American Journal of Operations Research, 2013, 3, 53-64
http://dx.doi.org/10.4236/ajor.2013.31004 Published Online January 2013 (http://www.scirp.org/journal/ajor)

Constraint Optimal Selection Techniques (COSTs) for Linear Programming*

Goh Saito#, H. W. Corley, Jay M. Rosenberger
IMSE Department, The University of Texas at Arlington, Arlington, USA
Email: #goh.saito@mavs.uta.edu

Received June 28, 2012; revised July 30, 2012; accepted August 15, 2012

ABSTRACT

We describe a new active-set, cutting-plane Constraint Optimal Selection Technique (COST) for solving general linear programming problems. We describe strategies to bound the initial problem and simultaneously add multiple constraints. We give an interpretation of the new COST's selection rule, which considers both the depth of constraints as well as their angles from the objective function. We provide computational comparisons of the COST with existing linear programming algorithms, including other COSTs in the literature, for some large-scale problems. Finally, we discuss conclusions and future research.

Keywords: Linear Programming; Large-Scale Linear Programming; Cutting Planes; Active-Set Methods; Constraint Selection; COSTs

1. Introduction

1.1. The General Linear Programming Problem

Linear programming is a tool for optimizing numerous real-world problems such as the allocation problem. Consider a general linear program (LP) as the following problem P

$$\text{maximize } z = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \tag{1}$$

$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \tag{2}$$

$$\boldsymbol{x} \ge \boldsymbol{0}, \tag{3}$$

where z represents the objective function for n variables

$$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

and the expression (2) describes m rows of constraints for n variables

$$\begin{bmatrix} a_1^{\mathrm{T}} \\ a_2^{\mathrm{T}} \\ \vdots \\ a_m^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \le \begin{bmatrix} b_1, \\ b_2, \\ \vdots \\ b_m, \end{bmatrix}.$$

Furthermore, the vector $\boldsymbol{0}$ in (3) is a column vector of zeros of appropriate dimension according to context. The dual of P is considered the standard minimization LP problem. We focus here on the maximization case.

A COST RAD [1] utilizing multi-bound and multi-cut techniques was developed by Saito *et al*. [2] for nonnegative linear programs (NNLPs). In NNLPs, $\boldsymbol{a}_i \ge \boldsymbol{0}$ and $\boldsymbol{a}_i \ne \boldsymbol{0}, \forall i = 1, \cdots, m; \boldsymbol{b} > \boldsymbol{0}$; and $\boldsymbol{c} > \boldsymbol{0}$. However in LPs, the components of $\boldsymbol{A}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ are not restricted to be nonnegative numbers.

Though simplex pivoting algorithms and polynomial interior-point barrier-function methods represent the two principal solution approaches to solve problem P [3], there is no single best algorithm. For either method, we can always formulate an instance of P for which the method performs poorly [4]. However, simplex methods remain the dominant approach because they have advantages over interior-point methods, such as efficient post-optimality analysis of pivoting algorithms, application of cutting-plane methods, and delayed column generation. Current simplex algorithms are often inadequate, though, for solving a large-scale LPs because of their insufficient computational speeds. In particular, emerging technologies require computer solutions in nearly real time for problems involving millions of constraints or variables. Hence faster techniques are needed. The COST of this paper represents a viable such approach.

1.2. Background and Literature Review

An active-set framework for solving LPs will be analo-

*All the authors contributed equally to this research.
#Corresponding author.

gous to that of Saito *et al*. [2] for NNLPs. We begin with a relaxation of P, with a single artificial bounding constraint such as $\mathbf{1}\boldsymbol{x} \le M$ or $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \le M$ for sufficiently large M so as not to reduce the feasible region of P.

A series of relaxations $P_r, r = 0,1,2,\cdots,$ of P is formed by adding one or more violating constraints from set (2). The constraints that have been added are called *operative constraints*, while constraints that still remain in (2) are called *inoperative constraints*. Eventually a solution $\boldsymbol{x}_r^*$ of P is obtained when none of the inoperative constraints are violated; *i.e*., for no inoperative constraint i is $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}_r^* - b_i > 0$. In the LP COSTs of this paper we explore 1) the ordering of a set of inoperable constraints for possibly adding them to the current operable constraints and 2) the actual selection of a group of such constraints to be added at an iteration.

Active-set approaches have been studied in the past, including those by Stone [5], Thompson *et al*. [6], Adler *et al*. [7], Zeleny [8], Myers and Shih [9], and Curet [10], with the term "constraint selection technique" used in Myers and Shih [9]. Adler *et al*. [7] added constraints randomly, without any selection criteria. Zeleny [8] added a constraint that was most violated by the problem P_r to form P_{r+1}. These methods are called SUB and VIOL here, respectively, as in Saito *et al*. [2]. Also, VIOL, which is a standard pricing method for delayed column generation in terms of the dual [11], is identical to the Priority Constraint Method of Thompson *et al*. [6]. In all of these approaches, constraints were added one at a time.

More recent work on constraint selection has focused on choosing the violated inoperative constraints considered most likely to be binding at optimality for the original problem P according to a particular constraint selection criterion. In the cosine criterion, the angle between normal vector $\boldsymbol{a}_i$ of (2) and normal vector $\boldsymbol{c}$ of (1) as measured by the cosine,

$$\cos(\boldsymbol{a}_i, \boldsymbol{c}) = \frac{\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{c}}{\|\boldsymbol{a}_i\|\|\boldsymbol{c}\|},$$

is considered. Naylor and Sell ([12], pp. 273-274), for example, suggest that a constraint with a larger cosine value may be more likely to be binding at optimality. Pan [13,14] applied the cosine criterion to pivot rules of the simplex algorithm as the "most-obtuse-angle" rule. The cosine criterion has also been utilized to obtain an initial basis for the simplex algorithm by Trigos *et al*. [15] and Junior *et al*. [16]. Corley *et al*. [1,17] chose for P_{r+1} a single inoperative constraint $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \le b_i$ of P_r violating $\boldsymbol{x}_r^*$ and having the largest $\cos(\boldsymbol{a}_i, \boldsymbol{c})$.

In Saito *et al*. [2] the COST RAD was developed for NNLPs. It was based on the following two geometric factors. Factor I is the angle that the a constraint's normal vector $\boldsymbol{a}_i$ formed with the normal vector $\boldsymbol{c}$ of the objective function. Factor II is the depth of the cut that constraint i removes as a violated inoperative constraint of P_r. From these two factors the constraint selection metric

$$\mathrm{RAD}(\boldsymbol{a}_i, b_i, \boldsymbol{c}) = \frac{\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{c}}{b_i}, \tag{4}$$

was developed. This constraint selection metric was utilized in conjunction with a multi-bound and multi-cut technique [2] in which multiple constraints were effectively selected from a RAD-ordered set of inoperative violating constraints for forming each P_r. In this paper we rename the COST RAD of [2] for NNLPs as NRAD.

