A Build-Up Interior-Point Method for Linear Programming: Affine Scaling Form

George B. Dantzig † and *Yinyu Ye* ‡

February 1990

Revised July 1991

Abstract

We propose a *build-up* interior-point method for solving an m equation n variable linear program which has the same convergence properties as its well known analogue Dikin's algorithm in dual affine form but may require less computational effort. It differs from Dikin's algorithm in that the ellipsoid chosen to generate the improving direction in dual space is constructed from only a subset of the dual constraints. We also present some preliminary computational results for this method.

Key words: linear programming, interior method, affine scaling scaling, active column.

Abbreviated title: A Build-Up Interior-Point Method for LP

---

† Department of Operations Research, Stanford University, Stanford, CA 94305

‡ Department of Management Sciences, The University of Iowa, Iowa City, IA 52242. Research supported in part by NSF Grant DDM-8922636 and an Interdisciplinary Research Grant from The University of Iowa Center for Advanced Studies.

1. Steps of Dikin's Algorithm

We are concerned with the linear program whose primal form is

$$(\mathcal{P}) \qquad \begin{aligned} &\text{minimize} && \overline{z} = cx \\ &\text{subject to} && x \in \{x \in R^n : Ax = b,\ x \geq 0\}, \end{aligned} \tag{1.1}$$

where $c \in R^n$, $A \in R^{m \times n}$, and $0 \neq b \in R^m$ are given. Its dual form is

$$(\mathcal{D}) \qquad \begin{aligned} &\text{maximize} && \underline{z} = yb \\ &\text{subject to} && y \in \{y \in R^m : c - yA \geq 0\}. \end{aligned} \tag{1.2}$$

We denote the dual slack variables by

$$u = c - yA \geq 0. \tag{1.3}$$

When feasible solutions exist for both $(\mathcal{P})$ and $(\mathcal{D})$, then feasible solutions x and (y, u) are optimal for $(\mathcal{P})$ and $(\mathcal{D})$ if and only if the vector

$$\text{Diag}(u)x = 0, \tag{1.4}$$

see e.g. Dantzig [4]. $\text{Diag}(u)$ denotes the diagonal matrix with diagonal u.

In order to show convergence for our build-up method, we need to refer to specific parts of the proof for Dikin's algorithm [5], and the best way we found to do so is to first give an *outline* of the latter in the next two sections. (A detailed proof in the context of this paper can be found in [4]). Proofs of Dikin's affine scaling algorithm [6] were further developed by Barnes [2], Tsuchiya [12], Vanderbei and Lagarias [13], and Vanderbei et al. [14], among others. (Some computational results of Dikin's dual affine scaling algorithm are given by Adler et al. [1] and Monma and Morton [9].) We will present a set of assumptions and inductive conditions upon which a proof of convergence can be based and show how these extend to our variant; hence details of their results with modifications provide a proof that our build-up algorithm also converges to optimal solutions to $(\mathcal{P})$ and $(\mathcal{D})$. We then show that our starting rules for initiating a minor cycle within an iteration imply convergence in a finite number of iterations.

Each iteration t of the Dikin algorithm starts with an interior dual solution y^t. Next an ellipsoid subproblem $(\mathcal{E})$ centered at y^t is solved:

$$(\mathcal{E}) \qquad \begin{array}{ll} \text{maximize} & \underline{z} = yb \\ \text{subject to} & y \in \mathcal{E} = \{y \in R^m : \|(y - y^t)AD^{-1}\| \leq 1\}, \end{array} \tag{2.0}$$

where the Euclidean norm of a vector v is denoted by $\|v\|$ and $D = \mathrm{Diag}(u^t)$. The optimal solution to $(\mathcal{E})$ becomes y^{t+1} of the next iteration. We will review later the proof that y^{t+1} lies in the interior of $(\mathcal{D})$. It is computed by (2.2) below. At the same time, a solution x^t is computed by (2.3) which satisfies $Ax^t = b$; x^t may or may not be a feasible solution to $(\mathcal{P})$. As $t \to \infty$, we will show that x^t tends towards being feasible and also becomes more and more complementary to:

$$u^t = c - y^t A > 0. \tag{2.1}$$

The algorithm starts with an interior solution y^1 given. The iterates are then computed by

$$y^{t+1} = y^t + (\Delta b)^{-1/2}\Delta, \tag{2.2}$$

$$x^t = D^{-2}A^T\Delta^T \tag{2.3}$$

where

$$\Delta = b^T(AD^{-2}A^T)^{-1} \quad \text{and} \quad D = \mathrm{Diag}(u^t). \tag{2.4}$$

The superscript ^{T} denotes transpose.

In place of (2.2), an adjusted y^{t+1} is often used that makes a bigger step:

$$\text{adj.} \quad y^{t+1} = y^t + \alpha(\Delta b)^{-1/2}\Delta, \quad \alpha \geq 1,$$

where α is chosen so that adj. y^{t+1} is .9 of the way in the direction Δ from unadjusted y^{t+1} given by (2.2) to the boundary of $(\mathcal{D})$. The proof of convergence is almost exactly the same. For our development, we assume $\alpha = 1$.

2. Motivation for the Build-Up Algorithm

The improved point $y^{t+1} = y^t + \bar{\Delta} = y^t + (\Delta b)^{-1/2}\Delta$ results from solving the ellipsoid constrained subproblem (2.0). The computation of $b^T(AD^{-2}A^T)^{-1}$ involves, however, all the columns of A even though, in practical problems, most columns have little effect on the shape and size of the ellipsoid and hence on the location of the optimal point $y = y^{t+1}$. Still other columns may affect its location but may do so adversely. This suggests that a good part of the computational work could be bypassed if one knew (or had a good guess about) which columns to drop temporarily on a given iteration as non-promising or non-active. The ellipsoid based on fewer columns contains $\mathcal{E}$, and hence its use accelerates convergence. Another motivation connected with the first step of the build-up algorithm is a rule for selecting a promising set of basic columns; our experimental results support our belief that it will speed up convergence of some practical problems, especially those with a large number of columns. Several versions of the rule lead to finite convergence, see Theorem 7.

