63	81.67	79.73	1.02	67.74
gesa2	7.37	7.37	7.37	0	43.73	NA	NA	NA	NA	51
gesa2-o	NA	NA	NA	NA	51	NA	NA	NA	NA	51
liu	139.59	162.79	147.83	8.94	127	250	310.15	269.41	21.7	152.1
manna81	NA	NA	NA	NA	51	NA	NA	NA	NA	51
mas74	27.43	30.11	27.88	1.09	272.91	21.78	21.79	21.79	0	46.13
mas76	4.3	4.55	4.34	0.1	49.93	4.3	6.41	5.71	1.09	45.24
mkc	57.36	57.36	57.36	0	52.06	57.3	57.31	57.31	0	52.05
modglob	0.22	0.63	0.29	0.17	61.04	0.23	0.23	0.23	0	180.87
noswot	7.31	7.31	7.31	0	44.55	4.87	9.75	8.94	1.99	45.29
opt1217	0	0	0	0	0.83	0	0	0	0	0.83
p2756	NA	NA	NA	NA	51	NA	NA	NA	NA	51
pp08aCUTS	18.09	19.59	18.73	0.55	56.08	19.59	34.96	23.19	5.8	52.99
pp08a	11.15	15.64	13.28	1.84	74.7	10.88	13.6	12.72	0.98	87.25
rout	56.19	56.19	56.19	0	51.87	20.14	47.03	36.17	12.99	51.02
set1ch	20.94	22.01	21.12	0.44	61.66	20.94	22.01	21.3	0.55	68.99
seymour	1.65	1.89	1.69	0.1	49.28	2.83	2.83	2.83	0	43.81
timtab1	NA	NA	NA	NA	51	NA	NA	NA	NA	51
timtab2	NA	NA	NA	NA	51	NA	NA	NA	NA	51
tr12-30	15.55	18.39	17.37	0.96	58.93	16.89	18.7	17.67	0.59	201.62
vpm2	14.54	34.54	26.36	7.25	57.42	30.9	52.72	43.02	7.15	58.3
Avg	23.03	29.12	26.64	2.35	64.93	28.02	35.89	31.47	3.06	76.47

[1] NA: Statistics not available.

Method	DW2					RR				
Problem	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	MinGap	MaxGap	AvgGap	StDevGap	AvgStep
10teams	0	0	0	0	77.69	0	0	0	0	44.17
a1c1s1	31.89	32.25	32.19	0.15	55.95	31.89	32.25	32.19	0.15	47.88
aflow30a	10.53	17.87	14.45	2.88	87.07	11.39	14.59	12.82	1.57	130.35
aflow40b	19.17	35.35	27.75	5.95	134.69	26.54	45.2	37.67	6.42	130.64
cap6000	0.22	0.34	0.3	0.06	230.41	0.34	2.22	1.7	0.7	262.8
danoint	2.80	8.64	5.15	2.06	110.75	3.56	6.6	5.46	1.34	77.6
disctom	0	0	0	0	1.83	0	0	0	0	0.83
fixnet6	75.67	80.26	78.81	1.68	67.37	72.88	81.24	77.35	3.59	75.90
gesa2	NA	NA	NA	NA	42.26	NA	NA	NA	NA	51
gesa2-o	NA	NA	NA	NA	41.83	NA	NA	NA	NA	51
liu	235.32	283.78	250.08	20.19	136.63	253.41	344.28	276.87	35.08	127.31
manna81	NA	NA	NA	NA	68.02	NA	NA	NA	NA	51
mas74	21.79	21.79	21.79	0	37.71	21.79	21.79	21.79	0	46.13
mas76	6.41	9.42	7.92	1.65	38.64	6.41	6.41	6.41	0	45.24
mkc	57.31	57.31	57.31	0	55.35	57.18	57.31	57.29	0.05	59.05
modglob	0.23	0.23	0.23	0	131.13	0.23	0.23	0.23	0	164.37
noswot	7.31	9.75	9.34	1.00	41.41	9.75	9.75	9.75	0	44.96
opt1217	0	0	0	0	0.68	0	0	0	0	0.83
p2756	NA	NA	NA	NA	43.73	NA	NA	NA	NA	51
pp08aCUTS	14.14	17.82	16.5	1.36	63.39	20.13	21.22	21.04	0.44	51.04
pp08a	9.38	13.6	11.49	1.39	80.69	10.88	13.6	12.67	1.15	109.43
rout	3.37	35.34	26.36	12.34	68.24	27.21	56.32	44.21	14	56.74
set1ch	16.78	19.31	18.47	0.92	89.58	20.94	23.88	21.79	1.15	63.81
seymour	2.83	2.83	2.83	0	159.78	2.83	3.30	2.95	0.2	43.97
timtab1	NA	NA	NA	NA	40.83	NA	NA	NA	NA	51
timtab2	NA	NA	NA	NA	40.87	NA	NA	NA	NA	51
tr12-30	16.99	18.85	17.79	0.63	156.12	18.16	18.73	18.47	0.21	151.09
vpm2	30.9	41.81	36.05	5.68	75.71	30.9	45.45	41.51	5.45	64.32
Avg	25.59	32.12	28.85	2.63	77.80	28.47	36.56	31.92	3.23	75.16

[1] NA: Statistics not available.

