Can Interior Solutions Help in Solving Mixed Integer Programs?

Sanjay Mehrotra
joint work with Kuo-Ling Huang, Utku Koc, Zhifeng Li

Industrial Engineering and Management Sciences
Northwestern University

June 3, 2014

Giant Orion carrying servant Cedalion on his shoulders

The Concept of a Vertex

GEORGE B. DANTZIG, Programming in a linear structure, *Econometrica* 17 (1949) 73–74. Copyright © 1949 by the Econometric Society.

GEORGE B. DANTZIG, Maximization of a linear function of variables subject to linear inequalities, in T.C. KOOPMANS, ed., *Activity Analysis of Production and Allocation*, New York: John Wiley & Sons, 1951, Chapter XXI, pp. 339–347. Copyright © 1951 by the Yale University Press.

The Concept of Cutting Planes

OUTLINE OF AN ALGORITHM FOR INTEGER SOLUTIONS TO LINEAR PROGRAMS

BY RALPH E. GOMORY[1]

Communicated by A. W. Tucker, May 3, 1958

SOLVING LINEAR PROGRAMMING PROBLEMS IN INTEGERS[1]

BY

RALPH E. GOMORY

[1] This work was supported in part by the Princeton–IBM mathematics research project.

[2] More recently a FORTRAN program on an IBM 704 has been used to run problems up to $m = n = 15$. The problems (only a few were run) ran rapidly.

The Concept of Branching on Hyperplanes

MATHEMATICS OF OPERATIONS RESEARCH
Vol 8, No. 4, November 1983
Printed in U.S.A.

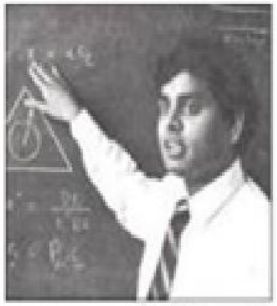
INTEGER PROGRAMMING WITH A FIXED NUMBER OF VARIABLES*

H. W. LENSTRA, JR.

Universiteit van Amsterdam

It is shown that the integer linear programming problem with a fixed number of variables is polynomially solvable. The proof depends on methods from geometry of numbers.

Interior Points, Log-barrier, and Ellipsoidal Rounding



A New Polynomial-Time Algorithm for Linear Programming

N. Karmarkar

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Careful and Efficient Implementations of Algorithms

IBM ILOG CPLEX V12.1

User's Manual for CPLEX

IBM

Processor Speed and Parallel Computing


Outline: Towards a Framework for Solving MIP in a Parallel Computing Environment

1. **Reduce the Number of Branches**. Modified Lenstra's Algorithm for solving Convex IP feasibility problems
   - Barrier Solutions
   - LLL Lattice Basis Reduction
   - Computational experience with a LLL-based heuristic for branching disjunctions (MICP)

Outline: Towards a Framework for Solving MIP in a Parallel Computing Environment

1. **Reduce the Number of Branches**. Modified Lenstra's Algorithm for solving Convex IP feasibility problems
   - Barrier Solutions
   - LLL Lattice Basis Reduction
   - Computational experience with a LLL-based heuristic for branching disjunctions (MICP)
2. **Generate feasible Integer Solutions**. Random Walks in the Interior
   - Walk-and-round heuristic for MILP
   - Walk-relax-round heuristic for MICP
   - Walk-vertex-round heuristic for MILP
   - Parallel walks and integer solutions

Outline: Towards a Framework for Solving MIP in a Parallel Computing Environment

1. **Reduce the Number of Branches**. Modified Lenstra's Algorithm for solving Convex IP feasibility problems
   - Barrier Solutions
   - LLL Lattice Basis Reduction
   - Computational experience with a LLL-based heuristic for branching disjunctions (MICP)
2. **Generate feasible Integer Solutions**. Random Walks in the Interior
   - Walk-and-round heuristic for MILP
   - Walk-relax-round heuristic for MICP
   - Walk-vertex-round heuristic for MILP
   - Parallel walks and integer solutions
3. **Cutting the Non-optimal Vertex Solutions**.
   - Can it help?
   - Limited experience with Gomory and Split Cuts as in Coin-OR
4. Concluding Remarks

Problem Formulation:

General Mixed Integer Convex Programming (MICP) Feasibility problem:

$$\begin{aligned}\mathcal{C} \quad = \quad & \{Rx = r, \\ & c_i(x) \leq 0, \text{ for } i = 1, \cdots, m, \\ & x_i \in \mathbb{Z}_+, i = 1, \cdots, n, \\ & x_i \in \mathbb{R}, i = n+1, \cdots, n+\bar{n}.\}\end{aligned}$$

where $c_i(x) : \mathbb{R}^{n+\bar{n}} \longrightarrow \mathbb{R}$ for $i = 1, \cdots, m$ are convex functions.

$$\text{Is } \mathcal{C} \bigcap \mathbb{Z}^n \neq \varnothing?$$

We will use notation $\mathcal{P}$ whenever $\mathcal{C}$ is a polyhedra, i.e., $g_i(\cdot)$ are linear (MILP); we will use notation $\mathcal{Y}$ whenever the polyhedral set is assumed to be full dimensional.

Basic Steps in Lenstra's Algorithm

S1. *(Assume $\mathcal{Y}$ is full dimensional with dimension k, and the Problem is Linear Pure Integer Program)* **Full Dimensionality Check**: If $\mathcal{Y}$ is not full dimensional, then project $\mathcal{Y}$ to a lower dimension

S2. **Ellipsoidal Rounding**:
Find a positive definite matrix $D \in \mathbb{R}^{k\times k}$, and a center d, so that the ellipsoidal approximation of $\mathcal{Y}$: $\mathcal{E}(w, D) := \{y \in \mathbb{R}^k \mid \|y - d\|_D \leq 1\}$ satisfies

$$\mathcal{E}(w, D) \subseteq \mathcal{Y} \subseteq \mathcal{E}(w, \gamma D), \gamma < 1.$$

$\gamma = O(1/k)$ if log-barrier solution is used.

S3. **Basis Reduction for finding a thin direction**:
Given a D find a reduced basis $b_1, \ldots, b_k$ of the lattice $\mathbb{Z}^k$ in $\|\cdot\|_D$.

S4. **Feasibility Check** Let $b_1, \ldots, b_k$ be reduced basis vectors from S3. Write $d = \sum_{i=1}^k \beta_i b_i$ and compute $\hat{y} = \sum_{i=1}^k \lfloor \beta_i \rceil b_i$. If $\hat{y} \in \mathcal{Y}$, we have found an integer feasible solution. Otherwise, find a primitive vector u satisfying $u^T b_i = 0$ for $i = 1, \ldots, k$.

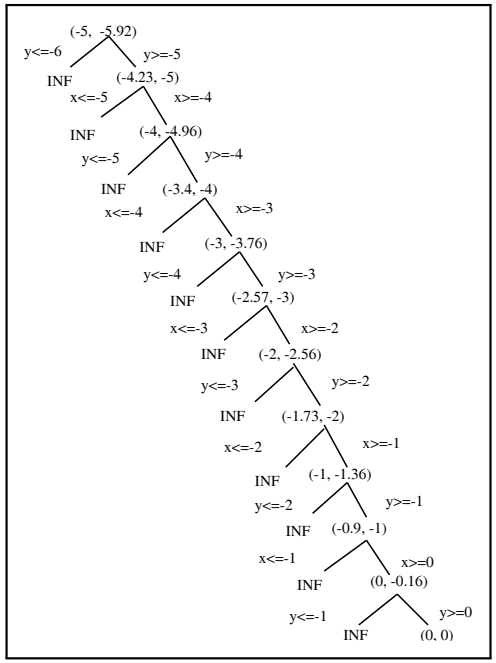
S5. **Branching on hyperplanes**: Add hyperplanes $u^T x = \mu$ for $\mu \in \mathbb{Z}$ for which $\mathcal{Y} \cap \{u^T x = \mu\}$ is feasible; go to Step 1

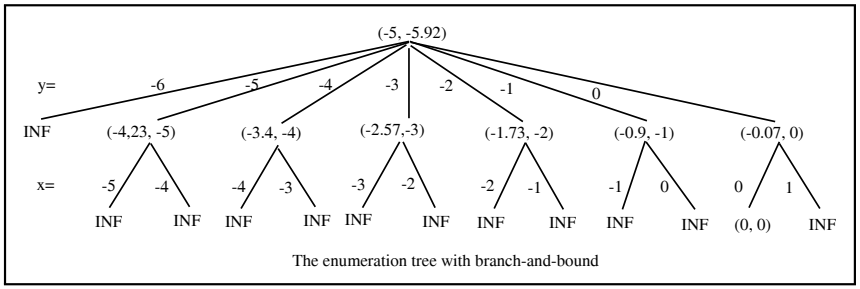
S5'. We can use split-disjunctions instead of branching hyperplanes.

When is Branching on Hyperplane Useful?

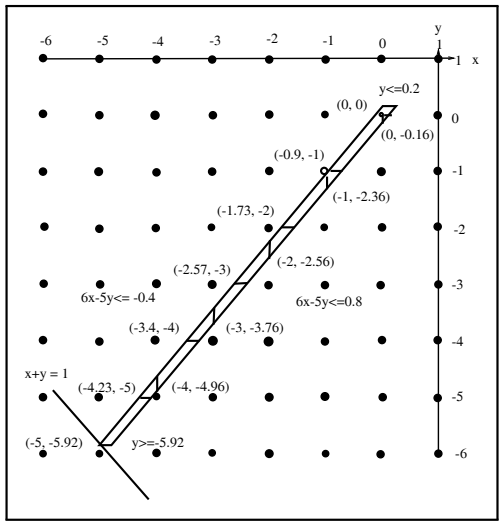
$$\begin{aligned} \min_{x,y\in\mathbb{Z}} \quad & x + y \\ \text{subject to} \quad & 0.4 \le 6x - 5y \le 0.8 \\ & -5.92 \le y \le 0.2 \end{aligned}$$

The solution of relaxation problem is at $(-5, -5.92)$, and the optimal solution is $(0, 0)$.





