SERIAL AND PARALLEL SOLUTION OF LARGE SCALE LINEAR PROGRAMS BY AUGMENTED LAGRANGIAN SUCCESSIVE OVERRELAXATION

by

R. De Leone and O. L. Mangasarian

Computer Sciences Technical Report #701

June 1987

Serial and Parallel Solution of Large Scale Linear Programs by Augmented Lagrangian Successive Overrelaxation[1)]

R. De Leone & O. L. Mangasarian

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53706

Technical Report #701

June 1987

Revised November 1987

Abstract. Serial and parallel successive overrelaxation (SOR) methods are proposed for the solution of the augmented Lagrangian formulation of the dual of a linear program. With the proposed serial version of the method we have solved linear programs with as many as 125,000 constraints and 500,000 variables in less than 72 hours on a MicroVax II. A parallel implementation of the method was carried out on a Sequent Balance 21000 multiprocessor with speedup efficiency of over 65% for problem sizes of up to 10,000 constraints, 40,000 variables and 1,400,000 nonzero matrix elements.

Key Words: Linear programming, SOR, augmented Lagrangian, parallel algorithms

Abbreviated Title: SOR Solution of Linear Programs

[1)] This material is based on research supported by National Science Foundation Grants DCR-8420963 and DCR-8521228 and Air Force Office of Scientific Research Grants AFOSR-86-0172 and AFOSR-86-0255.

1. Introduction

In [8, 9, 10, 12,] successive overrelation methods are proposed for solving the dual of the problem of finding the least 2-norm solution of a linear program. This leads to an exterior penalty formulation of the dual of the original linear program with the interesting property that the penalty parameter need not approach zero in order to obtain an exact solution of the primal linear program [1, 14]. Thus the penalty parameter need only be less than a certain threshold value in order to obtain an exact solution to the primal linear program. However, the penalty parameter must approach zero in order to obtain a solution to the dual problem. Although this approach has been used effectively in conjunction with successive overrelaxation methods both on serial [10] and parallel machines [11, 13], we propose here the use of an augmented Lagrangian on the dual problem instead of an exterior penalty function in order to alleviate the twin difficulties of determining the threshold value of the penalty parameter required for an exact primal solution, and the need for the penalty parameter to approach zero in order to obtain a dual solution. The first proposal for using an augmented Lagrangian formulation for solving linear programs was made in [22]. In [18] Polyak and Tretiyakov made the remarkable discovery that after a finite number of steps of the augmented Lagrangian algorithm, an exact solution to the primal and dual linear programs is obtained. In [3] Golshtein proposed a projected Gauss-Seidel method in conjunction with an augmented Lagrangian formulation and gave computational results for linear programs with sizes up to 352 variables and 166 constraints. No convergence proofs of the projected Gauss-Seidel method was given in [3], nor of the closely related iterative method of Syrov and Churkreidze in [18]. We propose here the use of a projected successive overrelaxation method in conjunction with an augmented Lagrangian formulation. The convergence of the projected SOR method established in [7] is general enough to cover both a serial and a parallel implementation of the method. Since SOR methods are inherently serial in nature, their parallelization is not a routine matter. In [11, 13] two related methods were proposed for the parallelization of SOR methods. The more recent method [13] utilizes an unreduced relaxation factor interval of (0, 2) which we shall employ here with an augmented Lagrangian algorithm for the dual linear program.

The paper is organized as follows. In Section 2 we give the necessary theoretical background and convergence results for the proposed augmented Lagrangian method applied to the dual linear program. In Section 3 we describe a serial SOR implementation of the method and establish its convergence. In Section 4 we describe our parallel SOR implementation and in Section 5 we present computational results for both the serial and parallel methods.

We briefly describe our notation now. For a vector x in the n-dimensional real space R^n, x_+ will denote the vector in R^n with components $(x_+)_i = \max\,\{x_i, 0\}$, $i = 1, \ldots, n$. The scalar product of two vectors x and y in R^n will be simply denoted by xy. For $1 \leq p \leq \infty$, the p-norm $\left(\sum_{i=1}^{n}\left|x_i\right|^p\right)^{1/p}$ of a vector in R^n will be denoted by $\|x\|_p$. For the 2-norm the subscript 2 will be dropped. R^n_+ will denote the nonnegative orthant or the set of points in R^n with nonnegative components, while $R^{m\times n}$ will denote the set of all $m \times n$ real matrices. For $A \in R^{m\times n}$, A^T will denote the transpose, A_i will denote the ith row, A_{ij} the element in row i and column j, and for $I \subset \{1, \ldots, m\}$, $J \subset \{1, \ldots, n\}$, A_I will denote the submatrix of A with rows A_i, $i \in I$, while A_{IJ} will denote the submatrix of A with elements A_{ij}, $i \in I$, $j \in J$. Similarly for $x \in R^n$ and $I_\ell \subset \{1, \ldots, n\}$, x_{I_ℓ} will denote x_i, $i \in I_\ell$. The set $\{I_1, I_2, \ldots, I_K\}$ is said to be a <u>consecutive</u> partition of $\{1, \ldots, n\}$ if it is a partition of $\{1, \ldots, n\}$ such that $i < j$ for $i \in I_\ell$, $j \in I_{\ell+1}$ and $\ell = 1, \ldots, k-1$. Here and throughout the symbols $:=$ and $=:$ denote definition of the term on the left and right sides of each symbol respectively.