Table 15: Walk-and-Round Performance on COR@L Library

Method	HR				DW1			
Problem	Obj	Gap	Time	Step	Obj	Gap	Time	Step
neos5	1	0	0.3	9	1	0	0.26	7
neos-584851	-11	0	4.96	15	-11	0	15.92	29
neos-881765	0	0	11.02	3	0	0	2.25	1
neos-905856	NA	NA	209.13	51	NA	NA	111.27	51
neos-1121679	77	?	1.26	85	67	?	2.66	199
neos-1211578	-77	0	1.4	15	-77	0	4.59	36
neos-1228986	-123	0	2.2	22	-123	0	4.25	31
neos-1337489	-77	0	1.49	15	-77	0	4.62	36
neos-1420205	40	0	5.68	50	79	97.5	3.15	108
neos-1430701	-76.3333	2.13	25.93	82	-76	2.56	34.33	113
neos-1440447	-98	2	16.33	74	-100	0	8.03	36
neos-1460246	2606	12.08	19.2	75	2606	12.08	26.28	85
neos-1480121	43	0	0.17	4	43	0	0.12	2
ran14x18_1	4265	16.29	5.56	52	4607	25.61	13.1	108
rlp1	18	1900	2.86	51	22	2300	4.61	52
roy	3218.96	0.31	2.51	58	3218.96	0.31	2.51	53
neos14	92490	24.42	12.69	87	92567.3	24.53	16.04	105
neos15	108777.6	39.64	13.17	107	107280	37.72	14.48	101
neos-1489999	354	0	38.57	65	354	0	74.53	135
neos-911880	88.6	77.72	14.11	105	89.45	79.42	21.77	181
neos-504815	7793.66	239.41	42.91	168	7227.35	214.74	63.21	236
bienst1	46.75	0	35.37	44	46.75	0	8.6	17
bienst2	56.1667	2.86	57.92	68	54.8571	0.47	56.99	63
mcf2	67.6667	3.04	323.86	128	70.2	6.9	285.04	115
neos-911970	77.96	54.15	16.55	86	80.94	60.04	27.32	170
neos-631517	1.25E+07	11.15	18.16	51	1.27E+07	12.4	15.36	51
neos-1439395	-180	1.09	31.13	74	-180	1.09	27.01	69
22433	21477	0	0.69	8	21477	0	0.43	4
neos-1620807	6	0	0.39	1	6	0	0.34	1
neos-1346382	-176	1.12	17.15	59	-176	1.12	21.86	71
neos-1426635	-176	1.12	16.97	59	-176	1.12	22	71
23588	8166	0.93	19.6	64	8223	1.64	22.2	55
neos-512201	1790.74	248.68	113.43	288	1851.76	260.56	64.56	173
neos-504674	12129.1	233.59	113.6	315	20779.1	471.5	16.88	52
neos-582605	1	?	46.8	54	1	?	48.7	51
neos-1225589	2.43E+09	97.68	26.62	58	4.84E+09	293.13	36.74	84
neos-631164	1.30E+07	18.9	23.33	51	1.31E+07	19.38	19.37	60
neos-955215	567.3	27.05	58.42	140	572	28.1	16.87	52
neos-1440460	-178	1.11	64.2	102	-177.2	1.55	60.71	126
neos-1056905	28	?	33.85	61	29	?	37.55	85
neos-1595230	9	0	6.13	18	9	0	5.44	14
neos-1429461	-101	0.98	93.51	127	-100	1.96	52.08	100
neos-1467067	-103	0.64	18.38	74	-103	0.64	17.05	64
neos-522351	18101.3	1.17	64.71	75	19684.9	10.02	45	60
neos-538867	272	122.95	17.93	81	337	176.22	40.02	96
neos-1200887	-74	0	11.14	26	-72	2.7	37.87	68
neos-1616732	160	?	392.28	77	159	?	605.86	124
neos-825075	-272	0	2.82	12	-272	0	2.43	9
neos-933815	955.7	25.79	37.49	78	972.4	27.99	48.87	123
neos-538916	298	122.38	44.3	89	333	148.5	41.52	82
neos-906865	3175	0	1.87	7	3175	0	1.6	3
neos-863472	11.7541	18.18	63.27	96	11.8101	18.74	20.17	57
binkar10_1	6879.61	2.03	37.87	70	6826.39	1.24	54.75	121
neos11	9	0	12.24	1	9	0	10.41	1
neos-503737	50	0	242.17	34	50	0	40.77	18
neos-593853	1.34E+09	14.21	196.26	149	1.34E+09	13.99	128.1	200
neos-598183	18610.9	0.98	185.89	231	18643.86	1.16	52.2	196
neos-603073	16947	0.93	87.14	132	17096.4	1.82	70.22	152
neos-848150	0	0	35.15	1	0	0	10.18	1
neos-892255	14	0	1.5	3	14	0	4.51	4
neos-933364	817.3	7.47	76.16	124	814.9	7.16	62.63	119
neos-934184	817.3	7.47	75.76	124	814.9	7.16	62.9	119
neos-942323	18	20	191.43	77	18	20	109.84	90
neos-1311124	-181	0.54	60.15	79	-181	0.54	52.94	71
neos-1426662	-44	15.38	68.66	68	-44	15.38	66.25	67
neos-1427181	-102	1.92	112.31	76	-102	1.92	144.07	126
neos-1427261	-127	2.3	193.07	109	-126	3.07	167.27	117
neos-1429185	-76	2.56	78.35	83	-76	2.56	96.41	126
neos-1436709	-128	0.77	115.15	116	-127	1.55	166.31	217
neos-1437164	10	25	24.21	65	10	25	20.95	79
neos-1440457	-179	0.55	190.77	129	-179	0.55	218.79	189
neos-1442119	-180	1.09	122.62	102	-180	1.09	125.94	143
neos-1442657	-154	1.28	85.34	87	-154	1.28	101.18	136
neos-1603512	5	∞	39.59	63	5	∞	88.71	51
Avg		50.19	54.59	76.41		65.38	45.51	83.19

[1] NA: Statistics not available.
[2] ?: Unknown bound.

