MULTI-SWEEP ASYNCHRONOUS PARALLEL SUCCESSIVE OVERRELAXATION FOR THE NONSYMMETRIC LINEAR COMPLEMENTARITY PROBLEM

by

R. De Leone, O. L. Mangasarian & T.-H. Shiau

Computer Sciences Technical Report #792

September 1988

Multi-Sweep Asynchronous Parallel Successive Overrelaxation for the Nonsymmetric Linear Complementarity Problem[1)]

R. De Leone, O. L. Mangasarian & T.-H. Shiau[2)]

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53706

Technical Report #792

September 1988

Revised March 14, 1989

Abstract. Convergence is established for the **multi-sweep** asynchronous parallel successive overrelaxation (SOR) algorithm for the **nonsymmetric** linear complementarity problem. The algorithm was originally introduced in [4] for the symmetric linear complementarity problem. Computational tests show the superiority of the multi-sweep asynchronous SOR algorithm over its single-sweep counterpart on both symmetric and nonsymmetric linear complementarity problems.

Key Words: Linear complementarity, parallel algorithm, asynchronous algorithm, SOR

Abbreviated Title: Asynchronous Parallel SOR for LCP

[1)] This material is based on research supported by National Science Foundation Grants CCR-8723091 and DCR-8521228 and Air Force Office of Scientific Research Grants AFOSR-86-0172 and AFOSR-86-0124.

[2)] Department of Computer and Information Science, New Jersey Institute of Technology, 323 Dr. Martin Luther King Blvd., Newark, NJ 07102.

1. Introduction

Our concern here is the asynchronous parallel solution of the linear complementarity problem [3]

$$Mz + q \geq 0,\ z \geq 0,\ z(Mz + q) = 0 \tag{1.1}$$

where M is a given real (possibly nonsymmetric) matrix in $R^{n\times n}$ and q is a given vector in R^n. The asynchronous parallel algorithm under consideration here was first proposed in [4] and its convergence was established there for the case when the matrix M is symmetric and the parallel processors were synchronized after a single sweep through the variables of the problem. By contrast we shall establish here convergence of the same algorithm with no assumption of symmetry on M and for the multi-sweep case where the parallel processors are synchronized only after at most k sweeps through the variables of the problem. The integer k is finite, but may be arbitrarily large. Although synchronization is not essential for convergence, it provides a practical means of enforcing the finite delay condition (2.2b) of Algorithm 2.1. In [4] the multi-sweep algorithm was tested computationally on symmetric linear complementarity problems, even though its convergence was not established there, and was found to be superior to the single-sweep algorithm. Therefore it seems highly desirable to establish convergence for the multi-sweep algorithm.

We briefly summarize now the results of the paper. In Section 2 we state the multi-sweep asynchronous parallel algorithm (Algorithm 2.1) in a different format from [4] and establish its convergence (Theorem 2.5) for a relaxation factor ω interval of $(0, 1]$ under the assumption that M has a positive diagonal and the comparison matrix $C(M)$ has a nonnegative inverse. (The comparison matrix $C(M)$ is obtained from M by taking absolute values of the diagonal elements and the negative of the absolute values for the off-diagonal elements.) When M has a positive diagonal and is strictly diagonally dominant, its comparison matrix $C(M)$ has a nonnegative inverse [6, Theorem 6.2.17].

In Section 3 we give numerical test results carried on the Sequent Symmetry S81 multiprocessor [7]. These results indicate that the multi-sweep asynchronous algorithm is superior to the single-sweep asynchronous algorithm.

We briefly describe our notation now. For a vector z in the n-dimensional real space R^n, z_+ will denote the vector with components $(z_+)_i := \max\{z_i, 0\}$, $i = 1, \ldots, n$. The scalar product of two vectors x and y in R^n will be denoted by xy. The 2-norm $(zz)^{\frac{1}{2}}$ of a vector z in R^n, will be denoted by $\|z\|$. 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 $m \times n$ real matrices. For $A \in R^{m\times n}$, A^T will denote the transpose, A_i will denote the ith row and A_{ij} the element in row i and column j. For $B \in R^{n\times n}$, $\rho(B)$ will denote its spectral radius. For a vector $x \in R^n$, $|x|$ will denote the vector with components $|x_i|$, $i = 1, \ldots, n$, and similarly for $A \in R^{m+n}$, $|A|$ will denote the matrix with absolute value elements $|A_{ij}|$, $i = 1, \ldots, m$, $j = 1, \ldots, n$.