The enumeration tree with branch-and-bound



Lenstra's Result using LLL Reduced Lattice Basis

Definition

Given a direction u and a convex set $\mathcal{C}$, the integer width of $\mathcal{C}$ is given by:
$\mathcal{W}_I(u, \mathcal{C}) := \lfloor \max \{u^T x | x \in \mathcal{C}\} \rfloor - \lceil \min \{u^T x | x \in \mathcal{C}\} \rceil + 1.$

Theorem

(Lenstra '82) For the choice of u in Lenstra's algorithm, $\mathcal{W}_I(u, \mathcal{Y}) \leq 2k(k+1)2^{k(k-1)/4}$.

Lenstra's Result using LLL Reduced Lattice Basis

Definition

Given a direction u and a convex set $\mathcal{C}$, the integer width of $\mathcal{C}$ is given by:
$\mathcal{W}_I(u, \mathcal{C}) := \lfloor \max \{u^T x | x \in \mathcal{C}\} \rfloor - \lceil \min \{u^T x | x \in \mathcal{C}\} \rceil + 1.$

Theorem

(Lenstra '82) For the choice of u in Lenstra's algorithm, $\mathcal{W}_I(u, \mathcal{Y}) \leq 2k(k+1)2^{k(k-1)/4}$.

Definition

The lattice generated by $B := [b_1, \ldots, b_k]$, $b_i \in \mathbb{R}^n$, $n \geq k$ is the set
$\mathcal{L}(B) := \{x \in \mathbb{R}^n \mid x = \sum_{i=1}^{k} \mathbb{Z} b_i\}$. *$B$ is a basis of $\mathcal{L}(B)$ if it is minimal.*

Definition

Let E (e.g., D) be as positive semi-definite matrix, $\|b\|_E^2 = b^E b \neq 0$ for vectors b of interest. Let $\hat{B} = [\hat{b}_1, \ldots, \hat{b}_k]$ be the Gram-Schmidt orthogonal basis computed by the orthogonalization $\hat{b}_i = b_i - \sum_{j=1}^{i-1} \Gamma_{j,i} \hat{b}_j, i = 1, \ldots, k,$ where $\Gamma_{j,i} = b_i^T E \hat{b}_j / \|\hat{b}_j\|_E^2,$ and $\hat{b}_1 = b_1,$ It is assumed that $\|\cdot\|_E \neq 0$ for the vectors of interest. A basis $b_1, \ldots, b_k$ of a lattice $\mathcal{L}(B)$ is called an LLL-reduced basis, *for $\delta \in (\frac{1}{4}, 1)$, if it satisfies:*

LLL1. $|\Gamma_{j,i}| \leq 1/2$ *for* $1 \leq j < i \leq k$. *(Size Reduced)*

LLL2. $\|\hat{b}_{i+1}\|_E^2 \geq (\delta - \Gamma_{i,i+1}^2)\|\hat{b}_i\|_E^2$, $|\Gamma_{i,i+1}| \leq 1/2$, *for* $i = 1, \ldots, k-1$. *(2-Reduced)*

Lenstra, Lenstra, Lovász Basis Reduction Algorithm

Algorithm: The LLL Basis Reduction Algorithm w.r.t. Ellipsoidal Norm

{

INPUT: A basis $b_1, b_2, \ldots, b_k$;

OUTPUT: An LLL reduced basis;

Initialize δ and compute Gram-Schmidt orthogonal basis $\hat{b}_1, \ldots, \hat{b}_k$;

WHILE $i \leq k$ DO

IF (2-reduced condition is violated) $\|\hat{b}_i\|_E^2 < (\delta - \Gamma_{i-1,i}^2)\|\hat{b}_{i-1}\|_E^2$ THEN

SWAP b_i and b_{i-1}, and update Γ;

[Size Reduction] Size reduce vectors $b_k, \ldots, b_i$;

IF $i > 2$ THEN $i \leftarrow \max\{i-1, 1\}$;

ELSE

$i \leftarrow i + 1$;

}

Figure 1: A Basic Description of the LLL Basis Reduction Algorithm

Lenstra, Lenstra, Lovász Basis Reduction Algorithm

Algorithm: The LLL Basis Reduction Algorithm w.r.t. Ellipsoidal Norm

{

INPUT: A basis $b_1, b_2, \ldots, b_k$;

OUTPUT: An LLL reduced basis;

Initialize δ and compute Gram-Schmidt orthogonal basis $\hat{b}_1, \ldots, \hat{b}_k$;

WHILE $i \leq k$ DO

IF (2-reduced condition is violated) $\|\hat{b}_i\|_E^2 < (\delta - \Gamma_{i-1,i}^2)\|\hat{b}_{i-1}\|_E^2$ THEN

SWAP b_i and b_{i-1}, and update Γ;

[Size Reduction] Size reduce vectors $b_k, \ldots, b_i$;

IF $i > 2$ THEN $i \leftarrow \max\{i-1, 1\}$;

ELSE

$i \leftarrow i + 1$;

}

Figure 1: A Basic Description of the LLL Basis Reduction Algorithm

- *LLL algorithm is polynomial time, but $\hat{b}$ are dense; and computations require long integer arithmetic.*

The Legacy

- Lovász and Scarf ['92] developed a Generalized Basis Reduction (GBR) algorithm.
- Cook, Rutherford, Scarf and Shallcross ['93] implemented GBR algorithm for some hard network design problems.
- Wang ['97] implemented GBR algorithm for LP and NLP ($\leq$ 100 integer variables).
- Owen, Mehrotra ['01] Experimental results on using heuristically computed disjunctions in branch-and-bound
- Gao, Zhang ['02] implemented Lenstra's algorithm.
- Aardal, Lenstra ['02] solved some hard equality constrained integer Knapsacks.
- G Pataki, M Tural, EB Wong ['10] Basis reduction and the complexity of branch-and-bound
- M Karamanov, G Cornujols ['11] Branching on general disjunctions
- G Nannicini, G Cornujols, M Karamanov, L Libert ['11] Branching on Split Disjunctions.

MICP: Ellipsoidal Rounding in Original Space

Recall that Feasibility-MICP is to find

$$\mathcal{C} := \left\{ x := \begin{pmatrix} x_z \\ x_c \end{pmatrix} \middle| \; x_z \in \mathbb{Z}^n_+,\; x_c \in \mathbb{R}^{\bar{n}}, Rx = r, c_i(x) \leq 0, i = 1, \ldots, l \right\},$$

or to show that no such solution exists.

Theorem

Let $f(x)$ be a self-concordant barrier associated with a compact convex set $\mathcal{C}$ having a non-empty relative interior. Then,

$$\mathcal{E}(w, \nabla^2 f(w)) \subseteq \mathcal{C} \subseteq \mathcal{E}\left(w, \gamma \nabla^2 f(w)\right),$$

where w is the barrier-center associated with $f(x)$, and $\gamma = 1/(4\theta + 1)$.

MICP: Ellipsoidal Rounding in Original Space

Recall that Feasibility-MICP is to find

$$\mathcal{C} := \left\{ x := \begin{pmatrix} x_z \\ x_c \end{pmatrix} \middle| \; x_z \in \mathbb{Z}^n_+,\; x_c \in \mathbb{R}^{\bar{n}}, Rx = r, c_i(x) \leq 0, i = 1, \ldots, l \right\},$$

or to show that no such solution exists.

Theorem

Let $f(x)$ be a self-concordant barrier associated with a compact convex set $\mathcal{C}$ having a non-empty relative interior. Then,

$$\mathcal{E}(w, \nabla^2 f(w)) \subseteq \mathcal{C} \subseteq \mathcal{E}\left(w, \gamma \nabla^2 f(w)\right),$$

where w is the barrier-center associated with $f(x)$, and $\gamma = 1/(4\theta + 1)$.

In fact, it is sufficient to find an approximate $\tilde{w}$. Let $q := -\nabla^2 f(\tilde{w})^{-1} f(\tilde{w})$, and $\|q\|_{\nabla^2 f(x)} \leq 1/4$. Then,

$$\mathcal{E}(\tilde{w}, 2\nabla^2 f(\tilde{w})) \subseteq \mathcal{C} \subseteq \mathcal{E}\left(\tilde{w}, \gamma \nabla^2 f(\tilde{w})\right),$$

where $\gamma = \frac{1}{4(2\theta+1)}$. If log-barrier center is used, $\theta = O(n + \bar{n})$ for LP, SOCP, SDP, e.g. Note that $c_i(\cdot)$ need not be differentiable.

MICP: Lattice Basis Reduction in the Original Space using an Adjoint Lattice (Mehrotra & Li '11)

Let $R = \begin{bmatrix} A : 0 \\ B : C \end{bmatrix}$, $r = \begin{pmatrix} b \\ a \end{pmatrix}$. Without loss of generality assume that the columns of A correspond to the integer variables, and the columns of C correspond to the real variables; and assume that A and C have full row rank. If C does not has a full row rank, we have a π such that $\pi^T C = 0$.