2. Theoretical Background

We consider the linear program

$$\min \; cx \quad \text{subject to} \quad Ax \geq b, \; x \geq 0 \tag{2.1}$$

where $c \in R^n$, $b \in R^m$ and $A \in R^{m \times n}$ and its dual

$$\max_{(u,v) \geq 0} bu \quad \text{subject to} \quad v = -A^T u + c \tag{2.2}$$

For simplicity we exclude trivial constraints with $A_i = 0$. In [8, 9] the exterior penalty problem associated with the dual problem (2.2)

$$\max_{(u,v) \geq 0} \varepsilon bu - \frac{1}{2} \left\| A^T u + v - c \right\|^2 \tag{2.3}$$

was solved by an SOR procedure for a sufficiently small value of the penalty parameter ε to obtain $\big(u(\varepsilon), v(\varepsilon)\big)$. The unique least 2-norm solution $\bar{x}$ of the linear program (2.1) was obtained by using the equation

$$x(\varepsilon) = \frac{1}{\varepsilon} \left(A^T u(\varepsilon) + v(\varepsilon) - c \right) \tag{2.4}$$

which relates an optimal solution $\big(u(\varepsilon), v(\varepsilon)\big)$ of the dual penalty problem (2.3) and the unique solution $x(\varepsilon)$ of the corresponding quadratic primal problem [5]

$$\min \; cx + \frac{\varepsilon}{2} xx \quad \text{subject to} \quad Ax \geq b, \; x \geq 0 \tag{2.5}$$

In particular it follows [14] that the least 2-norm solution $\bar{x}$ of the linear program (2.1) is related to $x(\varepsilon)$ of (2.4) by

$$\bar{x} = x(\varepsilon) \quad \text{for} \quad \varepsilon \in (0, \bar{\varepsilon}] \quad \text{for some} \quad \bar{\varepsilon} > 0 \tag{2.6}$$

Thus the penalty parameter ε of the dual penalty problem (2.3) is the perturbation parameter of the perturbed primal problem (2.5). In order to avoid possible difficulties associated with determining the threshold value $\bar{\varepsilon}$ we consider instead of the exterior penalty problem (2.3) the augmented Lagrangian associated with the dual linear program (2.2)

$$L(u, v, x, \gamma) := bu - \frac{1}{2\gamma} \left\| A^T u + v - c \right\|^2 - x(A^T u + v - c) \tag{2.7}$$

It is a standard result [6, 2, 19] that for any $\gamma > 0$, a primal-dual solution $(\hat{x}, \hat{u}, \hat{v})$ of (2.1)-(2.2) is equivalent to a stationary point of the following saddlepoint problem of (2.7): Find an $(\hat{x}, \hat{u}, \hat{v}) \in R^n \times R^m_+ \times R^n_+$ such that for all $x \in R^n$ and all $(u, v) \in R^m_+ \times R^n_+$,

$$L(u, v, \hat{x}, \gamma) \leq L(\hat{u}, \hat{v}, \hat{x}, \gamma) \leq L(\hat{u}, \hat{v}, x, \gamma) \tag{2.8}$$

The standard augmented Lagrangian algorithm [2, 19] consists of a maximization step in the (u, v) space $R^m_+ \times R^n_+$ followed by an unconstrained gradient descent step in the x space R^n. In particular we have the following.

2.1 Augmented Lagrangian Algorithm

Start with any $x^0 \in R^n$. Having x^i determine x^{i+1} as follows

$$\begin{cases} \textbf{(a) } L(u^i, v^i, x^i, \gamma^i) = \max\limits_{(u,v)\geq 0} L(u, v, x^i, \gamma^i) \\ \textbf{(b) } x^{i+1} = x^i - \dfrac{1}{\gamma^i} \nabla_x L(u^i, v^i, x^i, \gamma^i) = x^i + \dfrac{1}{\gamma^i}(A^T u^i + v^i - c) \end{cases} \tag{2.9}$$

where $\{\gamma^i\}$ is any bounded sequence of positive numbers.

For this iterative linear programming algorithm Polyak and Tretiyakov [18] have given the following important finite convergence theorem.

2.2 Augmented Lagrangian Algorithm Finite Termination Theorem [18]

For any bounded positive sequence $\{\gamma^i\}$ and $x^0 \in R^n$, Algorithm 2.1 is finite, that is there exists an integer k such that (x^k, u^k, v^k) solve the dual linear programs (2.1)-(2.2). Furthermore for each x^0 there exists a $\hat{\gamma} > 0$ such that for $0 < \gamma^0 < \hat{\gamma}$, the method will terminate in one step, that is (x^1, u^1, v^1) will solve the dual linear programs (2.1)-(2.2).

Note that in the above theorem, two exact maximizations over the (u, v) space $R^m_+ \times R^n_+$ are required in order to obtain $(x^{i+1}, u^{i+1}, v^{i+1})$ from x^i.

We note that since by duality theory [5]

$$\max_{(u,v)\geq 0} L(u, v, x, \gamma) = \min_x \left\{ cz + \frac{\gamma}{2} \|z - x\|^2 \middle| Az \geq b,\ z \geq 0 \right\} =: \phi(x) \tag{2.10}$$

the Augmented Lagrangian Algorithm 2.1 is equivalent to the following gradient method applied to the proximal point function $\phi(x)$

$$\begin{cases} x^{i+1} = x^i - \dfrac{1}{\gamma^i} \nabla \phi(x^i) = \text{Prox}\,(x^i) \\ \text{where Prox}\,(x^i) \text{ is the solution of} \\ \min\limits_z \left\{ cz + \dfrac{\gamma^i}{2} \|z - x^i\|^2 \middle| Az \geq b,\ z \geq 0 \right\} \end{cases} \tag{2.11}$$

Bertsekas [1] and Rockafellar [21] also obtain finite termination for (2.11) from proximal point theory considerations for the linear programming case.

