A PARALLEL ASYNCHRONOUS SUCCESSIVE OVERRELAXATION ALGORITHM FOR SOLVING THE LINEAR COMPLEMENTARITY PROBLEM

by

Karen M. Thompson

Computer Sciences Technical Report #705

July 1987

A PARALLEL ASYNCHRONOUS SUCCESSIVE OVERRELAXATION ALGORITHM FOR SOLVING THE LINEAR COMPLEMENTARITY PROBLEM

Karen M. Thompson

Computer Sciences Department

University of Wisconsin, Madison, Wisconsin 53706

Abstract We present a parallel asynchronous successive overrelaxation algorithm for the solution of symmetric linear complementarity problems and linear programs. A distinguishing feature of this algorithm is that processors need not communicate after each update of the solution vector and therefore processor idle time can be avoided. The proposed parallel algorithm is applied to finding least 2–norm solutions of linear programs. Improvement is observed over the synchronized version of the algorithm, the parallel gradient projection successive overrelaxation algorithm.

Key Words: Parallel algorithms, SOR, gradient projection, linear complementarity problem, linear programming

Abbreviated Title: Asynchronous SOR Solution of LCPs and LPs.

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

1. INTRODUCTION

In this chapter we propose a parallel asynchronous successive overrelaxation algorithm to solve the linear complementarity problem.

$$Mz + q \geq 0, z \geq 0, z(Mz + q) = 0$$

where M is a $n \times n$ symmetric matrix and q is a vector in $\mathbb{R}^n$. The serial successive overrelaxation (SOR) iterate is as follows:

$$z^{i+1} = (z^i - \omega E(Mz^i + q + K(z^{i+1} - z^i)))_+$$

where E is a positive diagonal matrix in $\mathbb{R}^{n \times n}$,ω is some positive number, and K is some substitution operator such as the lower or upper triangular part of M.

Related work has been done on parallel iterative methods for solving the system of equations

$$Ax = b$$

Missirlis, [Missirlis85] proposed a Jacobi–type parallel iterative scheme based on finding an approximation of the Neumann series to A^{-1}. Conrad and Wallach, [Conrad & Wallach77], propose a parallel Gauss–Seidel method based on breaking the matrix A into blocks. They show convergence for diagonally dominant systems. An asynchronous algorithm was proposed by [Barlow & Evans82] for matrices A which have the special property

$$A = \begin{bmatrix} I & B \\ B & I \end{bmatrix}$$

Previously proposed parallel SOR algorithms for solving the LCP have been described in [Mangasarian & DeLeone86a] and [Mangasarian & DeLeone86b]. Both methods required synchronization after each update of z. The proposed algorithms require communications only after asynchronous multiple updates of z. Pang and Yang, [Pang & Yang87], propose a parallel asynchronous algorithm which breaks the matrix M into blocks and makes multiple iterations in parallel. The proposed algorithm is an asynchronous version of [Mangasarian & De Leone86a].

The significance of developing an asynchronous algorithm lies in the need to develop efficient parallel algorithms. Ideally, the idle time of all processors as well as redundant computing time should be zero. If an algorithm consisted of p independent tasks which required the same amount of computing time we could assign the work to p different processors and complete the work p-times as fast. However, optimization algorithms have serial components, thereby requiring communication between the processors. If communication must occur often then communication costs become a large part of the computation time. Therefore it's desirable to communicate as little as possible without degrading the convergence properties of the algorithm.

In this chapter we propose a more general version of GPSOR presented in [Mangasarian & De Leone86b], that can be implemented in an asynchronous way. Like Pang and Yang's work we divide the matrix M into blocks and perform multiple iterations in parallel. However, Pang and Yang's method requires a two stage regular splitting of M which can only be guaranteed for positive definite M. In contrast our method requires positive semidefinite M.

In Section 2 we discuss the previous work of Mangasarian and DeLeone found in [Mangasarian & DeLeone86b]. Our principal contribution is contained in Sections 3, 4, and 5 where we discuss the proposed Asynchronous SOR (ASOR) algorithm. Section 3 develops optimality conditions for the proposed ASOR. In Section 4 we specify the algorithm and present convergence results. Finally in Section 5 computational results are presented based on tests performed on generated linear programs formulated as symmetric LCPs.

We briefly describe our notation now. For a vector z in the n–dimensional real space $\mathbb{R}^n$, z_+ will denote the vector in $\mathbb{R}^n$ with components $(z_+)_i = \max\{z_i, 0\}$, $i = 1, \ldots, n$. The scalar product of two vectors x and y in $\mathbb{R}^n$ will be simply denoted by xy. For $z \in \mathbb{R}^n$, $\| z \| = (z^T z)^{\frac{1}{2}}$ is the standard Euclidean norm. We also have $\| z \|_M = (zMz)^{1/2}$. $\mathbb{R}^{m \times n}$ will denote the set of all $m \times n$ real matrices. For $A \in \mathbb{R}^{m \times n}, A^T$ will denote the transpose, A_i will denote the ith row, A_{ij} the element in row i and column j, and for $\mathcal{I} \in \{1, \ldots, m\}, \mathcal{J} \in \{1, \ldots, n\}$, $A_{\mathcal{I}}$ will denote the submatrix of A with rows A_i, $i \in \mathcal{I}$, while $A_{\mathcal{I}\mathcal{J}}$ will denote the submatrix of A with elements A_{ij}, $i \in \mathcal{I}, j \in \mathcal{J}$.

2. THE PARALLEL GPSOR ALGORITHM

We start by recalling that the LCP is equivalent to finding $z \in \mathbb{R}^n$ such that

$$z = (z - \omega(Mz + q))_+ \text{ for some } \omega > 0 \tag{2.1}$$

The proof of this can be found in [Mangasarian77]. The above relationship suggests an obvious algorithm. Given $z^i \in \mathbb{R}^n$,

$$z^{i+1} = (z^i - \omega(Mz^i + q))_+$$

If M is symmetric this algorithm is a gradient projection algorithm. We can modify this algorithm by updating z component by component and using the most current information to update subsequent components. Therefore the modified algorithm would be

$$p(z) = (z - \omega E(Mz + q + K(p(z) - z))_+ \tag{2.2}$$

where ω is some positive number, E is a positive diagonal matrix in $\mathbb{R}^{n\times n}$ and K is some substitution operator such as the strictly lower or upper triangular part of M. Clearly, a fixed point of this algorithm also solves the LCP.