Zikan and Cottle [16] propose the *box* method to select a subset of "likely" columns to keep in. Their approach replaces the ellipsoid by a box. Given an interior point y^t in $(\mathcal{D})$, they choose m columns from A corresponding to the m closest hyperplanes in $(\mathcal{D})$ to form a "box" or parallelepiped around y^t. Then the moving direction is generated by maximizing yb subject to the box constraints.

Another approach, the *build-down* scheme, proposed by one of the authors, [15], is also designed to reduce the computational burden. The algorithm begins with a full set of columns. Then an eliminating criterion is applied which identifies columns guaranteed to never be basic in any optimal solution. Via this criterion, A is eventually built-down to the optimal basis (when it is unique) and does so in polynomial time.

The main idea in this paper, in contrast to the build-down scheme, can be described as follows. At the start of each major iteration t, we are given an interior iterate y^t. A selection of m dual constraints is made using an "order-columns" rule as to which constraints show "the most promise" of being active in the optimal dual solution. An ellipsoid centered at y^t is then inscribed in convex region defined by these promising constraints and an improving direction $\bar{\Delta}$ computed that points to the optimal point $y^t + \bar{\Delta}$ on the ellipsoid boundary. Minor cycling within a major iteration is then started.

During a minor cycle, the constraints selected to define the ellipsoid centered at y^t is built up to include (whenever there is one) the constraint which first blocks feasible movement from y^t to $y^t + \bar{\Delta}$. If one blocks, it is used to augment the set of promising constraints and the ellipsoid is revised; the improving direction $\bar{\Delta}$ is recomputed by means of a rank-one update, and the minor cycle repeated until none blocks movement from y^t to $y^t + \bar{\Delta}$. When none blocks, the minor cycling ends. $y^{t+1} = y^t + \bar{\Delta}$ initiates the next major iteration. Major iterations stop when an optimum solution is reached. We prove this will occur in a finite number of iterations.

Our goal is to compare this variant with the affine scaling method of Dikin [5] (in dual form). The well known theoretical result is that the iterates of Dikin's algorithm (y^t, x^t) converge as $t \to \infty$ to $(\bar{y}^*, \bar{x}^*)$, the optimal dual and primal feasible solutions. Proofs of convergence can be found in [2][14] and under somewhat weaker assumptions in [12][13]; proofs that the convergence ratio $\rho^t = (\bar{y}^* b - y^{t+1} b)/(\bar{y}^* b - y^t b)$ is asymptotically bounded above by $1 - 1/\sqrt{m}$ as $t \to \infty$ are given in [2][6].

3. Outline Proof of Convergence of Dikin's Algorithm

Assumptions:

(A0) $b \neq 0$, $c \neq 0$, $n > m$, and every subset of m columns from A has rank m.

(A1) An interior feasible dual solution y^1 is given.

(A2) A feasible primal solution exists.

(A3) Every feasible dual basic solution is nondegenerate.

(A4) Every primal basic solution, feasible or not, is nondegenerate.

The assumptions imply the primal feasible region is bounded, but no assumption is made about boundedness of the dual space here.

It can be shown that the detailed steps of the algorithm, (2.2) to (2.4) are legal and hence can be executed iteratively, namely:

(i) D^{-1} exists; (3.1)

(ii) $(AD^{-2}A^T)^{-1}$ exists; (3.2)

(iii) $\Delta b > 0$; (3.3)

(iv) $\bar{\Delta} = (\Delta b)^{-1/2}\Delta$ is an improving direction, i.e., $y^{t+1}b > y^t b$; (3.4)

(v) $u^{t+1} = u^t - \|Dx^t\|^{-1}(D^2x^t)^T$. (3.5)

(vi) $0 < u^{t+1} \leq 2u^t$. (3.6)

We omit the detailed proof here. The proof is based on several important relationships during the iterative process. First,

$$Ax^t = b. \tag{4.1}$$

Second,

$$0 < \|Dx^t\|^2 = \Delta b \tag{4.2}$$

and

$$y^{t+1}b - y^t b = (\Delta b)^{1/2} = \|Dx^t\| > 0. \tag{4.3}$$

Last,

$$u^{t+1} = u^t - (\Delta b)^{-1/2}\Delta A = u^t - \|Dx^t\|^{-1}(D^2x^t)^T. \tag{4.4}$$

The property in (3.6) that $u^{t+1} \le 2u^t$ is not needed for iteratively executing the algorithm but will be used in the proof of convergence which we now present in the form of several theorems and lemmas.

But first some **definitions**: A *primal solution* is any x satisfying $Ax = b$. A *dual solution* is any (y, u) satisfying $yA + u = c$. A *primal* solution is feasible if $x \ge 0$; a *dual* solution is feasible if $yA \le c$ or $u \ge 0$. Neither primal nor dual solutions need be feasible; however, all dual solutions we will be considering will be feasible ones. A partition of indices $j = 1, 2, ..., n$ into two sets consisting of m indices and the remaining $n-m$ indices will be denoted by (B, N). The set A_B of columns corresponding to B is non-singular by assumption (A0) and hence forms a basis in the space of the columns of A.

The *basic* primal and dual solutions associated with some partition (B, N) will be denoted by $\bar{x}$ and $(\bar{y}, \bar{u})$. By definition, the primal solution $\bar{x} = (\bar{x}_B, \bar{x}_N)$ associated with (B, N) is *basic*, if $\bar{x}_N = 0$. The nondegeneracy assumption (A4) also states that

$$|\bar{x}_j| > 0 \quad \text{for} \quad j \in B. \tag{5.1}$$

By definition, the dual basic solution complementary to $\bar{x}$ is $(\bar{y}, \bar{u})$ if $\bar{y}A_B = c_B$ or $u_B = 0$. The non-degeneracy assumption (A3) asserts that the dual basic solution satisfies $|\bar{u}_j| > 0$ for $j \in N$; in particular, if the dual basic solution is *feasible*, then

$$\bar{u}_N > 0. \tag{5.2}$$

Primal-dual solutions are called *complementary* if $x_j u_j = 0$ for $j = 1, 2, ..., n$.