Theorem

Let Z be a basis of the kernel lattice of A: $\{q \in \mathbb{Z}^n : v^T Aq = 0, v \in \mathbb{Z}^m\}$, Z^* satisfy $Z^T Z^* = I$ and have integer elements; and Λ^* be the lattice generated by Z^*. Let $P_{RQ^{-1/2}} = [I - Q^{-1/2} R^T (RQ^{-1} R^T)^{-1} AR^{-1/2}]$. Suppose that the column vectors of $Q^{-1/2} \begin{bmatrix} Z^* \\ 0 \end{bmatrix}$ are LLL-reduced under $E := P_{RQ^{-1/2}}$ norm, and no column of Z^* satisfies $\mathcal{W}\left(\begin{pmatrix} u \\ 0 \end{pmatrix}, \mathcal{C}\right) \leq \gamma(3/\sqrt{2})^{n+\bar{n}}$. Then, using Z in a rounding procedure we must produce a feasible integer solution of (MICP). □

Simple Approximations of Adjoint Lattice

Remark

- *Recall that simple single variable disjunctive branching uses a fractional variable for branching.*
- *Strong branching selects a variable with best local improvement using a heuristic merit function (e.g., lower bound).*
- *So, how about using a subset of fractional variables to form an approximation to adjoint lattice; and then using best local improvement for branching direction selection?*

Let $\mathcal{C}_- := \left\{x | \alpha^T x \leq \beta, x \in \mathcal{C}\right\}$ and $\mathcal{C}_+ := \left\{x \mid \alpha^T x \geq \beta + 1, x \in \mathcal{C}\right\}$.
Let the quality of disjunction (α, β) be given by

$$z_{\alpha,\beta} := (1-\mu)\min\{z^-_{\alpha,\beta}(\mathcal{C}^i), z^+_{\alpha,\beta}(\mathcal{C}^i)\} + \mu \max\{z^-_{\alpha,\beta}(\mathcal{C}^i), z^+_{\alpha,\beta}(\mathcal{C}^i)\},$$

where $z^-_{\alpha,\beta}(\mathcal{C}^i) := \min_{\tilde{x}^i \in \mathcal{C}^i_-}$ and $z^+_{\alpha,\beta}(\mathcal{C}^i) := \min_{\tilde{x}^i \in \mathcal{C}^i_+}$

CMU-IBM Test Set

Altered CMU-IBM test mixed binary to create mixed integer test problems as follows. Consider the following formulation was considered

$$
\begin{aligned}
\min \quad & c(x) \\
\text{s.t.} \quad & c_i(x) \leq 0, i = 1, \ldots, m, \\
& f - r_l e \leq Ax \leq g + r_u e, \\
& l_z - c_l e \leq x_z \leq u_z + c_u e, \\
& l_c \leq x_c \leq u_c, \\
& x_z \in \mathbb{Z}^n \text{ and } x_c \in \mathbb{R}^{\bar{n}},
\end{aligned}
$$

where f and g respectively represents the vectors of the original lower and upper bounds for constraints Ax, and l_z and l_c respectively represents the vectors of the original lower and upper bounds for the integer variables x_z. Two different sets of the expanded values (r_l, r_u, c_l, c_u) are used in our experiment. These are given in Table 1. The only exception is for problems CLay0203H, CLay0204H, and CLay0303H, for which we use $c_l = 0$, i.e., the lower bounds of variables are not expanded.

	r_l	r_u	c_l	c_u
Test Set 1	2	2	5	5
Test Set 2	5	5	10	10

Table 1: General MICP test problems

Some Implementation Details: i-Optimize Package

- Continuous Relaxation Solver. Homogeneous Primal-Dual Potential Reduction based Predictor-Corrector Method. Self-developed, and used with the following termination criteria: Let rv_p^k, rv_d^k, and rv_g^k be the vectors of the residuals corresponding to the kth iterate. An (approximate) optimal solution is obtained if

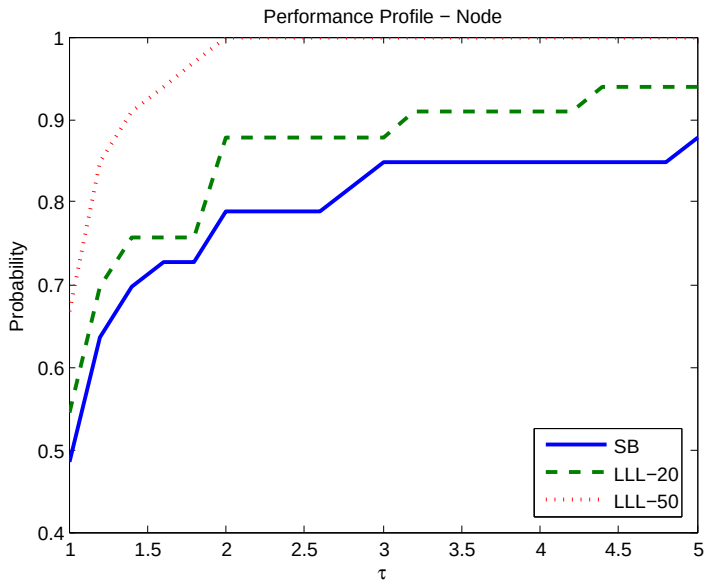
$$\begin{aligned} \frac{\|res_p^k\|}{\max\left\{1, \|res_p^0\|\right\}} &< 10^{-8}, \\ \frac{\|res_d^k\|}{\max\left\{1, \|res_d^0\|\right\}} &< 10^{-8}, \\ \frac{|res_g^k|}{\max\left\{1, |res_g^0|\right\}} &< 10^{-8}. \end{aligned}$$

  The problem is (near) infeasible if

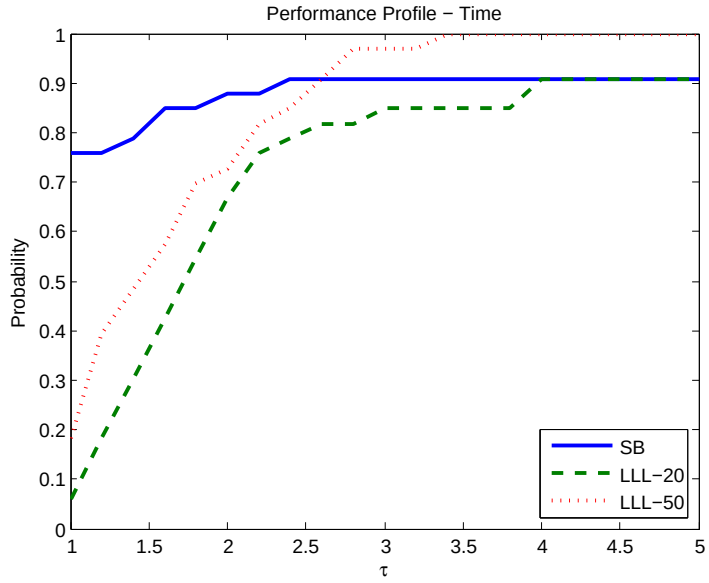
$$\frac{\mu^k}{\mu^0} < 10^{-8}, \quad \text{and} \quad \frac{\tau^k}{\min\left\{1, \kappa^k\right\}} < \frac{\tau^0}{\kappa^0} 10^{-12},$$

  where τ and κ represent homogeneous variables introduced in the homogeneous reformulation.
- Used the above solver in Phase I with null objective. Set inequality constraints to equality, if the slack to the bound is less than 10^{-8} at the end of phase I. Phase II: Newton iteration on KKT-conditions of the barrier problem. We terminate the algorithm if $\|Xs - e\|_\infty \leq 10^{-1}$.

Performance Profiles for 33 test problems:(2,2,5,5) using Strong Branching, LLL-20 and LLL-50 ($\delta = .99$ in LLL Algorithm)



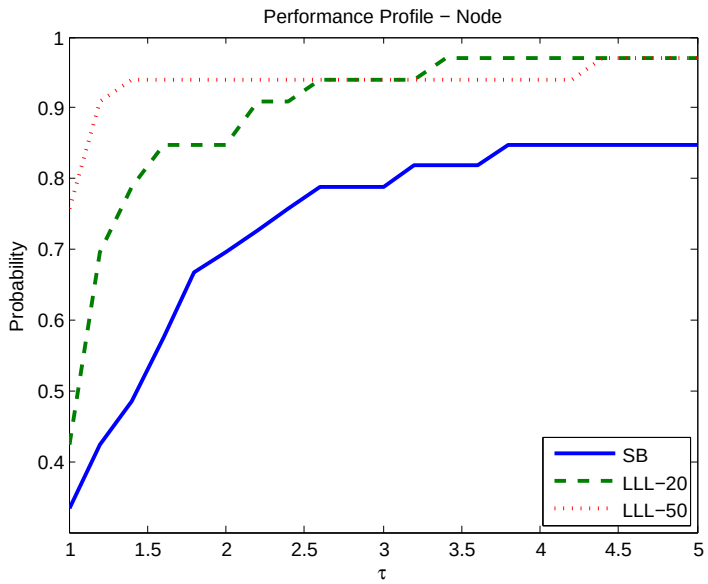
Node Count Performance



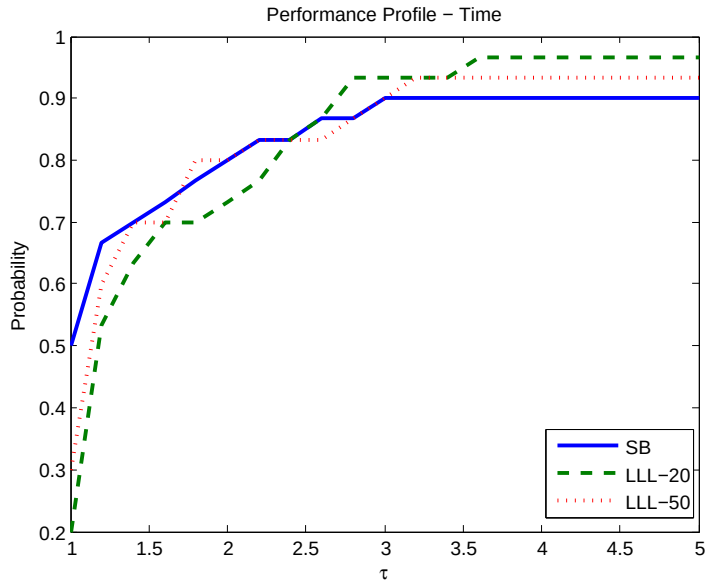
CPU time performance using **single** core in a processor

Only 5-7% time is spent in LLL-basis reduction, which was implemented using floating point calculations.