The GPSOR algorithm is based on the following optimality conditions presented in [Mangasarian & De Leone86b].

Theorem 2.1. *(GPSOR optimality condition) Let M,K,$E \in \mathbb{R}^{n\times n}$, $q \in \mathbb{R}^n$, $\omega > 0$ such that E is a positive diagonal matrix.*

(a) If z solves the LCP and $(\omega E)^{-1} + K$ is positive definite then $p(z) = z$ where $p(z)$ is a solution of (2.2).

(b) If $p(z)$ solves (2.2) and $p(z) = z$ then z solves the LCP.

Proof :

See [Mangasarian & De Leone86b] ∎

Consider the SOR iterate (2.2). If we choose $K = L$, where L is the strictly lower triangular part of M, we see that $p_j^{i+1}(z)$ replaces z_j^i during the computation of $p_l^{i+1}(z)$ for all $l > j$. This selection of K produces an algorithm which is sequential in nature. The idea Mangasarian and De Leone use to develop a parallel GPSOR is to find an appropriate K which will allow the algorithm to be split into parallel parts. Therefore K is chosen to be a block diagonal matrix consisting of the strictly lower triangular part of the principal submatrix of each horizontal partition of the matrix M. Then the new algorithm can be easily distributed on to a number of processors equal to the number of submatrices in the horizontal partition of M. This choice of K was first proposed in [Mangasarian & De Leone86a] for a different SOR procedure which didn't involve a stepsize. Therefore, to get convergence an ω restricted to an interval depending on E and K must be chosen. In practice this resulted in small ω which resulted in a slow algorithm. GPSOR allows a choice of ω in the range (0,2) when M is positive semidefinite. To specify parallel GPSOR, partition the matrix M into p contiguous horizontal blocks as follows:

$$M := \begin{bmatrix} M_{\mathcal{I}_1} \\ M_{\mathcal{I}_2} \\ \vdots \\ M_{\mathcal{I}_p} \end{bmatrix}$$

where the blocks $M_{\mathcal{I}_j}$ correspond to the variables $z_{\mathcal{I}_j}$ and $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p\}$ is a consecutive partition of $\{1, 2, \ldots, n\}$. Now partition $M_{\mathcal{I}_j}$ as follows:

$$M_{\mathcal{I}_j} := [M_{\mathcal{I}_j\mathcal{I}_j} \quad M_{\mathcal{I}_j\mathcal{I}_j^c}]$$

where $\mathcal{I}_j^c$ is the complement of $\mathcal{I}_j$ in $\{1, 2, \ldots, p\}$. Therefore, $M_{\mathcal{I}_j\mathcal{I}_j}$ is a principal square submatrix of M. Now

$$M_{\mathcal{I}_j\mathcal{I}_j} := L_{\mathcal{I}_j\mathcal{I}_j} + D_{\mathcal{I}_j\mathcal{I}_j} + U_{\mathcal{I}_j\mathcal{I}_j}$$

where $L_{\mathcal{I}_j\mathcal{I}_j}$ and $U_{\mathcal{I}_j\mathcal{I}_j}$ are the strictly lower and upper triangular part of $M_{\mathcal{I}_j\mathcal{I}_j}$, and $D_{\mathcal{I}_j\mathcal{I}_j}$ is the diagonal part of $M_{\mathcal{I}_j\mathcal{I}_j}$.

Now define K as follows:

$$K := \begin{bmatrix} L_{\mathcal{I}_1\mathcal{I}_1} & & & \\ & L_{\mathcal{I}_2\mathcal{I}_2} & & \\ & & \ddots & \\ & & & L_{\mathcal{I}_p\mathcal{I}_p} \end{bmatrix} \tag{2.3}$$

Algorithm 2.1 can now be performed for each row block $\mathcal{I}_j$ for $j = 1, \ldots, p$ simultaneously. We are now ready to specify the algorithm.

Algorithm 2.1 Parallel GPSOR Algorithm for the LCP

Let $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p\}$ be a consecutive partition of $\{1, 2, \ldots, n\}$,let E be a positive diagonal matrix in $\mathbb{R}^{n\times n}$ and let $z^0 \geq 0$. For $i = 0, 1, 2, \ldots$ do the following

Direction Generation Define the direction

$$d^i := p(z^i) - z^i := \begin{pmatrix} p_{\mathcal{I}_1}(z^i) - z^i_{\mathcal{I}_1} \\ \vdots \\ p_{\mathcal{I}_p}(z^i) - z^i_{\mathcal{I}_p} \end{pmatrix} \tag{2.4}$$

such that $p(z^i)$ satisfies

$$p_{\mathcal{I}_j}(z^i) = (z_{\mathcal{I}_j} - \omega E_{\mathcal{I}_j\mathcal{I}_j}(M_{\mathcal{I}_j} z^i + q_{\mathcal{I}_j} + L_{\mathcal{I}_j\mathcal{I}_j}(z^{i+1}_{\mathcal{I}_j} - z^i_{\mathcal{I}_j})))_+ \tag{2.5}$$

for $j = 1, \ldots p$ where $\omega > 0$ is chosen so that for some $\gamma > 0$

$$z_{\mathcal{I}_j}((\omega E_{\mathcal{I}_j})^{-1} + L_{\mathcal{I}_j\mathcal{I}_j})z_{\mathcal{I}_j} \geq \gamma \parallel z_{\mathcal{I}_j} \parallel^2 \quad \forall z_{\mathcal{I}_j} \tag{2.6}$$

for $j = 1, \ldots, p$

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

Stepsize Generation

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

where

$$f(z^i + \lambda^i d^i) = \min_{\lambda}\{f(z^i + \lambda d^i) | z^i + \lambda d^i \geq 0\} \tag{2.7}$$

The direction generation can be performed in parallel on p processors. The stepsize generation is performed and the new value of z^{i+1} is shared between p processors.