1.3. Contribution

Although NRAD performs extremely well for NNLPs, we show here that its superiority over traditional algorithms is less dramatic for the general LP (1)-(3). Hence, the contribution of this paper includes using the principles of NRAD to develop a new COST for the general LP, which we refer to as GRAD. Even though GRAD and NRAD share similar principles, GRAD is a significant modification of NRAD. Indeed, GRAD solves general LPs seven times faster on the average than NRAD in our computational experiments.

The remainder of this paper is organized as follows. GRAD with multi-cuts is developed in Section 2, and an interpretation is given. In Section 3, we present computational results where GRAD is compared to the CPLEX simplex methods, CPLEX barrier method, and the active-set approaches SUB, VIOL, and NRAD. In Section 4 we offer conclusions and discuss future research.

2. The COST GRAD

2.1. NNLP vs LP

An active-set framework for solving general LPs will be analogous to that for NNLPs. However, GRAD is not an immediate extension of NRAD since LPs do not have some of the useful properties of NNLP problems. For example, the origin $\boldsymbol{x} = \mathbf{0}$ is no longer guaranteed to be feasible for LP problems. Moreover, the optimal solution $\boldsymbol{x}^*$ may not lie in the same orthant as the normal $\boldsymbol{a}_i$ to a constraint. We must thus modify NRAD for NNLP to GRAD for LP in order to emulate the underlying reasoning of NRAD based on Factors I and II efficiently.

2.2. Constraint Selection Criterion

Boundedness of NNLP could be assured by adding multiple constraints from (2) until no column of $\boldsymbol{A}$ is a zero vector. However, this is not the case for LP. Therefore an initial bounded problem P_0 is formed by adding a bounding constraint such as $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \le M$, along with

some constraints from (2), as described in Section 2.3. P_0 is then solved to obtain an initial solution $\boldsymbol{x}_0^*$. P_{r+1} is generated by adding one or more inoperative constraints of P_r that maximize the constraint selection metric for LP among all inoperative constraints of P_r violating $\boldsymbol{x}_r^*$. Define this constraint selection metric as

$$\mathrm{GRAD}(\boldsymbol{a}_i, b_i, \boldsymbol{c}) = \sum_{j=1, c_j>0}^{n} \frac{a_{ij} c_j}{b_i^+} - \sum_{j=1, c_j<0}^{n} \frac{-a_{ij}}{b_i^+}, \tag{5}$$

where

$$b_i^+ = \begin{cases} b_i - \min\limits_{k=1,\cdots,m} [b_k] + \varepsilon, \text{if } \min\limits_{k=1,\cdots,m} [b_k] \le 0 \\ b_i, \text{otherwise,} \end{cases} \tag{6}$$

and ε is a small positive constant. Thus GRAD seeks i^* such that

$$i^* \in \underset{i \notin \mathrm{OPERATIVE}}{\arg\max} \left(\mathrm{GRAD}(\boldsymbol{a}_i, b_i, \boldsymbol{c}) \middle| a_i^{\mathrm{T}} x_r^* > b_i \right).$$

The first term in (5) is a quantity that invokes Factor I and Factor II analogous to NRAD, while the second term is a quantity that invokes Factor II. In (6), values of b_i are shifted by $\min\limits_{k=1,\cdots,m} [b_i]$ if the minimum value is non-positive. Hence b_i^+ is always positive, and each term in (5) contributes additively to the criterion. GRAD (5) becomes the same as NRAD (4) when $\boldsymbol{b} > \boldsymbol{0}$ and $\boldsymbol{c} > \boldsymbol{0}$. Therefore it could be utilized to effectively solve NNLPs as well. Although equality constraints are not considered here, it should be noted that equality constraints could be included in P_0.

2.3. Multi-Bound and Multi-Cut for LP

The boundedness of the initial problem P_0 for NNLP is obtained by adding multiple constraints from (2) ordered by decreasing value of NRAD until no column of $\boldsymbol{A}$ is a zero vector. Although this approach does not guarantee boundedness for LP, a generalization was found to be effective here.

For the COST GRAD, an initial bounded problem P_0 is formed by adding an artificial bounding constraint such as $\boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \le M$, as well as multiple constraints from (2) ordered by decreasing value of GRAD, until each column of $\boldsymbol{A}$ has at least one positive and at least one negative coefficient (Step 1). After an optimal solution to the initial bounded problem is obtained by the primal simplex method (Step 2), subsequent iterations are solved by the dual simplex method (Step 3). Moreover, after the solution of P_0 and each subsequent P_r, constraints are again added in groups. P_{r+1} is formed by selecting inoperative constraints in decreasing order of GRAD until both a positive coefficient and a negative coefficient are included for each variable x_j (Step 3, lines 7-27). The following pseudocode describes the COST GRAD with the new multi-cut technique.

Step 1—Identify constraints to form the initial problem P_0.
1: **for** $i = 1 \to m$ **do**
2: **if** $\exists j \left| a_{ij} > 0 \right.$ **then**
3: $\mathrm{POSITIVE}_a \leftarrow \mathrm{POSITIVE}_a \cup \{i\}$
4: **end if**
5: **if** $\exists j \left| a_{ij} < 0 \right.$ **then**
6: $\mathrm{NEGATIVE}_a \leftarrow \mathrm{NEGATIVE}_a \cup \{i\}$
7: **end if**
8: **end for**
9: $\mathrm{OPERATIVE} \leftarrow \varnothing$, $\mathrm{EXPLORED} \leftarrow \mathrm{OPERATIVE}$, $\boldsymbol{a}^{*\mathrm{positive}} \leftarrow 0$, $\boldsymbol{a}^{*\mathrm{negative}} \leftarrow 0$
10: **while** $\boldsymbol{a}^{*\mathrm{positive}} \not> 0$ **and** $\boldsymbol{a}^{*\mathrm{negative}} \not> 0$ **and** $\mathrm{EXPLORED} \subset \{1, \cdots, m\}$ **do**
11: Let $i^* \in \underset{i \notin \mathrm{EXPLORED}}{\mathrm{argmax}} \mathrm{GRAD}(\boldsymbol{a}_i, b_i, \mathbf{c})$
12: $\mathrm{EXPLORED} \leftarrow \mathrm{EXPLORED} \cup \{i^*\}$
13: **if** $\exists j \left| a_j^{*\mathrm{positive}} = 0 \right.$ **and** $a_{i^* j} > 0$ **then**
14: $\mathrm{OPERATIVE} \leftarrow \mathrm{OPERATIVE} \cup \{i^*\}$
15: **if** $\mathrm{POSITIVE}_a \subseteq \mathrm{EXPLORED}$ **then**
16: $\mathbf{a}^+ \leftarrow 1$ //case if there are no more constraints with $a_{ij} > 0$
17: **else**