Theorem 1. The primal-dual iterates $\{u^t, x^t\}$ tend towards complementarity, i.e., for $j = 1, 2, ..., n$

$$u_j^t x_j^t \to 0 \quad \text{as} \quad t \to \infty. \tag{6}$$

We omit the proof, which is based on relation (4.3).

Theorem 2. Given S_0, any infinite subsequence of $T = \{1, 2, ..., \infty\}$, there exists a sub-subsequence $S_* \subset S_0$ and a partition (B, N) such that $x^t = (x^t_B, x^t_N)$ tends to the primal basic solution $(\bar{x}_B, \bar{x}_N = 0)$ and (y^t, u^t_B, u^t_N) tends to the complementary dual basic feasible solution $(\bar{y}, \bar{u}_B = 0, \bar{u}_N > 0)$ using t in S_*, i.e., as $t \to \infty$ when t is restricted to successive t in the subsequence S_*.

Since our convergence result given in Theorems 2 does not require the dual feasible region to be bounded, we give below a detailed proof which is very different from the one in [13]. It is convenient to break up the proof into three lemmas.

Lemma 2.1. Given $S_0 \subset T = \{1, 2, ..., \infty\}$, there exists a sub-subsequence $S_* \subset S_0$ and a partition (B, N) such that either, for $j \in N$, $u^t_j \geq \epsilon_* > 0$ for all t in S_*, or, for $j \in B$, $u^t_j \to 0$ using t in S_*.

Proof of Lemma 2.1. Define inductively for $j = 1, 2, ..., n$ the nested set of infinite subsequences $S_0 \supset S_1 \supset S_2 ... \supset S_n$ as follows:

> initialize $j := 1$; S_0 given; $\epsilon_* := \infty$;
>
> **while** $j \leq n$ **do begin**
>
> $\epsilon_j := \inf(u^t_j)$ using t in S_{j-1}; (note $\epsilon_j \geq 0$)
>
> if $\epsilon_j > 0$ then $S_j := S_{j-1}$;
>
> if $\epsilon_j = 0$ then $S_j :=$ any infinite subsequence of S_{j-1} such that
>
> $u^t_j \to 0$ using t in this subsequence;
>
> $j := j + 1$;
>
> **end while**;
>
> $S_* := S_n$; $\epsilon_* := \min(\epsilon_j$ such that $\epsilon_j > 0)$.

Let $\bar{m}$ be the number of indices j such that $u_j^t \to 0$ using t in S_*. For the remaining $n - \bar{m}$ indices j, $u_j^t \geq \epsilon_* > 0$ for all t in S_*. We have two possibilities:

Case 1: Either $\bar{m} \leq m$, in which case there exists a subset N of $n - m$ indices j such that for j in N, $u_j^t \geq \epsilon_*$ for all t in S_*; the partition (B, N) is then defined by letting B denote the remaining indices j.

Case 2: Or the alternative $\bar{m} > m$, in which case there exists a subset B of m indices such that for $j \in B$, $u_j^t \to 0$ using t in S_*. The partition (B, N) is then defined by letting N denote the remaining indices j. Q.E.D.

This completes the proof of Lemma 2.1. The two lemmas that follow establish that $\bar{m} = m$ and iterates (y^t, x^t) converge using t in S_*. The complementary basic solutions associated with (B, N) are denoted by $(\bar{x}, \bar{y})$.

Lemma 2.2. If Lemma 2.1 resulted in Case 1, then Theorem 2 is true.

Proof of Lemma 2.2. Under Case 1 there is a partition (B, N) such that for $j \in N$ that $u_j^t \geq \epsilon_* > 0$ for all t in S_*. Since $(x_j^t u_j^t) \to 0$ as $t \to \infty$ by Theorem 1 and $u_j^t \geq \epsilon_* > 0$ for $j \in N$ and all t in S_*, it follows that $x_N^t \to 0 = \bar{x}_N$ using t in S_*, and it also follows that $x_B^t = A_B^{-1}(b - A_N x_N^t) \to A_B^{-1} b = \bar{x}_B$ using t in S_*. By the nondegeneracy assumption (A4), $|\bar{x}_j| > 0$ for $j \in B$. Again noting $(x_j^t u_j^t) \to 0$ as $t \to \infty$, it follows using t in S_* that $u_B^t \to 0 = \bar{u}_B$. Finally, we have $y^t = (c_B - u_B^t) A_B^{-1} \to c_B A_B^{-1} = \bar{y}$ using t in S_* and hence $u_N^t = c_N - y^t A_N \to c_N - \bar{y} A_N = \bar{u}_N$ using t in S_*. Moreover, since $u_N^t > 0$, $u_N^t \to \bar{u}_N \geq 0$. By the nondegeneracy assumption (A3) or (5.2), $\bar{u}_N \geq 0$ implies $\bar{u}_N > 0$.

Hence using t in S_*, $x^t \to \bar{x}$, the basic primal solution with respect to (B, N), and iterates (y^t, u^t) converge for t in S_* to the basic dual feasible solution $(\bar{y}, \bar{u}_B = 0, \bar{u}_N > 0)$. Thus Theorem 2 is proved for Case 1. Q.E.D.

Lemma 2.3. If Lemma 2.1 resulted in Case 2, then Theorem 2 is true.