With this background we are prepared now to state and prove the convergence of our serial and parallel SOR algorithms.

3. Serial Successive Overrelaxation Algorithm

The proposed serial algorithm consists of applying the projected SOR method of [7] to the maximization step (2.9a) of the Augmented Lagrangian Algorithm 2.1 for a decreasing sequence of positive numbers $\{\gamma^i\}$. It follows from the Finite Termination Theorem 2.2 and a theorem of Pang [16, Theorem 3.1], that for any x^i the projected SOR method will generate a sequence of points converging to an x^{i+1} that solves the primal linear program (2.1), provided that γ^i is sufficiently small. There are no easily implementable and theoretically justifiable ways of determining how to choose γ^i sufficiently small [20, 4], however we shall prescribe some computationally effective ways for doing that.

The serial projected SOR method has been proposed [7, 9] for solving the quadratic minimization problem

$$\min_{z\geq 0} \theta(z) := \min_{z\geq 0} \frac{1}{2} zMz + qz \tag{3.1}$$

where $M \in R^{k\times k}$ is symmetric and positive semidefinite. This is precisely problem (2.9a) of Algorithm 2.1 if we make the identifications

$$M := \frac{1}{\gamma^i}\begin{bmatrix} A \\ I \end{bmatrix}\begin{bmatrix} A^T & I \end{bmatrix}, \quad q := \begin{bmatrix} A(x^i - \frac{c}{\gamma^i}) - b \\ x^i - \frac{c}{\gamma^i} \end{bmatrix}, \quad z := \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.2}$$

The projected SOR algorithm consists of the following.

$$\begin{gathered} z_j^{t+1} = \left(z_j^t - \omega\left(\nabla^2\theta(z^t)\right)_{jj}^{-1}\nabla_{z_j}\theta\left(z_1^{t+1},\ldots,z_{j-1}^{t+1}, z_j^t,\ldots,z_k^t\right)\right)_+ \\ \omega \in (0,\,2),\ j = 1,\ldots,k \end{gathered} \tag{3.3}$$

More specifically for $\theta(z) = \frac{1}{2}\, zMz + qz$ we have the following.

3.1 Serial SOR Algorithm for $\min\limits_{z\geq 0} \frac{1}{2}\mathbf{zMz} + \mathbf{qz}$

Choose $z^0 \in R^k_+$, $\omega \in (0,\,2)$. Having z^t compute z^{t+1} as follows

$$\begin{gathered} z_j^{t+1} = \left(z_j^t - \omega M_{jj}^{-1}\Big(\sum_{\substack{\ell=1\\ \text{for } j>1}}^{j-1} M_{j\ell}\, z_\ell^{t+1} + \sum_{\ell=j}^{k} M_{j\ell} z_\ell^t + q_j\Big)\right)_+ \\ j = 1,\ldots,k \end{gathered} \tag{3.4}$$

We are ready to state and establish the convergence of our augmented Lagrangian serial SOR algorithm.

3.2 Augmented Lagrangian Serial SOR Algorithm

Let $\{\gamma^i\} \downarrow \bar{\gamma}$ for some $\bar{\gamma} > 0$, let $\{\delta^i\} \downarrow 0$ and let $x^0 \in R^n$. Having x^i determine x^{i+1} as follows:

(a) Apply the Serial SOR Algorithm 3.1 to solve (2.9a) with the identifications (3.2), and let $(u^i(t), v^i(t))$ be the t iterate of this SOR algorithm. Stop if for some $t = t^i$ the following inequality is satisfied.

$$\begin{aligned}(3.5)\quad &\left|u^i(t)\nabla_u L(u^i(t), v^i(t), x^i, \gamma^i)\right| + \left|v^i(t)\nabla_v L(u^i(t), v^i(t), x^i, \gamma^i)\right| \\ &\quad + \left\|(\nabla_u L(u^i(t), v^i(t), x^i, \gamma^i)_+\right\| + \left\|\left(\nabla_v L(u^i(t), v^i(t), x^i, \gamma^i)\right)_+\right\| \le \delta^i\end{aligned}$$

(b) Set $x^{i+1} = x^i(t^i)$ where

$$(3.6)\qquad x^i(t) := x^i + \frac{1}{\gamma^i}(A^T u^i(t) + v^i(t) - c)$$

3.3 Augmented Lagrangian Serial SOR Convergence Theorem

Let $\{\gamma^i\} \downarrow \bar{\gamma} > 0$ be a sufficiently rapidly decreasing sequence of positive numbers and $\bar{\gamma}$ sufficiently small. Then either

(a) For some integer k, the sequence $\{x^k(t)\}$ converges to an $\bar{x}^k$ that solves the linear program (2.1), or

(b) For each subsequence of $\{(x^i, u^i(t^i), v^i(t^i))\}$ converging to some $(\bar{x}, \bar{u}, \bar{v})$, the corresponding subsequence $\{x^{i+1} = x^i(t^i)\}$ converges to an $\hat{x}$ such that $\hat{x}$ solves the linear program (2.1). If $\hat{x} = \bar{x}$, then $(\bar{u}, \bar{v})$ solves the dual linear program (2.2).

Proof Either the inequality (3.5) of the algorithm is satisfied at each iteration i for some t^i or not. Accordingly we have the alternatives (b) and (a) below respectively.