18: $\boldsymbol{a}^{+} \leftarrow \left[a^{+}_{i^*1},\cdots,a^{+}_{i^*n}\right]$ where $a^{+}_{i^*j} = \begin{cases}1, \text{if } a_{i^*j} > 0\\ 0, \text{otherwise}\end{cases}, \forall j = 1,\cdots,n$

19: **end if**

20: **end if**

21: **if** $\exists j \left| a_j^{*\text{negative}} = 0 \right.$ **and** $a_{i^*j} < 0$ **then**

22: OPERATIVE $\leftarrow$ OPERATIVE $\cup \left\{i^*\right\}$

23: **if** NEGATIVE$_a \subseteq$ EXPLORED **then**

24: $\boldsymbol{a}^{-} \leftarrow 1$ // case if there are no more constraints with $a_{ij} < 0$

25: **else**

26: $\boldsymbol{a}^{-} \leftarrow \left[a^{-}_{i^*1},\cdots,a^{-}_{i^*n}\right]$ where $a^{-}_{i^*j} = \begin{cases}1, \text{if } a_{i^*j} < 0\\ 0, \text{otherwise}\end{cases}, \forall j = 1,\cdots,n$

27: **end if**

28: **end if**

29: $\boldsymbol{a}^{*\text{positive}} \leftarrow \boldsymbol{a}^{*\text{positive}} + \boldsymbol{a}^{+}$, $\boldsymbol{a}^{*\text{negative}} \leftarrow \boldsymbol{a}^{*\text{negative}} + \boldsymbol{a}^{-}$

30: **end while**

Step 2—Using the primal simplex method, obtain an optimal solution x_0^* for the initial bounded problem P_0

$$\text{maximize } z = \mathbf{c}^{\mathrm{T}}\boldsymbol{x} \tag{7}$$

$$\text{subject to } \mathbf{c}^{\mathrm{T}}\boldsymbol{x} \le M \tag{8}$$

$$\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} \le b_i, \forall i \in \text{OPERATIVE} \tag{9}$$

$$x \ge \mathbf{0}. \tag{10}$$

Step 3—Perform the following iterations until an optimal solution to problem P is found.

1: $r \leftarrow 0$, STOP $\leftarrow$ **false**

2: **while** STOP = **false do**

3: **if** $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}_{\boldsymbol{r}}^* \le b_i \forall i \notin$ OPERATIVE **then**

4: STOP $\leftarrow$ **true** // $\boldsymbol{x}_r^*$ is an optimal solution to P.

5: **else**

6: EXPLORED $\leftarrow$ OPERATIVE, $\boldsymbol{a}^{*\text{positive}} \leftarrow 0$, $\boldsymbol{a}^{*\text{negative}} \leftarrow 0$

7: **while** $\boldsymbol{a}^{*\text{positive}} \not> 0$ **and** $\boldsymbol{a}^{*\text{negative}} \not> 0$ **and** EXPLORED $\subset \left\{i = 1,\cdots,m \left| \boldsymbol{a}_i^T \boldsymbol{x}_r^* > b_i \right.\right\}$ **do**

8: Let $i^* \in \underset{i \notin \text{EXPLORED}}{\text{argmax}}\ \text{GRAD}\left(\boldsymbol{a}_i, b_i, \mathbf{c} \left| \boldsymbol{a}_i^T \boldsymbol{x}_r^* > b_i \right.\right)$

9: EXPLORED $\leftarrow$ EXPLORED $\cup \left\{i^*\right\}$

10: **if** $\exists j \left| a_j^{*\text{positive}} = 0 \right.$ **and** $a_{i^*j} > 0$ **then**

11: OPERATIVE $\leftarrow$ OPERATIVE $\cup \left\{i^*\right\}$

12: **if** POSITIVE$_a \subseteq$ EXPLORED **then**

13: $\boldsymbol{a}^{+} \leftarrow \mathbf{1}$ //case if there are no more constraints with $a_{ij} > 0$

14: **else**

15: $a^{+} \leftarrow \left[a^{+}_{i^*1},\cdots,a^{+}_{i^*n}\right]$ where $a^{+}_{i^*j} = \begin{cases}1, \text{if } a_{i^*j} > 0\\ 0, \text{otherwise}\end{cases}, \forall j = 1,\cdots,n$

16: **end if**

17: **end if**

18: **if** $\exists j \left| a_j^{*\text{negative}} = 0 \right.$ **and** $a_{i^*j} < 0$ **then**

19: OPERATIVE $\leftarrow$ OPERATIVE $\bigcup \left\{i^*\right\}$

20: **if** NEGATIVE$_a \subseteq$ EXPLORED **then**
21: $\boldsymbol{a}^- \leftarrow 1$ case if there are no more constraints with $a_{ij} < 0$
22: **else**
23: $\boldsymbol{a}^- \leftarrow \left[a^-_{i^*1}, \cdots, a^-_{i^*n}\right]$ where $a^-_{i^*j} = \begin{cases} 1, \text{if } a_{i^*j} < 0 \\ 0, \text{otherwise} \end{cases}, \forall j = 1, \cdots, n$
24: **end if**
25: **end if**
26: $\boldsymbol{a}^{*\text{positive}} \leftarrow \boldsymbol{a}^{*\text{positive}} + \boldsymbol{a}^+$, $\boldsymbol{a}^{*\text{negative}} \leftarrow \boldsymbol{a}^{*\text{negative}} + \boldsymbol{a}^-$
27: **end while**
28: $r \leftarrow r + 1$
29: Solve P_r defined by (7)-(10) using the dual simplex method to obtain x^*_r.
30: **end if**
31: **end while**

2.4. Interpretation of GRAD

The interpretation of NRAD in Saito *et al*. [2] utilized the fact that an NNLP always results in a positive value for $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{c}$. However for LP of (1)-(3), the intersection of $\boldsymbol{c}$, drawn from the origin, and $\boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x} = b_i$ may not necessarily lie in the feasible region. Moreover, the c_j of NNLP are all positive. The GRAD constraint selection metric of (5) thus must be modified from $\mathrm{RAD}(\boldsymbol{a}_i, b_i, \boldsymbol{c})$ in (4) to account for these facts.