Proof of Lemma 2.3. Under Case 2, there is a partition (B, N) such that, for $j \in B$, $u_B^t \to 0$ using t in S_*. As in final part of proof of Lemma 2.2, $y^t = (c_B - u_B^t)A_B^{-1} \to c_B A_B^{-1} = \bar{y}$ using t in S_*. Moreover, since $u_N^t > 0$, $u_N^t \to \bar{u}_N \geq 0$. By (A3) or (5.2), $\bar{u}_N \geq 0$ implies $\bar{u}_N > 0$. Thus $u_N^t \to \bar{u}_N > 0$ using t in S_*. But $u_N^t > 0$ and $\bar{u}_N > 0$ imply for $j \in N$ there exists an $\epsilon_* > 0$ such that $u_N^t \geq \epsilon_* > 0$ for all t in S_*. But these were the conditions we assumed in the previous lemma, hence its conclusions apply, proving Theorem 2 for Case 2 as well. Q.E.D.

Theorem 3. As $t \to \infty$, (x^t, y^t) converge to primal basic and complementary dual basic feasible solutions $(\bar{x}^*, \bar{y}^*)$.

The proof of Theorem 3 is very similar to the one given in [14]. Let $S_0 = T = \{1, 2, ..., \infty\}$ in Theorem 2, and let $(\bar{x}^*, \bar{y}^*)$ be the basic primal solution and complementary basic dual feasible solution that (x^t, y^t) converge to using t in S_*. Then, either $y^t \to \bar{y}^*$ as $t \to \infty$ establishing convergence, or there exists an infinite subsequence $T_1 = \{t_1, t_2, ...\} \subset S_*$ and a successor subsequence $T_2 = \{t_1 + 1, t_2 + 1, ...\}$ such that $y^t \to \bar{y}^*$ for $t = t_k \in T_1$, and $y^t \to \bar{y}^p$ for $t = t_k + 1 \in T_2$ where $(\bar{x}^p, \bar{y}^p)$ are *different* basic primal and complementary basic dual feasible solutions from $(\bar{x}^*, \bar{y}^*)$. However, the later possibility can't happen since by (3.6), $u_r^{t+1} \leq 2u_r^t$ for $r \in B$ and $t = t_k \in T_1$.

Theorem 4. Dikin's algorithm converges to optimal basic primal and dual solutions.

Again, we omit the proof here. It is based on the relation

$$u_j^{t+1} = u_j^t - \frac{(u_j^t)^2 x_j^t}{\|Dx^t\|} = u_j^t(1 - \frac{u_j^t x_j^t}{\|Dx^t\|}). \tag{7}$$

Theorem 5. The ratio of convergence

$$\rho^t = \frac{\bar{y}^* b - y^{t+1} b}{\bar{y}^* b - y^t b} \leq 1 - (\frac{1}{m} + \epsilon^t)^{1/2} \tag{8}$$

where $\epsilon^t \to 0$ as $t \to \infty$, i.e., ρ^t is asymptotically $\leq 1 - (1/m)^{1/2}$.

The proof can be found in [2][4][6].

4. The Build-Up Affine Scaling Algorithm

At the start of each major iteration t, we have given an interior $y = y^t$ with $u = c - y^t A > 0$. We order the columns by some criterion such as (9.0) below which measures the "promise" of a column $A_{.j}$ being in the optimal basis. The subroutine is as follows:

choose-m-promising-columns:

Reorder indices $(1, 2, ..., n)$ to $(j_1, j_2, ..., j_n)$ such that

$$u^t_{j_1} \leq u^t_{j_2} \leq .... \leq u^t_{j_n}, \tag{9.0}$$

$$\beta^t := (j_1, j_2, ..., j_m) \tag{9.1}$$

Later we will show that this rule and some other alternate rules for choosing the starting $\beta = \beta^t$ all lead to finite convergence, see Theorem 7. In practical applications, the assumption that every set of m columns is independent is usually not correct and (9.1) is modified to read: choose β as the indices of the first m most promising *independent* columns. β stands for "build up" indices; during a major iteration the initial set of indices (9.1) is built up by additional indices. When β is used as a subscript, it means that the indices j are restricted to β.

The second subroutine is

solve-starting-basis-(β):

Here β is the set of indices of the m most promising columns. A_β at the start of a major iteration is nonsingular and therefore may serve as a basis. The detailed steps are:

$$x_\beta = (A_\beta)^{-1}b; \quad y = c_\beta(A_\beta)^{-1}; \tag{10.0}$$

$$r = \arg\min_{j_i}(x_\beta). \tag{10.1}$$

The third subroutine is

solve-ellipsoid-(y^t, β):

Here β refers to the initial set of promising indices plus those built-up so far during the minor cycles within a major iteration. The ellipsoid subproblem to solve is

$$(\mathcal{E}_\beta) \qquad \text{maximize} \quad yb \quad \text{subject to} \quad y \in \mathcal{E}_\beta = \{y : \|(y - y^t)A_\beta D_\beta^{-1}\| \leq 1\}.$$

The subroutine steps to solve $(\mathcal{E}_\beta)$ are:

$$D_\beta = \mathrm{Diag}(u_\beta^t); \tag{11.1}$$

$$\Delta = b^T(A_\beta D_\beta^{-2} A_\beta^T)^{-1}; \tag{11.2}$$

$$\bar{\Delta} = (\Delta b)^{-1/2}\Delta. \tag{11.3}$$

The fourth subroutine is

find-blocking-constraint-(s):

At the start of a major iteration, *ratio* is set equal to $+\infty$ and the subroutine scans from y^t in the direction $\bar{\Delta}$ looking for the first blocking constraint $j = s$; at the start of each minor cycle, *ratio* is set equal to 1 and the subroutine then scans the line segment y^t to $y^t + \bar{\Delta}$ looking for the first blocking constraint $j = s$. If no blocking constraint is encountered, $s = 0$. The detailed searching steps are:

$s := 0$;

$v := \bar{\Delta}A$;

for $j := 1$ **to** n **do begin**

  if $((j \notin \beta)$ and $(v_j > 0)$ and $(u_j^t/v_j \leq ratio))$ **then begin**

    $ratio := u_j^t/v_j; \quad s := j;$

  end;

end for.

