INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 36, 255–271 (1993)

AN APPLICATION OF PRECONDITIONED CONJUGATE GRADIENTS TO RELATIVE PLACEMENT IN CHIP DESIGN

CHRISTIAN KREDLER AND CHRISTIAN ZILLOBER

Institut für Angewandte Mathematik und Statistik, Technische Universität München, Arcisstr 21, D-8000 München 2, Germany

FRANK JOHANNES AND GEORG SIGL

Lehrstuhl für Rechnergestütztes Entwerfen, Technische Universität München, Arcisstr 21, D-8000 München 2, Germany

SUMMARY

In distance geometry problems and many other applications, we are faced with the optimization of high-dimensional quadratic functions subject to linear equality constraints. A new approach is presented that projects the constraints, preserving sparsity properties of the original quadratic form such that well-known preconditioning techniques for the conjugate gradient method remain applicable. Very-large-scale cell placement problems in chip design have been solved successfully with diagonal and incomplete Cholesky preconditioning. Numerical results produced by a FORTRAN 77 program illustrate the good behaviour of the algorithm.

1. INTRODUCTION

Quadratic programming with either non-negativity or linear equality constraints is a standard model type in quite a large number of applications: we mention only discretization of certain optimal control problems or many kinds of distance geometry approaches. In this paper, we discuss preconditioned conjugate gradient (CG) methods for large-scale quadratic optimization problems with linear equality constraints. Our new algorithm has been developed for an initial global placement phase in chip design. The presented CG method enables a simultaneous treatment of all cells and is applicable to general-cell and standard-cell circuits of several thousand cells and nets. Now it is no longer necessary to partition the chip area and to solve stand-alone smaller optimization problems. This formerly used local strategy frequently neglects the connections between the subproblems, leading to models which are often unrealistic.

In Section 2 we formulate the constrained problem. Section 3 discussses the various techniques of preconditioning. Further, a new CG method for preconditioning in an affine subspace is presented. Section 4 contains the model formulation for the chip placement problem and some numerical results.

2. QUADRATIC OPTIMIZATION SUBJECT TO LINEAR EQUALITY CONSTRAINTS

Consider the quadratic program

$$\min_{\mathbf{x}} \{\tfrac{1}{2}\,\mathbf{x}^{\mathrm{T}}\mathbf{C}\mathbf{x} - \mathbf{d}^{\mathrm{T}}\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}\} \tag{1}$$

0029–5981/93/020255–17$13.50
© 1993 by John Wiley & Sons, Ltd.

Received 5 June 1990
Revised 1 April 1992

where

$$\mathbf{C} \in \mathbb{R}^{n,n}, \quad \mathbf{C} > \mathbf{0} \text{ (positive definite)}, \quad \mathbf{d}, \mathbf{x} \in \mathbb{R}^n$$

$$\mathbf{A} \in \mathbb{R}^{r,n}, \quad \operatorname{rank}(\mathbf{A}) = r < n, \quad \mathbf{b} \in \mathbb{R}^r$$

and the superscript T denotes transposition.

For the feasible region there exist the following equivalent representations:

$$\mathbf{S} \overset{\text{def}}{=} \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}\} = \{\mathbf{x} = \mathbf{x}_0 + \mathbf{W}\mathbf{z} \mid \mathbf{z} \in \mathbb{R}^{n-r}\} \tag{2}$$

where $\mathbf{x}_0 \in \mathbb{R}^n$ is a feasible basic solution ($\mathbf{A}\mathbf{x}_0 = \mathbf{b}$) and $\mathbf{W} \in \mathbb{R}^{n,n-r}$, $\mathbf{A}\mathbf{W} = \mathbf{0}$. $\mathbf{W}$ can be assumed to have orthonormal columns; hence, $\mathbf{W}^T\mathbf{W} = \mathbf{I}_{n-r}$. For $\mathbf{C} > \mathbf{0}$ the original problem (1) as well as the following transformed problem (3) have a unique minimum:

$$\begin{aligned} &\min_{\mathbf{x}} \{\tfrac{1}{2}(\mathbf{x}_0 + \mathbf{W}\mathbf{z})^T\mathbf{C}(\mathbf{x}_0 + \mathbf{W}\mathbf{z}) - \mathbf{d}^T(\mathbf{x}_0 + \mathbf{W}\mathbf{z})\} \\ &= \min_{\mathbf{z}} \{\gamma + \tfrac{1}{2}\mathbf{z}^T\mathbf{W}^T\mathbf{C}\mathbf{W}\mathbf{z} + (\mathbf{x}_0^T\mathbf{C} - \mathbf{d}^T)\mathbf{W}\mathbf{z}\} \end{aligned} \tag{3}$$

where $\gamma = \frac{1}{2}\mathbf{x}_0^T\mathbf{C}\mathbf{x}_0 - \mathbf{d}^T\mathbf{x}_0$.

The n-dimensional problem with r constraints has been reduced to a $(n - r)$-dimensional problem without constraints.

The well-known classical CG algorithm for the quadratic programs (1) or (3) is not stated explicitly. It might be taken from Reference 1 (p. 523) or as a special case of algorithm 3.4 with $\mathbf{W} = \mathbf{I}$ and $\mathbf{D} = \mathbf{I}$.

For problem (3), this algorithm computes a vector $\mathbf{z} \in \mathbb{R}^{n-r}$ which solves the transformed optimization problem. The corresponding x-co-ordinates can then be computed by the relation $\mathbf{x} = \mathbf{x}_0 + \mathbf{W}\mathbf{z}$.

The matrix $\mathbf{W}$ is not needed explicitly. We use only products of the form

(a) $\mathbf{W}\mathbf{z}$, $\mathbf{z} \in \mathbb{R}^{n-r}$,

(b) $\mathbf{W}^T\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^n$.

For that purpose we apply r Householder transformations to the matrix $\mathbf{A}^T$:

$$\underbrace{\mathbf{P}_r \cdot \mathbf{P}_{r-1} \cdot \ldots \cdot \mathbf{P}_1}_{\overset{\text{def}}{=} \mathbf{P}^T} \cdot \mathbf{A}^T = \begin{bmatrix} \mathbf{R} \\ \hline \mathbf{0} \end{bmatrix} \tag{4}$$

where $\mathbf{R} \in \mathbb{R}^{r,r}$, $\mathbf{P}_i \in \mathbb{R}^{n,n}$, $i = 1, \ldots, r$.

Partitioning $\mathbf{P} = (\mathbf{Q} | \mathbf{W})$, $\mathbf{Q} \in \mathbb{R}^{n,r}$, $\mathbf{W} \in \mathbb{R}^{n,n-r}$, $\mathbf{Q}$ and $\mathbf{W}$ represent mappings into span($\mathbf{A}^T$) and the orthogonal complement span$(\mathbf{A}^T)^\perp$, respectively. Hence, (a) and (b) can be computed from $\mathbf{P}\mathbf{x}$ and $\mathbf{P}^T\mathbf{x}$, respectively, by partitioning and supplementing with zeros as follows:

$$\mathbf{z} \in \mathbb{R}^{n-r} \Rightarrow \tilde{\mathbf{z}} \overset{\text{def}}{=} \begin{Bmatrix} \mathbf{0} \\ \hline \mathbf{z} \end{Bmatrix}\begin{matrix} r \\ n-r \end{matrix} \in \mathbb{R}^n \Rightarrow \mathbf{P}\tilde{\mathbf{z}} = \mathbf{W}\mathbf{z} \tag{5}$$

$$\mathbf{x} \in \mathbb{R}^n \Rightarrow \mathbf{P}^T\mathbf{x} = \begin{Bmatrix} \mathbf{Q}^T\mathbf{x} \\ \hline \mathbf{W}^T\mathbf{x} \end{Bmatrix}\begin{matrix} r \\ n-r \end{matrix} \tag{6}$$

Since $\mathbf{P}$ is a product of Householder matrices, we have $\mathbf{P}^T\mathbf{P} = \mathbf{I}_n$ and

$$\mathbf{P}^T\mathbf{P} = \left[\frac{\mathbf{Q}^T}{\mathbf{W}^T}\right]\cdot[\mathbf{Q}|\mathbf{W}] = \left[\begin{array}{c|c} \mathbf{Q}^T\mathbf{Q} & \mathbf{Q}^T\mathbf{W} \\ \hline \mathbf{W}^T\mathbf{Q} & \mathbf{W}^T\mathbf{W} \end{array}\right] = \left[\begin{array}{c|c} \mathbf{I}_r & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{I}_{n-r} \end{array}\right] = \mathbf{I}_n \tag{7}$$

we obtain, among others, $\mathbf{W}^T\mathbf{W} = \mathbf{I}_{n-r}$.