(a) For some iteration $i = k$ the inequality (3.5) is never satisfied. Hence by Pang's Theorem 3.1 [16], since $L(u, v, x^k, \gamma^k)$ is by duality theory bounded above for $(u, v) \ge 0$ by $cx + \gamma^k\|x - x^k\|^2$ for any $x \ge 0$ such that $Ax \ge b$, it follows that the sequence

$$(3.7)\qquad \{x^k(t)\} := \left\{x^k + \frac{A^T u^k(t) + v^k(t) - c}{\gamma^k}\right\},\ t = 0, 1, 2, \ldots,$$

where t is the SOR iteration index, converges to a vector $\bar{x}^k$ defined by

$$\bar{x}^k := x^k + \frac{A^T\bar{u}^k + \bar{v}^k - c}{\gamma^k} \tag{3.8}$$

for some $(\bar{u}^k, \bar{v}^k)$ which solves $\max\limits_{(u,v)\geq 0} L(u, v, x^k, \gamma^k)$. Note however that $\{u^k(t),\, v^k(t)\}$ need not converge to $(\bar{u}^k, \bar{v}^k)$. If γ^k is sufficiently small (and this is what is meant by requiring that $\{\gamma^i\}$ decreases sufficiently rapidly) it follows by Theorem 2.2 that $x^{k+1} = \bar{x}^k$, is a solution of the primal linear program (2.1). (Note that $(\bar{u}^k, \bar{v}^k)$ need not be a solution of the dual linear program (2.2). To obtain such a dual optimal $(\tilde{u}^k, \tilde{v}^k)$ we need to solve $\min\limits_{(u,v)\geq 0} L(u, v, \bar{x}^k, \gamma^k)$ exactly. See Corollary 3.5 below.)

(b) If inequality (3.5) holds for each i for some t^i, then since $\{\delta^i\} \downarrow 0$ and $\{\gamma^i\} \downarrow \bar{\gamma} > 0$, we have that for any subsequence of $\{(x^i, u^i(t^i),\, v^i(t^i))\}$ converging to some $(\bar{x}, \bar{u}, \bar{v})$, the point $(\bar{u}, \bar{v})$ solves the problem $\min\limits_{(u,v)\geq 0} L(u, v, \bar{x}, \bar{\gamma})$ (because $\{\delta^i\} \downarrow 0$), and moreover the corresponding subsequence $\{x^{i+1}\}$ defined by (3.6) and part (b) of Algorithm 3.2 converges to an $\hat{x}$ defined by

$$\hat{x} := \bar{x} + \frac{A^T\bar{u} + \bar{v} - c}{\bar{\gamma}} \tag{3.9}$$

If $\bar{\gamma}$ is sufficiently small, then it follows by Theorem 2.2 that $\hat{x}$ solves the linear program (2.1). If in addition $\hat{x} = \bar{x}$, then $(\bar{u}, \bar{v})$ is feasible for the dual linear program (2.2) and is also optimal because $b\bar{u} = c\hat{x}$. ∎

It is useful to point out that when the sequence $\{(u^i(t),\, v^i(t))\}$ of Algorithm 3.2 is bounded then it has an accumulation point and by Theorem 2.1 [7], each such accumulation point satisfies the optimality conditions for (2.9a). Consequently for each i inequality (3.5) of Algorithm 3.2 is satisfied after a finite number of steps of part (a) of the algorithm. However by Theorem 2(ii) of [9] we have that if the linear program (2.1) satisfies the Slater constraint qualification, then the sequence $\{(u^i(t),\, v^i(t))\}$ is indeed bounded. Therefore we have the following.

3.4 Augmented Lagrangian Serial SOR Convergence Corollary

Let the linear program (2.1) satisfy a Slater constraint qualification, that is $Ax > b$ for some $x \geq 0$. Then Theorem 3.3 holds with outcome (b).

Another useful observation follows from the fact [18, Lemma 1], [19] that minimizing the augmented Lagrangian $L(u, v, x, \gamma)$ with an optimal value of x and any $\gamma > 0$ gives a solution to the dual linear program (2.2). Hence we have the following.

3.5 Dual LP SOR Solution Corollary

Under the assumptions of Theorem 3.3 a solution to the dual linear program (2.2) can be obtained for either outcome (a) or (b) of Theorem 3.3 by solving respectively

(a) $\min\limits_{(u,v)\geq 0} L(u, v, \bar{x}^k, \gamma^k)$

or

(b) $\min\limits_{(u,v)\geq 0} L(u, v, \hat{x}, \bar{\gamma})$

We note immediately that we have left open the procedure by which the sequence $\{\gamma^i\}$ is decreased. This is an inherent theoretical difficulty that arises when using inexact minimization in the subproblems of proximal point or augmented Lagrangian algorithms. Thus the approximate minimization criteria of [21, Criteria A, B, A', B'] are not implementable for our problem, while the assumptions of [2, Section 2.5] are not verifiable for our problem. Computationally we have overcome this difficulty by using the following scheme, often used for updating the penalty parameter in augmented Lagrangian algorithm

$$\gamma^{i+1} = \begin{cases} \gamma^i \text{ if } \left\|x^{i+1} - x^i\right\| \leq \mu\left\|x^i - x^{i-1}\right\|, \; 0 < \mu < 1 \\ \nu\gamma^i \text{ otherwise, } \; 0 < \nu < 1 \end{cases} \tag{3.10}$$

This scheme works effectively for the solution of very large sparse linear programs as our computational results indicate.