The following results are from [Mangasarian & De Leone86b].

Theorem 2.2. *(Convergence of the Parallel GPSOR Algorithm) Either the sequence $\{z^i\}$ generated by the Parallel GPSOR Algorithm 2.1 terminates at a solution of the LCP or each of its accumulation points solves the LCP.*

Corollary 2.6. *Condition (2.6) of Algorithm 2.1 hold with either of the following two assumptions:*

(i) $0 < \omega < \min_{j=1,\dots,p} \min_{i\in\mathcal{I}_j} 2/E_{ii} \sum_{\substack{l\in\mathcal{I}_j \\ l\neq i}} \mid M_{il} \mid$

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

3. OPTIMALITY CONDITIONS FOR ASOR

Parallel GPSOR must be synchronized after every iteration. On a loosely coupled network this would involve communicating information after every iteration. Because the SOR iteration is relatively cheap communication costs may be a large proportion of the total time, thus causing the algorithm to be inefficient. Therefore, it would be advantageous to construct an algorithm that is asynchronous.

The proposed asynchronous algorithm is a multi-sweep GPSOR. The idea of the algorithm is that each processor makes multiple updates of $z_{\mathcal{I}_j}$ before sharing information. Assuming K is defined as in (2.3) we have

$$p(z) := \begin{pmatrix} p^{k_1}_{\mathcal{I}_1}(z) \\ \vdots \\ p^{k_p}_{\mathcal{I}_p}(z) \end{pmatrix}$$

where

$$p^{k_j}_{\mathcal{I}_j}(z) = \Big(p^{k_j-1}_{\mathcal{I}_j}(z) - \omega E_{\mathcal{I}_j\mathcal{I}_j}(M_{\mathcal{I}_j} p^{k_j-1}_{\mathcal{I}_j}(z) + q_{\mathcal{I}_j} + L_{\mathcal{I}_j\mathcal{I}_j}(p^{k_j}_{\mathcal{I}_j}(z) - p^{k_j-1}_{\mathcal{I}_j}(z)))\Big)_+ \tag{3.1}$$

We now develop optimality conditions for ASOR.

Theorem 3.1. *(Optimality Conditions for Asynchronous SOR)*

Let $M, K, E \in \mathbb{R}^{n\times n}$, $q \in \mathbb{R}^n$, $\omega > 0$ such that E is a positive diagonal matrix.

(a) (Necessity)If z solves the LCP and $(\omega E)^{-1} + K$ is positive definite then p(z)=z.

(b) (Sufficiency) If $p(z) = z$, $(\omega E_{\mathcal{I}_j})^{-1} + L_{\mathcal{I}_j\mathcal{I}_j}$ is positive definite and $(\omega E_{\mathcal{I}_j\mathcal{I}_j})^{-1} + L_{\mathcal{I}_j\mathcal{I}_j} - \frac{M_{\mathcal{I}_j\mathcal{I}_j}}{2}$ is positive definite $\forall j$ then z solves the LCP.

Proof :

Part (a) follows from the part (a) in Theorem 2.1. We now prove part (b).

Suppose $p^{k_j}_{\mathcal{I}_j}(z) = z_{\mathcal{I}_j}$ $\forall j$. If $k_j = 1$ $\forall j$ then the result follows from part b of Theorem 2.1 because this is equivalent to communicating after each update. Therefore we will show that if $p^{k_j}_{\mathcal{I}_j}(z) = z_{\mathcal{I}_j}$ then $p^1_{\mathcal{I}_j} = z_{\mathcal{I}_j}$ in which case the result follows from Theorem 2.1.

Suppose that $z_{\mathcal{I}_j} \neq p^1_{\mathcal{I}_j}(z)$ but $z_{\mathcal{I}_j} = p^{k_j}_{\mathcal{I}_j}(z)$. Then $z_{\mathcal{I}_j}$ is an accumulation point for the following iterate:

$$z^{i+1}_{\mathcal{I}_j} = (z^i_{\mathcal{I}_j} - \omega E_{\mathcal{I}_j\mathcal{I}_j}(M_{\mathcal{I}_j\mathcal{I}_j} z^i_{\mathcal{I}_j} + q_{\mathcal{I}_j} + M_{\mathcal{I}_j\mathcal{I}^c_j} z_{\mathcal{I}^c_j} + L_{\mathcal{I}_j\mathcal{I}_j}(z^{i+1}_{\mathcal{I}_j} - z^i_{\mathcal{I}_j})))_+$$

where $\{\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_p\}$ is a consecutive partition of $\{1, 2, \ldots, n\}$ and $\mathcal{I}_j^c$ is the complement of $\mathcal{I}_j$ in $\{1, 2, \ldots, p\}$. Then by Theorem 2.1 of [Mangasarian77] we have that $z_{\mathcal{I}_j}$ solves the following LCP:

$$w_{\mathcal{I}_j} = M_{\mathcal{I}_j\mathcal{I}_j} z_{\mathcal{I}_j} + q_{\mathcal{I}_j} + M_{\mathcal{I}_j\mathcal{I}_j^c} z_{\mathcal{I}_j^c} \geq 0$$

$$w_{\mathcal{I}_j} \times z_{\mathcal{I}_j} = 0$$

$$z_{\mathcal{I}_j} \geq 0$$

since $(\omega E_{\mathcal{I}_j\mathcal{I}_j})^{-1} + L_{\mathcal{I}_j\mathcal{I}_j} - \frac{M_{\mathcal{I}_j\mathcal{I}_j}}{2}$ is positive definite. But by Theorem 2.1 part a and $(\omega E_{\mathcal{I}_j\mathcal{I}_j})^{-1} + K_{\mathcal{I}_j\mathcal{I}_j}$ positive definite this contradicts the fact that $z_{\mathcal{I}_j} \neq p^1_{\mathcal{I}_j}(z)$. Therefore from Theorem 2.1(b) z solves the LCP. ∎

4. ASYNCHRONOUS SOR ALGORITHM

We are now prepared to specify the algorithm.