2. The Multi-Sweep Asynchronous Parallel Successive Overrelaxation Algorithm

In [4] a dynamic asynchronous successive overrelaxation (DASOR) was proposed where the assignment of tasks to processors was done dynamically by employing the first available processor for each task that arose. In DASOR the processors were synchronized after a single sweep through the n components of the vector z, while in DASOR10 [4] the processors were synchronized after 10 sweeps. We will now present a model which will handle DASORk for any finite integer k. Unlike [4] where the convergence proof was based on a forcing-function argument [5], the proof here is based on the point-of-attraction result for asynchronous iterative methods of Baudet [1]. Chazan and Miranker [2] first proposed this approach for linear equations. In [4] the iterate is modeled by $\{z^i\}$, where each z^i denotes the new point after completion of one (DASOR) or ten (DASOR10) sweeps through the components of z. Here instead we shall use $\{x^i\}$ to denote all instances of the vector z, even including those in the middle of a sweep. More specifically let $x^0 = z^0$, and let $t_1 \leq t_2 \leq \cdots \leq t_r \leq \cdots$, be the instants of time at which a new value of the component x_{j_r} becomes available at t_r. Note that we do not exclude the possibility of more than one component becoming available simultaneously, because t_{r+1} can equal t_r. Note also that the order in which the components are computed is determined dynamically by the algorithms, that is the component to be processed next by a freed processor is typically the component $x_{\ell+1}$ where ℓ is the highest index of a component being currently processed. We are now ready to state our algorithm.

2.1 k-Sweep Dynamic Asynchronous SOR (DASORk) Let $x^0 \in R^n_+$. Then

$$x_j^{i+1} = \begin{cases} x_j^i & \text{if} \quad j \neq j_r \\ \left(x_j^i - \omega E_{jj}(M_j y^i + q_j)\right)_+ & \text{otherwise} \end{cases} \tag{2.1}$$

where j_r is the index of the component being currently computed, E is a positive diagonal matrix, y^i is the vector of components of previous iterates used for computing x_j^{i+1}, that is

$$y^i := \left[x_1^{s(i,1)}, x_2^{s(i,2)}, \ldots, x_\ell^{s(i,\ell)}, \ldots, x_n^{s(i,n)}\right] \tag{2.2a}$$

and

$$\min\{0,\, i-k\} \le s(i,\ell) \le i \quad \forall i,\, \ell \tag{2.2b}$$

Furthermore, each index $j \in \{1,\ldots,n\}$ appears infinitely often in the sequence $\{j_r\}$.

One way to implement (2.2b) in a parallel environment is to synchronize the processors after at most k sweeps through the variables.

Note that both DASOR and DASOR10 of [4] are special cases of Algorithm 2.1, and that z^i of [4] is a subsequence of $\{x^i\}$ of the above algorithm. In view of Theorem 1 of Baudet [1], the sequence $\{x^i\}$ of (2.1) converges if the following operator of iteration (2.1) is a contraction:

$$F(x) := \big(x - \omega E(Mx+q)\big)_+ \tag{2.3}$$

For clarity, we state below a weaker version of Baudet's theorem suitable for our purposes.

2.2 Theorem (Baudet [1]). Let the sequence $\{x^i\}$ be generated by the asynchronous iterative method with bounded delay defined by Algorithm 2.1. If $F(x)$, as defined by (2.3), is a contracting operator, that is there exists a nonnegative matrix $A \in R^{n\times n}$ with spectral radius $\rho(A) < 1$ such that for all $x,\ y \in R^n$

$$|F(y) - F(x)| \le A|x-y|\,, \tag{2.4}$$

then $\{x_i\}$ converges to the unique fixed point $\bar{x} = F(\bar{x})$ of F from any starting point x^0.

We shall derive sufficient conditions for (2.4). We begin with two lemmas.

2.3 Lemma Let M have a positive diagonal D, let $E = D^{-1}$. Then

$$|F(x) - F(y)| \le \big((1-\omega)I + \omega D^{-1}|L+U|\big)|x-y|$$

for all x, y, and $0 < \omega \le 1$, where L and U are respectively the strictly lower and upper triangular submatrices of M.