For LP, the basic idea for determining whether a constraint from (2) is likely to be binding at optimality is described as follows. Given an objective function

$$\max z = c_1x_1 + c_2x_2 + \cdots + c_nx_n,$$

observe that z is maximized when c_j and x_j are large. Hence, a larger value of c_j is more likely to yield a larger value of x_j. This relationship implies that the left-hand side of the constraint is likely to be larger for larger values of

$$\sum_{j=1, c_j>0}^{n} a_{ij}c_j.$$

For a_j with $c_j < 0$, it is hard to predict the value of x_j in a solution. Consequently, we assume that the x_j in which $c_j < 0$ are all equally likely and have the nominal value 1. The left-hand side is now

$$\sum_{j=1, c_j>0}^{n} a_{ij}c_j + \sum_{j=1, c_j<0}^{n} a_{ij}.$$

As for the right-hand side of the constraint, a small b_i makes a constraint more likely to be binding. We thus divide the left-hand side by b_i to measure the ith constraint's likelihood of being binding at optimality, resulting in

$$\sum_{j=1, c_j>0}^{n} \frac{a_{ij}c_j}{b_i} + \sum_{j=1, c_j<0}^{n} \frac{a_{ij}}{b_i},$$

which is essentially GRAD.

GRAD can also be derived from NRAD. Note that the term

$$\sum_{j=1, c_j>0}^{n} \frac{a_{ij}c_j}{b_i^+}$$

in (5) is simply RAD for an NNLP. For LP we add a term involving the j for which the c_j are negative. Consider (11) below.

$$\sum_{j=1, c_j>0}^{n} \frac{a_{ij}c_j}{b_i^+} - \sum_{j=1, c_j<0}^{n} \frac{a_{ij}c_j}{b_i^+}. \tag{11}$$

The second term in (11) makes sense from the point of view that the expression (11) results in a higher value when a_{ij} and c_j are both negative, and b_i is large. However, it is found in Section 3.3.1 that the constraint selection metric performed better when the second term was

$$\sum_{j=1, c_j<0}^{n} \frac{-a_{ij}}{b_i^+},$$

as shown in (5).

3. Computational Experiments

The COST GRAD (5) was compared with the CPLEX primal simplex method, the CPLEX dual simplex method, the polynomial interior-point CPLEX barrier method, as well as the previously defined constraint selection techniques SUB, VIOL, and NRAD (4). GRAD, NRAD, SUB, and VIOL utilized the CPLEX dual simplex solver to solve each new relaxed problem P_{r+1}.

3.1. Problem Instances

A set of 105 randomly generated LP was constructed. The LP problems were generated with 1000 variables (n) and 200,000 constraints (m) having various densities ranging from 0.005 to 1. Randomly generated real

numbers between 1 and 5, or between −1 and −5 were assigned to elements of $\boldsymbol{A}$. To assure that the randomly generated LP had a feasible solution, a feasible solution $\boldsymbol{x}^*$ (not all elements of $\boldsymbol{x}^*$ were nonzero) was randomly generated to derive random $\boldsymbol{b}$, where $\boldsymbol{A}\boldsymbol{x}^* \le \boldsymbol{b}$. Then a feasible solution $\boldsymbol{y}^*$ (having the same number of nonzero elements as $\boldsymbol{x}^*$) was randomly generated to derive random $\boldsymbol{c}$, where $\boldsymbol{y}^*\boldsymbol{A} \ge \boldsymbol{c}$. The ratio of the number of positive and negative elements of $\boldsymbol{A}$ was one. The number of nonzero a_{ij} in each constraint was binomially distributed $B(n, p = \text{density})$. Additionally, we required each constraint to have at least two nonzero a_{ij} so that a constraint would not become a simple upper or lower bound on a variable. At each of the 21 densities, 5 random LP were generated. **Table 1** summarizes the generated LP.

Table 1. Randomly generated general LP problem set [18].

Number of variables	1000
Number of constraints	200,000
Range of a_{ij}	$1 \le$ random real ≤ 5, or $-5 \le$ random real ≤ -1
Fraction of positive a_{ij}	0.5

Problem instance	Average of 5 instances of LP at each density										
	Density	Number of nonzero a_{ij} in a constraint			b_i			c_j			Number of binding constraints at optimality
	Mean	Min	Mean	Max	Min	Mean	Max	Min	Mean	Max	Mean
1-5	0.00505	2.0	5.1	17.6	−2.5	1.0	4.4	−19.2	−1.0	16.0	836.0
6-10	0.00602	2.0	6.0	19.6	−2.4	1.0	4.3	−18.2	−0.9	15.1	834.2
11-15	0.00701	2.0	7.0	23.0	−2.2	1.0	4.2	−19.0	−0.9	13.7	827.6
16-20	0.00801	2.0	8.0	23.2	−2.0	1.0	4.1	−19.8	−1.0	14.6	812.4
21-25	0.00900	2.0	9.0	25.4	−1.8	1.0	3.9	−17.7	−1.1	14.8	803.6
26-30	0.01000	2.0	10.0	27.6	−1.8	1.0	3.8	−17.1	−0.9	13.1	807.0
31-35	0.02000	4.0	20.0	43.2	−1.6	1.0	3.6	−15.2	−1.0	12.9	754.0
36-40	0.03001	8.8	30.0	60.2	−1.3	1.0	3.3	−13.3	−1.0	11.6	723.0
41-45	0.04000	15.8	40.0	70.6	−1.3	1.0	3.2	−11.6	−1.1	9.5	694.4
46-50	0.04999	22.0	50.0	82.8	−1.2	1.0	3.2	−13.1	−0.9	10.0	696.0
51-55	0.06000	29.8	60.0	97.8	−1.0	1.0	3.0	−11.1	−1.0	9.7	659.8
56-60	0.07000	38.0	70.0	110.4	−1.1	1.0	3.1	−9.9	−1.0	9.4	664.8
61-65	0.08001	43.8	80.0	123.6	−1.0	1.0	2.9	−12.0	−1.0	9.3	647.4
66-70	0.08999	53.2	90.0	133.6	−0.9	1.0	2.9	−9.7	−1.0	8.2	630.0
71-75	0.10000	61.0	100.0	145.0	−0.9	1.0	2.9	−11.5	−1.0	9.2	633.6
76-80	0.20001	146.2	200.0	261.4	−0.8	1.0	2.8	−9.2	−1.0	8.1	578.6
81-85	0.30001	236.8	300.0	369.6	−0.6	1.0	2.6	−7.9	−1.0	5.6	546.6
86-90	0.40000	332.6	400.0	471.4	−0.6	1.0	2.6	−9.1	−1.0	6.8	530.0
91-95	0.50001	429.6	500.0	574.0	−0.7	1.0	2.6	−8.6	−1.0	6.6	514.0
96-100	0.75000	688.8	750.0	811.4	−0.6	1.0	2.5	−8.4	−1.0	5.8	472.2
101-105	1.00000	999.0	1000.0	1000.0	−0.6	1.0	2.6	−6.5	−1.0	5.1	432.6

3.2. CPLEX Preprocessing

As in Saito *et al*. [2], the CPLEX preprocessing parameters PREIND (preprocessing presolve indicator) and PREDUAL (preprocessing dual) had to be chosen appropriately. The default parameter settings of PREIND = 1 (ON) and PREDUAL = 0 (AUTO) were used for CPU times of the CPLEX primal simplex method, the CPLEX dual simplex method, and the CPLEX barrier method. No CPLEX preprocessing was implemented [used PREIND = 0 (OFF) and PREDUAL = −1 (OFF)] by the CPLEX primal simplex and dual simplex solvers as part of GRAD, NRAD, SUB, and VIOL.

