CENTER FOR
PARALLEL OPTIMIZATION

INTERIOR DUAL PROXIMAL POINT ALGORITHM USING PRECONDITIONED CONJUGATE GRADIENT

by

Rudy Setiono

Computer Sciences Technical Report #951

July 1990

Interior Dual Proximal Point Algorithm Using Preconditioned Conjugate Gradient

Rudy Setiono*

July 1990

Abstract

Serial and parallel implementations of the interior dual proximal point algorithm for the solution of large linear programs are described. A preconditioned conjugate gradient method is used to solve the linear system of equations that arises at each interior point iteration. Numerical results for a set of multicommodity network flow problems are given. For larger problems preconditioned conjugate gradient method outperforms direct methods of solution. In fact it is impossible to handle very large problems by direct methods.

1 Introduction

The interior dual proximal point algorithm using preconditioned conjugate gradient we are proposing here is based on the interior dual proximal point (IDPP) algorithm for solving linear programs, which is described in detail in [Setiono, 1989]. We begin by summarizing the IDPP algorithm briefly here.

We consider the linear program

$$\min_x cx \text{ s.t. } Ax = b,\ 0 \leq x \leq d \tag{1}$$

*Computer Sciences Department, University of Wisconsin, 1210 West Dayton Street, Madison, WI 53706. Research supported by National Science Foundation Grants DCR-8521228 and CCR-8723091 and Air Force Office of Scientific Research Grants AFOSR-86-0172 and AFOSR-89-0410

which is equivalent to

$$\min_{x,y} cx \tag{2}$$

subject to

$$\begin{pmatrix} A & 0 \\ I & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ d \end{pmatrix}$$
$$x, y \geq 0$$

This is a problem with $m + n$ constraints in $2n$ variables. The dual of the linear program (1) is

$$\min_{u,v} bu + dv \tag{3}$$

subject to

$$\begin{aligned} A^t u + v &\leq c \\ v &\leq 0 \end{aligned}$$

The primal proximal point minimization problems corresponding to the primal linear program (2) is

$$\min_{x,y} cx + \frac{\epsilon^i}{2} \left\| x - x^i \right\|^2 + \frac{\epsilon^i}{2} \left\| y - y^i \right\|^2 \tag{4}$$

subject to

$$\begin{pmatrix} A & 0 \\ I & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ d \end{pmatrix}$$
$$x, y \geq 0$$

and its dual [Mangasarian, 1969] is

$$\min_{(w,z)\geq 0,u,v} \frac{1}{2} \left\| A^t u + v + w - c + \epsilon^i x^i \right\|^2 + \frac{1}{2} \left\| v + z + \epsilon^i y^i \right\|^2 - \epsilon^i bu - \epsilon^i dv \tag{5}$$

The barrier penalty minimization problem associated with the dual problem (5) is

$$\begin{aligned} \min_{u,v,w,z} G(u,v,w,z) := \frac{1}{2} \left\| A^t u + v + w - c + \epsilon^i x^i \right\|^2 &+ \frac{1}{2} \left\| v + z + \epsilon^i y^i \right\|^2 \\ -\epsilon^i bu - \epsilon^i dv - \gamma^i \sum_{j=1}^{n} \log w_j &- \eta^i \sum_{j=1}^{n} \log z_j \end{aligned} \tag{6}$$

where $\gamma^i > 0$ and $\eta^i > 0$.

We do not attempt to solve the above logarithmic penalty problem exactly. To get the next iterate, one Newton step with an appropriate stepsize to maintain the positivity of the variables is taken.

The optimality condition for the problem (6) is

$$\begin{aligned}\nabla G(u,v,w,z) &:= \begin{pmatrix} \nabla_u G(u,v,w,z) \\ \nabla_v G(u,v,w,z) \\ \nabla_w G(u,v,w,z) \\ \nabla_z G(u,v,w,z) \end{pmatrix} \\ &:= \begin{pmatrix} A\left(A^t u + v + w - c + \epsilon^i x^i\right) - \epsilon^i b \\ A^t u + 2v + w - c + \epsilon^i x^i + z + \epsilon^i y^i - \epsilon^i d \\ A^t u + v + w - c + \epsilon^i x^i - \gamma^i W^{-1} e \\ v + z + \epsilon^i y^i - \eta^i Z^{-1} e \end{pmatrix} = 0\end{aligned}$$

where $W = diag(w)$ and $Z = diag(z)$.