Algorithm 4.1

Direction Generation Define the direction

$$d^i := p(z^i) - z^i$$

Let $z^i := [z^i_{\mathcal{I}_j} \; z^i_{\mathcal{I}_j^c}]$. Define $p^0_{\mathcal{I}_j}(z^i) := z^i_{\mathcal{I}_j}$ Then $p^{k_j}_{\mathcal{I}_j}$ is defined as in (3.1). Choose $E_{\mathcal{I}_j\mathcal{I}_j}$ and $\omega > 0$ such that for some $\gamma_1, \gamma_2 > 0$

$$y(L_{\mathcal{I}_j\mathcal{I}_j} + (\omega E_{\mathcal{I}_j\mathcal{I}_j})^{-1} - \frac{M_{\mathcal{I}_j\mathcal{I}_j}}{2})y > \gamma_1 \parallel y \parallel^2 \quad \forall y \tag{2.17}$$

$$y(L_{\mathcal{I}_j\mathcal{I}_j} + (\omega E_{\mathcal{I}_j\mathcal{I}_j})^{-1})y > \gamma_2 \parallel y \parallel^2 \quad \forall y \tag{4.1}$$

stop if $d^i = 0$ else continue.

Step Generation

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

where

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

We can now use the optimality results to establish convergence for Algorithm 4.1.

Theorem 4.1. *(ASOR Convergence) Let M be symmetric and positive semidefinite. Either the sequence $\{z^i\}$ generated by Algorithm 2.3 terminates at a solution of the LCP or each accumulation point of $\{z^i\}$ solves the LCP.*

Proof :

The sequence terminates only if for some $i, p(z^i) = z^i$ in which case z^i solves the LCP. Suppose $\{z^i\}$ does not terminate and $\bar{z}$ is an accumulation point of $\{z^i\}$.

$$\begin{aligned} -\nabla f(z^i)d^i &= -\nabla f(z^i)(p^i(z^i) - z^i) \\ &= -\nabla_{\mathcal{I}_1} f(z^i)(p^{k_1}_{\mathcal{I}_1}(z^i) - z^i_{\mathcal{I}_1}) - \ldots - \nabla_{\mathcal{I}_p} f(z^i)(p^{k_p}_{\mathcal{I}_p}(z^i) - z_{\mathcal{I}_p}) \end{aligned}$$

Therefore define

$$f^i_{\mathcal{I}_j}(z_{\mathcal{I}_j}) = f(z^i_{\mathcal{I}_1}, z^i_{\mathcal{I}_2}, \ldots, z_{\mathcal{I}_j}, \ldots, z^i_{\mathcal{I}_j})$$

For ease of notation we'll drop the superscript i on $f^i_{\mathcal{I}_j}(z_{\mathcal{I}_j})$. Then

$$
\begin{aligned}
-\nabla_{\mathcal{I}_j} f(z^i)(p^{k_j}_{\mathcal{I}_j} - z^i_{\mathcal{I}_j}) &= f_{\mathcal{I}_j}(z^i_{\mathcal{I}_j}) - f_{\mathcal{I}_j}(p^{k_j}_{\mathcal{I}_j}(z_{\mathcal{I}_j})) + \frac{1}{2} \| p^{k_j}_{\mathcal{I}_j}(z_{\mathcal{I}_j}) - z^i_{\mathcal{I}_j} \|^2_{M_{\mathcal{I}_j\mathcal{I}_j}} \\
&\geq f_{\mathcal{I}_j}(z^i_{\mathcal{I}_j}) - f_{\mathcal{I}_j}(p^1_{\mathcal{I}_j}(z_{\mathcal{I}_j})) + \frac{1}{2} \| p^{k_j}_{\mathcal{I}_j}(z_{\mathcal{I}_j}) - z^i_{\mathcal{I}_j} \|^2_{M_{\mathcal{I}_j\mathcal{I}_j}} \\
&\qquad\qquad \text{(By (4.1) and Theorem 2.1 [Mangasarian77])} \\
&\geq \| p^1_{\mathcal{I}_j}(z^i_{\mathcal{I}_j}) - z^i_{\mathcal{I}_j} \|^2_{L_{\mathcal{I}_j\mathcal{I}_j} + (\omega E_{\mathcal{I}_j\mathcal{I}_j})^{-1} - M_{\mathcal{I}_j\mathcal{I}_j}/2} \\
&\qquad + \frac{1}{2} \| p^{k_j}_{\mathcal{I}_j}(z_{\mathcal{I}_j}) - z^i_{\mathcal{I}_j} \|^2_{M_{\mathcal{I}_j\mathcal{I}_j}} \\
&\qquad\qquad \text{(By Theorem 2.1 [Mangasarian77])} \\
&\geq \gamma_1 \| p^1_{\mathcal{I}_j}(z^i_{\mathcal{I}_j}) - z^i_{\mathcal{I}_j} \|^2 + \frac{1}{2} \| p^{k_j}_{\mathcal{I}_j}(z_{\mathcal{I}_j}) - z^i_{\mathcal{I}_j} \|^2_{M_{\mathcal{I}_j\mathcal{I}_J}} \\
&\qquad\qquad \text{(By (4.1))} \\
&\geq \gamma_1 \theta_{\mathcal{I}_j}(z_{\mathcal{I}_j})^2 \quad \text{(By positive semidefiniteness of } M\text{)}
\end{aligned}
$$