3.3. Computational Results

Comparisons of computational methods were performed with the IBM CPLEX 12.1 callable library on an Intel Core 2 Duo E8600 3.33 GHz workstation with a Linux 64-bit operating system and 8 GB of RAM. Computational test results of **Tables 2** through **5** were obtained by calling CPLEX commands from an application written in the programming language C. In these tables, each CPU time presented is an average computation time of solving five instances of randomly generated LP.

Computational results for the CPLEX primal simplex, dual simplex, and barrier solvers for the general LP set are presented in **Table 2**. CPU times for the COST GRAD with multi-cut, as well as the COST NRAD with multi-bound and multi-cut are shown for comparison. The CPU times for GRAD were faster than the CPLEX primal simplex, the CPLEX dual simplex, and the CPLEX barrier linear programming solvers at densities between 0.02 and 1. Between densities 0.005 and 0.01, CPLEX barrier was up to 4.0 times faster than GRAD. On average, GRAD was 7.0 times faster than NRAD applied to these non-NNLP problems and 14.6 times faster than the fastest CPLEX solver, which was the dual simplex.

3.3.1. Influences of the COST GRAD and Multi-Cut

In constructing a constraint selection metric for LP, a

Table 2.Comparison of computation times of CPLEX and COST RAD methods on LP problem set.

	CPLEX Primal Simplex	CPLEX Dual Simplex	CPLEX Barrier	GRAD with multi-cut	NRAD with multi-bound and multi-cut [2]
Presolve	On	On	On	Off	Off
Predual	Auto	Auto	Auto	Off	Off
Density	CPU TIME† (std. dev.), sec				
0.00505	41.1 (2.3)	21.6 (1.4)	2.4 (0.1)	7.9 (1.0)	15.0 (0.8)
0.00602	86.0 (8.2)	36.6 (1.1)	2.9 (0.1)	9.3 (0.5)	21.5 (4.7)
0.00701	134.8 (7.8)	49.3 (3.2)	4.6 (0.3)	13.4 (0.4)	27.4 (3.9)
0.00801	186.5 (11.1)	65.6 (4.3)	7.8 (0.4)	14.4 (0.9)	31.4 (3.6)
0.00900	215.3 (17.5)	84.8 (9.8)	9.3 (0.3)	12.6 (0.7)	33.9 (5.2)
0.01000	263.2 (18.1)	100.5 (11.9)	11.2 (0.2)	14.2 (0.4)	36.6 (5.3)
0.02000	412.0 (25.2)	223.3 (20.3)	27.4 (1.1)	23.3 (2.1)	69.1 (6.5)
0.03001	503.7 (46.3)	317.4 (28.9)	45.8 (1.3)	26.7 (1.5)	100.7 (8.5)
0.04000	575.2 (40.1)	389.2 (34.0)	64.9 (3.9)	27.4 (0.4)	103.6 (4.8)
0.04999	672.5 (94.3)	427.4 (37.2)	82.9 (3.9)	33.5 (1.6)	132.2 (7.0)
0.06000	718.3 (80.0)	497.9 (38.5)	99.9 (5.5)	31.8 (2.2)	130.4 (4.3)
0.07000	841.0 (119.0)	531.1 (44.2)	125.4 (9.8)	34.5 (1.8)	142.0 (4.7)
0.08001	835.0 (95.8)	570.5 (42.5)	145.0 (14.7)	32.9 (2.5)	150.7 (14.3)
0.08999	930.6 (111.0)	595.8 (50.0)	176.3 (27.0)	35.4 (1.0)	160.4 (10.3)
0.10000	1029.9 (131.4)	627.8 (21.5)	203.1 (11.0)	36.2 (4.6)	175.2 (12.6)
0.20001	1667.8 (293.8)	857.8 (68.5)	553.2 (21.2)	52.7 (3.5)	266.4 (18.9)
0.30001	1939.7 (185.5)	1016.9 (69.2)	1097.9 (90.4)	60.1 (6.8)	273.0 (22.1)
0.40000	2535.9 (857.5)	1173.3 (117.7)	1684.0 (124.8)	73.0 (6.3)	403.1 (48.0)
0.50001	2383.3 (466.3)	1421.3 (153.3)	2480.3 (156.1)	83.1 (5.6)	460.1 (31.9)
0.75000	2727.7 (287.1)	1775.3 (168.8)	5026.2 (217.4)	119.4 (11.8)	712.5 (62.9)
1.00000	2972.7 (397.2)	1904.8 (203.3)	8463.6 (950.4)	143.9 (6.2)	2600.9 (188.1)
Average (pooled standard deviation)	1032.0 (257.0)	604.2 (78.3)	967.3 (218.3)	42.2 (4.1)	287.9 (45.9)

†Average of 5 instances of LP at each density.

natural strategy might be to have the metric give priority to those constraints with either "as large positive a_{ij} and large positive c_j with small b_i as possible," or "as small negative a_{ij} and small negative c_j with large b_i as possible." However, when running LP problems utilizing NRAD (4), the constraint selection metric is

$$\mathrm{NRAD}(\boldsymbol{a}_i, b_i, \boldsymbol{c}) = \sum_{j=1, c_j>0}^{n} \frac{a_{ij}c_j}{b_i} + \sum_{j=1, c_j<0}^{n} \frac{a_{ij}c_j}{b_i}, \qquad (12)$$

if the terms for $c_j > 0$ and $c_j < 0$ are explicitly written out. The first term follows the above general strategy, except when b_i becomes negative. The second term should be subtracted, instead of added, from the first term in order for the constraint selection metric to take a higher value when giving priority to "as small negative a_{ij} and small negative c_j with large b_i as possible." The b_i in the second term should also be positive for the metric to work additively.

To examine the effect of changing the form of NRAD (12) to GRAD, several intermediate variations are tested and presented in **Table 3**. The results utilizing SUB are also shown in the table for comparison. The first variation,

Table 3. Comparison of computation times to illustrate the effects of muti-cut, NRAD and GRAD on LP problem set.