Remark. The efficiency of our method depends heavily on the costs for calculating $\mathbf{W}$ and for evaluating a projection $\mathbf{W}\mathbf{W}^T\mathbf{x}$. Generally, the Householder transformations (4) may be too expensive. However, in the application of Section 4, the rows of $\mathbf{A}$ are pairwise orthogonal and (4) can be specified trivially.

3. PRECONDITIONING

The technique of preconditioning is well-known and is described in many textbooks. Nevertheless, we give a short summary of the classical approach which is modified later to solve our specific problem in the linear subspace.

3.1. Preconditioning without constraints

A bound for the speed of convergence in the classical CG algorithm depends on the quotient of the largest and smallest eigenvalue of the (positive-definite) matrix $\mathbf{C}$ (or $\mathbf{W}^T\mathbf{C}\mathbf{W}$); see Reference 1 (p. 525):

$$\|\mathbf{z} - \mathbf{z}_k\|_{\mathbf{C}} \leqslant 2\|\mathbf{z} - \mathbf{z}_0\|_{\mathbf{C}}\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \tag{8}$$

where $\|\mathbf{z}\|_{\mathbf{C}}^2 \overset{\text{def}}{=} \mathbf{z}^T\mathbf{C}\mathbf{z}$, $\kappa = \text{cond}(\mathbf{C})$ is the condition number of $\mathbf{C}$ (or $\mathbf{W}^T\mathbf{C}\mathbf{W}$) and k denotes the iteration index. For the rest of this paper, $\|\cdot\|$ means the Euclidean norm.

Lemma 3.1. Assume $\mathbf{C} \in \mathbb{R}^{n,n}$, $\mathbf{C} > \mathbf{0}$ and $\mathbf{W} \in \mathbb{R}^{n,q}$, $q \leqslant n$ to have orthonormal columns. Then

(i) $\|\mathbf{W}\mathbf{z}\| = \|\mathbf{z}\|$

(ii) $\lambda_{\max}(\mathbf{W}^T\mathbf{C}\mathbf{W}) \leqslant \lambda_{\max}(\mathbf{C})$ (maximal eigenvalue)

(iii) $\lambda_{\min}(\mathbf{W}^T\mathbf{C}\mathbf{W}) \geqslant \lambda_{\min}(\mathbf{C})$ (smallest eigenvalue)

(iv) $\text{cond}(\mathbf{W}^T\mathbf{C}\mathbf{W}) \leqslant \text{cond}(\mathbf{C})$ (condition number)

Proof.

(i) clear.

(ii) $\lambda_{\max}(\mathbf{W}^T\mathbf{C}\mathbf{W}) = \max\limits_{\|\mathbf{z}\|=1} \mathbf{z}^T\mathbf{W}^T\mathbf{C}\mathbf{W}\mathbf{z} \leqslant \max\limits_{\|\mathbf{y}\|=1} \mathbf{y}^T\mathbf{C}\mathbf{y} = \lambda_{\max}(\mathbf{C})$

(iii) $\lambda_{\min}(\mathbf{W}^T\mathbf{C}\mathbf{W}) \geqslant \lambda_{\min}(\mathbf{C})$ analogously

(iv) $\text{cond}(\mathbf{W}^T\mathbf{C}\mathbf{W}) = \dfrac{\lambda_{\max}(\mathbf{W}^T\mathbf{C}\mathbf{W})}{\lambda_{\min}(\mathbf{W}^T\mathbf{C}\mathbf{W})} \leqslant \dfrac{\lambda_{\max}(\mathbf{C})}{\lambda_{\min}(\mathbf{C})} = \text{cond}(\mathbf{C})$

Corollary 3.2. The bound (8) on the speed of convergence of the CG method is not reduced when working in certain subspaces.