Method	DW2				RR			
Problem	Obj	Gap	Time	Step	Obj	Gap	Time	Step
neos5	15	0	0.31	7	1	0	0.3	6
neos-584851	-10	9.09	29.78	53	-11	0	16.5	28
neos-881765	0	0	2.19	1	0	0	2.23	1
neos-905856	NA	NA	105.6	51	NA	NA	112.21	51
neos-1121679	109	?	0.19	60	85	?	1.26	79
neos-1211578	-77	0	5.98	43	-77	0	5.62	44
neos-1228986	-123	0	1.56	14	-123	0	1.13	10
neos-1337489	-77	0	5.21	43	-77	0	5.35	44
neos-1420205	69	72.5	3.57	121	79	97.5	2.62	90
neos-1430701	-76.3333	2.13	28.93	95	-76	2.56	30.67	99
neos-1440447	-100	0	4.38	23	-96	4	17.21	76
neos-1460246	2606	12.08	25.15	88	2606	12.08	42.52	149
neos-1480121	43	0	0.12	2	43	0	0.12	2
ran14x18_1	4704	28.26	10.47	85	4657	26.98	19.78	158
rlp1	22	2300	4.43	52	22	2300	4.73	52
roy	3208.96	0	0.26	6	3218.96	0.31	2.42	53
neos14	93018.7	25.13	13.78	89	92358.2	24.24	14.44	96
neos15	104560	34.23	9.95	79	106996.5	37.35	20.99	151
neos-1489999	354	0	50.3	92	354	0	67.94	121
neos-911880	85.23	70.96	12.71	108	99.8	100.18	9.15	76
neos-504815	4493.81	95.7	152.65	500	4457.499	94.12	138.06	500
bienst1	46.75	0	13.2	20	46.75	0	4.88	8
bienst2	55	0.73	71.3	81	54.8571	0.47	83.05	89
mcf2	69	5.07	230.88	94	69	5.07	219.28	85
neos-911970	78.21	54.65	15.34	98	80.31	58.8	19.34	121
neos-631517	1.27E+07	12.4	15.5	51	1.27E+07	12.4	15.1	51
neos-1439395	-180	1.09	28.54	75	-179	1.64	36.11	97
22433	21477	0	0.36	4	21477	0	0.4	4
neos-1620807	6	0	0.33	1	6	0	1.44	1
neos-1346382	-176	1.12	18.5	59	-176	1.12	19.67	61
neos-1426635	-176	1.12	18.34	59	-176	1.12	20.1	61
23588	8329	2.95	0.45	51	8172	1.01	13.1	55
neos-512201	1217.984	137.16	206	500	1573.32	206.34	119.24	310
neos-504674	9798.77	169.5	150.13	427	10365.4	185.08	166.85	469
neos-582605	1	?	45.55	51	1	?	46.96	51
neos-1225589	5.00E+09	306.15	1.32	54	4.84E+09	293.13	152	356
neos-631164	1.30E+07	18.72	20.45	64	1.29E+07	18.1	29.41	96
neos-955215	572	28.1	16.69	52	572	28.1	17.01	52
neos-1440460	-178	1.11	35.2	80	-178	1.11	62.61	131
neos-1056905	28	?	38.78	85	29	?	28.23	54
neos-1595230	9	0	8.04	18	9	0	5.28	14
neos-1429461	-101	0.98	59.55	118	-100.444	1.52	67.26	130
neos-1467067	-103	0.64	20.05	77	-103	0.64	17.22	65
neos-522351	21376.4	19.48	2.11	54	18519.8	3.51	54.14	67
neos-538867	322	163.93	43.91	108	337	176.22	27.15	64
neos-1200887	-74	0	20.59	38	-72	2.7	32.46	57
neos-1616732	160	?	479.02	85	160	?	540.89	93
neos-825075	-272	0	2.17	8	-272	0	2.33	8
neos-933815	1007.8	32.65	21.4	52	1007.8	32.65	21.61	52
neos-538916	303	126.11	76.16	157	324	141.79	48.19	95
neos-906865	3175	0	1.48	3	3175	0	4.56	7
neos-863472	12.01409	20.8	32.82	116	11.8911	19.56	37.23	112
binkar10_1	6840.51	1.45	30.61	68	6840.51	1.45	51.63	113
neos11	9	0	10.08	1	9	0	10.38	1
neos-503737	50	0	158.84	32	50	0	121.46	31
neos-593853	1.33E+09	13.44	64.89	117	1.34E+09	14.55	160.22	126
neos-598183	18663.46	1.26	107.9	157	18670.51	1.3	63.81	167
neos-603073	17140.9	2.08	71.22	156	17091.1	1.79	135.72	283
neos-848150	0	0	10.14	1	0	0	10.16	1
neos-892255	14	0	4.61	4	14	0	6.27	5
neos-933364	834.5	9.74	46.96	97	812.3	6.82	100.19	195
neos-934184	834.5	9.74	47.08	97	812.3	6.82	100.77	195
neos-942323	17	13.33	143.26	120	17	13.33	118.56	99
neos-1311124	-181	0.54	60.33	88	-181	0.54	52.89	74
neos-1426662	-44	15.38	61.63	63	-44	15.38	66.25	66
neos-1427181	-102	1.92	94.05	86	-102	1.92	103.07	95
neos-1427261	-127	2.3	202.53	142	-127	2.3	215.51	149
neos-1429185	-75	3.84	94.43	130	-76	2.56	117.18	150
neos-1436709	-127	1.55	82.47	114	-126	2.32	99.26	131
neos-1437164	10	25	25.1	96	10	25	23.3	85
neos-1440457	-179	0.55	196.28	180	-179	0.55	144.47	124
neos-1442119	-180	1.09	217.8	260	-180	1.09	136.79	159
neos-1442657	-154	1.28	101.06	137	-150	3.84	55.68	79
neos-1603512	5	∞	88.94	51	5	∞	85.71	51
Avg		56.75	48.88	90.68		58.72	52.83	100.03

[1] NA: Statistics not available.
[2] ?: Unknown bound.