	Constraint selection metric†							
	SUB	SUB	NRAD	NRAD	NRAD variation 1 (13)	NRAD variation 2 (14)	NRAD variation 3 (11)	GRAD
	Multiple cuts for NNLP	Multiple cuts for LP	Multiple cuts for NNLP	Multiple cuts for LP	Multiple cuts for LP			
Density	CPU TIME‡, sec							
0.00505	16.2	14.5	15.0	13.9	14.0	10.1	11.2	9.4
0.00602	16.8	15.1	21.5	15.7	15.5	10.8	13.0	11.4
0.00701	22.9	17.9	27.4	18.6	22.1	12.4	18.5	15.3
0.00801	30.9	19.3	31.4	21.0	24.2	13.7	20.7	14.8
0.00900	31.2	22.4	33.9	22.4	21.7	14.2	17.9	13.4
0.01000	31.8	28.6	36.6	28.8	25.8	15.4	19.7	16.4
0.02000	73.5	57.0	69.1	53.4	53.3	23.9	34.7	25.8
0.03001	98.2	79.6	100.7	73.5	66.7	27.9	38.3	26.9
0.04000	111.4	83.4	103.6	75.8	76.4	29.6	40.7	27.6
0.04999	144.6	100.1	132.2	87.3	90.1	38.7	49.9	34.5
0.06000	148.5	106.3	130.4	91.7	93.3	35.9	47.1	32.5
0.07000	160.0	109.4	142.0	93.8	108.5	39.6	52.2	34.5
0.08001	179.0	116.6	150.7	98.4	98.5	37.5	49.4	32.5
0.08999	191.0	122.8	160.4	102.8	111.5	38.8	53.2	34.9
0.10000	201.1	133.5	175.2	111.4	112.5	41.6	57.7	36.2
0.20001	312.7	196.1	266.4	166.6	176.6	57.5	83.3	50.4
0.30001	389.0	246.0	273.0	168.5	195.4	59.2	98.2	55.8
0.40000	566.8	344.5	403.1	238.0	254.7	74.6	119.8	70.0
0.50001	734.8	431.0	460.1	284.9	299.5	90.9	136.1	79.4
0.75000	1158.4	614.8	712.5	379.7	430.7	113.0	186.2	107.2
1.00000	3662.7	774.6	2600.9	455.4	522.1	134.8	236.9	138.5
Average	394.4	173.0	287.9	123.9	134.0	43.8	65.9	41.3

†Used CPLEX preprocessing parameters of presolve = off and predual = off. ‡Average of 5 instances of LP at each density.

$$\text{NRAD}_{\text{variation1}}(\boldsymbol{a}_i, b_i, \boldsymbol{c}) = \sum_{j=1, c_j>0}^{n} \frac{a_{ij} c_j}{b_i}, \quad (13)$$

is a version only considering the $c_j > 0$ term of (12). The test problems of **Table 1** did not have any constraints with $b_i = 0$, therefore the case was not specially handled. The second version,

$$\text{NRAD}_{\text{variation2}}(\boldsymbol{a}_i, b_i, \boldsymbol{c}) = \sum_{j=1, c_j>0}^{n} \frac{a_{ij} c_j}{b_i^+}, \quad (14)$$

is (13) with b_i^+, which was defined in (6). Variation 3,

$$\text{NRAD}_{\text{variation3}}(\boldsymbol{a}_i, b_i, \boldsymbol{c}) = \sum_{j=1, c_j>0}^{n} \frac{a_{ij} c_j}{b_i^+} - \sum_{j=1, c_j<0}^{n} \frac{a_{ij} c_j}{b_i^+},$$

subtracts a term from (14) to give (11) above. For calculation of $b_i^+, \varepsilon = 10^{-10}$ was used for all results presented.

Results for SUB and NRAD from **Table 3** show that the multi-cut method for LP reduced CPU times by 56% to 57% over the multi-cut method for NNLP to support the importance of having both positive and negative a_{ij} for every variable x_j in forming a set of cuts for each iteration of an active-set method for LP. The rest of the comparisons are made with those methods utilizing the multi-cut method for LP.

For densities between 0.005 and 0.09, SUB, NRAD, and variation 1 (13) of NRAD performed about the same. Introducing the use of b_i^+ in (14), (5), and (11) significantly improved the CPU times over (13). Between (11) and (14), (14) which only considered the $c_j > 0$ term was faster. Going from (14) to GRAD, utilizing the term $\sum_{j=1, c_j<0}^{n} \frac{-a_{ij}}{b_i^+}$ improved the CPU time slightly more, 5.7% on average.

The COST GRAD with multiple cuts for LP was also tested on NNLP problem Set 1 from Saito *et al*. [2], as shown in **Table 4**. The results confirm that the methods perform equally for NNLP. Although the constraint selection metric becomes exactly the same between the two methods when running NNLP, a slight increase in CPU times for the COST GRAD occurs because of the time it takes for the algorithm to determine whether the problem is an NNLP (pseudocode in Section 2.3, Step 1, lines 1 through 8). In the case of solving NNLP with the GRAD, this check allows the multi-cut procedure to stop searching for negative a_{ij} if there are no constraints with negative a_{ij} in the inoperative set.

Table 4. Comparison of computation times of NRAD and GRAD on NNLP Set 1 from Saito *et al.* [2].

	CPU TIME†, sec	
Density	COST NRAD	COST GRAD
0.00505	2.1	2.1
0.00602	2.4	2.5
0.00701	2.7	2.7
0.00800	2.5	2.6
0.00900	2.8	2.8
0.01000	2.8	2.8
0.02000	3.1	3.2
0.03000	3.3	3.4
0.04001	3.4	3.5
0.05000	3.4	3.5
0.06000	3.2	3.4
0.06999	3.4	3.6
0.08001	3.3	3.6
0.08999	3.4	3.7
0.10000	3.3	3.7
0.19999	4.3	4.9
0.30001	4.9	5.8
0.40000	5.6	6.9
0.49998	6.7	8.3
0.75001	7.9	10.3
1.00000	8.1	11.5
Average	3.9	4.5

†Average of 5 instances of LP at each density. Used CPLEX preprocessing parameters of Presolve = off and Predual = off.

3.3.2. Number of Constraints Added

In **Table 5**, CPU times and the number of constraints added during computation of the test problems by GRAD (both single-cut and multi-cut versions) are compared with the constraint selection methods SUB and VIOL of Adler *et al.* [7] and Zeleny [8], respectively. "Number of Constraints Added" reflects the number of constraints added in the OPERATIVE set, but not the artificial bounding constraint $\boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \le M$. To implement SUB and VIOL as in previous work, a single bounding constraint $\boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} \le M = 10^{10}$ was used.