4. Parallel Successive Overrelaxation Algorithm

The key to our parallel SOR algorithm is the use of the parallel gradient projection SOR (GP-SOR) that we proposed in [13] for the solution of (3.1) and which we outline below. Partition the matrix M of (3.1) into r contiguous horizontal blocks as follows:

$$M =: \begin{bmatrix} M_{I_1} \\ M_{I_2} \\ \vdots \\ M_{I_r} \end{bmatrix} \tag{4.1}$$

where the blocks M_{I_j} correspond to the variables z_{I_j}, and $\{I_1, I_2, \ldots, I_r\}$ is a consecutive partition of $\{1, 2, \ldots, k\}$. Now partition M_{I_j} as follows

$$M_{I_j} =: \begin{bmatrix} M_{I_jI_j} & M_{I_j\bar{I}_j} \end{bmatrix} \tag{4.2}$$

where $\bar{I}_j$ is the complement of I_j in $\{1, 2, \ldots, k\}$. Thus $M_{I_jI_j}$ is a principal square submatrix of M with elements M_{st}, $s \in I_j$ and $t \in I_j$. We further partition $M_{I_jI_j}$ into the sum of its strictly lower triangular part $L_{I_jI_j}$, its diagonal part $D_{I_jI_j}$ and its strictly upper triangle part $U_{I_jI_j}$ as follows

$$M_{I_jI_j} =: L_{I_jI_j} + D_{I_jI_j} + U_{I_jI_j} \tag{4.3}$$

Now define a <u>block diagonal</u> matrix K as follows

$$K := \begin{bmatrix} L_{I_1I_1} & & & \\ & L_{I_2I_2} & & \\ & & \ddots & \\ & & & L_{I_rI_r} \end{bmatrix} \tag{4.4}$$

where each $L_{I_jI_j}$ is the strictly lower triangular part of $M_{I_jI_j}$. An SOR algorithm can now be performed for each row block I_j, $j = 1, \ldots, r$, simultaneously, that is in parallel. Note that the block diagonal matrix K replaces the traditional strictly lower triangular matrix of the serial SOR. Specifically we have the following.

4.1 Parallel GP-SOR Algorithm for (3.1)

Let $\{I_1, I_2, \ldots, I_r\}$ be a consecutive partition of $\{1, 2, \ldots, k\}$, let E be a positive diagonal matrix in $R^{k\times k}$ and let $z^0 \geq 0$. For $i = 0, 1, 2, \ldots,$ do the following

Direction Generation Define the direction

$$d^i := p(z^i) - z^i := \begin{pmatrix} p_{I_1}(z^i) - z^i_{I_1} \\ \vdots \\ p_{I_r}(z^i) - z^i_{I_r} \end{pmatrix} \tag{4.5}$$

such that $p(z^i)$ satisfies

$$p_{I_j}(z^i) = \left(z^i_{I_j} - \omega E_{I_jI_j}\left(M_{I_j}z^i + q_{I_j} + L_{I_jI_j}(p_{I_j}(z^i) - z^i_{I_j})\right)\right)_+, \; j = 1, ..., r \tag{4.6}$$

where $\omega > 0$ is chosen such that for some $\nu > 0$

$$z_{I_j}\left((\omega E_{I_jI_j})^{-1} + L_{I_jI_j}\right)z_{I_j} \geq \nu \left\|z_{I_j}\right\|^2, \; \forall z_{I_j}, \; j = 1, ..., r \tag{4.7}$$

Stop if $d^i = 0$, else continue.

Stepsize Generation $z^{i+1} = z^i + \lambda^i d^i$

where

$$f(z^i + \lambda^i d^i) = \min_{\lambda}\{f(z^i + \lambda d^i) | z^i + \lambda d^i \geq 0\} \tag{4.8}$$

4.2 Remark The principal part of this algorithm consists of the direction generation part (4.6), which can be performed in parallel on r processors. Once this is done the stepsize generation (4.8) is performed and the new value z^{i+1} is shared between the r processors.

The following convergence results were derived in [13] for Algorithm 4.1.

4.3 Theorem (Convergence of the Parallel GP-SOR Algorithm) Let M be symmetric and positive semidefinite. Either the sequence $\{z^i\}$ generated by the Parallel GP-SOR Algorithm 4.1 terminates at a solution of (3.1) or each of its accumulation points solves (3.1).

4.4 Corollary (Parallel GP-SOR special cases) Condition 4.7 of Algorithm 4.1 holds under either of the following two assumptions:

$$0 < \omega < \min_{j=1,\ldots,r} \; \min_{i \in I_j} \frac{2}{E_{ii} \sum\limits_{\substack{\ell \in I_j \\ \ell \neq i}} |M_{i\ell}|} \tag{4.9}$$

(4.10) $$0 < \omega < 2,\ E = D^{-1} \quad \text{and } M \text{ is positive semidefinite}$$

Our parallel augmented Lagrangian method consists of replacing the Serial SOR Algorithm 3.1 by the Parallel GP-SOR Algorithm 4.1 in Algorithm 3.2 with option (4.10) for the choice of ω and E. We formally state the algorithm below.

4.5 Augmented Lagrangian Parallel SOR Algorithm

Identical to Algorithm 3.2 except that the Serial SOR Algorithm 3.1 in Algorithm 3.2 is replaced by the Parallel GP-SOR Algorithm 4.1 with condition (4.10) above in force.