Let (u^i, v^i, w^i, z^i) be the current iterate. The linearization of the gradient function $\nabla G(u,v,w,z)$ around the point (u^i, v^i, w^i, z^i) yields the following system of linear equations

$$\begin{pmatrix} AA^t & A & A & 0 \\ A^t & 2I & I & I \\ A^t & I & I + \gamma^i (W^i)^{-2} & 0 \\ 0 & I & 0 & I + \eta^i (Z^i)^{-2} \end{pmatrix} \begin{pmatrix} u \\ v \\ w \\ z \end{pmatrix} = \begin{pmatrix} Ac + \epsilon^i b - \epsilon^i A x^i \\ c + \epsilon^i d - \epsilon^i x^i - \epsilon^i y^i \\ c + 2\gamma^i (W^i)^{-1} e - \epsilon^i x^i \\ 2\eta^i (Z^i)^{-1} e - \epsilon^i y^i \end{pmatrix} \tag{7}$$

For ease of notation, define the diagonal matrices

$$\begin{aligned} D_1 &:= \left(I + \gamma^i (W^i)^{-2}\right)^{-1} \\ D_2 &:= \left(I + \eta^i (Z^i)^{-2}\right)^{-1} \\ and\ E &:= (2I - D_1 - D_2)^{-1} \end{aligned}$$

and vectors

$$\begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix} := \begin{pmatrix} Ac + \epsilon^i b - \epsilon^i A x^i \\ c + \epsilon^i d - \epsilon^i x^i - \epsilon^i y^i \\ c + 2\gamma^i (W^i)^{-1} e - \epsilon^i x^i \\ 2\eta^i (Z^i)^{-1} e - \epsilon^i y^i \end{pmatrix}$$

We can obtain the solution of the $(m + 3n)$ dimensional system (7) by first solving the m dimensional linear system

$$A(I - D_1)(I - E(I - D_1))A^t u =$$

$$h_1 - AD_1h_3 - A(I - D_1)E(h_2 - D_1h_3 - D_2h_4) \tag{8}$$

for u. The other 3 unknowns can then be computed as follows

$$\begin{aligned} v &= E\left(h_2 - D_1h_3 - D_2h_4 - (I - D_1)A^t u\right) \\ w &= -D_1\left(A^t u + v - h_3\right) \\ z &= -D_2(v - h_4) \end{aligned}$$

If we let $(\overline{u}^i, \overline{v}^i, \overline{w}^i, \overline{z}^i)$ be the solution of the linear system (7), the new iterate is then determined as follows

$$\begin{aligned} u^{i+1} &= \overline{u}^i \\ v^{i+1} &= \overline{v}^i \\ w^{i+1} &= w^i + \lambda(\overline{w}^i - w^i) \\ z^{i+1} &= z^i + \lambda(\overline{z}^i - z^i) \\ x^{i+1} &= x^i + \frac{1}{\epsilon^i}\left(A^t\overline{u}^i + \overline{v}^i + \overline{w}^i - c\right) \\ y^{i+1} &= y^i + \frac{1}{\epsilon^i}\left(\overline{v}^i + \overline{z}^i\right) \end{aligned}$$

where λ is a stepsize that is computed to ensure the positivity of the variables w and z. The algorithm is then repeated with decreased values of parameters ϵ, γ and η.