The Main Program of the build-up affine scaling method is as follows:

Program;

Input: dual feasible solution y^1 and $u^1 = (c - y^1 A)^T > 0$;

Initialize $t := 1$ and $opt := false$;

While $((opt = false)$ **do begin**

choose-m-promising-columns;

solve-starting-basis-(β) for β and r;

if $((x_{j_r} \geq 0)$ and $(c - \pi A \geq 0))$ **then begin**

output ("optimum", x_β, $x_N = 0$, π); $opt := true$;

end;

if $(opt = false)$ **then begin**

$\bar{\Delta} := (A_\beta)^{-1}_{r.}$;

$ratio := +\infty$;

find-blocking-constraint-(s);

if $s = 0$ **then begin**

output ("error, primal infeasible"); $opt := true$;

end;

while $s > 0$ **do begin**

$\beta := \beta \cup \{s\}$;

solve-ellipsoid-(y^t, β) for $\bar{\Delta}$;

$ratio := 1$;

find-blocking-constraint-(s);

end (end of a minor cycle);

$x_j^t := (\Delta A_{.j})/(u_j^t)^2$ for $j \in \beta$;

$x_j^t := 0$ for $j \notin \beta$;

$y^t := y^t + \bar{\Delta}$; $u^t := c - y^t A$;

$t := t + 1$;

end if;

end while (end of a major iteration);

end program.

5. Proof of Finite Convergence of Build-Up Algorithm

Theorem 6. Independent of the rule used to *choose-m-promising-columns*, the iterative process either terminates in a finite number of iterations with the optimal primal and dual solutions or converges to them in the limit.

Proof of Theorem 6. Let us assume it does not converge in a finite number of iterations. Each major iteration t, after it checks and finds that the starting basis is not optimal, engages in a number of minor cycles building up β^t (the indices of the constraints used to define the ellipsoid $\mathcal{E}_\beta$) until there is no constraint blocking movement from y^t to $y^t + \bar{\Delta}$, the optimal point of $(\mathcal{E}_\beta)$: i.e., $y^t + \bar{\Delta}$ is an interior point of $(\mathcal{D})$. The count of minor cycles within a major iteration cannot be more than $n - m$ because this is the number of remaining constraints $yA_{.j} \le c_j$ whose index j might be used to augment starting β. Therefore minor cycling always terminates with a move from y^t to y^{t+1} where the formulae for this move are the analogs of (2.1) to (2.4) of Dikin's algorithm.

$$\beta = \beta^t, \quad D_\beta = \mathrm{Diag}(u^t_\beta), \tag{12.0}$$

$$\Delta = b^T (A_\beta D_\beta^{-2} A_\beta^T)^{-1}, \tag{12.1}$$

$$\bar{\Delta} = (\Delta b)^{-1/2} \Delta, \tag{12.2}$$

$$y^{t+1} = y^t + \bar{\Delta}, \tag{12.3}$$

$$x^t_\beta = D_\beta^{-2} A_\beta^T \Delta^T; \quad x^t_j = 0 \quad \text{for} \quad j \notin \beta. \tag{12.4}$$

In addition, we have the analogs to (4.1) and (4.2)

$$b = A_\beta x^t_\beta = A x^t, \tag{13.0}$$

$$0 < \|D_\beta x^t_\beta\|^2 = \Delta b, \quad \|D_\beta x^t_\beta\| = \|D x^t\|, \tag{13.1}$$

because $x_j^t = 0$ for $j \notin \beta$. The proofs that D_β^{-1} and the $(A_\beta D^{-2} A_\beta^T)^{-1}$ exist are the same as those without the subscript β, see (3.1) and (3.2) implying that steps (12.1) and (12.2) are legal. The proofs that

$$\Delta b > 0, \tag{13.2}$$

$$\bar{\Delta} = (\Delta b)^{-1/2}\Delta \quad \text{is an improving direction, and} \tag{13.3}$$

$$y^{t+1}b - y^t b = (\Delta b)^{1/2} = \|D_\beta x_\beta^t\| = \|Dx^t\| > 0 \tag{13.4}$$

follow the proofs of (3.3) and (3.4). The analogs to (3.5) and (3.6) are

$$u_\beta^{t+1} = u_\beta^t - \|D_\beta x_\beta^t\|^{-1}(D_\beta^2 x_\beta^t)^T, \tag{13.5}$$

$$0 < u_\beta^{t+1} \le 2u_\beta^t, \quad 0 < u^{t+1}, \tag{13.6}$$

where $u_j^{t+1} > 0$ for $j \notin \beta$ is true because the minor cycles do not terminate until there is no blocking constraint, i.e., $u_j^{t+1} = c_j - y^{t+1}A_{.j} > 0$ for all $j \notin \beta$.

Under the assumption of no finite termination, we now show that Theorems 1 to 5 of Dikin's algorithm are true for the build-up algorithm. However the reasons *why* they are true will need more explanation in places and this we will now proceed to do.

That $u_j^t x_j^t \to 0$ as $t \to \infty$, Theorem 1, follows from $(y^{t+1}b - y^t b) \to 0$ and (13.4). The proof of Theorem 2 is the same. It states that given any infinite subsequence S_0 of $T = \{1, 2, ..., \infty\}$, there exists a subsequence $S_* \subset S_0$ such that (x^t, y^t) converge using t in S_* to $(\bar{x}, \bar{y})$, a pair of basic primal and complementary basic dual feasible solutions associated with some partition (B, N).