Hence,

$$
-\nabla f(z_i) d^i \geq \gamma(\theta_{\mathcal{I}_1}(z^i_{\mathcal{I}_1})^2 + \dots \theta_{\mathcal{I}_p}(z^i_{\mathcal{I}_p})^2) = \gamma\theta(z^i)^2 \geq 0
$$

Now let $\{z^{i_j}\}$ be a subsequence converging to $\bar{z}$. We claim that $p(z^{i_j})$ is bounded. Suppose $p(z^{i_j})$ is not bounded and suppose

$$
p(z^{i_j}) = \begin{pmatrix} p^1_{\mathcal{I}_1}(z^{i_j}) \\ p^1_{\mathcal{I}_2}(z^{i_j}) \\ \vdots \\ p^1_{\mathcal{I}_p}(z^{i_j}) \end{pmatrix}
$$

Then

$$\lim_{j\to\infty} \frac{p^1_{\mathcal{I}_k}(z^{i_j})}{\| p^1_{\mathcal{I}_k}(z^{i_j}) \|} = \lim_{j\to\infty}\Big(-\omega E_{\mathcal{I}_k\mathcal{I}_k} L_{\mathcal{I}_k\mathcal{I}_k} \frac{p^1_{\mathcal{I}_k}(z^{i_j})}{\| p^1_{\mathcal{I}_k}(z^{i_j}) \|}\Big)_+$$

and hence for an accumulation point $\bar{p}$ we have

$$\bar{p} = (-\omega E_{\mathcal{I}_k\mathcal{I}_k} L_{\mathcal{I}_k\mathcal{I}_k}\bar{p})_+ \quad \bar{p} \neq 0$$

This however is equivalent to

$$0 \neq \bar{p} \geq 0, \bar{p}((\omega E_{\mathcal{I}_k\mathcal{I}_k})^{-1} + L_{\mathcal{I}_k\mathcal{I}_k})\bar{p} = 0$$

$$((\omega E_{\mathcal{I}_k\mathcal{I}_k})^{-1} + K_{\mathcal{I}_k\mathcal{I}_k})\bar{p} \geq 0$$

But this contradicts (4.2). By induction on l for $p^l_{\mathcal{I}_k}(z^{i_j})$ we see that $p(z^{i_j})$ is bounded. Without loss of generality let $\{z^{i_j}, p(z^{i_j})\} \to \{\bar{z}, \bar{p}\}$ and hence $\bar{d} = \bar{p} - \bar{z}$. Since $p(z^{i_j}) \geq 0$

$$z^{i_j} + \lambda d^{i_j} = (1-\lambda)z^{i_j} + \lambda p(z^{i_j}) \geq 0 \quad \text{for} \quad 0 \leq \lambda \leq 1$$

Consequently

$$f(z^{i_j} + \lambda d^{i_j}) \geq f(z^{i_{j+1}}) \geq f(z^{i_j+1}) \quad \text{for} \quad 0 \leq \lambda \leq 1$$

letting $j \to \infty$ we get

$$f(\bar{z} + \lambda\bar{d}) \geq f(\bar{z}) \quad 0 \leq \lambda 1$$

Hence $\nabla f(\bar{z})\bar{d} \geq 0$ and

$$0 \geq -\nabla f(\bar{z})\bar{d} = \lim_{j\to\infty} -\nabla f(z^{i_j})d^{i_j} \geq \lim_{j\to\infty} \gamma\theta(z^{i_j})^2 \geq 0$$

Therefore $\theta(\bar{z}) = 0$. Then by Theorem 3.1 and conditions (4.1) and (4.2) $\bar{z}$ solves the LCP. ▌

When E is chosen appropriately we have the following special case.

Corollary 4.1. *(Special Case) Conditions (4.1) and (4.2) of Algorithm 4.1 hold when M is positive semidefinite $E_{\mathcal{I}_j\mathcal{I}_j} = D^{-1}_{\mathcal{I}_j\mathcal{I}_j} > 0$ and $0 < \omega < 2$.*

Proof :

(i)

$$
\begin{aligned}
&z((\omega E_{\mathcal{I}_j\mathcal{I}_j})^{-1} + L_{\mathcal{I}_j\mathcal{I}_j} - \frac{M_{\mathcal{I}_j\mathcal{I}_j}}{2})z = \\
&\qquad = z(\omega^{-1} D_{\mathcal{I}_j\mathcal{I}_j} + L_{\mathcal{I}_j\mathcal{I}_j} - \frac{L_{\mathcal{I}_j\mathcal{I}_j} + U_{\mathcal{I}_j\mathcal{I}_j} + D_{\mathcal{I}_j\mathcal{I}_j}}{2})z \\
&\qquad = \frac{1}{2} z((2\omega^{-1} - 1)D_{\mathcal{I}_j\mathcal{I}_j})z \geq \gamma \parallel z \parallel^2
\end{aligned}
$$

where the last inequality follows form the positivity of $D_{\mathcal{I}_j\mathcal{I}_j}$ and $0 < \omega < 2$.

(ii)

$$
\begin{aligned}
z((\omega E_{\mathcal{I}_j\mathcal{I}_j})^{-1} + L_{\mathcal{I}_j\mathcal{I}_j})z &= z(\omega^{-1} D_{\mathcal{I}_j\mathcal{I}_j} + \frac{L_{\mathcal{I}_j\mathcal{I}_j} + U_{\mathcal{I}_j\mathcal{I}_j}}{2})z \\
&= \frac{1}{2} z((2\omega^{-1} - 1)D_{\mathcal{I}_j\mathcal{I}_j} + M_{\mathcal{I}_j\mathcal{I}_j}) \geq \gamma \parallel z \parallel^2
\end{aligned}
$$

where the last inequality follows from the positive semidefiniteness of M, the positivity of D and $0 < \omega < 2$. ∎

5. Computational Experience

The ASOR algorithm was tested on a symmetric LCP formulation of a linear programming problem. We state the LP in standard form.