In [Setiono, 1989] we reported the numerical results from our implementation of the IDPP algorithm for small and medium sized linear programs. For these problems, a direct pivotal method such as that provided by the Yale Sparse Matrix Package (YSMP) [Eisenstat et al, 1977 & 1982] has been proven to be a very effective method for solving (8) as our computational results indicated. We recall that YSMP computes the lower unit triangular matrix L and a positive diagonal matrix D such that $PAGA^tP^t = LDL^t$. The matrix P is a permutation matrix designed to reduce the number of fill-ins in L. For linear problems with tens of thousands of constraints however, the number of fill-ins in the factor matrix L can be overwhelming even

when matrix A is relatively sparse. For this reason, we propose an alternative approach for solving the linear system (8) for very large but sparse problems. This approach is the iterative preconditioned conjugate gradient method. We will describe the method and the preconditioning matrix that we use in Section 2. Computational results for large multicommodity network flow problems are presented for serial implementation in Section 3 and for parallel implementation on the Sequent Symmetry S81 in Section 4. We also describe in Section 4 how the structure of the constraint matrix of these large multicommodity network flow problems allows us to solve these problems using parallel architecture and thereby obtaining a speedup efficiency in the range of 75% to 96%. In Section 5 a brief summary of the paper is given.

2 Preconditioned Conjugate Gradient Method

The conjugate gradient method is often used to compute the solution of the linear system of equations

$$Mx = b \tag{9}$$

where M is an m *by* m symmetric positive definite matrix. The conjugate gradient method is better suited than the direct methods for solving the linear system when M is large and/or dense, since the factorization done by the direct method may produce a matrix that is much more dense than M.

When all computations are performed with exact arithmetic, it is well known that the conjugate gradient method converges in k iterations, where $k \leq m$ is the number of distinct eigenvalues of M [Gill et al, 1981]. However, in practice the method may take many more iterations to find the solution of the linear system because of rounding errors. Our approach for reducing the iteration number is to find a nonsingular symmetric matrix C such that the matrix $\hat{M} = CMC^t$ is better conditioned than M and to apply the conjugate gradient method to the transformed linear system

$$\hat{M}\hat{x} = \hat{b} \tag{10}$$

where $\hat{x} = (C^t)^{-1}x$ and $\hat{b} = Cb$. If we define the preconditioning matrix $W = (C^tC)^{-1}$, we have the so-called Preconditioned Conjugate Gradient Method [Ortega, 1988 page 179]

Preconditioned Conjugate Gradient for solving linear systems: Mx = b

- **Choose** x^0
- **Compute** $r^0 = b - Mx^0$
- **Solve** $W\tilde{r}^0 = r^0$ **for** $\tilde{r}^0$
- **Let** $p^0 = \tilde{r}^0$
- **Set** $k = 0$
- (*) $\alpha^k = - < \tilde{r}^k, r^k > / < p^k, Mp^k >$
  - $x^{k+1} = x^k + \alpha^k p^k$
  - $r^{k+1} = r^k + \alpha^k Mp^k$
  - **If** $\|r^{k+1}\| \leq \epsilon$, **stop**
  - **Else Solve** $W\tilde{r}^{k+1} = r^{k+1}$
  - $\beta^k = - < \tilde{r}^{k+1}, r^{k+1} > / < \tilde{r}^k, r^k >$
  - $p^{k+1} = \tilde{r}^{k+1} + \beta^k p^k$
  - $k = k + 1$
  - **Go to (*)**

In each iteration of the PCG method, we need to solve the linear system

$$W\tilde{r} = r \tag{11}$$

for $\tilde{r}$. It is very crucial that this step be done efficiently. The simplest case is when W is the identity matrix. In this case the PCG reduces to ordinary conjugate gradient method and we have not done anything to improve the condition of the matrix M. On the other hand if W is chosen such that $W = LL^t$ where L is the Cholesky factor of M, then the PCG method will converge in just one step and the method reduces to a direct method.

Our aim is to find a preconditioning matrix such that is the linear system (11) is easy to solve and that the matrix $\hat{M}$ is better conditioned that M. With these goals in mind, we decided to use the incomplete Cholesky factor

of M for our preconditioning matrix. If we denote the incomplete Cholesky factor of M by $\hat{L}$, then the preconditioning matrix W is defined as $W = \hat{L}\hat{L}^t$. The matrix $\hat{L}$ can be computed in a similar fashion as the complete Cholesky factor of M, except that if $M_{i,j} = 0$, we force $\hat{L}_{i,j} = 0$. More specifically, given a symmetric positive definite matrix M the following algorithm overwrites the lower half of M by its incomplete Cholesky factor.