Next to show Theorem 3 for the build-up algorithm that (x^t, y^t) converges as $t \to \infty$ to a pair of basic primal and complementary basic dual feasible solutions, we need only to verify the last step where relation (3.6) that $u_r^{t+1} \le 2u_r^t$ for $r \in B$ and $t = t_k \in T_1$ is used to obtain a contradiction. However the analog of (3.6) for the build-up algorithm is (13.6) which states $u_r^{t+1} \leq 2u_r^t$ provided $r \in \beta = \beta^t$. We claim that except for a finite subsequence of t in T_1 that r is indeed in β^t; if so then this is clearly sufficient to complete the proof of Theorem 3. Suppose on the contrary there is an infinite subsequence R_1 in T_1 such that, for t in R_1, r not in β^t. Since $R_1 \subset T_1$ is infinite, its convergence properties are the same as T_1 and therefore $x_r^t \to \bar{x}_r^*$ and $|\bar{x}_r^*| \neq 0$ because r is basic index in (B, N). On the other hand, $x_r^t \to \bar{x}_r^* = 0$ for t in R_1 because the algorithm sets all $x_j^t = 0$ for $j \notin \beta^t$ which contradicts $|\bar{x}_r^*| \neq 0$ for $r \in B$.

Also requiring explanation is Theorem 4 which asserts that (x^t, y^t) converge to complementary optimal basic primal and dual solutions. The proof assumes, on the contrary, $\bar{x}^*$ is not feasible and that there is some basic index $j = r$ such that $x_r^t \to \bar{x}_r^* < 0$ as $t \to \infty$. To get a contradiction, relation (7) was used; its analog (13.5) is applicable provided $r \in \beta^t$. However there can be at most a finite count of t such that basic $r \notin \beta^t$ and a corresponding count of $x_r^t = 0$, because if the set S of $r \notin \beta^t$ were infinite, this would imply $x_r^t \to \bar{x}_r^* = 0$, but this limit is the same as that for all t, contradicting $|\bar{x}_r^*| \neq 0$. Therefore there exists a finite t_0 such that for all $t > t_0$ that $B \in \beta^t$ and (13.6) is true for all $r \in B$, and the rest of the proof of the Theorem 4 can now be applied without further change.

Finally the proof of Theorem 5 about the asymptotic ratio of convergence being $\rho^t \leq 1 - (1/m)^{1/2}$ may be used without change. Q.E.D.

This completes the proof of Theorem 6 which asserts that if the build-up algorithm does not detect an optimal basic feasible solution at start of some major iteration t and terminate, then it will converge in the limit to such a solution.

We now present several alternative rules that rank the promise of column j being in the optimal basis according to the ratio $(x_j^t)^\mu/(u_j^t)^\nu$ from high to low:

Rule 1 $(\mu, \nu) = (0, 1)$:

$$(1/u_{j_1}^t) \geq (1/u_{j_2}^t) \geq ... \geq (1/u_{j_n}^t).$$

Rule 2 $(\mu, \nu) = (1, 0)$:

$$(x_{j_1}^{t-1}/1) \geq (x_{j_2}^{t-1}/1) \geq ... \geq (x_{j_n}^{t-1}/1).$$

Rule 3 $(\mu, \nu) = (1, 1)$:

$$(x_{j_1}^{t-1}/u_{j_1}^t) \geq (x_{j_2}^{t-1}/u_{j_2}^t) \geq ... \geq (x_{j_n}^{t-1}/u_{j_n}^t).$$

Theorem 7. If the starting $\beta^t = (j_1, j_2, ..., j_m)$ is chosen by any of the above rules, the build-up algorithm will terminate in a finite number of iterations with the optimal primal and dual solutions.

Proof of Theorem 7. Let the optimal partition be (B, N). Assume on the contrary that the particular rule chosen does not lead to finite termination, then by Theorem 6 as $t \to \infty$, (x_B^t, x_N^t) converge to optimal $(\bar{x}_B^* > 0, \bar{x}_N^* = 0)$; and (u_B^t, u_N^t) converge to $(\bar{u}_B^* = 0, \bar{u}_N^* > 0)$. Let

$$\epsilon_0 = (1/2) \min(\bar{x}_{i \in B}^*, \bar{u}_{j \in N}^*).$$

There exists a t_0 such that for all $t > t_0$:

$$u_j^t < \epsilon_0 \quad j \in B \quad \text{and} \quad u_j^t > \epsilon_0 \quad j \in N.$$

$$x_j^t > \epsilon_0 \quad j \in B \quad \text{and} \quad x_j^t < \epsilon_0 \quad j \in N.$$

$$x_j^t/u_j^t > 1 \quad j \in B \quad \text{and} \quad x_j^t/u_j^t < 1 \quad j \in N.$$

Therefore, contrary to the assumption of convergence in an infinite number of iterations, we see that under any one of the three rules, the basis associated with the starting β^t for all $t > t_0$ is optimal and the algorithm will detect this fact by iteration $t \leq t_0$. Q.E.D.

6. Number of Minor Cycles

The count $h^t \geq m$, the number of indices in $\beta = \beta^t$ build up by the end of major iteration t, is not fixed and can vary up and down from one iteration to the next. We expect in early iterations that h^t will be somewhat large, but as t increases h^t will likely be close to m. Also, one may allow several columns to enter A_β simultaneously, instead of one at a time, and allow some columns deleted from A_β in order to keep the size of A_β under control. In the following, we show, at least heuristically, that the size of A_β might be controlled to no more than $2m$ for problems with $n > 2m$. This is based on the following theorem:

Theorem 8. Let $\Omega = \{y \in R^m : c - yA \geq 0\}$ where $A \in R^{m\times n}$ and $n = 2m+1$, and let y^0 be any interior point in Ω. Also, let A_β denote an $m \times (2m)$ submatrix of A. Then, there exists a particular A_β such that

$$\mathcal{E}_\beta = \{y : \|(y - y^0)A_\beta D_\beta^{-1}\| \leq 1\} \subset \Omega$$

where $D_\beta = \mathrm{Diag}(c_\beta - y^0 A_\beta)$.