5. Computational Results

Our algorithms were tested on random linear programs which were generated as follows. First the matrix A of the linear program (2.1) was generated. Each of its nonzero elements was a random number generated by a uniform distribution on the interval [-100, 100]. The number of nonzero elements in each row was in accordance to a prescribed density and the random position of each nonzero element was determined according to a uniform distribution on the column indices of the matrix. Next a primal-dual solution vector $(\bar{x}, \bar{u})$ was generated from a uniform distribution on the interval [0, 10] with 80% of the components being nonzero. Finally the vectors b and c were chosen so that $(\bar{x}, \bar{u})$ is optimal.

In Figure 1 we give a summary of our computational results for 6 problems with the number of constraints varying between 25,000 and 125,000 and the number of variables varying between 100,000 and 500,000. All tests were performed on a MicroVax II with 16 megabytes of memory and an expanded disk swap space. We are not aware of any other linear programming software that can handle problems of the size that we have attempted on a comparable machine. MINOS [15], a state-of-the-art pivotal linear programming package, cannot handle any of the problems listed in Figure 1 because they are too big for the machine memory using the MINOS configuration. The largest problem attempted on the MicroVax II with MINOS was a problem with 5000 constraints, 20,000 variables and a matrix density of 0.2% with about 200,000 nonzero elements. MINOS was used with the standard partial pricing and scaling options. After 3150 iterations and 49 hours 54 minutes of machine time, the point was infeasible and the objective function was in error by 59% of the exact minimum. By comparison our Algorithm 3.2 solved the same problem in 1 hour and 4 seconds with a primal-dual objective function accuracy of 7 figures, and relative accuracy of not less than 10^{-9} as defined in Figure 1.

The Parallel SOR Algorithm 4.5 was implemented on the Sequent Balance 21000, a multiprocessor that incorporates eight NS32032 processors running at 10MHz, each with a floating point unit, memory management and an 8-kbyte cache sharing a global memory via a 32-bit wide pipelined bus. The machine has 8-Mbytes of physical memory. The operating system DYNIX, is a version of Berkeley 4.2 bsd unix. The computational results are depicted in Figures 2 and 3.

Figure 2 shows the total computing time versus number of processors for four different densities: d=1, 2, 7 and 10 percent for a linear program with 1000 constraints and 4000 variables. All problems were solved to a 7-figure accuracy of the primal-dual objective function and relative accuracy better than 10^{-7} as defined in Figure 1. We observe that the optimal number of processors, that is the one that solves the problem in minimum total time, increases with density as expected. This number is 3 for 1% and 2% densities, 6 for 7% density, and 7 or more for 10% density. This means that for denser problems, a larger number of processors is needed in order to arrive at the shortest solution time. This also means that for denser problems, the communication cost does not become a dominant and hence prohibitive factor until a larger number of processors are used.

In Figure 3 we show results for the case with 10,000 constraints and 40,000 variables, with density of 0.35% and about 1,400,000 nonzero elements. To our knowledge this is one of the largest linear programs solved on this relatively modest sized multiprocessor. One of the reasons that we were able to solve larger problems on the MicroVax II, is that the latter had twice the total memory size of the Balance 21000 and furthermore, the MicroVax was essentially a single-user machine dedicated to the serial SOR algorithm. The optimal number of processors for the low-density case shown in Figure 3 is 4. Just as in the cases of Figure 2, the optimal number of processors should increase with problem density.

We conclude with some observations on the speedup efficiency of our Parallel SOR Algorithm 4.5. We define the speedup efficiency $E(r)$ as the ratio of the actual to the theoretical speedup of the algorithm using r processors instead of 1 processor, thus

$$E(r) := \frac{T(1)}{rT(r)} \tag{5.1}$$

where $T(r)$ is the total time for solving a given problem using r parallel processors. Figure 4 shows the speedup efficiencies for a typical case of a linear program with 1000 constraints, 4000 variables and 2% density. The reason why some efficiencies are over 100% was pointed out in [13]. The explanation is that our Parallel SOR Algorithm 4.5 changes with the number of processors used, because the matrix K defined by (4.4) changes with the number of blocks into which M is divided. Thus we are not comparing identical algorithms when we evaluate the ration $T(1)/rT(r)$ of (5.1). Nevertheless the expression is a valid measure of efficiency in the sense of comparing the theoretical reduced time $T(1)/r$ to the observed time $T(r)$ for an algorithm with a variable K that depends on the partition of M. If the matrix K is held fixed for $r = 1$ and $r > 1$, then we obtain efficiencies for identical algorithms, and they would all be less than 100%. This was demonstrated in [13]. Nevertheless the present efficiencies of over 100% are indeed very encouraging and also give the additional and somewhat surprising result that a serial implementation of our r-block Parallel SOR Algorithm 4.5 on a single machine, will give for some r a better computing time than the single block Serial SOR Algorithm 3.2. For the specific case of Figure 4, a serial implementation of the r-block parallel SOR with r=2, 3, 4 and 5 will be faster than the single block serial SOR.