- For $i = 1, 2, \ldots m$ do
  - $M_{i,i} = \left(M_{i,i} - \sum_{k=1}^{i-1} M_{i,k}^2\right)^{\frac{1}{2}}$
  - For $j = i+1, \ldots m$ do
    if $M_{j,i} \neq 0$ then
    $$M_{j,i} = \left(M_{j,i} - \sum_{k=1}^{i-1} M_{j,k}M_{i,k}\right)/M_{i,i}$$

The m square roots operations that are done above can be avoided if we compute

$$W = \overline{L}\,\overline{D}\,\overline{L}^t \tag{12}$$

where $\overline{L}$ is a unit lower triangular matrix and $\overline{D}$ is a diagonal matrix. The modified algorithm is

- For $i = 1, 2, \ldots m$ do
  - $M_{i,i} = M_{i,i} - \sum_{k=1}^{i-1} M_{i,k}^2 M_{k,k}$
  - For $j = i+1, \ldots m$ do
    if $M_{j,i} \neq 0$ then
    $$M_{j,i} = \left(M_{j,i} - \sum_{k=1}^{i-1} M_{j,k}M_{i,k}M_{k,k}\right)/M_{i,i}$$

The main advantage of using incomplete Cholesky factors for preconditioning is that these factors preserve the sparsity structure of the original matrix M. This is very crucial when M is large but relatively sparse. Such matrices arise from our implementation of the interior point algorithms for large multicommodity network flow problems. In the next section we shall describe our computational experience with the Preconditioned Conjugate Gradient method for solving these multicommodity network flow problems.

3 Numerical results

The Interior Dual Proximal Point (IDPP) algorithm using Preconditioned Conjugate Gradient method was tested on the Patient Distribution System (PDS) problems. These are multicommodity network flow problems that were developed by the staff of the Military Airlift Command at Scott Air Force Base. The sizes of these problems are listed in Table 1.

We recall that the linear system of equations that needs to be solved at each interior point step is of the form

$$Mu = A(I - D_1)\,(I - E(I - D1))\,A^t u = z$$

The matrix M was reordered using the reversed Cuthill-McKee ordering [Duff et al, 1986] to reduce its bandwidth before its incomplete Cholesky factor was computed. Let us denote the reversed Cuthill-McKee reordering matrix by P. This reordering was done once at the start of the IDPP. At each iteration of the IDPP, the incomplete Cholesky factor of the matrix PMP^t is computed and the conjugate gradient iterations were executed until the norm of the residual $\|Mu - z\|$ is less than $5.d - 08$.

All computations were done on the Astronautics ZS-1 pipeline vector machine [Smith, 1989]. The code was written in FORTRAN and all computations were done in double precision. Table 2 shows the objective values and the primal infeasibility of the solutions obtained by IDPP using Preconditioned Conjugate Gradient. The optimal objective values match to at least 8 digit for problems upto PDS-10 when compared to the objective values obtained by the IDPP algorithm using YSMP [Setiono, 1989]. The primal infeasibility of the solution was computed as follows.

$$\|Ax_+ - b, (-x)_+, (x_+ - d)_+\|_\infty$$

where z_+ denotes the orthogonal projection of z onto the nonnegative orthant.

Using the PCG method, we were able to solve the PDS-20 problem that we could not solve using the direct method. The solution times for the direct (YSMP) and PCG methods are listed in Table 3. We observe that as the problem dimensions increase, the PCG method actually outperforms YSMP.

The number of conjugate gradient iterations increases as the interior point iterates converge to the solution of the linear program and the matrix M

becomes increasingly ill-conditioned. The third column of Table 4 shows the average number of conjugate gradient iterations PCG took to solve the linear systems. The second column of the table shows the dimensions of these linear systems which are the number of constraints of the linear programs. Note that even as the problem dimension increases, the number of conjugate gradient iterations are always between 2 to 3 % of the matrix dimension.