Table 16: 30 Runs of Walk-and-Round Performance on COR@L Library

Method	HR					DW1				
Problem	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	MinGap	MaxGap	AvgGap	StDevGap	AvgStep
neos5	0	0	0	0	9.13	0	0	0	0	7.7
neos-584851	0	9.09	0.91	2.77	28.73	0	9.09	1.82	3.7	28.6
neos-881765	0	0	0	0	3	0	0	0	0	1
neos-905856	62.5	75	66.67	7.22	53.7	50	62.5	56.25	8.84	53.73
neos-1121679	?	?	?	?	104.27	?	?	?	?	103.1
neos-1211578	0	1.29	0.26	0.52	30.97	0	1.29	0.52	0.64	41.67
neos-1228986	0	0.81	0.03	0.15	25.8	0	0	0	0	26.57
neos-1337489	0	1.29	0.26	0.52	30.97	0	1.29	0.52	0.64	41.67
neos-1420205	0	15	1	3.81	20.33	72.5	397.5	129.5	109.08	82.53
neos-1430701	1.28	2.56	2.26	0.48	86.9	1.28	3.84	2.79	0.59	106.77
neos-1440447	0	2	0.67	0.96	49.33	0	2	1.45	0.85	83.63
neos-1460246	12.08	12.17	12.09	0.02	97.97	12.08	12.17	12.09	0.02	98.67
neos-1480121	0	0	0	0	4	0	0	0	0	2
ran14x18_1	13.72	16.29	16.2	0.47	53.53	24.25	31.53	28.48	2.41	102.87
rlp1	1900	1900	1900	0	51	2300	2300	2300	0	52
roy	0.31	0.31	0.31	0	57.23	0.31	0.31	0.31	0	53
neos14	20.52	32.85	25.08	2.01	109.3	20.16	27.45	24.73	1.81	123.73
neos15	33.61	41.83	38.61	2.14	78.27	35.02	42.8	38.81	1.92	88.6
neos-1489999	0	0	0	0	55.73	0	0	0	0	117.27
neos-911880	65.92	113.64	89.9	12.71	136.1	55.87	108.19	83.73	13.12	134.03
neos-504815	129.24	362.12	220.71	49.99	274.73	116.68	364.57	158.34	48.98	445.13
bienst1	0	0	0	0	20.3	0	0	0	0	19.13
bienst2	0	3.08	1.11	0.86	99.63	0	2.56	1.05	0.77	103.23
mcf2	1.11	7.74	5.39	1.7	84.2	2.03	8.62	5.54	1.81	91.03
neos-911970	37.66	87.57	62.39	12.33	133.53	47.82	83.1	61.64	10.55	125.87
neos-631517	11.15	11.15	11.15	0	51	12.4	12.4	12.4	0	51
neos-1439395	0.91	1.09	1.08	0.05	76.1	1.09	1.64	1.13	0.14	106.7
22433	0	0	0	0	7.77	0	0	0	0	3.83
neos-1620807	0	0	0	0	1	0	0	0	0	1
neos-1346382	1.12	1.12	1.12	0	59.13	1.12	1.12	1.12	0	72.2
neos-1426635	1.12	1.12	1.12	0	59.13	1.12	1.12	1.12	0	72.2
23588	0	1.01	0.47	0.39	98.33	0.79	2	1.66	0.51	54.83
neos-512201	147.25	299.62	234.75	38.11	329.9	132.86	268.98	203.32	34.62	402.73
neos-504674	183.9	469.93	250.11	73.98	338.9	161.92	471.5	313.45	141.1	283.67
neos-582605	?	?	?	?	54	?	?	?	?	51
neos-1225589	28.43	115.25	63.27	31.54	58.77	293.13	293.13	293.13	0	112.33
neos-631164	18.9	18.9	18.9	0	51	17.42	20.17	19.07	0.66	99.7
neos-955215	13.93	41.9	27.85	5.95	80.03	25.15	28.1	28	0.54	53.1
neos-1440460	0.55	1.11	0.84	0.27	107.3	0.55	1.66	1.46	0.3	115
neos-1056905	?	?	?	?	100.7	?	?	?	?	75.33
neos-1595230	0	0	0	0	13.5	0	0	0	0	14.07
neos-1429461	0.98	0.98	0.98	0	107.3	0.98	1.96	1.67	0.46	107
neos-1467067	0.64	0.64	0.64	0	81.33	0.64	0.64	0.64	0	65.77
neos-522351	0.4	1.23	0.81	0.24	117.4	2.97	10.02	3.64	1.22	70.23
neos-538867	122.95	122.95	122.95	0	51	65.57	217.21	163.25	27.23	90.17
neos-1200887	0	2.7	1.31	1.31	58.1	0	2.7	1.67	1.16	63.13
neos-1616732	?	?	?	?	79.37	?	?	?	?	90.87
neos-825075	0	0	0	0	12.27	0	0	0	0	8.27
neos-933815	25.79	39.45	37.15	4.45	62.33	27.79	32.65	32.02	1.5	58.03
neos-538916	51.49	201.49	119.18	31.73	110.03	85.07	174.62	138.03	23.38	97.07
neos-906865	0	0	0	0	7.1	0	0	0	0	3.67
neos-863472	17.52	24.15	21.12	1.77	96.3	18.21	100	23.18	14.59	96.17
binkar10_1	1.45	5.01	2.11	0.57	90.93	1.24	1.45	1.42	0.06	98.87
neos11	0	0	0	0	1	0	0	0	0	1
neos-503737	0	0	0	0	27.67	0	0	0	0	25.33
neos-593853	9.08	62.35	26.9	13.81	133.17	8.71	337.89	42.15	77.32	123.6
neos-598183	0.9	15.35	1.78	2.57	142	1.09	2.51	1.48	0.36	133
neos-603073	0.25	18.14	8.14	7.8	135.13	1.36	18.38	3.51	4.11	143.03
neos-848150	0	0	0	0	1	0	0	0	0	1
neos-892255	0	0	0	0	3	0	0	0	0	3.83
neos-933364	4.04	12.77	7.44	2.12	111.67	4.06	9.64	6.95	1.29	120.8
neos-934184	4.04	12.77	7.44	2.12	111.67	4.06	9.64	6.95	1.29	120.8
neos-942323	13.33	26.66	15.33	3.57	93.43	13.33	26.66	19.33	4.75	91.3
neos-1311124	0.54	0.54	0.54	0	85.8	0.54	1.09	0.56	0.1	82.13
neos-1426662	15.38	15.38	15.38	0	67.37	15.38	15.38	15.38	0	66.13
neos-1427181	1.92	1.92	1.92	0	89	1.92	1.92	1.92	0	99.7
neos-1427261	2.3	3.07	2.38	0.23	139.17	2.3	3.84	2.61	0.52	146.6
neos-1429185	2.56	3.84	2.69	0.39	97.07	2.56	3.84	3.03	0.63	122.4
neos-1436709	0.77	1.55	0.95	0.32	111.43	0.77	3.1	1.59	0.56	165.2
neos-1437164	12.5	37.5	21.25	7.45	75.83	12.5	37.5	23.33	6.34	82.2
neos-1440457	0.41	1.11	0.56	0.11	134.53	0.55	1.11	0.82	0.28	166.2
neos-1442119	0.54	1.09	0.82	0.25	117.03	0.91	2.19	1.28	0.3	156.23
neos-1442657	0.96	1.28	1.26	0.08	86.57	0.96	2.56	1.29	0.26	119.27
neos-1603512	∞	∞	∞	?	61	∞	∞	∞	?	51
Avg	42.84	61.56	50.13	4.74	79.86	53.07	81.13	62.13	7.98	90.4