Although the single-cut SUB performed comparably with the CPLEX dual simplex with the default preprocessing parameter settings in solving NNLP, SUB is much slower when solving LP. However, as shown above in **Table 3**, the CPU times for SUB greatly improves to 173.0 seconds from 4515.6 seconds on average, faster than 604.2 seconds for the CPLEX dual simplex, once the multi-cut procedure is incorporated.

For the NNLP Set 1 in Saito *et al*. [2], the 25.1

Table 5. Comparison of computation times of COST GRAD and non-COST methods, SUB and VIOL on LP problem set.

Density	CPU TIME†, sec						Number of added constraints (and number of iterations r for multi-cut methods)†						Constraints binding at Optimality as a % of constraints added†					
	SUB		VIOL		GRAD		SUB		VIOL		GRAD		SUB		VIOL		GRAD	
	SC‡	MC‖	SC§	MC‖	SC¶	MC‖	SC‡	MC‖	SC§	MC‖	SC¶	MC‖	SC‡	MC‖	SC§	MC‖	SC¶	MC‖
0.00505	246.0	14.5	120.1	15.1	149.1	9.4	8041	7363 (12.8)	2911	7002 (11.8)	6133	6129 (10.8)	10.4	11.4	28.7	11.9	13.6	13.6
0.00602	324.8	15.1	145.8	17.6	188.0	11.4	8154	7264 (12.8)	2904	6751 (12.4)	6094	5801 (11.0)	10.2	11.5	28.7	12.4	13.7	14.4
0.00701	408.7	17.9	166.7	18.5	223.4	15.3	8,205	7193 (13.6)	2834	6393 (12.8)	6014	5638 (11.6)	10.1	11.5	29.2	12.9	13.8	14.7
0.00801	458.6	19.3	179.7	20.9	256.1	14.8	8165	7068 (14.2)	2767	6144 (12.4)	5985	5514 (11.4)	10.0	11.5	29.4	13.2	13.6	14.7
0.00900	545.4	22.4	193.4	23.0	296.0	13.4	8129	7001 (15.0)	2685	6031 (13.6)	5845	5387 (12.0)	9.9	11.5	29.9	13.3	13.7	14.9
0.01000	628.5	28.6	217.4	24.3	299.6	16.4	8228	7009 (16.0)	2,705	5725 (13.4)	5726	5143 (12.0)	9.8	11.5	29.8	14.1	14.1	15.7
0.02000	1463	57.0	302.2	31.5	491.9	25.8	7974	6597 (21.8)	2341	4423 (15.0)	5208	4465 (15.4)	9.5	11.4	32.2	17.0	14.5	16.9
0.03001	2521	79.6	356.8	37.2	536.2	26.9	7570	6237 (26.4)	2163	3875 (17.4)	4557	3826 (16.6)	9.6	11.6	33.4	18.7	15.9	18.9
0.04000	2552	83.4	346.9	36.7	600.2	27.6	7347	6024 (30.8)	2055	3496 (18.6)	4386	3685 (20.4)	9.5	11.5	33.8	19.9	15.8	18.8
0.04999	3344	100.1	400.1	41.4	724.3	34.5	7354	6112 (36.8)	2009	3318 (20.8)	4317	3634 (22.4)	9.5	11.4	34.6	21.0	16.1	19.2
0.06000	3346	106.3	378.5	38.8	644.2	32.5	7067	5864 (40.2)	1860	3028 (21.6)	3949	3326 (24.0)	9.3	11.3	35.5	21.8	16.7	19.8
0.07000	4480	109.4	414.9	41.0	654.1	34.5	6923	5780 (44.4)	1843	2905 (23.2)	3927	3323 (26.0)	9.6	11.5	36.1	22.9	16.9	20.0
0.08001	4357	116.6	426.4	40.1	708.3	32.5	6851	5737 (48.6)	1803	2724 (24.0)	3600	3035 (26.4)	9.5	11.3	35.9	23.8	18.0	21.3
0.08999	4448	122.8	428.1	42.3	595.8	34.9	6727	5658 (52.8)	1743	2636 (25.6)	3555	3029 (28.8)	9.4	11.1	36.1	23.9	17.7	20.8
0.10000	4379	133.5	449.2	45.1	643.3	36.2	6730	5681 (58.2)	1734	2563 (26.8)	3500	2992 (30.8)	9.4	11.2	36.5	24.7	18.1	21.2
0.20001	7666	196.1	650.5	69.4	758.9	50.4	6226	5415 (97.2)	1561	2089 (37.8)	3145	2742 (50.0)	9.3	10.7	37.1	27.7	18.4	21.1
0.30001	7499	246.0	758.1	93.8	806.6	55.8	5814	5137 (131.8)	1393	1775 (45.8)	2787	2467 (63.6)	9.4	10.6	39.2	30.8	19.6	22.2
0.40000	10,947	344.5	947.9	134.7	1020	70.0	5814	5248 (176.6)	1359	1684 (55.6)	2641	2381 (79.6)	9.1	10.1	39.0	31.5	20.1	22.3
0.50001	11,929	431.0	1104	178.6	1050	79.4	5764	5225 (217.2)	1318	1573 (64.0)	2482	2264 (93.6)	8.9	9.8	39.0	32.7	20.7	22.7
0.75000	12,697	614.8	1420	313.4	1100	107.2	5386	4988 (315.2)	1192	1387 (83.6)	2201	2030 (127.4)	8.8	9.5	39.6	34.0	21.4	23.3
1.00000	10,585	774.6	1,621	478.0	1183	138.5	5055	4725 (415.6)	1070	1245 (102.0)	2040	1913 (168.4)	8.6	9.2	40.4	34.8	21.2	22.6
Average	4516	173.0	525.1	82.9	615.7	41.3	7025	6063 (85.6)	2012	3656 (31.3)	4195	3749 (41.1)	9.6	11.1	33.4	18.4	16.0	17.9

†Average of 5 instances of LP at each density; ‡One constraint was added per iteration r [7]. $\mathbf{c}^{\mathrm{T}}\boldsymbol{x} \le M = 10^{10}$ was used as the bounding constraint; ‖Multi-cut technique for LP was applied with $\mathbf{c}^{\mathrm{T}}\boldsymbol{x} \le M = 10^{10}$ as the bounding constraint; §One constraint was added per iteration r [8]. $\mathbf{c}^{\mathrm{T}}\boldsymbol{x} \le M = 10^{10}$ was used as the bounding constraint; ¶One constraint was added per iteration r. $\mathbf{c}^{\mathrm{T}}\boldsymbol{x} \le M = 10^{10}$ was used as the bounding constraint.

seconds of the single-cut version of NRAD was faster than 118.5 second for VIOL on average, whereas for the general LP set, the 615.7 seconds of the single-cut version of GRAD was slower than 525.1 seconds for VIOL on average. However the COST GRAD, which incorporates multi-cut, outperformed VIOL with multi-cut. The respective times were compared at 41.3 seconds vs 82.9 seconds on average.