Proof

$$\begin{aligned} F(x) - F(y) &= \big(x - \omega E(Mx+q)\big)_+ - \big(y - \omega E(My+q)\big)_+ \\ &\le \big((I - \omega EM)(x-y)\big)_+ \ \text{(Since } a_+ - b_+ \le (a-b)_+\text{)}. \end{aligned}$$

Taking the plus function of both sides gives

$$\big(F(x)-F(y)\big)_+ \le \big((I-\omega EM)(x-y)\big)_+$$

Similarly by exchanging x and y, we have

$$\big(F(y)-F(x)\big)_+ \le \big((I-\omega EM)(y-x)\big)_+$$

Summing up these two inequalities, we have (since $a_+ + (-a)_+ = |a|$)

$$\begin{aligned}
|F(x)-F(y)| &\le |(I-\omega EM)(x-y)| \\
&\le |(I-\omega E(D+L+U)|\,|x-y| \\
&= |(I-\omega I-\omega D^{-1}(L+U)|\,|x-y| \quad (\text{Since } \mathrm{E}=\mathrm{D}^{-1}) \\
&\le \big((1-\omega)I+\omega D^{-1}|L+U|\big)\,|x-y| \quad \blacksquare
\end{aligned}$$

2.4 Lemma For $0<\omega\le 1$,

$$\rho(A_\omega) \le (1-\omega)+\omega\rho(B) \tag{2.5}$$

where

$$A_\omega := (1-\omega)I+\omega B,\ \ B := D^{-1}|L+U| \tag{2.6}$$

Proof By the definition of A_ω it suffices to show that

$$|\lambda| \le (1-\omega)+\omega\rho(B) \tag{2.7}$$

for all eigenvalues λ of A_ω. Let

$$A_\omega x = \lambda x,\ \ x\neq 0$$

Then

$$\big((1-\omega)I+\omega B\big)x = \lambda x$$

and

$$Bx = \frac{\lambda-(1-\omega)}{\omega}\,x$$

Hence $\frac{\lambda-(1-\omega)}{\omega}$ is an eigenvalue of B and therefore $\left|\frac{\lambda-(1-\omega)}{\omega}\right| \leq \rho(B)$, from which (2.7) follows. ∎

Recall that the comparison matrix $C(M)$ of M is defined [6]

$$C(M)_{ij} := \begin{cases} |M_{ii}| & \text{if} \quad i = j \\ -|M_{ij}| & \text{if} \quad i \neq j \end{cases} \tag{2.8}$$

That is $C(M) = |D| - |L + U|$. By using results on regular splitting of matrices [6] and the above lemmas, we obtain our principal convergence theorem.

2.5 DASORk Convergence Theorem Let M have a positive diagonal and let either M be strictly diagonally dominant, or let $C(M)$ be an M-matrix (i.e. $C(M)^{-1} \geq 0$). Then

(i) $\rho(B) < 1$, where B is defined by (2.6).

(ii) For $0 < \omega \leq 1$, $\rho(A_\omega) < 1$ and F is a contracting operator, where A_ω and F are defined by (2.6) and (2.3) respectively.

(iii) The sequence $\{x^i\}$ of Algorithm 2.1 converges to the unique solution of the linear complementarity problem (1.1).

Proof By Theorem 6.2.17 of [6] strict diagonal dominance implies that $C(M)$ is an M-matrix. Hence we need establish the theorem under this assumption only.

(i) By the Regular Splitting Theorem 7.1.3 of [6], $\rho(B) < 1$.

(ii) This follows from (i) and the last two lemmas.

(iii) It follows by (ii) and Theorem 2.2 that $\{x_i\}$ converges to the unique fixed point of F. By Lemma 2.1 of [5] such a fixed point is a solution of the linear complementarity problem (1.1). ∎

By minor modifications of the above arguments we can show that the Convergence Theorem 2.5 also holds for $0 < \omega < \frac{2}{1+\rho(B)}$, where B is defined by (2.6). Note that under the assumption of Theorem 2.5, we have that $2 > \frac{2}{1+\rho(B)} > 1$.