Performance Profiles for 33 test problems:(5,5,10,10) using Strong Branching, LLL-20 and LLL-50 ($\delta = .99$ in LLL Algorithm)



Node Count Performance



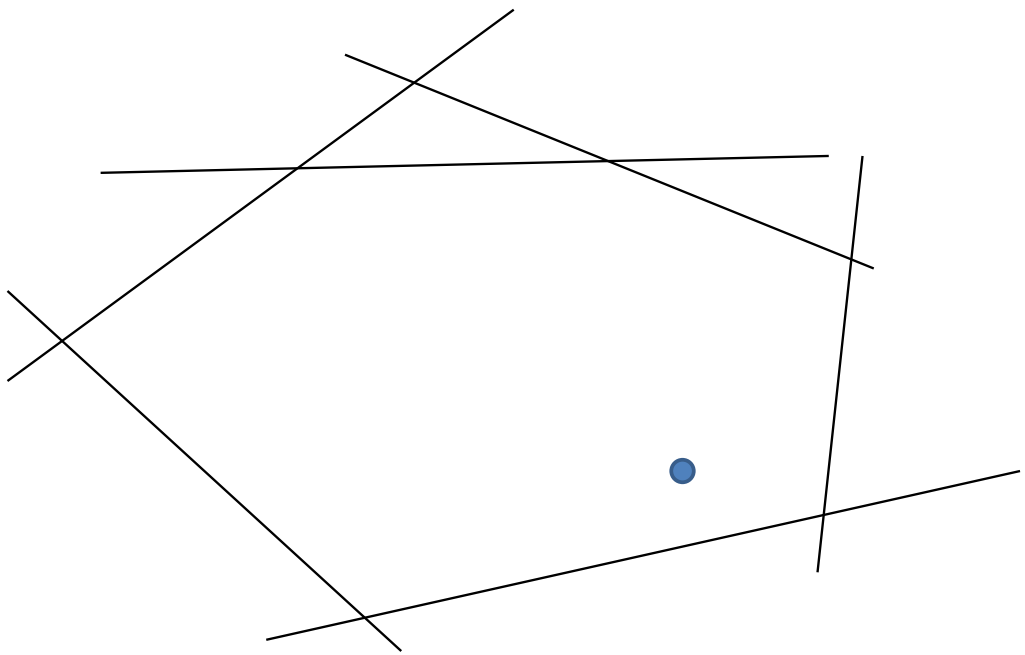
CPU time performance using single core

Only 5-7% time is spent in LLL-basis reduction, which was implemented using floating point calculations.

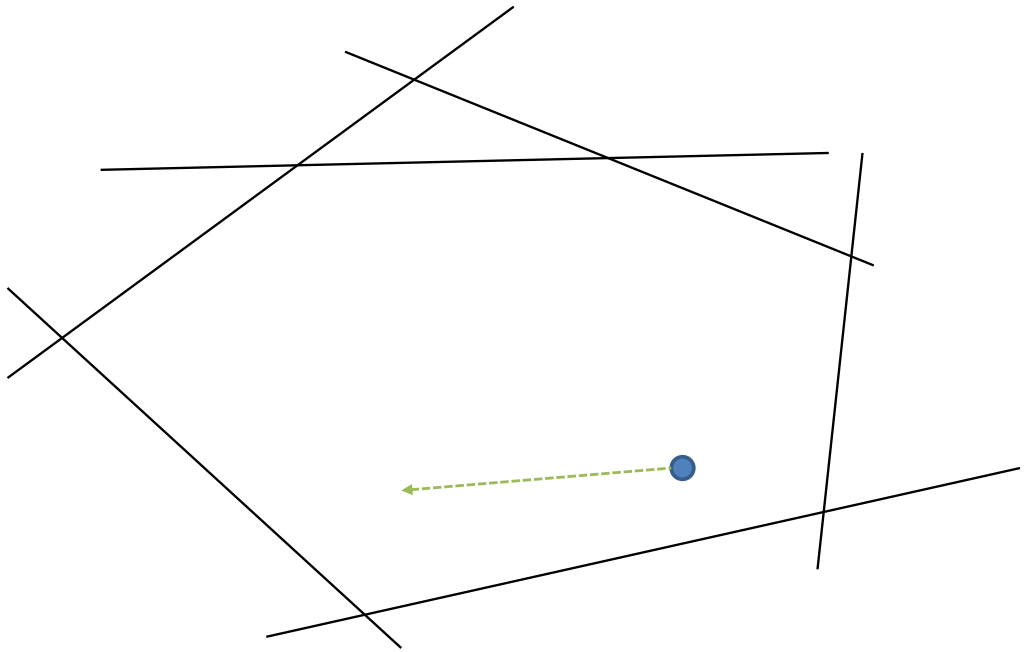
Problem	Most Fractional		Strong Branching		LLL-20		LLL-50	
	Node	Time	Node	Time	Node	Time	Node	Time
CLay0203H-(2,2,0,5)	4654	512.	44	38.	44	72.	44	68.27
CLay0203M-(2,2,5,5)	5984	146	82	33.	**62**	41.	62	41.87
CLay0204H-(2,2,0,5)	64342	7201. †	250	395.	**132**	421.	138	454.36
CLay0205M-(2,2,5,5)	133470	7205. †	5096	7209. †	326	1031.	**176**	772.44
CLay0303H-(2,2,0,5)	1404	174.	82	89.	**74**	226.	**74**	185.15
CLay0303M-(2,2,5,5)	12152	423.	228	757.	152	875.	**82**	486.93
CLay0304M-(2,2,5,5)	167222	7204. †	912	477.	912	480.	912	489.52
FLay02H-(2,2,5,5)	20	0.43	10	0.96	10	1.88	10	1.53
FLay02M-(2,2,5,5)	20	0.16	10	0.34	10	0.67	10	0.56
FLay03H-(2,2,5,5)	66	3.53	42	19.87	**30**	29.92	**30**	23.38
FLay03M-(2,2,5,5)	74	1.01	**36**	4.02	46	11.84	46	9.68
FLay04H-(2,2,5,5)	50878	7205.44†	246	390.23	**34**	174.18	60	180.72
FLay04M-(2,2,5,5)	1150	24.01	104	35.81	**92**	60.23	98	57.88
FLay05H-(2,2,5,5)	35150	7145.70	152	644.39	100	862.84	**78**	519.52
FLay05M-(2,2,5,5)	36284	1089.87	788	543.41	480	592.70	**264**	272.49
SLay04H-(2,2,5,5)	204	8.	**130**	92.	254	356.	254	307.67
SLay04M-(2,2,5,5)	842	15.	240	48.	**212**	108.	**212**	101.63
SLay05H-(2,2,5,5)	95928	7201. †	**266**	499.	800	2793.	332	928.30
SLay05M-(2,2,5,5)	1656	40.	146	70.	442	274.	**104**	113.82
SLay06H-(2,2,5,5)	56076	7204. †	2326	2850.	**468**	2027.	712	2155.46
SLay06M-(2,2,5,5)	234910	7203. †	6502	7201. †	5938	7201. †	**380**	762.15
fo7-(2,2,5,5)	143194	7205.04†	750	589.24	750	662.83	750	691.07
fo7_2-(2,2,5,5)	151518	7202.09†	516	517.98	516	574.35	516	604.88
fo8-(2,2,5,5)	118354	7205.60†	1670	3542.17	1670	3862.11	**1668**	4260.26
Syn05M-(2,2,5,5)	10	0.	10	0.	10	0.	10	1.16
Syn05M02M-(2,2,5,5)	224	7.	22	8.	22	18.	22	22.26
Syn05M03M-(2,2,5,5)	5554	360.	**60**	40.	62	82.	62	113.19
Syn05M04M-(2,2,5,5)	74810	7201. †	130	178.	**128**	324.	**128**	417.95
Syn10M-(2,2,5,5)	246	5.	**38**	5.	40	7.	40	7.83
Syn10M02M-(2,2,5,5)	87148	7182.	**208**	415.	218	728.	218	887.61
Syn15M-(2,2,5,5)	166	6.	**34**	11.	36	13.	36	14.44
Syn20M-(2,2,5,5)	830	39.	54	25.	54	32.	54	33.48
trimloss2-(2,2,5,5)	439508	7202.02†	5096	1426.02	1700	1223.81	**334**	178.30