Prob. No.	No. constr. $m \times 10^{-3}$	No. var. $n \times 10^{-3}$	Nonzero elements per row	Total Nonzero elements $\times 10^{-5}$	Iter	Time hr:min	Fig. Accur. Obj. Func. P	D	Log_{10} Rel. Accur. 1	2	3	4
							(a)	(b)	(c)	(d)	(e)	(f)
1	25	100	20	5	309	2:31	7	7	-7	-10	-9	-10
2	30	120	25	7.5	392	4:34	7	7	-10	-14	-12	-14
3	40	160	20	8	363	4:56	7	7	-9	-11	-7	-8
4	50	200	16	8	525	7:47	7	7	-9	-8	-9	-8
5	100	400	12	12	967	22:35	5	7	-3	-5	-5	-7
6	125	500	9	11.25	3100	71:40	6	7	-3	-5	-5	-7

(a) Number of correct figures in primal objective

(b) Number of correct figures in dual objective

(c) $\|(-Ax+b)_+\|_\infty / \|(b)_+\|_\infty$ (Relative Accuracy)

(d) $\|(A^Tu-c)_+\|_\infty / \|(-c)_+\|_\infty$ (Relative Accuracy)

(e) $|cx - c\bar{x}|/|cx + c\bar{x}|$, $\bar{x}$: exact, x: computed (Relative Accuracy)

(f) $|bu - b\bar{u}|/|bu + b\bar{u}|$, $\bar{u}$: exact, u: computed (Relative Accuracy)

Fig. 1. MICROVAX II: Serial SOR Algorithm 3.2 test results.

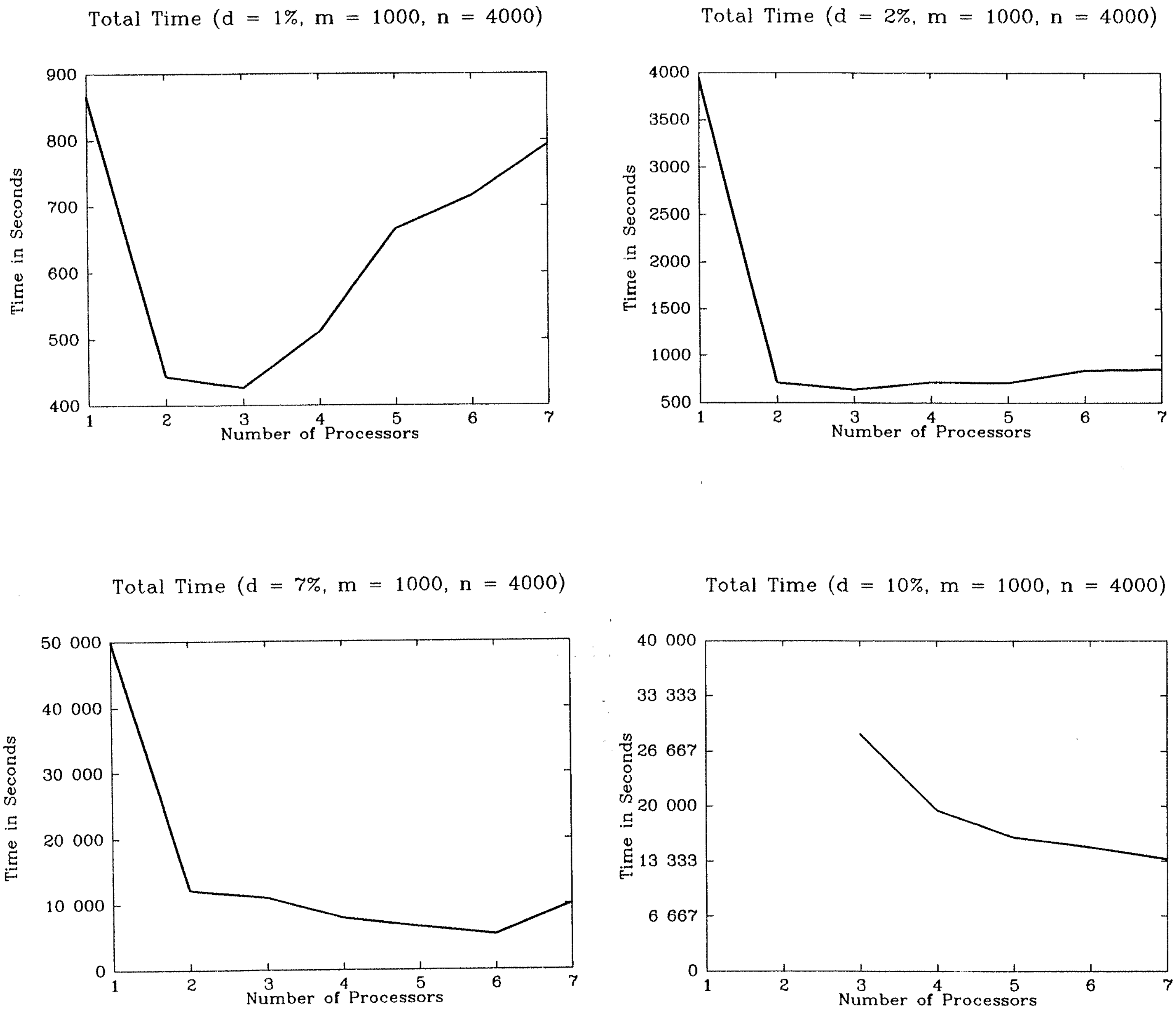

Fig. 2. BALANCE 21000: Total time for Parallel SOR Algorithm 4.5 to solve linear program versus number of processors for various densities d. (Average of 4 randomly generated cases with 1000 constraints and 4000 variables.)