3. Computational Results

Computational testing of the k-Sweep Asynchronous SOR Algorithm 2.1 was carried out on the Sequent Symmetry S81 multiprocessor [7] with 14 tightly coupled 32-bit 80386 Intel microprocessors that share a 40-Megabyte physical memory (256-Megabyte virtual) and a single copy of DYNIX, an enhanced version of the UNIX operating system. Each 80386 processor is accompanied by an 80387 floating point unit, 64-Kilobyte cache memory and a Weitek 1167 floating point accelerator chip.

All test problems were randomly generated with either strictly diagonally dominant matrices or not, and with a prescribed density d of nonzero randomly placed elements in each row except for one fully dense row. The role of the fully dense row was to simulate imbalance in problem data. All nonzero matrix elements were picked from a uniform distribution on [-0.1, 0.1]. Four sets of test results are depicted in Figures 1 to 4 for a linear complementarity problem with a nonsymmetric real matrix of order 100,000 $\times$ 100,000 and row density of 0.01%, except for one fully dense row. The relaxation factor ω used in all runs was 0.9. All the conditions of the linear complementarity problem (1.1) were satisfied to an accuracy of 10^{-8} by all the solutions obtained. Figures 1 and 2 are for a strictly diagonally matrix with $M_{ii} = 1.001 \sum_{i \neq j} |M_{ij}|$ for all i. For test problems of Figure 1, the fully dense row was placed in position $n-1$, whereas for problems of Figure 2, the fully dense row was randomly placed. As expected, the improvement in algorithm performance as a function of number of sweeps before synchronization was more pronounced when the dense row was placed in the next to the last row in the matrix. The reason for this is that for this case all processors except the one processing the dense row have to wait (for the processing of the dense row) more frequently when the number of sweeps is smaller. For the case of the randomly placed dense row, the wait is not as long, because many sparse rows can be processed in each sweep while the dense row is being processed. Hence increasing the number of sweeps before synchronization may not have as pronounced an improvement as in the case of Figure 1. Nevertheless a downward trend in time versus number of sweeps is still noticeable in the graph of Figure 2 for the case of randomly placed dense row.

In Figures 3 and 4 we give results for test problems similar to those of Figures 1 and 2 except that the matrix M is not strictly diagonally dominant, thus $M_{ii} =$

$0.4 \sum_{i \neq j} |M_{ij}|$ for all i. The results for these problems are similar to those for the diagonally dominant test cases of Figures 1 and 2.

We make the following conclusions regarding our computational results:

(a) Improvement of as much as 29% in computation time can be achieved by increasing the number of sweeps before processors are synchronized.

(b) The Dynamic Asynchronous SOR Algorithm 2.1 is sufficiently robust to converge even for problems with matrices which are not diagonally dominant.

(c) Average efficiency of the parallelization on speedup (that is actual speedup divided by ideal speedup) is in the 68% to 71% range.

References

1. G. M. Baudet: "Asynchronous iterative methods for multiprocessors", Journal of the Association for Computer Machinery 25, 1978, 226-244.

2. D. Chazan & W. Miranker: "Chaotic relaxation", Linear Algebra and Its Applications 2, 1969, 199-222.

3. R. W. Cottle & G. B. Dantzig: "Complementary pivot theory in mathematical programming", Linear Algebra and Its Applications 1, 1968, 103-125.

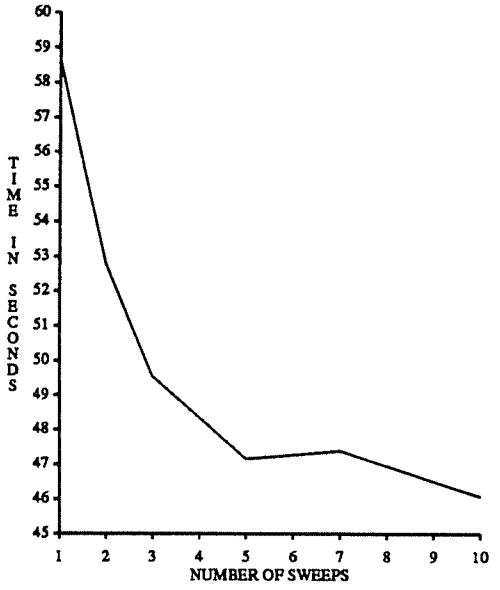
4. R. De Leone & O. L. Mangasarian: "Asynchronous parallel successive overrelaxation for the symmetric linear complementarity problem", Mathematical Programming, Series B, 42, 1988, 347-361.