In general, a method that makes use of *posterior* information such as VIOL adds fewer constraints and thus adds a higher percentage of binding constraints at optimality. But this comes at a cost of extra computation time required to rank the set of inoperative constraints at every iteration r. The data in **Table 5** confirmed that single-cut VIOL added the fewest number of constraints (2012 on average). The advantage of not re-sorting the constraints at every r for a *prior* method, *i.e*. GRAD, became apparent when multi-cut is applied. In multi-cut VIOL, violating inoperative constraints had to be re-sorted in descending order of violation at every iteration r.

Comparing the CPU times with and without the multiple cuts, the reduction in CPU times was greater for GRAD than in NRAD. For NRAD, the reduction was about six-fold (from 25.1 to 3.9 seconds) on average. The CPU times for GRAD reduced about 14-fold (from 615.7 seconds to 41.3 seconds).

4. Conclusions

A COST GRAD with multi-cut for general LP was developed here. An interpretation of GRAD was given, and the new technique was tested on a set of large-scale randomly generated LP with $m \gg n$. For densities between 0.02 and 1, GRAD outperformed the CPLEX primal simplex, dual simplex, and barrier solvers for LP (maximization) with long-and-narrow $\boldsymbol{A}$ matrices. Moreover, GRAD for LP still maintains the attractive features of the dual simplex method such as a basis, shadow prices, reduced costs, and sensitivity analysis [5]. As with the case for NRAD, GRAD also retains the post-optimality advantages of pivoting algorithms useful for integer programming. As a practical matter, the CPLEX presolve routines, which as noted in Saito *et al*. [2] account for most of the speed of the CPLEX solvers, are proprietary. However, this paper places GRAD in the public domain. Further research may improve GRAD towards the efficiency that NRAD demonstrated with NNLP. Incorporating new techniques may also be of interest. In particular, incorporating a method to better approximate the feasible region for general LP, as well as simultaneously addressing both the primal and dual problems, could conceivably improve COST GRAD by adding both constraints and variables.

Another area of exploration is the utilization of local posterior information [1] obtained from each $\boldsymbol{x}_r^*$ in addition to the global GRAD information for constraints obtained prior to the active-set iterations. It is conceivable that the rationale behind NRAD and GRAD could also lead to better integer programming cutting planes. Finally, it should be noted that any COST such as GRAD is a polynomial algorithm if the CPLEX barrier solver is used to solve each new subproblem with added constraints instead of the primal simplex or the dual simplex. Such a COST, however, performs extremely poorly in practice.

5. Acknowledgements

We gratefully acknowledge the Texas Advanced Research Program for supporting this material under Grant No. 003656-0197-2003.

REFERENCES

[1] H. W. Corley and J. M. Rosenberger, "System, Method and Apparatus for Allocating Resources by Constraint Selection," US Patent No. 8082549, 2011. http://patft1.uspto.gov/netacgi/nph-Parser?patentnumber=8082549

[2] G. Saito, H. W. Corley, J. M. Rosenberger and T.-K. Sung, "Constraint Optimal Selection Techniques (COSTs) for Nonnegative Linear Programming Problems," Technical Report, The University of Texas, Arlington, 2012. http://www.uta.edu/cosmos/TechReports/COSMOS-04-02.pdf

[3] M. J. Todd, "The Many Facets of Linear Programming," *Mathematical Programming*, Vol. 91, No. 3, 2002, pp. 417-436.

[4] J. M. Rosenberger, E. L. Johnson and G. L. Nemhauser, "Rerouting Aircraft for Aircraft Recovery," *Transportation Science*, Vol. 37, No. 4, 2003, pp. 408-421.

[5] J. J. Stone, "The Cross-Section Method: An Algorithm for Linear Programming," Rand Corporation Memorandum P-1490, 1958.

[6] G. L. Thompson, F. M. Tonge and S. Zionts, "Techniques for Removing Nonbinding Constraints and Extraneous Variables from Linear Programming Problems," *Management Science*, Vol. 12, No. 7, 1966, pp. 588-608.

[7] I. Adler, R. Karp and R. Shamir, "A Family of Simplex Variants Solving an $m \times d$ Linear Program in Expected Number of Pivots Steps Depending on d Only," *Mathematics of Operations Research*, Vol. 11, No. 4, 1986, pp. 570-590.

[8] M. Zeleny, "An External Reconstruction Approach (ERA) to Linear Programming," *Computers & Operations Research*, Vol. 13, No. 1, 1986, pp. 95-100.

[9] D. C. Myers and W. Shih, "A Constraint Selection Technique for a Class of Linear Programs," *Operations Research Letters*, Vol. 7, No. 4, 1988, pp. 191-195.

[10] N. D. Curet, "A Primal-Dual Simplex Method for Linear

Programs," *Operations Research Letters*, Vol. 13, No. 4, 1993, pp. 223-237.

[11] M. S. Bazaraa, J. J. Jarvis and H. D. Sherali, "Linear Programming and Network Flows," 3rd Edition, John Wiley, New York, 2005.

[12] A. W. Naylor and G. R. Sell, "Linear Operator Theory in Engineering and Science," Springer-Verlag, New York, 1982. doi:10.1007/978-1-4612-5773-8

[13] P.-Q. Pan, "Practical Finite Pivoting Rules for the Simplex Method," *Operations-Research-Spektrum*, Vol. 12, No. 4, 1990, pp. 219-225.

[14] P.-Q. Pan, "A Simplex-Like Method with Bisection for Linear Programming," *Optimization*, Vol. 22, No. 5, 1991, pp. 717-743.

[15] F. Trigos, J. Frausto-Solis and R. R. Rivera-Lopez, "A Simplex-Cosine Method for Solving Hard Linear Problems," *Advances in Simulation, System Theory and Systems Engineering*, Vol. 70X, 2002, pp. 27-32.

[16] H. Vieira Jr. and M. P. E. Lins, "An Improved Initial Basis for the Simplex Algorithm," *Computers & Operations Research*, Vol. 32, No. 8, 2005, pp. 1983-1993.

[17] H. W. Corley, J. M. Rosenberger, W.-C. Yeh and T.-K. Sung, "The Cosine Simplex Algorithm," *The International Journal of Advanced Manufacturing Technology*, Vol. 27, No. 9, 2006, pp. 1047-1050.

[18] IMSE Library, The University of Texas, Arlington. http://imselib.uta.edu