4 Parallel implementation of IDPP-PCG

Most of the steps in the Preconditioned Conjugate Gradient method for solving linear systems of equations and in the Interior Dual Proximal Point method for solving linear programs consist of vector-vector addition or matrix-vector multiplication. Both of these operations are relatively easy to be executed in parallel. To achieve good utilization of a parallel machine therefore, it is very important that the following two steps:

1. the computations of the preconditioning matrix $\overline{L}$ and
2. the solutions of the system of equations $\overline{L}\,\overline{D}\,\overline{L}^t y = d$

can be done efficiently in parallel. The preconditioning matrix $\overline{L}$ needs to be computed at each IDPP iteration and the solutions of $\overline{L}\,\overline{D}\,\overline{L}^t y = d$ needs to be computed at each PCG iteration.

The structure of the matrix ADA^t for the PDS problems (Figure 1) leads us to 2 approaches for computing the preconditioning matrix $\overline{L}$. The first approach is the **Block Diagonal Preconditioning**. The preconditioning matrix $\overline{L}$ is obtained by computing the incomplete Cholesky factors of the block diagonal part of the matrix ADA^t. Since the block diagonal part of this matrix consists of 12 blocks, its incomplete Cholesky factors can be computed in parallel using 2,3,4,6 or 12 processors.

If we divided the lower half of the matrix ADA^t into 12 vertical blocks, the Cholesky factors of the first 11 blocks can be computed simultaneously using 11 processors. When this is completed, the processors can then work on the remaining one block. This is our second approach, which we called the **Total Matrix Preconditioning**.

Our computational experience using these two approaches are taken from a shared memory multiprocessors, the Sequent Symmetry S81. Table 5 shows

the results for the solutions of the PDS problems using the IDPP algorithm with the Block Diagonal Preconditioned Conjugate Gradient. The results for these problems using the IDPP algorithm with the Total Matrix Preconditioned Conjugate Gradient are shown in Table 6. In these 2 tables we list the time to solve the problems, the average number of conjugate gradient iterations and the efficiency when more than 1 processors are employed. The efficiency for n processors $E(n)$ is computed as follows:

$$E(n) = (T(1)/(n * T(n)) * 100\%$$

where $T(1)$ and $T(n)$ are the CPU times required by 1 processor and n processors respectively.

5 Summary

We have presented numerical results from our serial and parallel implementation of the Interior Dual Proximal Point algorithm using the Preconditioned Conjugate Gradient method. At each iteration of the IDPP algorithm, the Newton direction was obtained by solving a system of linear equations of the from $ADA^t x = b$. . Solving this linear systems by the preconditioned conjugate gradient has enabled us to solve large linear programs which we could not solve by direct pivotal methods. For linear programs with block structured constraints, such as the Patient Distribution System problems, the solution of the linear system was obtained after relatively low number of conjugate gradient iterations. The preconditioning matrix used to speedup the conjugate gradient method was the incomplete Cholesky factor of the matrix ADA^t. This factor was easy to compute and did not introduce any fill-in.

For the parallel implementation of the Interior Dual Proximal Point Algorithm, we considered 2 different preconditioning approaches. The first was the Block Diagonal Preconditioned Conjugate Gradient method. In this approach, we computed the incomplete Cholesky factor of the block diagonal part of the matrix ADA^t. Using this approach, 2,3,4,6 and 12 processors were used and the efficiencies that we obtained ranged from 82% to 96%. The second approach was the Total Matrix Preconditioned Conjugate Gradient. With this approach the incomplete Cholesky factor of the entire matrix

ADA^t was computed. When 11 processors were used, the efficiencies obtained using this approach ranged from 75% for PDS-1 problem to 84% for the PDS-6 problem. We note that although the efficiency of the parallelization of the Block Diagonal PCG was better than that of the Total Matrix PCG, the latter solved the problems in shorter times.

6 References

[1] Duff, I.S., Erisman, A.M. and Reid, J.K. (1986). *Direct methods for sparse matrices.* Clarendon Press, Oxford, England.

[2] Eisenstat, S.C., Gursky, M.C., Schultz. M.H. and Sherman, A.H. (1982). Yale Sparse Matrix Package I: The symmetric codes. *International Journal for Numerical Methods in Engineering.* Vol. 18, pp. 1145-1151.

[3] Eisenstat, S.C., Gursky, M.C., Schultz, M.H. and Sherman, A.H. (1977). Yale Sparse Matrix Package I: The symmetric codes. Research Report # 112, Yale University, CT.