[1] ?: Unknown upper bound or statistics not available.

Method	DW2					RR				
Problem	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	MinGap	MaxGap	AvgGap	StDevGap	AvgStep
neos5	0	0	0	0	8.43	0	0	0	0	8.23
neos-584851	0	9.09	1.52	3.45	25.23	0	9.09	0.91	2.77	19.97
neos-881765	0	0	0	0	1	0	0	0	0	1
neos-905856	87.5	87.5	87.5	0	51.9	37.5	75	56.25	26.52	51.67
neos-1121679	?	?	?	?	76.03	?	?	?	?	91.13
neos-1211578	0	1.29	0.34	0.58	34.73	0	1.29	0.47	0.63	41.7
neos-1228986	0	0.81	0.03	0.15	24.67	0	0	0	0	21.33
neos-1337489	0	1.29	0.34	0.58	34.73	0	1.29	0.47	0.63	41.7
neos-1420205	0	122.5	75.58	22.58	110.67	72.5	397.5	128.85	100.13	81.03
neos-1430701	2.13	3.84	2.87	0.6	103.07	2.13	3.84	2.83	0.58	98.07
neos-1440447	0	4	1.4	1.22	68.63	0	4	1.87	1.17	72.1
neos-1460246	12.08	12.17	12.08	0.02	102.97	12.08	12.17	12.09	0.02	102.03
neos-1480121	0	0	0	0	2	0	0	0	0	2
ran14x18_1	19.83	31.53	27.23	3.46	110.9	22.4	31.53	28.67	2.27	93.57
rlp1	2200	2300	2210	30.51	64.93	2300	2300	2300	0	52
roy	0	0.31	0.01	0.06	15.73	0.31	0.31	0.31	0	53
neos14	22.48	27.15	24.43	1.31	144.43	21.75	27.69	25.2	1.3	126.5
neos15	28.52	36.23	32.88	1.98	157.4	34.83	42.78	38.9	2.09	102.97
neos-1489999	0	0	0	0	112.97	0	0	0	0	118.4
neos-911880	66.79	117.46	89.49	13.19	122.33	52.16	121.51	83.69	17.53	133.2
neos-504815	83.29	227.55	116.61	28.68	429.27	94.12	372.26	166.72	59.63	383.1
bienst1	0	0	0	0	13.8	0	0	0	0	13.87
bienst2	0	3.47	1.21	0.77	107.9	0	2.86	1.17	0.7	94.1
mcf2	1.45	8.62	5.11	1.61	85.77	1.26	8.12	5.57	1.46	81.47
neos-911970	37.98	89.27	65.2	12.74	118.3	44.6	84.72	62.51	11.15	122.53
neos-631517	12.4	12.4	12.4	0	51	12.4	12.4	12.4	0	51
neos-1439395	1.09	1.64	1.18	0.21	101.5	1	1.64	1.13	0.13	111.37
22433	0	0.2	0.04	0.08	13.1	0	0	0	0	3.97
neos-1620807	0	0	0	0	1	0	0	0	0	1
neos-1346382	1.12	1.12	1.12	0	72.47	1.12	1.12	1.12	0	69.87
neos-1426635	1.12	1.12	1.12	0	72.47	1.12	1.12	1.12	0	69.87
23588	0.74	2.95	1.86	0.74	55.4	0.79	2	1.69	0.51	54.67
neos-512201	137.16	219.08	172.85	24.71	415.6	163.71	277	210.48	26.96	378.33
neos-504674	157.51	207.13	181.07	13.66	422.13	165.04	471.5	365.73	137.83	204
neos-582605	?	?	?	?	51	?	?	?	?	51
neos-1225589	246.85	306.15	298.85	15.21	64.43	253.84	293.13	291.82	7.17	114.6
neos-631164	17.48	21.1	19.26	0.86	98.3	16.4	20.22	18.77	0.92	102.37
neos-955215	17.67	28.1	27.28	2.18	59.5	23.98	28.1	27.96	0.75	53.77
neos-1440460	1	2.22	1.42	0.3	105.73	1.11	2.22	1.42	0.3	107.2
neos-1056905	?	?	?	?	68.47	?	?	?	?	75.67
neos-1595230	0	0	0	0	12.5	0	0	0	0	15.03
neos-1429461	0.98	2.94	1.49	0.55	111.07	0.98	1.96	1.39	0.48	118.57
neos-1467067	0.64	0.64	0.64	0	66.77	0.64	0.64	0.64	0	66.67
neos-522351	2.65	59.16	28.02	18.61	57.17	2.97	10.02	3.61	1.23	68.1
neos-538867	102.45	217.21	151.5	27.15	97.27	90.16	213.11	158.74	32.7	103.43
neos-1200887	0	2.7	1.62	1.3	56.4	0	2.7	2.34	0.86	66.6
neos-1616732	?	?	?	?	96.5	?	?	?	?	86.53
neos-825075	0	0	0	0	8.63	0	0	0	0	7.8
neos-933815	25.88	32.65	32.04	1.6	56.77	29.55	32.65	32.36	0.72	58.67
neos-538916	104.47	185.82	142.53	21.95	109	89.55	167.16	134.17	17.18	102.77
neos-906865	0	0	0	0	3.6	0	0	0	0	3.73
neos-863472	17.65	26.29	20.63	1.75	95.07	17.52	23.76	20.01	1.57	107.23
binkar10_1	0.99	2.01	1.39	0.16	105.1	1.2	1.45	1.4	0.08	110.77
neos11	0	0	0	0	1	0	0	0	0	1
neos-503737	0	0	0	0	28.4	0	0	0	0	25.8
neos-593853	8.36	613.93	59.69	127.72	88.5	11.52	154.42	24.57	29.77	113.9
neos-598183	1.05	2.85	1.51	0.33	123.33	1.1	2.46	1.43	0.27	137.7
neos-603073	1.12	18.21	3.31	3.13	145.5	0.85	18.68	5.71	6.28	160.5
neos-848150	0	0	0	0	1	0	0	0	0	1
neos-892255	0	0	0	0	3.93	0	0	0	0	3.83
neos-933364	4.42	12.29	7.43	1.55	114.7	4.04	10.07	7.37	1.38	117.9
neos-934184	4.42	12.29	7.43	1.55	114.7	4.04	10.07	7.37	1.38	117.9
neos-942323	13.33	26.66	17.33	3.76	100	13.33	20	17.33	3.32	90.2
neos-1311124	0.54	0.54	0.54	0	82.47	0.54	0.54	0.54	0	78
neos-1426662	15.38	15.38	15.38	0	64.37	15.38	15.38	15.38	0	65.13
neos-1427181	1.92	1.92	1.92	0	97.4	1.92	1.92	1.92	0	97.13
neos-1427261	2.3	3.07	2.43	0.29	155.9	2.3	3.46	2.62	0.39	148
neos-1429185	2.56	5.98	3.63	0.98	113.07	2.56	5.12	3.39	0.81	113.67
neos-1436709	0.77	2.32	1.78	0.42	135.97	0.77	3.1	1.78	0.5	144.47
neos-1437164	12.5	37.5	22.92	6.63	84.2	12.5	37.5	23.75	8.27	84.93
neos-1440457	0.55	1.11	0.68	0.24	168.07	0.55	1.11	0.7	0.25	169.6
neos-1442119	0.91	2.19	1.28	0.38	171.3	0.82	3.57	1.39	0.64	157.8
neos-1442657	0.96	2.56	1.31	0.24	115	0.96	3.84	1.48	0.6	109.17
neos-1603512	∞	∞	∞	?	51	∞	∞	∞	?	51
Avg	49.93	74.82	57.55	5.91	91.48	53.01	77.59	62.72	7.14	87.54