Problem	Most Fractional		Strong Branching		LLL-20		LLL-50	
	Node	Time	Node	Time	Node	Time	Node	Time
CLay0203H-(5,5,0,10)	15100	1329.	118	100.	**78**	82.	**78**	80.
CLay0203M-(5,5,10,10)	200226	7207. †	286	95.	**78**	48.	**78**	47.
CLay0204H-(5,5,0,10)	63608	7202. †	1402	1666.	652	1620.	**254**	679.
CLay0205M-(5,5,10,10)	160398	7201. †	2734	7201.	2006	4435.	**1640**	5745.
CLay0303H-(5,5,0,10)	550	55.	**96**	168.	110	195.	110	194.
CLay0303M-(5,5,10,10)	217780	7206. †	3846	2259.	580	675.	**270**	292.
CLay0304M-(5,5,10,10)	127632	7200. †	7956	7200. †	7011	7207 †	6589	7201. †
FLay02H-(5,5,10,10)	104	2.00	20	1.63	**8**	1.15	**8**	1.11
FLay02M-(5,5,10,10)	24	0.22	18	0.57	**8**	0.55	**8**	0.56
FLay03H-(5,5,10,10)	110	5.22	36	15.72	**18**	17.77	**18**	17.51
FLay03M-(5,5,10,10)	15202	190.22	770	82.16	**136**	27.97	**136**	27.68
FLay04H-(5,5,10,10)	4546	486.32	146	255.22	**48**	141.21	260	822.55
FLay04M-(5,5,10,10)	308808	7208.07†	198	55.04	**190**	109.62	**190**	96.81
FLay05H-(5,5,10,10)	18102	3241.99	202	781.63	172	1060.86	**124**	776.27
FLay05M-(5,5,10,10)	228118	7201.60†	998	633.85	1174	1007.78	**766**	1025.19
SLay04H-(5,5,10,10)	9878	497.	234	87.	**112**	93.	**112**	97.
SLay04M-(5,5,10,10)	377996	7203. †	6316	1675.	486	239.	**454**	186.
SLay05H-(5,5,10,10)	105068	7204. †	5304	7202. †	1364	2806.	**420**	1054.
SLay05M-(5,5,10,10)	225296	7204. †	9830	7201. †	14326	7200. †	6712	7201. †
SLay06H-(5,5,10,10)	73442	7215. †	**406**	1193.	2116	5989.	1757	6892.
SLay06M-(5,5,10,10)	144164	7206. †	4516	7202. †	4180	7201. †	3038	7203. †
fo7-(5,5, 10, 10)	186758	7205.41†	**2696**	3520.69	4214	5582.31	3766	5129.92
fo7_2-(5,5, 10, 10)	184944	7200.56†	4782	7200.63†	4290	7201.39†	4144	7201.21†
fo8-(5,5, 10, 10)	149456	7201.95†	2240	7203.40†	2106	7212.03†	1922	7203.55†
Syn05M-(5,5,10,10)	6	0.	6	0.	6	0.	6	0.
Syn05M02M-(5,5,10,10)	672	23.	**44**	19.	52	51.	52	53.
Syn05M03M-(5,5,10,10)	8798	524.	**142**	170.	162	596.	164	506.
Syn05M04M-(5,5,10,10)	87146	7201. †	**436**	1053.	516	2666.	514	3323.
Syn10M-(5,5,10,10)	110	2.	30	4.	30	5.	30	5.
Syn10M02M-(5,5,10,10)	97160	7208. †	**1616**	2949.	1626	6034.	1626	5979.
Syn15M-(5,5,10,10)	222	8.	34	10.	34	12.	34	13.
Syn20M-(5,5,10,10)	632	28.	62	34.	62	36.	62	38.
trimloss2-(5,5,10,10)	433110	7202.07†	60	17.99	**36**	17.51	**36**	19.80

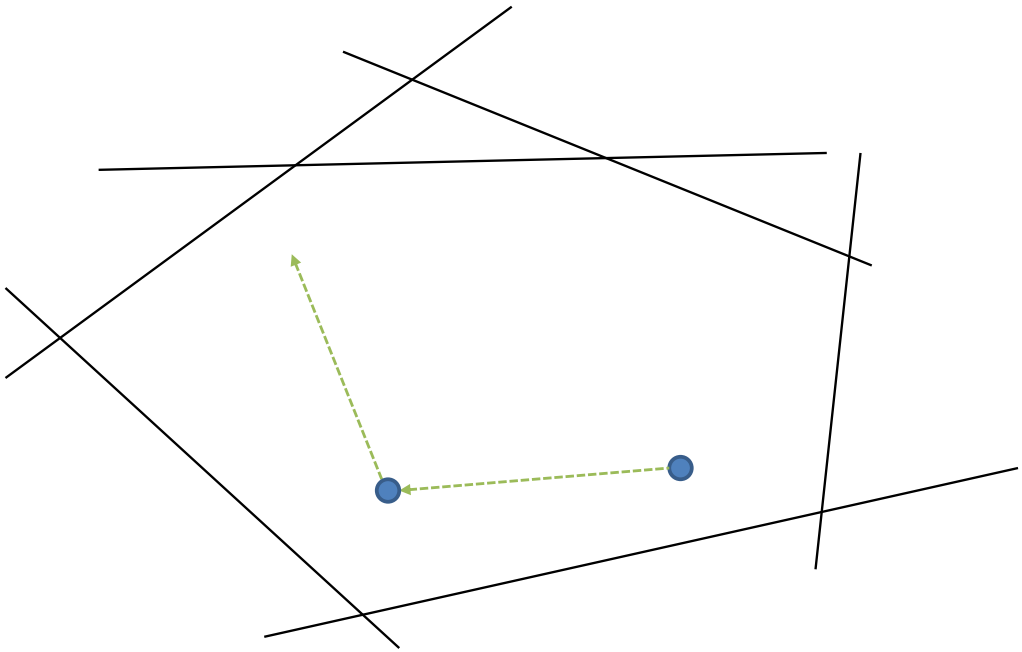
Random Walk: A Basic Illustration



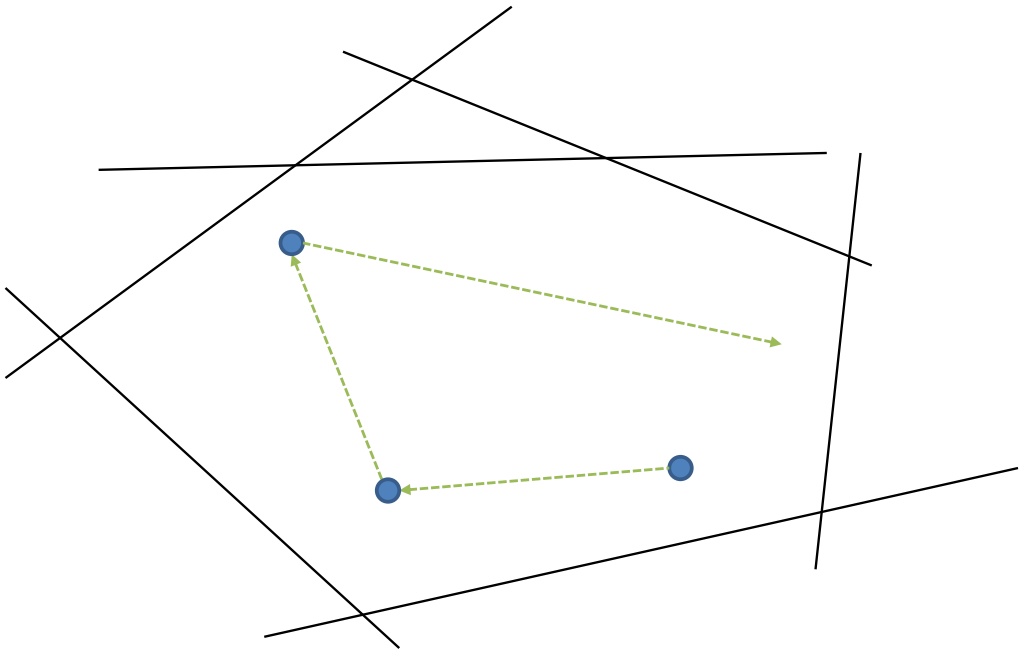
Random Walk: A Basic Illustration



Random Walk: A Basic Illustration



Random Walk: A Basic Illustration



Random Walk: 30+ Years of History

- A Markov Chain. May be used to uniformly sample on a polytope.
- Robert Smith [1984] introduced Hit and Run Walk
  - Generates a random direction on uniform sphere
  - Takes a random step in this direction
- Lovász (1999): Hit-and-run mixes fast. *Mathematical Programming* 86(3)
  - Number of HR steps required to sample nearly uniformly from a polytope is polynomial in problem dimension.
- Bertsimas and Vempala (2004): Use RW to give a polynomial time algorithm for convex programs.
- Kannan and Narayanan (2009): A lazy random walk using Dikin ellipsoid approximation at each step mixed in strongly polynomial time, when started from near a center point. They use it to give a new algorithm for LP.
- Can they help in solving integer programs?

Projected Hit-and-Run Walk

Let $\mathcal{P} := \{x | \mathcal{P}_I \cap \mathcal{P}_E\}$, where $\mathcal{P}_I := \{x | A_I x \leq b_I, x \geq 0\}$ and $\mathcal{P}_E := \{x | A_E x = b_E\}$. We first compute a random direction d in $\mathbb{R}^{n+\bar{n}}$. Next we orthogonally project d onto the affine space given by the equality constraints, yielding direction p. Finally, we compute a random point along $\{x + \lambda p\} \bigcap \mathcal{P}$.