[4] Gill, P.E., Murray, W., and Wright, M.H. (1981). *Practical Optimization.* Academic Press, Orlando, FL.

[5] Mangasarian, O.L. (1969). *Nonlinear Programming.* Mc Graw-Hill, New York.

[6] Ortega, J.M. (1988) Introduction to parallel and vector solution of linear systems. Plenum Press, New York.

[7] Setiono, R. (1989) An interior dual proximal point algorithm for linear programs. Technical Report #879, Computer Sciences Department, University of Wisconsin-Madison, WI.

[8] Smith, J.E. (1989) Dynamic Instruction Scheduling and the Astronautics ZS-1. *Computer*, July, pp. 21-35.

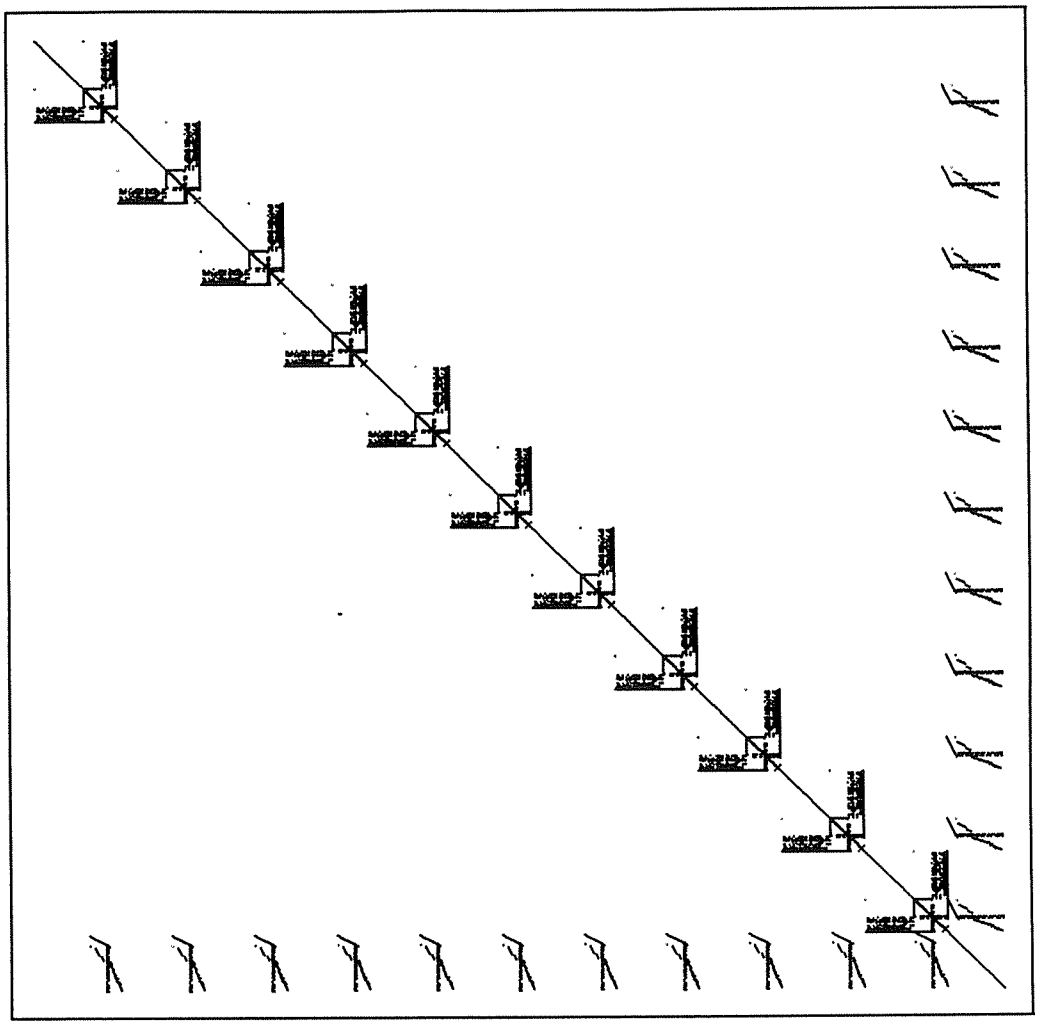

Figure 1: **Nonzero structure of** ADA^t

- Problem Name : **PDS-1**
- Matrix Dimension : **1473 by 1473**
- Nonzero Elements : **11307**
- Density : **0.52 %**