[1] ?: Unknown upper bound or statistics not available.

Table 17: 30 Runs of Walk-and-Round Performance on COR@L Library with Expanded Regions

Method	HR					DW1				
Problem	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	MinGap	MaxGap	AvgGap	StDevGap	AvgStep
neos5	0	0	0	0	10.33	0	0	0	0	8.47
neos-584851	0	9.09	0.91	2.77	23.43	0	9.09	0.61	2.31	24.77
neos-881765	0	0	0	0	3	0	0	0	0	9
neos-905856	75	75	75	?	52.5	62.5	62.5	62.5	0	52.43
neos-1121679	?	?	?	?	106.8	?	?	?	?	85.97
neos-1211578	0	1.29	0.3	0.55	33.4	0	1.29	0.28	0.53	34.03
neos-1228986	0	0.81	0.03	0.15	23.5	0	0	0	0	25.47
neos-1337489	0	1.29	0.3	0.55	33.4	0	1.29	0.28	0.53	34.03
neos-1420205	0	15	5	7.19	42.3	72.5	355	181	108.99	110.47
neos-1430701	1.28	2.56	2.13	0.51	93.6	2.13	3.84	2.8	0.51	95.97
neos-1440447	0	2	0.6	0.89	54.8	0	4	2	1.02	83.2
neos-1460246	12.08	12.17	12.09	0.03	92.43	12.08	12.17	12.08	0.02	84.23
neos-1480121	0	0	0	0	4	0	0	0	0	2
ran14x18_1	10.48	16.29	16.1	1.06	52.63	26.59	31.53	29.33	1.59	95.7
rlp1	1900	1900	1900	0	51	1900	1900	1900	0	51
roy	0.31	0.31	0.31	0	57.13	0.31	0.31	0.31	0	53
neos14	21.58	37.62	24.93	2.92	104.1	22.92	30.5	26.13	2.1	87.1
neos15	32.35	40.93	37.26	2.33	92.1	35	42.2	38.73	1.91	120.1
neos-1489999	10.73	10.73	10.73	0	1	0	4.8	0.98	1.11	16.37
neos-911880	59.18	135.11	88.46	15.93	145.13	59.08	111.94	88.57	13.52	122.5
neos-504815	101.1	372.72	219.34	64.9	325.67	103.6	244.05	155.05	37.77	437.63
bienst1	0	0.53	0.02	0.1	21.27	0	0	0	0	16.3
bienst2	0.12	3.78	1.39	0.82	99.77	0	2.93	1.2	0.72	98.27
mcf2	1.11	7.37	4.76	1.64	93.43	0.92	7.87	5.02	1.87	82.23
neos-911970	43.55	101.35	65.95	14.48	124.07	37.21	99	69.47	10.64	110.77
neos-631517	16.11	16.43	16.4	0.08	55.9	12.86	15.81	15.01	0.87	70.47
neos-1439395	0.91	1.09	1.08	0.05	73.53	0.91	1.64	1.17	0.2	113.37
22433	0	0	0	0	6	0	0	0	0	2
neos-1620807	0	0	0	0	1	0	0	0	0	1
neos-1346382	1.12	1.12	1.12	0	61.6	1.12	1.12	1.12	0	62.83
neos-1426635	1.12	1.12	1.12	0	61.6	1.12	1.12	1.12	0	62.83
23588	0.11	0.64	0.23	0.15	83.77	0.64	0.75	0.68	0.05	57.03
neos-512201	164.99	309.47	233.62	35.61	338.93	155.16	262.94	208.57	29.93	367.77
neos-504674	170.55	303.92	210.1	27.49	398.53	158.48	194.42	173.75	9.95	460.33
neos-582605	?	?	?	?	51	?	?	?	?	51
neos-1225589	30.27	128.88	61.85	29.94	58.83	228.69	281.66	265.44	9.86	248.8