$$
\begin{aligned}
&\min \quad cx \\
&\text{st.} \\
&\qquad Ax \geq b \\
&\qquad x \geq 0
\end{aligned}
$$

where $c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}$and $b \in \mathbb{R}^m$. The LCP formulation due to Mangasarian [Mangasarian84] is based on the following observation. Consider the following quadratic programming problem.

$$\min \quad cx + \frac{\epsilon}{2} \parallel x \parallel_2^2$$

$$\text{st.}$$

$$Ax \geq b$$

$$x \geq 0$$

The unique solution of the above QP is the least 2–norm solution of the LP provided that $\epsilon \in (0, \bar{\epsilon}]$ for some $\bar{\epsilon} > 0$. We take the dual of QP and get the following dual program.

$$\max_{(u,v) \geq 0} \quad cx + \frac{\epsilon}{2} \parallel x \parallel_2^2 + u(b - Ax) - vx$$

$$\text{st.}$$

$$c + \epsilon x - A^T u - v = 0$$

where u and v are dual variables in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively. After making appropriate substitutions we get the following minimization of a quadratic function over nonnegativity constraints

$$\min_{(u,v) \geq 0} \quad \frac{1}{2\epsilon} \parallel A^T u + v - c \parallel_2^2 - bu$$

Then the LCP formulation falls out by constructing the KKT conditions.

The algorithm was tested on the CRYSTAL multi–computer. CRYSTAL is a set of 20 VAX – 11/750 computers with 2 megabytes of memory each connected by a 80 megabit/sec Proteon ProNet token ring. We randomly generate an LP which is known to be solvable. We choose $\epsilon = .2$ and ω varies between 1.2 and 1.8. Primal feasibilities were calculated to an accuracy of 10^{-5}. The primal objective function values were accurate to four significant digits.

An important question for implementing ASOR is how many updates of z_{I_j} should be carried out in processor j before sharing information. Pang and Yang, [Pang & Yang87], suggest using a progressive tolerance. Therefore we iterate in parallel until the difference between successive iterates is less than a certain tolerance based on a progressive accuracy strategy. Pang and Yang report success on an implementation on the IBM 4381 and the CRAY X-MP/24, however, they compare their method with an earlier version of a parallel SOR algorithm, [Mangasarian & De Leone86a], which is not as effective as GPSOR.

In our experience we have noted that it is not always wise to select $k_j > 1$. We can think of the parallel sweeps as being part of an inner iteration while that step plus the final line search is an outer iteration. Therefore the more sweeps we do in the inner iteration the more time consuming one outer iteration is. If the marginal improvement due to extra inner iterations is very small it may not be worthwhile to choose $k_j > 1$. Our experience has been that near the optimal point one should use only one sweep.

We incorporated the following strategy. A master node is in charge of determining the number of sweeps at a given iteration. If

$$\frac{\| u^{i+1} - u^{i} \|}{\| u^{i} - u^{i-1} \|} > r$$

use one sweep. Otherwise, use two sweeps. In our computations we use $r = .25$. We note that since we were able to balance the load evenly among the processors there was no need to have each node carry out a different number of sweeps. However, this can be changed in case of unbalanced distributions.

Computational results are presented for LPs of size 125 x 500, 250 x 1000, and 1000 x 4000. The two smaller problems have density 5% while the 1000 x 4000 problem has density .8%. Figures 5.1, 5.2, and 5.3 graph time against number of processors. Each graph represents an average over three random problems. We note that in these cases ASOR tends to provide slight improvements over GPSOR. If this algorithm was implemented on a parallel system with a slower communication medium such as a network of workstations these differences may become more pronounced. However, on a shared memory processor the savings of ASOR diminish.

In Figures 5.2 and 5.3 we note that two processors solve the problem more than twice as fast as one processor. This is due to the fact that the substitution operator K changes as we move to more processors. Therefore we are running a different algorithm as opposed to simply distributing the program. In these cases there was a drop in iteration number which accounts for the extra speedup. This leads us to a certain side benefit of studying parallel algorithms. That is we can discover faster serial algorithms! In other words if we use the substitution operator K used in the two processor case but run it serially we should produce a faster serial algorithm. This phenomenon was originally discussed in [Mangasarian & De Leone86b].

Finally we note that further research may provide a more sophisticated criterion to determine how many multiple sweeps should be computed at a given iteration. Such a criterion may provide substantial improvement in the algorithm.