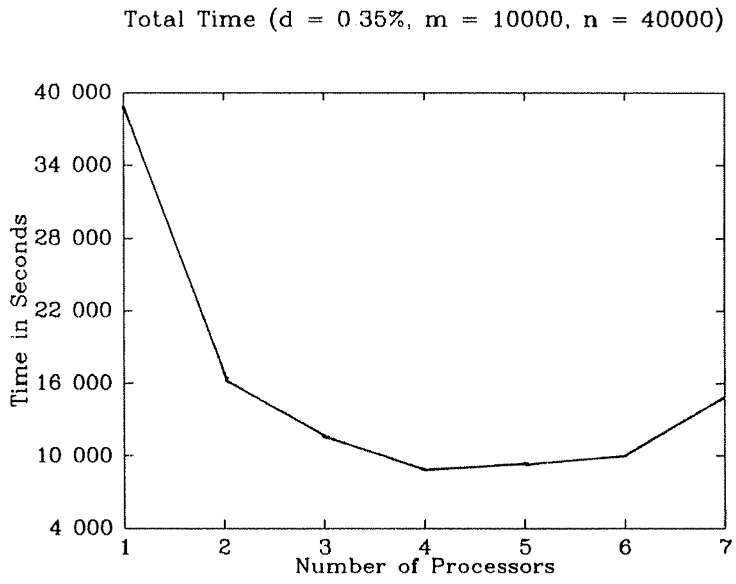

Fig. 3. BALANCE 21000: Total time for Parallel SOR Algorithm 4.5 to solve linear program versus number of processors. (d=density, m=number of constraints, n=number of variables.)

Density $d = 2\%$, $m = 1000$, $n = 4000$		
No. Processes r	Time Sec. $T(r)$	Speedup Efficiency $E(r) = T(1)/rT(r)$
1	3946	——
2	709	278%
3	638	206%
4	716	138%
5	711	111%
6	840	78%
7	850	66%

Fig. 4. BALANCE 21000: Speedup efficiency $E(r)$ for the Parallel SOR Algorithm 4.5 for an LP with 1000 constraints and 4000 variables and 2% density.

References

1. D. P. Bertsekas: "Necessary and sufficient conditions for a penalty method to be exact", Mathematical Programming 9, 1975, 87-99.

2. D. P. Bertsekas: "Constrained optimization and Lagrange multiplier methods", Academic Press, New York 1982.

3. E. G. Golshtein: "An iterative linear programming algorithm based on the use of modified Lagrange function", US-USSR Seminar on Distribution and Production Problems of Industry and Enterprise, October 15-17, 1980, University of Maryland, College Park, Proceedings Published in College Park, Maryland 1982.

4. Y. Y. Lin & J. S. Pang: "Iterative methods for large convex quadratic programs: A survey", SIAM Journal on Control and Optimization 25, 1987, 383-411.

5. O. L. Mangasarian: "Nonlinear programming", McGraw-Hill, New York 1969.

6. O. L. Mangasarian: "Unconstrained Lagrangians in nonlinear programming", SIAM Journal on Control 13, 1975, 772-791.

7. O. L. Mangasarian: "Solution of symmetric linear complementarity problems by iterative methods", Journal of Optimization Theory and Applications 22, 1977, 465-485.

8. O. L. Mangasarian: "Iterative solution of linear programs", SIAM Journal on Numerical Analysis 18, 1981, 606-614.

9. O. L. Mangasarian: "Sparsity-preserving SOR algorithms for separable quadratic and linear programs", Computers and Operations Research 11, 1984, 105-112.

10. O. L. Mangasarian: "Normal solution of linear programs", Mathematical Programming Study 22, 1984, 206-216.

11. O. L. Mangasarian & R. De Leone: "Parallel successive overrelaxation methods for symmetric linear complementarity problems and linear programs", Journal of Optimization Theory and Applications 54(3), September 1987.

12. O. L. Mangasarian & R. De Leone: "Error bounds for strongly convex programs and (super)linearly convergent iterative schemes for the least 2-norm solution of linear programs", Applied Mathematics and Optimization, 1988.

13. O. L. Mangasarian & R. De Leone: "Parallel gradient projection successive overrelaxation for symmetric linear complementarity problems and linear programs", University of Wisconsin Computer Sciences Department Tech. Rept. 659, August 1987, submitted to Annals of Operations Research.

14. O. L. Mangasarian & R. R. Meyer: "Nonlinear perturbation of linear programs", SIAM Journal on Control and Optimization 17, 1979, 745-752.

15. B. A. Murtagh & M. A. Saunders: "MINOS 5.0 user's guide", Stanford University Technical Report SOL 83.20, December 1983.

16. J. S. Pang: "More results on the convergence of iterative methods for the symmetric linear complementarity problem", Journal of Optimization Theory and Applications 49, 1986, 107-134.

17. B. T. Polyak: "Introduction to optimization", Optimization Software, Inc., New York 1987 (translated from Russian).

18. B. T. Polyak & N. V. Tretiyakov: "Concerning an iterative method for linear programming and its economic interpretation", Economics and Mathematical Methods 8(5) 1972, 740-751 (Russian).

19. R. T. Rockafellar: "A dual approach to solving nonlinear programming problems by unconstrained optimization", Mathematical Programming 5, 1973, 354-373.

20. R. T. Rockafellar: "Monotone operators and the proximal point algorithm", SIAM Journal on Control and Optimization 14, 1976, 877-898.

21. R. T. Rockafellar: "Augmented Lagrangians and applications of the proximal point algorithm in convex programming", Mathematics of Operations Research 1, 1976, 97-116.

22. Yu. P. Syrov & Sh. S. Churkreidze: "Questions on the optimization of inter-branch and inter-regional connections in the planning for the growth of a unified national-economic system", Institute of National Economy of Irkutsk, Irkutsk 1970 (Russian).