Algorithm *Projected 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: Pick a uniform random direction d in $\mathcal{P}_I$.
4: **if** $A_E \neq \{\phi\}$ **then**
5: Compute p by solving $\min_{A_E p = 0} \|p - d\|^2$.
6: **else**
7: Set $p := d$.
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 $k := k + 1$.
12: **end while**

Projected and Modified Dikin Walk

Consider the polytope in the form $\mathcal{P} := \{x := (y, s) | A_I y + s = b_I, A_E y = b_E, (y, s) \geq 0\} \subseteq \mathbb{R}^{n+\bar{n}}$ where s represents the vector of slacks for the constraint $A_I y \leq b_I$. The log-barrier analytic center of $\mathcal{P}$ is defined as a solution of

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

Algorithm *Modified Dikin Walk*

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

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

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

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**

Fischetti, Glover, Lodi (2005): The Feasibility Pump

From a given point $x \in \mathcal{P}$, FP heuristically searches for a point x^* that is as close as possible to a rounded integer solution $\tilde{x}$ of x by solving an l_1-norm 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

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

Let $\hat{x}^{k*}$ be the optimum solution of the above problem. If $f^k(\hat{x}^{k*}) = 0$, we have a feasible solution of 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 (e.g., randomly flipping bits) are implemented to avoid cycling.

Some Implementation Details: i-Optimize Package

- Analytic Center Computations. Settings as before.
- Used COIN-OR Branch-and-Cut (CBC) version 2.3 for feasibility pump computations with default settings. Disabled prepress and initialSolve routines.
- l_1-norm minimization limited to 500 for LP optimum and 75 for all RW points (maximum steps 500).
- FP COIN-OR l_1-norm minimization limited to 10,000; with 30 re-tries
- CPU time for pure-FP, and RW-FPs are calibrated to be nearly equal.
- MIPLIB 2003 (28 small and medium size) problems and COR@AL (74 small and medium size) problems.
- MIPLIB 2003 (4 pure binary, 18 mixed binary, 6 mixed integer); COR@L (11 pure binary, 62 mixed binary, one general integer)
- Sliding objective cut, once a feasible solution is found

Performance of Walk-and-Round Heuristics

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

LPI-FP: FP Starting from LP optimal solution obtained by interior solver

LPS-FP: FP Starting from LP optimal solution obtained by simplex method (Cplex)

AC-FP: FP Staring from Analytic Center

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

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	75.97	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

HR: Projected Hit and Run

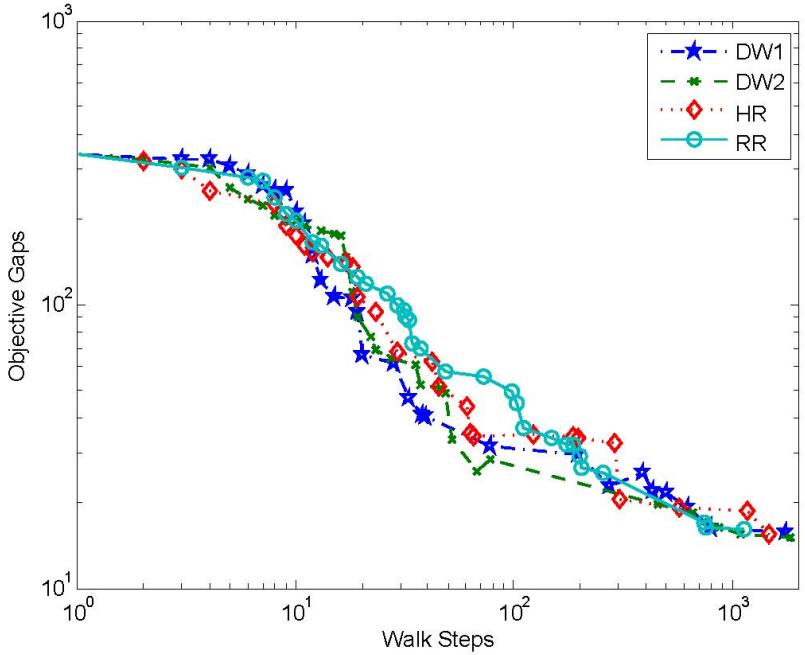
DW1: Short step Projected Dikin Walk

DW2: Long step Projected Dikin Walk

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

RR: Random Ray; PropLP: Fraction of time in LP Solution; PropAC: Fracton of time in Analytic Center computation; PropFP: Proportion of time in FP

Performance of Long Walks



Problem: aflow30a

Feasibility Pump For MICPs

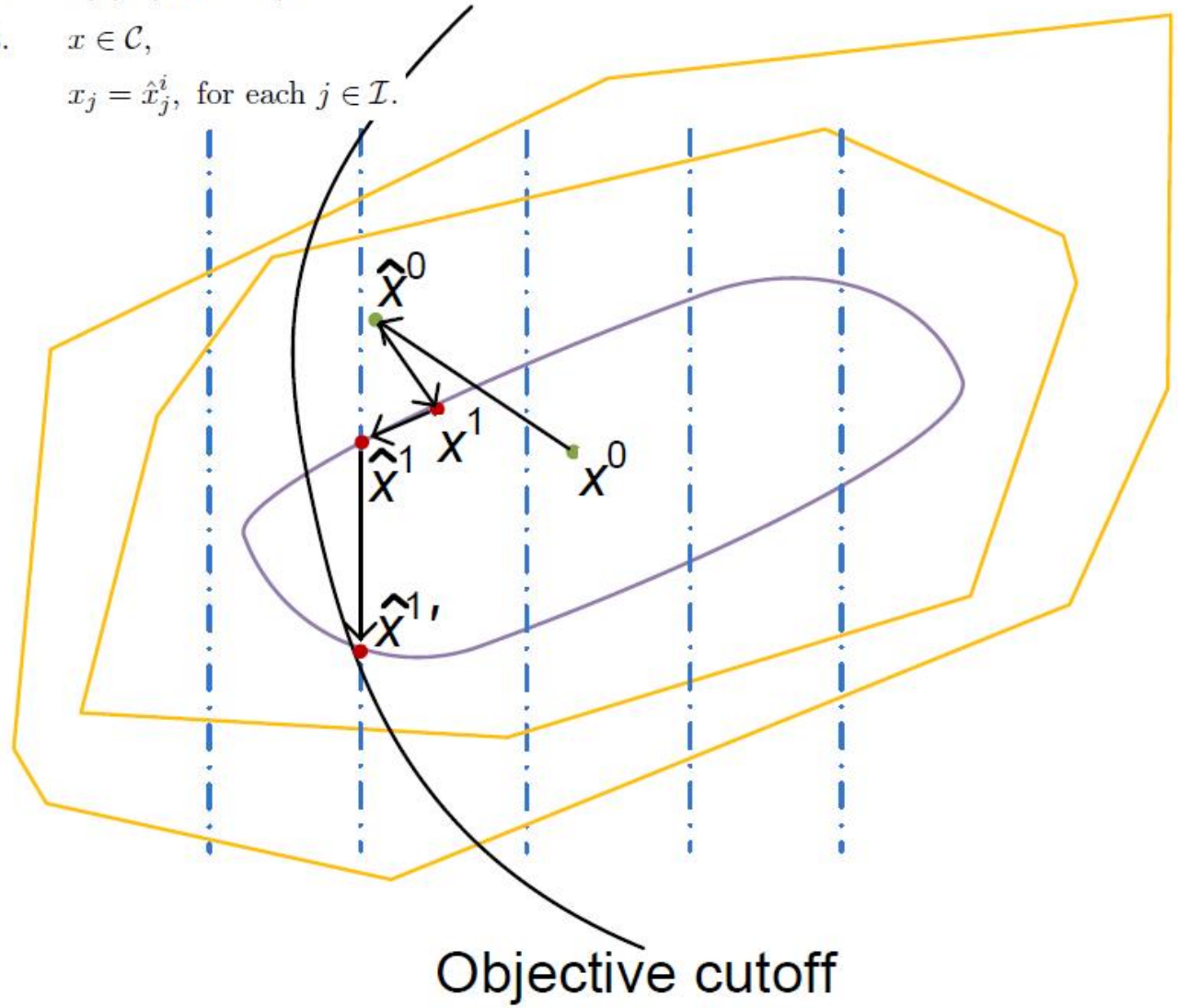
Bonami, Cornuéjols, Lodi, and Margot (2009) proposed a variant of FP for MICPs. In their method,

1. $x^k \in \mathcal{C}$ is generated in the same way as in the basic FP heuristic,
2. whereas $\hat{x}^k$ is generated by solving a l_1-norm sub-MILP.
3. sub-MILP uses outer approximation (OA) of convex constraints at all points of the sequence $\left\{x^l\right\}_{l=0,\ldots,k}$

$$
\begin{aligned}
\text{min} \quad & \sum_{j \in \mathcal{I}} |x_j - x_j^k| \quad \text{(FP-OA)} \\
\text{s.t.} \quad & c_i(x^l) + \nabla c_i(x^l)(x - x^l) \geq 0, \ i = 1, \ldots, \hat{m} \ \text{ and } l = 0, \ldots, k, \\
& \nabla c(x)(x - x^{k*}) \leq c(x^{k*}) - \epsilon, \ \text{if the (best) integer solution } x^{k*} \text{ is available,} \\
& Rx = r, \\
& x_j \in \mathbb{Z} \text{ for each } j \in \mathcal{I}, \\
& 0 \leq x \leq u.
\end{aligned}
$$

If (FP-OA) is empty, i.e, no integer solution exists, then FP terminates the search and claims that the available best integer solution is optimum (within a specific tolerance ϵ), or the MICP is infeasible.

$$\begin{aligned} \min \quad & c(x) \quad \text{(FP-POL)} \\ \text{s.t.} \quad & x \in \mathcal{C}, \\ & x_j = \hat{x}^i_j, \text{ for each } j \in \mathcal{I}. \end{aligned}$$



Can Walking help MICPs?