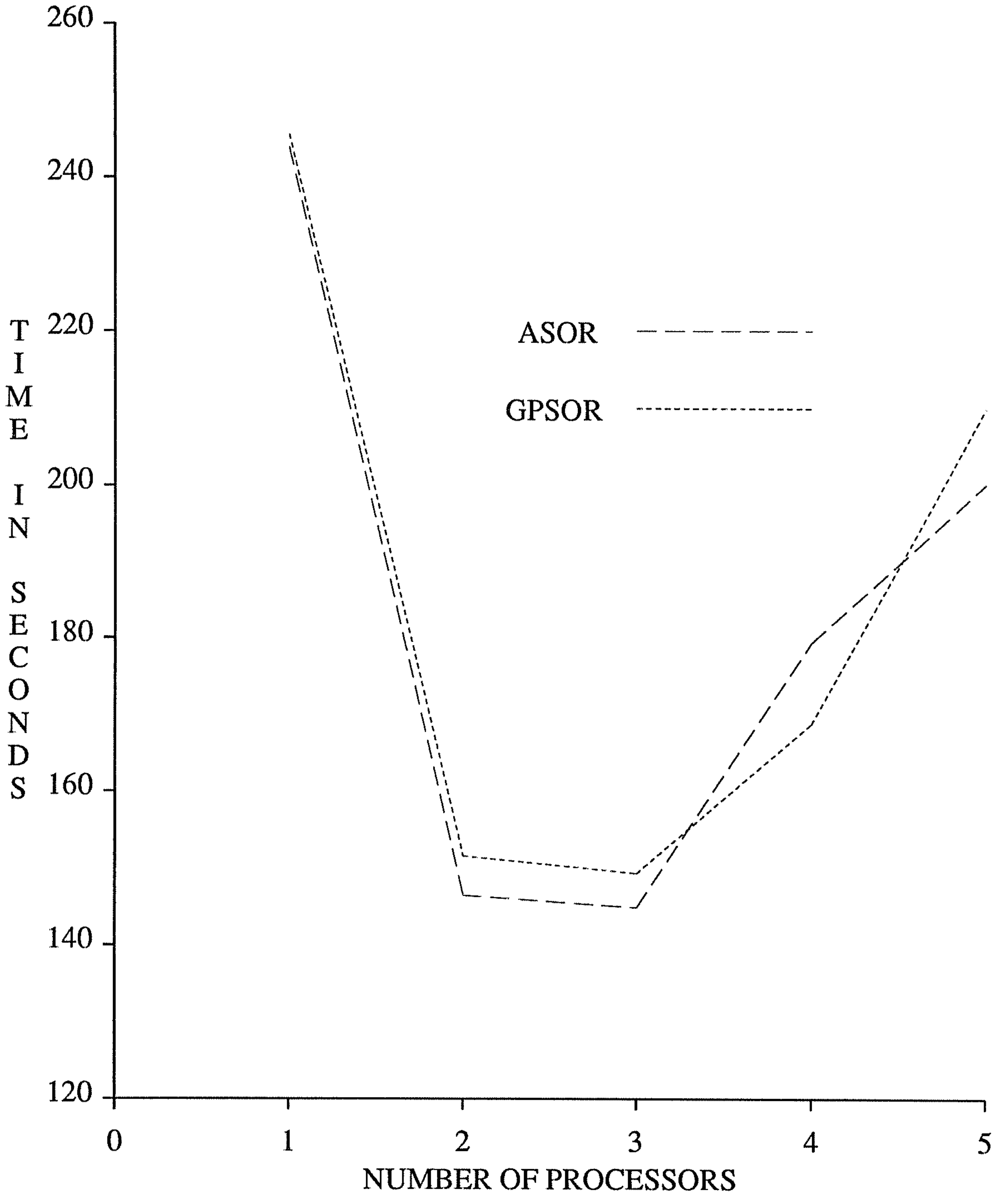

Figure 5.1 Total Time (d=5%, m=125, n=500)

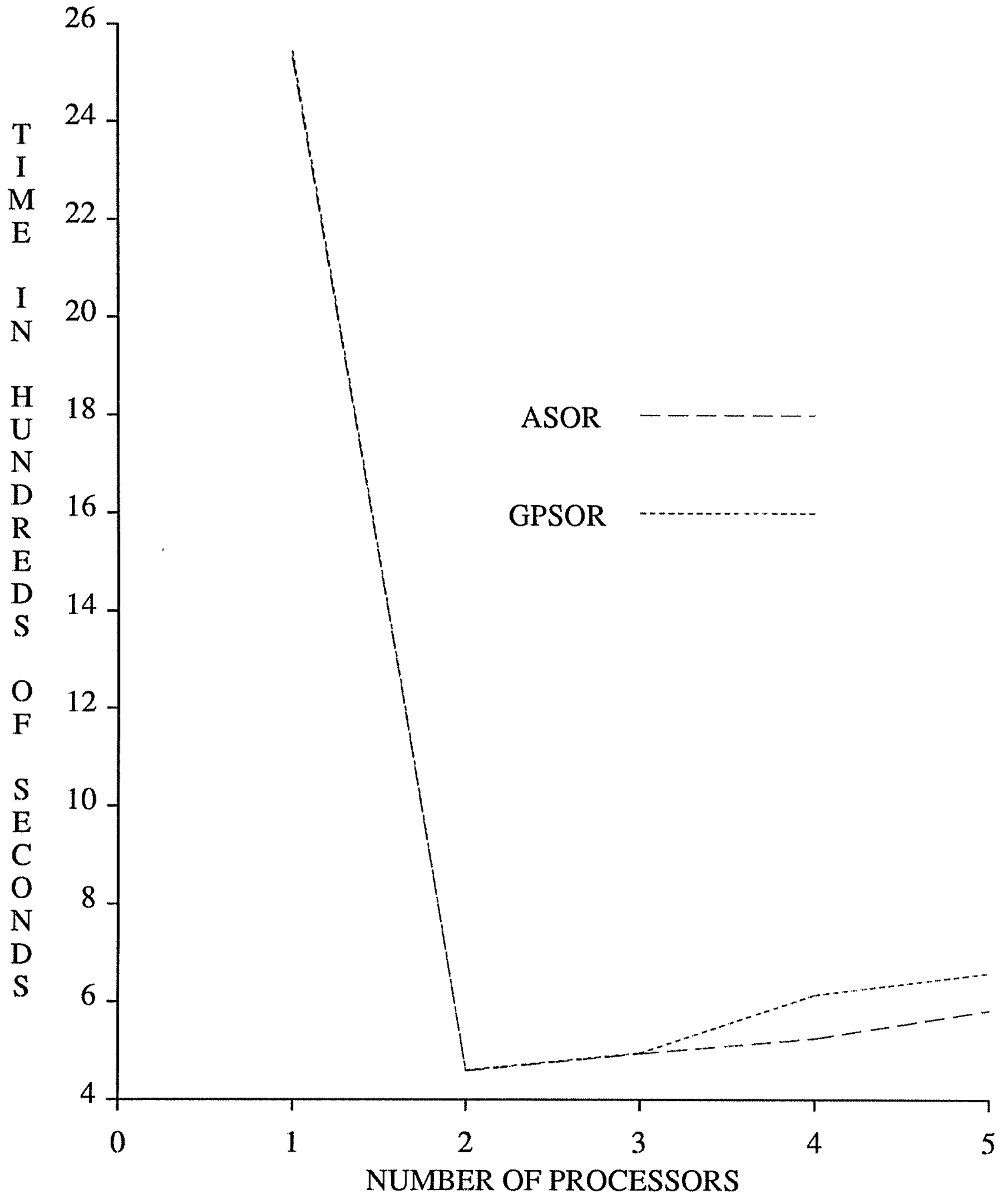

Figure 5.2 Total Time (d=5%, m=250, n=1000)