Proof. We know that

$$\mathcal{E}_\beta \subset \{y : c_\beta - yA_\beta \geq 0\}.$$

Therefore, we only need to show that

$$\mathcal{E}_\beta \subset \{y : c_s - yA_{.s} \geq 0\} \quad \text{for} \quad s \notin \beta.$$

Let $a_j = A_{.j}$, $D = \mathrm{Diag}(c - y^0 A)$ and let σ be the vector whose j-th component is

$$\sigma_j = a_j^T (AD^{-2}A^T)^{-1} a_j / D_{jj}^2 > 0,$$

which are diagonal components of the projection matrix

$$D^{-1}A^T(AD^{-2}A^T)^{-1}AD^{-1}.$$

The trace of the projection matrix satisfies

$$\sum_{j=1}^{2m+1} \sigma_j = m.$$

Therefore, there exists at least one $j = s$ such that $\sigma_s < 1/2$. Define β to be the $2m$ indices which excludes s. We show $\mathcal{E}_\beta \subset \{y : c_s - ya_s > 0\}$, i.e., constraint s does not block $\mathcal{E}_\beta$ in any direction. As shown in Reference [15], it suffices to prove that

$$\bar{\sigma}_s = a_s(A_\beta D_\beta^{-2} A_\beta^T)^{-1} a_s / D_{ss}^2 < 1.$$

This is because of the following argument. The maximal value of the problem:

$$\text{maximize} \quad (y - y^0)a_s \quad \text{subject to} \quad y \in \mathcal{E}_\beta$$

is exactly

$$(\bar{y} - y^0)a_s = \sqrt{a_s(A_\beta D_\beta^{-2} A_\beta^T)^{-1} a_s}$$

where $\bar{y}$ denotes the maximum. Thus, if

$$c_s - y^0 a_s = D_{ss} > \sqrt{a_s(A_\beta D_\beta^{-2} A_\beta^T)^{-1} a_s} = (\bar{y} - y^0)a_s,$$

then

$$c_s > \bar{y}a_s \geq ya_s \quad \text{for all} \quad y \in \mathcal{E}_\beta.$$

Note that

$$(AD^{-2}A^T)^{-1} = (A_\beta D_\beta^{-2} A_\beta^T + a_s a_s^T / D_{ss}^2)^{-1}$$

$$= (A_\beta D_\beta^{-2} A_\beta^T)^{-1} - (A_\beta D_\beta^{-2} A_\beta^T)^{-1} a_s a_s^T (A_\beta D_\beta^{-2} A_\beta^T)^{-1} / (D_{ss}^2(1 + \bar{\sigma}_s)). \qquad (14)$$

Multiplying (14) by a_s^T and a_s from the left and right, and then dividing it by D_{ss}^2, we have

$$\sigma_s = \bar{\sigma}_s - \bar{\sigma}_s^2 / (1 + \bar{\sigma}_s)$$

which yields

$$\bar{\sigma}_s = \sigma_s/(1 - \sigma_s).$$

Thus, $\sigma_s < 1/2$ indicates that

$$\bar{\sigma}_s < 1$$

This completes the proof of Theorem 8. Q.E.D.

Theorem 8 states that the ellipsoid $\mathcal{E}_\beta$ inscribed in polytope Ω with $2m + 1$ constraints can be constructed using at most $2m$ constraints, which suggests but does not establish that h^t will be at most $2m$. This helps when faced by an LP problem with $n > 2m$ because, if the number of indices in β becomes greater than $2m$, the calculation of $A_\beta D_\beta^{-2} A_\beta^T$ will become a bigger burden tending to that of the conventional interior-point algorithms.

A rank-one updating method should be used to efficiently refactorize

$$(A_\beta D_\beta^{-2} A_\beta^T)^{-1}$$

and recalculate Δ after A_β is augmented by a blocking dual constraint s. This efficiency can be achieved using any one of several techniques (see, for example, Gill et al. [7] and Ng [10]). Also, at the beginning of each major iteration we need to invert a basis typically in factorized form. If the basis differs from the one at the beginning of the previous major iteration by only few columns, we can apply a few rank-one updates to obtain its inverse in factorized form as well.

7. Computational Experiments

Here we report two sets of computational experiments conducted by Kaliski in his Ph.D. thesis at The University of Iowa [8] and by Núñez Araya in his Advanced Engineering thesis at Stanford University [11].

Experiment 1

Kaliski modified and implemented a variant of the build-up method for solving transportation problems. There were 2 major differences between his variant and our build-up method. Kaliski observed the minor iterations were almost always completed before the inclusion of $2m$ columns. Thus, instead of iteratively adding 1 column at a time and then resolving the augmented ellipsoid, he simply sorts out $2m$ promising columns and uses only this set to generate the improving direction. By eliminating the minor cycles, the work of repeatedly calculating rank one updates was avoided. The second major difference was the addition of a centering (or barrier) term into the generation of the search direction. Implemented as a dual potential reduction algorithm, this addition helps to ensure the generation of a primal feasible solution in the presence of the highly degenerate transportation problems.

Several other modifications are of interest. Included in the $2m$ columns which are selected, a minimal spanning tree (MST) is identified. This MST basis serves several functions in the variant. For example, the MST can be quickly checked for optimality in a "Simplex"-like manner. Also the MST is used as a preconditioner for the conjugate gradient method, the technique used to solve for the search direction. It was found that this preconditioner was essential to the efficient generation of this direction. Finally, the original integral cost vector was slightly, randomly perturbed to help avoid degeneracy.

For comparison purposes a dual potential reduction algorithm was implemented. This algorithm was identical to the variant described above except for the following:

the sort was modified to recover only a MST, and all (full) columns were included in the calculation of the improving direction.