neos-631164	18.64	18.64	18.64	0	51	17.07	18.34	18.23	0.31	54.83
neos-955215	16.86	38.32	25.69	5.3	83.73	27.03	28.1	28.06	0.2	53.07
neos-1440460	0.55	1.38	0.93	0.26	102.03	1.11	1.66	1.44	0.24	104.63
neos-1056905	?	?	?	?	96.27	?	?	?	?	52
neos-1595230	0	0	0	0	13.7	0	0	0	0	14.2
neos-1429461	0.98	0.98	0.98	0	108.13	0.98	1.96	1.63	0.47	105.8
neos-1467067	0.64	0.64	0.64	0	79.73	0.64	0.64	0.64	0	66.77
neos-522351	1.11	1.97	1.33	0.19	101.5	9.18	15.1	14.36	1.01	65.47
neos-538867	159.83	159.83	159.83	0	52	114.75	213.11	150.95	26.89	93.73
neos-1200887	0	2.7	1.74	1.28	58.33	0	2.7	2.03	1.11	63.47
neos-1616732	?	?	?	?	86.43	?	?	?	?	98.43
neos-825075	0	0	0	0	10.97	0	0	0	0	8.3
neos-933815	23.72	39.45	34.89	5.33	70.5	23.78	32.65	31.8	2.08	60.77
neos-538916	70.89	103.73	101.52	6.79	58.97	88.8	200	138.25	28.83	113.43
neos-906865	0	0	0	0	7.1	0	0	0	0	3.47
neos-863472	17.68	25.58	21.14	1.61	91.27	17.78	25.95	20.56	2.13	96.77
binkar10_1	1.82	2.28	1.96	0.17	89.6	0.99	2.03	1.38	0.17	116
neos11	0	0	0	0	1	0	0	0	0	1
neos-503737	0	0	0	0	25.1	0	0	0	0	24.07
neos-593853	8.13	74.98	35.57	15.85	164.1	9.69	428.84	47.71	90.21	102.4
neos-598183	15.35	15.35	15.35	0	53	15.86	15.86	15.86	0	51
neos-603073	61.64	61.69	61.64	0.01	72.97	61.64	61.64	61.64	0	53.8
neos-848150	0	0	0	0	1	0	0	0	0	1
neos-892255	0	0	0	0	3	0	0	0	0	3.83
neos-933364	3.54	12.49	7.05	2.04	117.93	4	10.35	7.46	1.54	106.47
neos-934184	3.54	12.49	7.05	2.04	117.93	4	10.35	7.46	1.54	106.47
neos-942323	13.33	26.66	16.22	4.17	93.43	13.33	26.66	19.33	3.65	88.53
neos-1311124	0.54	0.54	0.54	0	88	0.54	0.54	0.54	0	82.43
neos-1426662	15.38	15.38	15.38	0	68.43	15.38	15.38	15.38	0	65.63
neos-1427181	1.92	1.92	1.92	0	85.43	1.92	1.92	1.92	0	97.73
neos-1427261	2.3	3.07	2.38	0.23	144.2	2.3	3.07	2.45	0.31	150.3
neos-1429185	2.56	3.84	2.6	0.23	100.47	2.56	5.12	3.31	0.96	111.8
neos-1436709	0.77	1.55	1.02	0.35	110.5	1.55	4.65	2.12	0.86	146.57
neos-1437164	12.5	37.5	22.08	6.3	73.23	12.5	37.5	21.67	6.51	90.2
neos-1440457	0.55	0.55	0.55	0	140.47	0.55	1.11	0.7	0.25	163.43
neos-1442119	0.54	1.09	0.88	0.23	118.9	0.91	2.74	1.33	0.44	144.43
neos-1442657	0.96	1.28	1.25	0.1	85.5	0.96	3.84	1.39	0.53	115.47
neos-1603512	∞	∞	∞	?	51	∞	∞	∞	?	51
Avg	44.64	60.29	51.13	3.86	78.89	48.25	70.49	55.89	5.97	87.89

[1] ?: Unknown upper bound or statistics not available.