1. $x^k \in \mathcal{C}$ is generated in the same way as the basic heuristic, except Outer Approximation is at the points generated from walk-points
   - Start OAFP at the continuous relaxation solution. Perform a small number of (15) iterations to generate a crude feasible solution.
   - Build OA at random walk points. Perform a small number (10) iterations to generate an improved integer solution.
   - "Delete" all previously added OA constraints, and perform a longer walk-relax-round run while keeping OA constraints.
2. $\hat{x}^k$ is generated by solving the sub-MILP in Cplex
3. 58 MICP instances from CMU-IBM library. 9 MIQP instances (comparison with Cplex 12.5).
4. All other software settings as before. 5 minute limit.

Computational Results: Walk-Relax-Round Heuristic for MICP (58 Problems)

Method	Optimum Found	Optimum Proved	Mean-gap
OA-FP	45	39	3.34%
HR-OA-FP	50	40	1.26%
DW1-OA-FP	49	40	1.35%
DW2-OA-FP	51	40	1.11%

Long Runs of Walk-Relax-Round Heuristic for MICP

Method	HR				
Problem	Start Obj	Start Gap	End Obj	End Gap	Time
Trimolss7	23.2	49.68	18.9	21.94	‡
BatchS201210M	2302924	0.33	2295349*	0	1618.83
RSyn0830M03M	-1498.04	2.92	-1543.06*	0	1503.2
RSyn0840M03M	-2486.93	9.32	-2742.65*	0	1427.04
SLay10H	139824.1	7.91	130969.4	1.07	‡

Method	DW1				
Problem	Start Obj	Start Gap	End Obj	End Gap	Time
Trimolss7	23	48.39	18.4	18.71	‡
BatchS201210M	2345035	2.16	2295349*	0	1377.54
RSyn0830M03M	-1523.37	1.28	-1543.06*	0	985.91
RSyn0840M03M	-2513.14	8.37	-2742.65*	0	1440.42
SLay10H	144585	11.58	129971.1	0.3	‡

Method	DW2				
Problem	Start Obj	Start Gap	End Obj	End Gap	Time
Trimolss7	23	48.39	18.4	18.71	‡
BatchS201210M	2310423	0.66	2295349*	0	1820.36
RSyn0830M03M	-1526.02	1.1	-1543.06*	0	904.76
RSyn0840M03M	-2507.26	8.58	-2742.65*	0	1386.98
SLay10H	132926.2	2.58	130996.5	1.09	‡

Method	FP-OPT				
Problem	Start Obj	Start Gap	End Obj	End Gap	Time
Trimolss7	–	NA	18.6	20	‡
BatchS201210M	2324008	1.25	2295349*	0	1001.36
RSyn0830M03M	-1520.02	1.49	-1543.06*	0	1084.5
RSyn0840M03M	-2673.3	2.53	-2742.65*	0	1133.8
SLay10H	152751.8	17.88	131351.7	1.37	‡

[1] *: Proven optimality.
[2] ‡: One hour time limit reached.

Random Walk (DW2) for MIQPs and Cplex 12.1 (5 min time limit)

Method	Dikin Walk (DW2) – Long Variant				Cplex 12.1			
Problem	Obj	Gap	Time (Found)	Time (Proven)	Obj	Gap	Time (Found)	Time (Proven)
ibell3a	87875.03*	0	21.14	36.47	87875.03*	0	5.56	6.47
ibienst1	34.21	0	23.74	‡	34.21	0	226.87	‡
ilaser0	2412537.96	0.001	287.57	‡	2412505.12	0	33.24	‡
iportfolio	-0.49*	0	58.71	260.83	–	NA		‡
iqiu	-126.66	0	101.19	‡	-127.08*	0	140.95	206.87
isqp	-21000.45	0	154.45	‡	-21000.45	0	8.11	‡
isqp0	-20319.51	0	7.24	‡	-20319.51*	0	2.03	19.71
isqp1	-18992.68	0	8.48	‡	-18992.68*	0	3.74	36.47
itointqor	-1146.7*	0	118.14	287.61	-1146.7	0	141.5	‡

1 NA: Statistics not available.

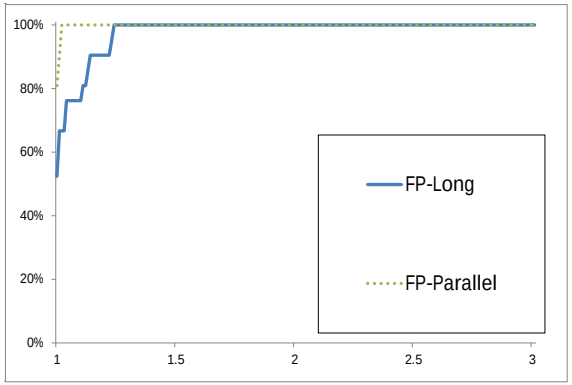
2 * : Proven optimality.

3 –: No solution found.

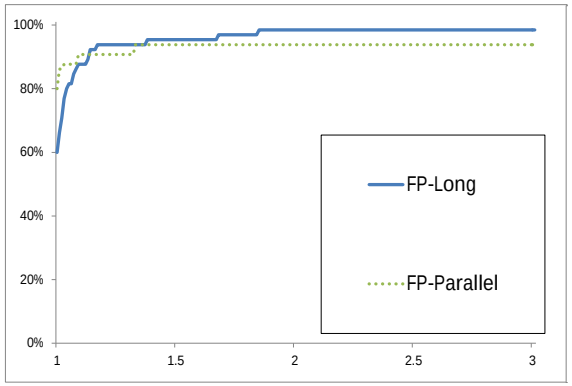
4 ‡: 5-minute time limit reached.

Parallel FP Versus Long-FP Run

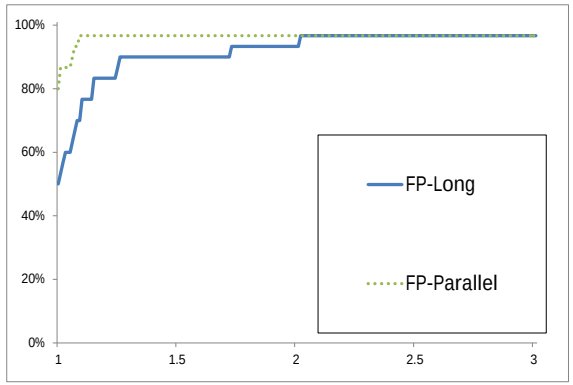
- Parallel-FP: In addition to the standard FP, generate rounded integer solutions $\hat{x}$ by random flipping



MIPLIB 2003 Problems (27)



MIPLIB 2010 Problems (67)



CORAL Problems (73)

FP-Parallel gets 1 min wall clock time. FP-Long gets upto 32 min wall clock time for problems in MIPLIB 2003 and CORL testsets. FP-Parallel gets 5 min wall clock time. FP-Long gets upto 32x5 min wall clock time for problems in MIPLIB 2010.

- It appears that there is benefit from objective value sharing among slave processors running FP concurrently.

Running Walks in Parallel: Walk-Vertex-Round Heuristic

- Use Dikin walk (long variant) to generate points for random cost directions.
- Obtain vertex solutions (Cplex 12.1 as a LP-solver)
- Run FP on the generated vertex solutions.
  1. Variant (FP-Parallel): All processors run FP with randomized flipping
  2. Variant (WVR1): 8 Processors run FP from optimum with randomized flipping; 24 Processors run FP on vertices generated from walk points. FP is run on

$$x^k := \arg\min\{d_r^T x : Rx = r, c^T x \leq \bar{z}\}, d_r = x_r - x_{ac}$$

  3. Variant (WVR2): 8 Processors run FP from optimum with randomized flipping; 24 Processors take a convex combination of random walk direction and true cost direction to generate a vertex solution, i.e., FP is run on $x^k := \arg\min\{d_r^T x : Rx = r, c^T x \leq \bar{z}\}$,

$$d_r = \lambda c + (1 - \lambda)(x_r - x_{ac})$$

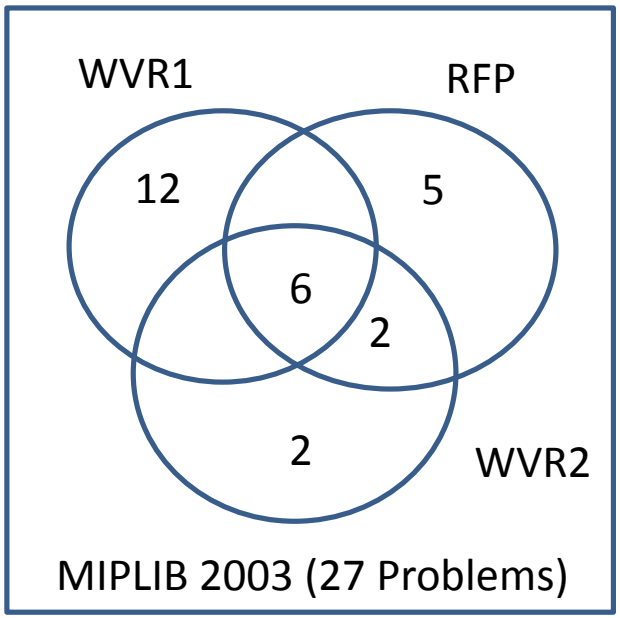
  .
- Implemented using PVM on a 32-node (hyper threaded to 64) shared memory machine, with 128 GB RAM.