Table 1 illustrates some computational results for solving dense transportation problems, where $n = O(m^2)$, on an IBM 3090-2E. The problem has the same number of supply nodes and demand nodes. The cost coefficients, supplies, and demands are randomly generated integers. Most of these problems are degenerate. In Table 1, n is the number of variables in the corresponding LP problem. (Thus, the number of nodes in the network is $2\sqrt{n}$.) Itrs is the number of major iterations required to find the optimal basis. CPU is the total CPU time in minutes. The second from last column is the ratio of the CPU times using full vs active columns. The last column is the CPU time required by NETFLO, a simplex code developed by Helgason and Kennington (Southern Methodist University), to solve these problems.

We note for this set of transportation problems NETFLO has the best performance. This was expected and may partially be explained by the fact that NETFLO uses all integer-based arithmetic, whereas the variant uses primarily double precision. Comparing the two affine scaling methods, we see that the build-up variant is superior to the other. Figure 1 of the next page graphically indicates the ratio of CPU times between using all columns versus active columns. The efficiency of the active vs full column approach grows dramatically as the problem size increases. This is indicated by the apparent linear growth in the CPU time ratio.

Experiment 2

Núñez Araya compared performances of four algorithms for solving three sets of test problems. The algorithms include Dikin's algorithm, Dantzig-Ye's build-up method described in this paper, a simplified build-up method, and the revised simplex method. The simplified build-up method, also known as nonshrinking ellipsoid method, consists in simply choosing the most promising m independent columns as a basis from which an improving direction is generated. Movement is restricted to within the interior of the dual feasible region. As in Kaliski's variant, no "build-up" process is used within a major iterations. Núñez Araya showed that the nonshrinking ellipsoid method has the same convergence properties as the original build-up method. He used the matroid greedy algorithm to select the first m independent columns in each major iteration.

The first set of test problems, Data 1, consists of 50 problems with diverse characteristics such as unboundedness, infeasibility, redundancy, and degeneracy. Most of these problems were collected by Dantzig. These problems are in fact relatively small: 3 to 28 rows and 4 to 40 columns. The other two data sets consist of random problems, which were generated in a manner similar to that in [13]. A and c consists of $m \times n$ and n uniform $[0, 1)$ independent elements, respectively. Assign $b = Ae$, where e is the vector of all ones. For each set, 50 problems were generated. In Data 2, m ranges from 20 to 30, and n from $m + 1$ to 200. In Data 3, m ranges from 30 to 50, and n from $m + 1$ to 100. All of the four programs are run on an EVEREX 386/25, IBM AT compatible, with an 8087 math coprocessor. Table 2 contains the total running times of each of the four algorithms. The time is in seconds.

Núñez Araya also used statistical fits to derive a formula for the running time in terms of m and n. The formula for Dikin's method is $0.002m^2n$, for the original build-up method is $0.002m^{2.3}n^{0.6}$, for the nonshrinking method is $0.001m^{2.7}n^{0.5}$, and for the simplex method is $10^{-4}m^{2.7}n^{0.6}$.

Although it may be too early to tell, these two computational experiments are evidence that interior-point algorithms can be improved using a build-up approach. Note that the build-up method can also be used for optimization problems where linear constraints are generated from an oracle during the course of the algorithm. The build-up approach underlines an important principle in optimization: decomposition, which we believe is essential for practical large-scale optimization.

References

[1] I. Adler, N. Karmarkar, M. G. C. Resende and G. Veiga, "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming* **44** (1989) 297-335.

[2] E. R. Barnes, "A variation on Karmarkar's algorithm for linear programming," *Mathematical Programming* **36** (1986) 174-182.

[3] G. B. Dantzig, *Linear Programming and Extensions* (Princeton University Press, Princeton, NJ, 1963).

[4] G. B. Dantzig and Y. Ye, "A build-up interior method for linear programming: affine scaling form," Technical Report SOL 90-4, Department of Operations Research, Stanford University (Stanford, CA, 1990).

[5] I. I. Dikin, "Iterative solution of problems of linear and quadratic programming," *Soviet Math. Doklady* **8** (1967) 674-675.

[6] I. I. Dikin, "On the speed of an iterative process," *Upravlyaemye Sistemi* **12** (1974) 54-60.

[7] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright, "Maintaining LU factors of a general sparse matrix," *Linear Algebra and its Applications* **88/99** (1987) 239-270.

[8] J. A. Kaliski, Ph.D Thesis (in preparation), The University of Iowa (Iowa City, IA, 1991).

[9] C. L. Monma and A. J. Morton, "Computational experimental with a dual affine variant of Karmarkar's method for linear programming," Technical Report, Bell Communications Research (Morristown, NJ, 1987).

[10] E. Ng, "On the solution of sparse linear least-squares problems," Presentation at Stanford, Mathematical Science Section, Oak Ridge National Labs (Oak Ridge,

TN, 1988).

[11] M. A. Núñez Araya, "Experiments using variants of Dikin's ellipsoid method for solving mathematical programs," Engineer Thesis, Department of Operations Research, Stanford University (Stanford, CA, 1990).

[12] T. Tsuchiya, "Global convergence of the affine scaling methods for degenerate linear programming problems," Technical Report, The Institute of Statistical Mathematics, 4-6-7 Minami-Azabu Minato-ku (Tokyo, Japan, 1990).

[13] R. J. Vanderbei and J. C. Lagarias, "I. I. Dikin's convergence result for the affine-scaling algorithm," in J. C. Lagarias and M. J. Todd (eds.) *Contemporary Mathematics 114: Mathematical Developments Arising from Linear Programming* (1990) 109-120.

[14] R. J. Vanderbei, M. S. Meketon and B. A. Freedman, "A modification of Karmarkar's linear programming algorithm," *Algorithmica* **1** (1986) 395-407.

[15] Y. Ye, "A 'build-down' scheme for linear programming," *Mathematical Programming* **46** (1990) 61-72.

[16] K. Zikan and R. W. Cottle, "The box method for linear programming: Part I—basic theory," Technical Report SOL 87-6, Department of Operations Research, Stanford University (Stanford, CA, 1987).