Method	HR					DW1				
Problem	MinGap	MaxGap	AvgGap	StDevGap	AvgStep	MinGap	MaxGap	AvgGap	StDevGap	AvgStep
neos5	0	0	0	0	8.37	0	0	0	0	8.3
neos-584851	0	9.09	0.61	2.31	25.37	0	9.09	0.3	1.66	25.57
neos-881765	0	0	0	0	9.4	0	0	0	0	13.77
neos-905856	?	?	?	?	51	62.5	62.5	62.5	?	51.67
neos-1121679	?	?	?	?	70.33	?	?	?	?	85.77
neos-1211578	0	1.29	0.22	0.49	29.3	0	1.29	0.17	0.45	27.03
neos-1228986	0	0	0	0	22.4	0	0	0	0	25.4
neos-1337489	0	1.29	0.22	0.49	29.3	0	1.29	0.17	0.45	27.03
neos-1420205	50	137.5	80.33	15.04	106.77	72.5	355	156.67	101.37	111.3
neos-1430701	2.13	3.84	3.12	0.64	88.07	1.28	3.84	2.65	0.57	99.6
neos-1440447	0	4	2.01	0.89	87.2	0	3	1.77	0.73	81.83
neos-1460246	12.08	12.17	12.09	0.03	79.8	12.08	12.08	12.08	0	88.17
neos-1480121	0	0	0	0	2	0	0	0	0	2
ran14x18_1	18.47	31.53	26.27	3.31	152.13	18.63	31.53	28.27	3.3	98.43
rlp1	1900	1900	1900	0	51	1900	1900	1900	0	51
roy	0	0.31	0.08	0.14	33.63	0.31	0.31	0.31	0	53
neos14	20.7	28.03	25	1.84	123.6	22.01	29.34	25.53	1.85	100.47
neos15	32.97	37.92	36.01	1.27	125.83	34.68	41.35	38.09	1.61	110.47
neos-1489999	0	5.64	1.32	1.57	16.5	0	4.23	1.17	1.09	15.8
neos-911880	56.62	108.33	84.88	13.51	130.53	63.24	112.26	84.54	13.33	124.47
neos-504815	93.28	138.04	108.92	12.58	440.93	104.12	365.74	152.86	52.93	440.57
bienst1	0	0	0	0	14.67	0	0	0	0	14.1
bienst2	0	2.56	1.25	0.61	93.1	0	2.56	0.98	0.7	91.47
mcf2	1.26	8.13	4.78	1.71	86.8	1.42	7.55	4.31	1.54	80.7
neos-911970	40.49	88.76	63.68	12.41	120.8	45.45	93.46	63.54	11.29	129.33
neos-631517	12.1	15.81	15.01	0.94	72.2	12.5	15.81	14.79	0.81	80.1
neos-1439395	1.09	1.64	1.2	0.22	99.3	0.91	1.64	1.12	0.14	110.7
22433	0	0	0	0	2	0	0	0	0	2
neos-1620807	0	0	0	0	1	0	0	0	0	1
neos-1346382	1.12	1.12	1.12	0	63.9	1.12	1.12	1.12	0	64.77
neos-1426635	1.12	1.12	1.12	0	63.9	1.12	1.12	1.12	0	64.77
23588	0.28	0.97	0.78	0.23	54.77	0.28	0.75	0.69	0.09	55.97
neos-512201	134.15	245.65	180.66	30.95	392.53	141.6	251.21	201.33	25.58	397.4
neos-504674	99.97	206.16	180.08	20.42	385.43	157.28	254.11	185.64	25.88	413.93
neos-582605	?	?	?	?	51	?	?	?	?	51
neos-1225589	223.67	306.15	286.66	22.63	63.33	252.73	275.71	265.08	5.78	240.37
neos-631164	17.58	18.34	18.25	0.2	58.37	17.5	18.34	18.24	0.23	58.07
neos-955215	20.42	28.1	27.81	1.41	53.1	25.3	28.1	27.87	0.71	55.03
neos-1440460	0.55	1.66	1.37	0.31	99.47	0.74	2.22	1.43	0.33	105.27
neos-1056905	?	?	?	?	54.87	?	?	?	?	52
neos-1595230	0	0	0	0	12.63	0	0	0	0	14.3
neos-1429461	0.98	2.94	1.47	0.56	112.5	0.98	1.96	1.63	0.47	104.83
neos-1467067	0.64	0.64	0.64	0	66.77	0.64	0.64	0.64	0	65.97
neos-522351	1.44	63.5	34.75	18.32	58.13	9.95	15.38	14.25	1.21	68.43
neos-538867	94.26	188.52	148.36	24.72	91.33	98.36	221.31	143.85	25.95	91.87
neos-1200887	0	2.7	1.8	1.19	63.67	0	2.7	1.62	1.3	55.37
neos-1616732	?	?	?	?	93.87	?	?	?	?	90.8
neos-825075	0	0	0	0	8	0	0	0	0	8.73
neos-933815	28.36	32.65	32.01	1.23	59.33	27.65	32.65	32.35	0.99	56.93
neos-538916	99.25	170.89	135.27	23.61	98.47	88.05	200.74	138.5	28.18	106.33
neos-906865	0	0	0	0	3.6	0	0	0	0	3.73
neos-863472	18.08	23.89	20.9	1.55	95.23	17.52	24.22	20.74	1.86	91.6
binkar10_1	1.24	2.03	1.43	0.18	111.57	1.24	2.03	1.53	0.24	95.3
neos11	0	0	0	0	1	0	0	0	0	1
neos-503737	0	4	0.13	0.73	25.73	0	0	0	0	23.5
neos-593853	8.75	410.11	28.73	72.2	98.4	7.74	267.54	24.25	46.28	120.67
neos-598183	14.77	15.86	15.8	0.24	51.97	15.86	15.86	15.86	0	51
neos-603073	61.64	61.64	61.64	0	55.73	61.64	61.64	61.64	0	54.7
neos-848150	0	0	0	0	1	0	0	0	0	1
neos-892255	0	0	0	0	3.83	0	0	0	0	3.9
neos-933364	4.34	10.54	7.02	1.4	123.43	4.58	11.76	7.21	1.46	122.33
neos-934184	4.34	10.54	7.02	1.4	123.43	4.58	11.76	7.21	1.46	122.33
neos-942323	13.33	26.66	18.67	4.07	91.5	13.33	26.66	19.33	3.65	81.77
neos-1311124	0.54	0.54	0.54	0	83.67	0.54	0.54	0.54	0	83.7
neos-1426662	15.38	15.38	15.38	0	64.63	15.38	15.38	15.38	0	64.13
neos-1427181	1.92	1.92	1.92	0	96.33	1.92	1.92	1.92	0	98.9
neos-1427261	2.3	3.07	2.48	0.33	150.77	2.3	3.84	2.53	0.41	153.83
neos-1429185	2.56	5.12	3.29	0.73	119.77	2.56	5.12	3.39	0.87	114.6
neos-1436709	0.77	3.1	1.7	0.6	150.93	1.16	3.1	1.83	0.54	144.1
neos-1437164	12.5	37.5	21.25	6.69	91.9	12.5	37.5	22.08	6.3	85.47
neos-1440457	0.55	1.11	0.84	0.28	160.03	0.55	1.38	0.82	0.3	160.07
neos-1442119	0.54	2.74	1.37	0.47	162.13	0.91	2.19	1.24	0.3	155.67
neos-1442657	0.96	3.84	1.46	0.57	111.5	1.28	2.99	1.38	0.38	110.67
neos-1603512	∞	∞	∞	?	51	∞	∞	∞	?	51
Avg	46.02	65.39	53.39	4.52	85.82	48.21	70.51	54.91	5.51	88.03

[1] ?: Unknown upper bound or statistics not available.