Problem	Rows	Columns	Up. Bound	Non-Zeros
1	1473	3816	605	8139
2	2953	7716	2134	16571
3	4593	12590	3839	27099
4	6372	18615	5555	39944
5	8099	24192	7370	51978
6	9881	29351	9240	63220
10	16558	49932	16148	107605
20	33798	108175	34888	232647

Table 1: **Patient Distribution System (PDS) Problem Data**

Problem	Objective Value	Primal Inf.
PDS-1 .	2.9083929918 e+10	8.2E-02
PDS-2	2.8857862390 e+10	1.0E-01
PDS-3	2.8597374836 e+10	4.4E-01
PDS-4	2.8341928705 e+10	2.8E-01
PDS-5	2.8054052684 e+10	6.1E-02
PDS-6	2.7761037757 e+10	9.0E-02
PDS-10	2.6727094978 e+10	1.1E-01
PDS-20	2.3821660488 e+10	3.6E-02

Table 2: **PDS : PCG Results**

Problem	YSMP	PCG
PDS-1	30 sec.	1 min. 13 sec.
PDS-2	1 min. 43 sec.	4 min. 10 sec.
PDS-3	4 min. 39 sec.	8 min. 52 sec.
PDS-4	12 min. 35 sec.	18 min. 15 sec.
PDS-5	35 min. 25 sec.	40 min. 7 sec.
PDS-6	51 min. 56 sec.	57 min. 43 sec.
PDS-10	4 hr. 41 min.	3 hr. 35 min.
PDS-20	n.a.	25 hr. 28 min.

Table 3: **Solution times: YSMP vs PCG (Astronautics ZS-1)**

Problem	Dimensions	CG iter.
PDS-1	1473	40
PDS-2	2953	69
PDS-3	4593	97
PDS-4	6372	118
PDS-5	8099	139
PDS-6	9881	224
PDS-10	16558	414
PDS-20	33798	984

Table 4: **Average number of conjugate gradient iterations**

Proc.	PDS-1	PDS-2	PDS-3	PDS-4	PDS-5	PDS-6
1	1485.83	5365.56	11887.44	27087.78	40175.13	79447.81
	67 iter.	113 iter.	148 iter.	204 iter.	226 iter.	373 iter.
2	774.98	2800.74	6222.30	14183.66	20976.07	41788.48
	95.86 %	95.79 %	95.53 %	95.49 %	95.76 %	95.06 %
3	526.15	1910.08	4213.21	9583.08	14169.17	28149.25
	94.13 %	93.63 %	94.05 %	94.22 %	94.51 %	94.08 %
4	400.46	1451.72	3211.90	7314.09	10804.08	21412.05
	92.76 %	92.40 %	92.53 %	92.59 %	92.96 %	92.76 %
6	274.90	991.90	2211.22	5071.91	7373.30	14653.03
	90.08 %	90.16 %	89.60 %	89.01 %	90.82 %	90.37 %
12	149.88	529.88	1175.05	2670.55	3966.54	7856.12
	82.61 %	84.38 %	84.30 %	84.51 %	84.40 %	84.27 %

Table 5: **Time (seconds), iteration and efficiency of IDPP using Block Diagonal PCG (Sequent S81)**

Proc.	PDS-1	PDS-2	PDS-3	PDS-4	PDS-5	PDS-6
1	1064.17	3575.88	9059.76	20147.97	29227.59	52936.08
	41 iter.	67 iter.	98 iter.	133 iter.	143 iter.	266 iter.
11	128.90	399.44	1012.66	2181.79	3152.11	5709.33
	75.05 %	81.38%	81.33%	83.94%	84.30%	84.29 %

Table 6: **Time (seconds), iteration and efficiency of IDPP using Total Matrix PCG (Sequent S81)**