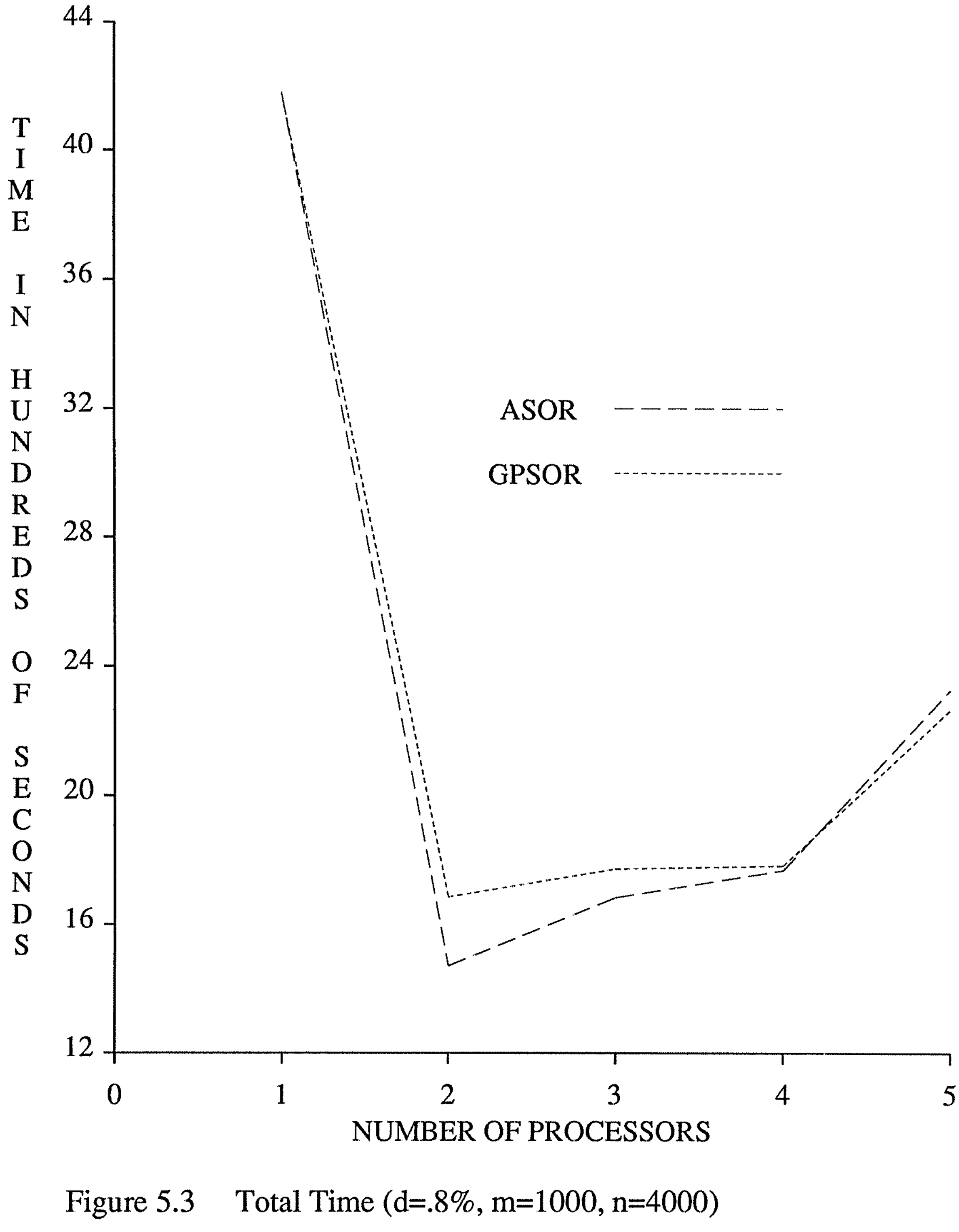

Figure 5.3 Total Time (d=.8%, m=1000, n=4000)

References

Barlow, R. H. and Evans, D. J. (1982) Parallel Algorithms for the Iterative Solution to Linear Systems, The Computer Journal, Vol. 25, No. 1, 56–6-.

Conrad, V. and Wallach, Y. (1977) Iterative Solution of Linear Equations on a Parallel Processor System, IEEE Transactions on Computers, C–26, 9, 838–847.

DeWitt, D. J., Finkel, R. and Solomon, M. (1984) The CRYSTAL Multicomputer: Design and Implementation Experience, Computer Sciences Technical Report #553, University of Wisconsin–Madison.

Mangasarian, O. L. (1977) Solution of Symmetric Linear Complementarity Problems by Iterative Methods, Journal of Optimization Theory and Applications, 22, 465–484.

Mangasarian, O. L. (1984) Sparsity–Preserving SOR Algorithms for Separable Quadratic and Linear Programs, Computers and Operations Research 11, 105–112.

Mangasarian, O. L. and De Leone, R. (1986a) Parallel Successive Overrelaxation Methods for Symmetric Linear Complementarity Problems and Linear Programs, Computer Sciences Technical Report #647, University of Wisconsin–Madison.

Mangasarian, O. L. and De Leone, R. (1986b) Parallel Gradient Projection Successive Overrelaxation for Symmetric Linear Complementarity Problems and

Linear Programs, Computer Sciences Technical Report #659, University of Wisconsin–Madison.

Missirlis, N. M. (1985) A Parallel Iterative System Solver, Linear Algebra and its Applications 65, 25-44.

Pang, J. S. and Yang, J. M. (1987) Two–Stage Parallel Iterative Methods for the Symmetric Linear Complementarity Problem, School of Management, The University of Texas at Dallas.