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

6. J. M. Ortega: "Numerical analysis, a second course", Academic Press, New York 1972.

7. Sequent Computer Systems, Inc.: "Symmetry technical summary", Beaverton, Oregon 97006, 1987.

n = 100,000; d = 0.01%; ω = 0.9



Average time versus number of sweeps of 5 cases below

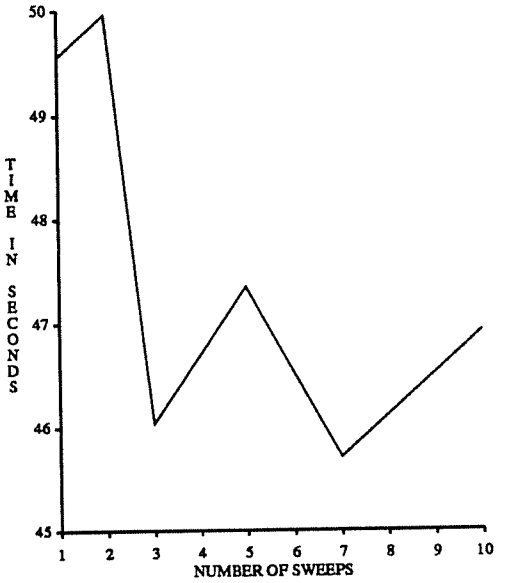
Prob. #	1 proc		10 processors											
	1 sweep		1 sweep		2 sweeps		3 sweeps		5 sweeps		7 sweeps		10 sweeps	
	it	sec	it	sec	it	sec	it	sec	it	sec	it	sec	it	sec
1	24	328.08	24	57.86	25	55.17	25	51.37	24	48.49	24	47.72	24	47.47
2	24	334.12	24	67.28	25	59.81	25	54.73	24	50.57	24	50.25	24	49.15
3	23	316.69	23	64.15	23	54.71	23	50.43	23	48.43	23	48.50	23	46.90
4	23	307.50	23	55.47	23	49.70	23	46.06	23	46.33	23	45.84	23	45.90
5	21	275.28	21	48.68	21	44.61	22	45.10	21	41.97	22	44.64	21	40.94
ave.	23.0	312.33	23.0	58.69	23.4	52.80	23.6	49.54	23.0	47.16	23.2	47.39	23.0	46.07

Row number of dense row = n–1

M_{ii} =1.001 $\sum_{i \neq j} | M_{ij} |$

Figure 1: Test results for the k-Sweep Dynamic Asynchronous Algorithm 2.1: Strictly diagonally dominant matrix with dense row in position n–1.

n = 100,000; d = 0.01%; $\omega = 0.9$



Average time versus number of sweeps of 5 cases below

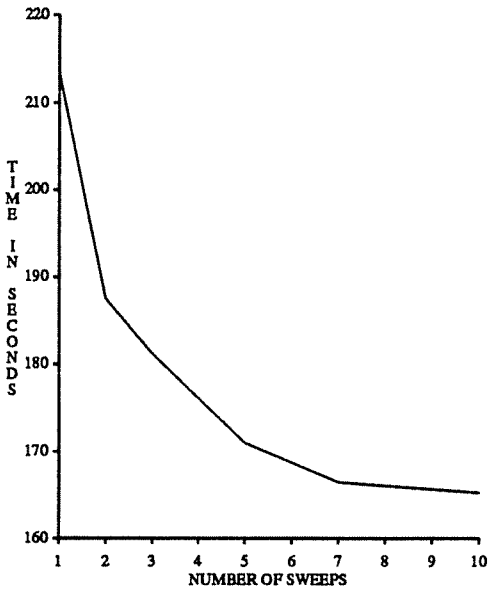
Prob. #	1 proc 1 sweep		10 processors											
			1 sweep		2 sweeps		3 sweeps		5 sweeps		7 sweeps		10 sweeps	
	it	sec	it	sec	it	sec	it	sec	it	sec	it	sec	it	sec
1	28	400.49	28	60.55	29	60.47	25	49.28	28	56.13	24	47.56	28	55.64
2	22	299.27	22	41.66	23	43.94	22	41.63	22	41.95	22	43.13	22	41.80
3	24	333.60	24	47.75	25	49.35	25	47.24	24	46.70	24	46.52	24	46.73
4	24	326.04	24	54.18	25	52.44	25	49.43	24	48.08	24	46.91	24	47.17
5	23	310.50	23	43.68	23	43.62	23	42.63	23	43.87	23	44.44	23	43.31
ave.	24.2	333.94	24.2	49.56	25.0	49.96	24.2	46.04	24.2	47.35	23.4	45.71	24.2	46.93