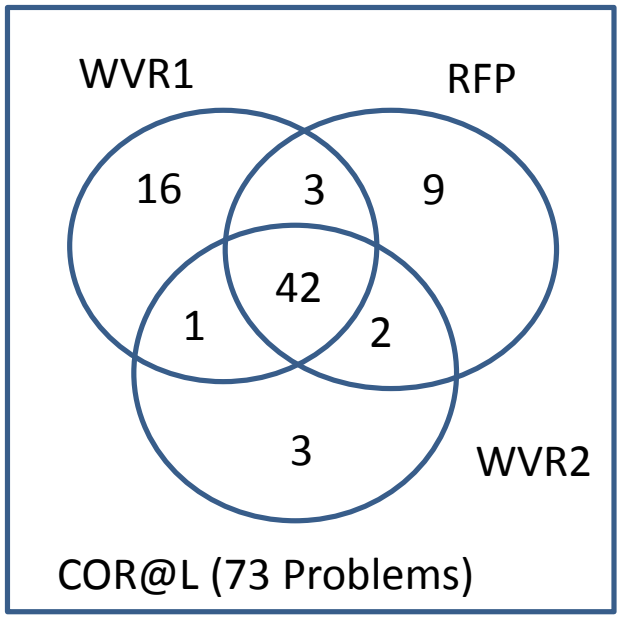
Performance of Walk-Vertex-Round Heuristics in Parallel

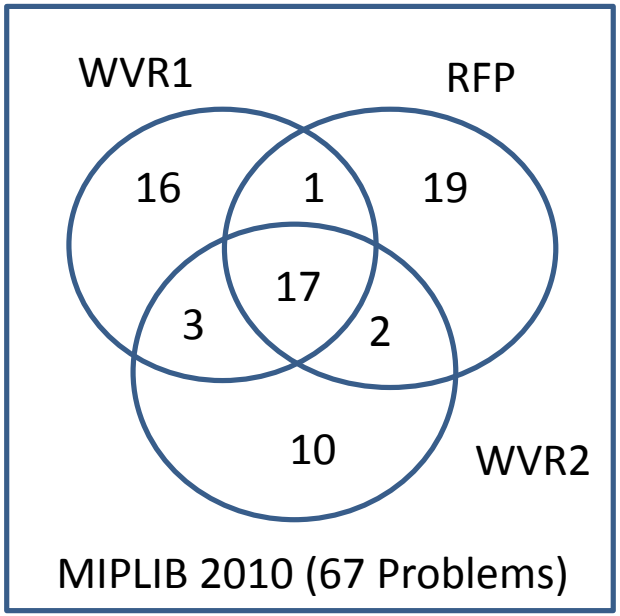
	COR@L (73 problems)			MIPLIB 2003 (27 problems)			MIPLIB 2010 (67 problems)		
	FP-Parallel	Parallel-WVR1	Parallel-WVR2	FP-Parallel	Parallel-WVR1	Parallel-WVR2	FP-Parallel	Parallel-WVR1	Parallel-WVR2
# Feasible found	72	73	72	27	27	27	61	65	62
# Optimal	24	25	23	6	5	6	15	17	15
# 5% gap	45	48	45	16	15	16	38	35	33
Avg gap	38.6%	37.9%	40.0%	17.4%	10.1%	11.1%	16.8%	13.0%	22.9%

One minute wall clock time for MIPLIB 2003 and COR@L problems.
5 min wall clock time for MIPLIB 2010 problems.

Performance of Walk-Vertex-Round Heuristics in Parallel







Performance of Walk-Vertex-Round Heuristics on Different Problem Sets
1 min wall-clock time limit for MIPLIB 2003 and COR@L problems
5 min wall-clock time limit for MIPLIB 2010 problems

The # of problems for which a heuristic wins is given in the Venn diagram.

Can Cuts Help towards Generating Heuristic Solutions?

All Gomory and Split cuts at optimal vertex, and best 5 cuts at non-optimal vertices were added.

Problem		FP (1 min)	FP (1 min) x 3 Round Cuts		
	Int-LB	Objective	FP(1st)	FP(3rd)	FNL-LB
10teams	917.0	924.0	924.0	924.0	924.0
a1c1s1	997.5	18337.7	18344.4	18344.4	2381.96
aflow30a	983.2	1923.0	2559	2559	1024.64
aflow40b	1005.7	10559.0	10559	10559	1032.6
cap6000	-2451540.0	-2443330.0	-2442910	**-2449620**	-2451440
danoint	62.6	70.2	69.5	69.5	62.6595
disktom	-5000.0	-5000.0	-5000	-5000	-5000
fixnet6	1200.9	6041.9	6041.9	6041.9	1443.1
liu	346.0	5033.0	4825.1	**4825**	453
mas74	10482.8	13259.5	13258.9	13258.9	10495.5
mas76	38893.9	41731.9	41139	41139	38940.7
mkc	-611.9	-253.1	-262.828	-262.828	-593.743
modglob	20430900.0	26412300.0	26410800	**26402900**	20522000
opt1217	-20.0	-16.0	-16	-16	-16.8019
p2756	2688.8	NA	NA	NA	3097.68
pp08aCUTS	5480.6	8080.0	8219.9	**8010**	6850.47
pp08a	2748.4	8609.8	8590	**8110**	6907.59
set1ch	32007.7	75575.0	75016.7	75016.7	49611.3
seymour	403.8	430.0	432	432	406.011
tr12-30	14210.4	178918.0	179121	179121	87805.1
vpm2	9.9	17.5	17.5	17.5	12.4402
neos-1480121	0.0	43.0	43	43	0
ran14x18	3016.9	4237.0	4239	4239	3365.16
rlp1	13.2	21.0	21	21	13.9938
bienst1	11.7	47.3	47	47	27.9354
neos-504674	472.8	19542.2	19621.3	**19621.2**	1521.5

What have we learned, and Where to go from here?

1. Do interior solutions help?
   - Use of ellipsoidal rounding, combined with limited LLL-reduction in the original space appears to help for harder MICPs
   - **The bottleneck step of evaluating disjunction quality is parallelizable**!
   - Use of points generated from random walk appear to help in generating heuristic solutions
   - Dikin walk points are slightly better
   - **Multiple short walks in parallel is generally as effective as one long walk**!
   - Jury is out on the value in using cuts at non-vertex solutions

What have we learned, and Where to go from here?

1. Do interior solutions help?
   - Use of ellipsoidal rounding, combined with limited LLL-reduction in the original space appears to help for harder MICPs
   - **The bottleneck step of evaluating disjunction quality is parallelizable**!
   - Use of points generated from random walk appear to help in generating heuristic solutions
   - Dikin walk points are slightly better
   - **Multiple short walks in parallel is generally as effective as one long walk**!
   - Jury is out on the value in using cuts at non-vertex solutions
2. What Next?
   - Other feasible solution finding heuristics
   - Managing the number of cuts
   - Combining heuristic solution generation, cuts, and branching – all in parallel
   - release i-Optimize package v1

What have we learned, and Where to go from here?

1. Do interior solutions help?
   - Use of ellipsoidal rounding, combined with limited LLL-reduction in the original space appears to help for harder MICPs
   - **The bottleneck step of evaluating disjunction quality is parallelizable**!
   - Use of points generated from random walk appear to help in generating heuristic solutions
   - Dikin walk points are slightly better
   - **Multiple short walks in parallel is generally as effective as one long walk**!
   - Jury is out on the value in using cuts at non-vertex solutions
2. What Next?
   - Other feasible solution finding heuristics
   - Managing the number of cuts
   - Combining heuristic solution generation, cuts, and branching – all in parallel
   - release i-Optimize package v1

References

1. S Mehrotra and Z Li (2011). Branching on hyperplane methods for mixed integer linear and convex programming using adjoint lattices. Journal of Global Optimization, 49(4):623-649.
2. S. Mehrotra and KL Huang (2012). Computational experience with a modified potential reduction algorithm for linear programming. Optimization Methods And Software, 27(4-5):865–891, 2012.
3. KL Huang and S. Mehrotra (2013). An empirical evaluation of walk-and-round heuristics for mixed integer linear programs. *Computational Optimization and Applications*, 55(3):545-570.
4. KL Huang and S. Mehrotra (2013). An empirical evaluation of a walk-relax-and-round heuristics for mixed integer convex programs. Technical report, Northwestern University (under review).
5. S Mehrotra and KL Huang (2013). On Implementing a General Disjunctive Branching Algorithm Using Lattice Basis Reduction for Mixed Integer Convex Programming. Technical report, Northwestern University (submitted).
6. KL Huang and S Mehrotra (2013). Solution of Monotone Complementarity and General Convex Programming using a modified Potential Reduction Interior Point Method. Technical report, Northwestern University (under review).
7. U Koc, KL Huang and S Mehrotra (2014). Concurrent Generation of Feasible Integer Solutions Using Random Walks in Parallel. Technical report, Northwestern University (in preparation).
8. U Koc and S Mehrotra (2014). A Cut-and-Branch framework for solving Mixed Integer Programs in Parallel (in preparation).