Row number of dense row randomly generated

$M_{ii} = 1.001 \sum_{i \neq j} | M_{ij} |$

Figure 2: Test results for the k-Sweep Dynamic Asynchronous Algorithm 2.1: Strictly diagonally dominant matrix with dense row randomly placed.

n = 100,000; d = 0.01%; ω = 0.9



Average time versus number of sweeps of 5 cases below

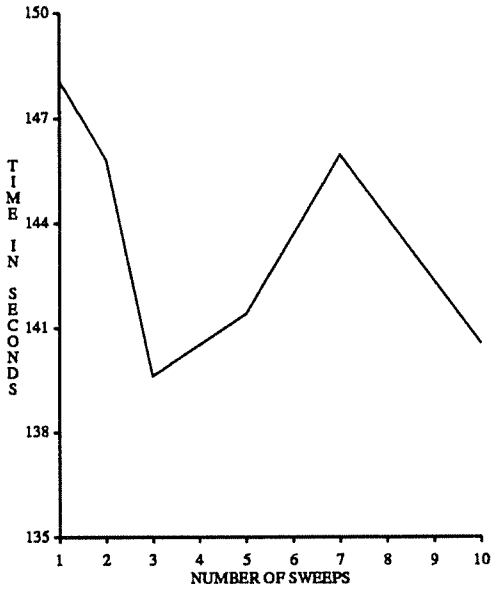
Prob. #	1 proc 1 sweep		10 processors											
			1 sweep		2 sweeps		3 sweeps		5 sweeps		7 sweeps		10 sweeps	
	it	sec	it	sec	it	sec	it	sec	it	sec	it	sec	it	sec
1	112	1514.89	112	307.62	113	260.98	115	248.26	112	229.80	113	223.44	112	220.23
2	82	1087.55	82	225.07	83	191.05	86	187.30	82	167.01	82	162.38	82	160.49
3	80	1002.72	80	179.57	81	165.25	82	166.35	81	155.52	85	160.42	81	151.82
4	81	1020.30	81	183.12	81	165.30	76	152.39	81	155.53	76	146.25	81	152.05
5	75	975.19	75	172.69	75	155.10	76	151.70	76	147.03	71	139.98	75	141.79
ave.	86.0	1120.13	86.0	213.61	86.6	187.51	87.0	181.20	86.4	170.98	85.4	166.49	86.2	165.28

Row number of dense row = n–1

$$M_{ii} = 0.4 \sum_{i \neq j} | M_{ij} |$$

Figure 3: Test results for the k-Sweep Dynamic Asynchronous Algorithm 2.1: Matrix not diagonally dominant matrix with dense row in position n–1.

n = 100,000; d = 0.01%; $\omega = 0.9$



Average time versus number of sweeps of 5 cases below

Prob. #	1 proc 1 sweep		10 processors											
			1 sweep		2 sweeps		3 sweeps		5 sweeps		7 sweeps		10 sweeps	
	it	sec	it	sec	it	sec	it	sec	it	sec	it	sec	it	sec
1	71	960.88	71	149.73	71	142.97	67	130.11	71	137.09	68	131.26	71	135.76
2	76	997.02	76	140.27	77	141.68	73	133.69	76	140.01	80	147.19	76	140.20
3	84	1114.30	84	161.87	85	160.27	79	148.99	84	157.08	80	148.53	84	155.78
4	71	935.03	70	151.65	73	145.82	76	146.55	71	136.69	85	158.97	71	135.14
5	74	981.99	74	136.87	75	138.27	76	138.80	74	136.18	78	143.80	74	135.93
ave.	75.2	997.84	75.0	148.08	76.2	145.82	74.2	139.63	75.2	141.41	78.2	145.95	75.2	140.56

Row number of dense row randomly generated

$$M_{ii} = 0.4 \sum_{i \neq j} | M_{ij} |$$

Figure 4: Test results for the k-Sweep Dynamic Asynchronous Algorithm 2.1: Matrix not diagonally dominant matrix with dense row randomly placed.