Given a starting point $\mathbf{z}_0$, convergence can be improved by scaling the matrix $\mathbf{C}$ such that κ is close to unity. The optimal convergence rate is attained at $\kappa = 1$ (only one iteration needed).

Obviously, we can expect an acceleration of the convergence by preconditioning the matrix **C**. We have to maintain the following two conditions:

(c) $\bar{\mathbf{C}} \overset{\text{def}}{=} \mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}$ has a better condition number than **C** ($\mathbf{D}^{-T}$ means $(\mathbf{D}^{-1})^T$ throughout).

(d) The system $\mathbf{D}\mathbf{D}^T\mathbf{t} = \mathbf{y}$ can be solved easily.

These requirements are consequences of the following well-known transformations. For clarity of presentation we relax the constraints for a moment and set

$$\mathbf{W} \overset{\text{def}}{=} \mathbf{I}, \quad \mathbf{x}_0 \overset{\text{def}}{=} \mathbf{0} \quad \text{and} \quad \mathbf{x} \overset{\text{def}}{=} \mathbf{z}, \quad \bar{\mathbf{x}} \overset{\text{def}}{=} \mathbf{D}^T\mathbf{x}$$

$$\min_{\mathbf{x}} \{\tfrac{1}{2}\mathbf{x}^T\mathbf{C}\mathbf{x} - \mathbf{d}^T\mathbf{x}\} = \min_{\mathbf{x}} \{\tfrac{1}{2}\mathbf{x}^T\mathbf{D}\mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}\mathbf{D}^T\mathbf{x} - \mathbf{d}^T\mathbf{D}^{-T}\mathbf{D}^T\mathbf{x}\} \quad (\text{if } \mathbf{D}^{-1} \text{ exists})$$

$$= \min_{\bar{\mathbf{x}}} \{\tfrac{1}{2}\bar{\mathbf{x}}^T\mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}\bar{\mathbf{x}} - \mathbf{d}^T\mathbf{D}^{-T}\bar{\mathbf{x}}\} = \min_{\bar{\mathbf{x}}} \{\tfrac{1}{2}\bar{\mathbf{x}}^T\bar{\mathbf{C}}\bar{\mathbf{x}} - \bar{\mathbf{d}}^T\bar{\mathbf{x}}\} \tag{9}$$

The algorithm for the modified program (9) can be taken from Reference 1 (p. 529) or as a special case of algorithm 3.4. with $\mathbf{W} = \mathbf{I}$.

Remark. An ideal choice would be $\mathbf{D} = \sqrt{\mathbf{C}}$ or $\mathbf{D} = \mathbf{L}$ (for $\mathbf{C} = \mathbf{L}\mathbf{L}^T$) leading to $\bar{\mathbf{C}} = \mathbf{I}$ with optimal convergence rate. In this case, we would have to solve $\mathbf{C}\mathbf{t} = \mathbf{y}$ as auxiliary problem in the algorithm mentioned above. But this is exactly the original one. In the literature essentially the following two kinds of preconditioning matrices **D** are discussed:

(i) diagonal matrices,
(ii) triangular matrices or products of triangular matrices.

Both kinds satisfy requirement (d), but we have to get along with losses in (c).

In References 1–4 many proposals for the choice of **D** are discussed. Each of these methods reduces the amount of computation when we can utilize special features of the matrix **C** (e.g. tridiagonality, block tridiagonality and so on).

In the application discussed below none of these features is apparent. Hence, we restricted ourselves to the following alternatives:

(i) $\mathbf{D} = \text{diag}(\sqrt{c_{11}}, \ldots, \sqrt{c_{nn}})$
(ii) Incomplete Cholesky factorization of the matrix **C** (see e.g. Reference 1, p. 530)

(i) is simple to implement, but gives only a small improvement in our application because the diagonal entries c_{ii} do not vary widely in size here.

The incomplete factorization generates approximate Cholesky factors for **C** ($\mathbf{C} \approx \mathbf{L}\mathbf{L}^T$), where additionally

$$l_{ij} = 0 \text{ if } c_{ij} = 0$$

These factors can be computed with numerical stability (and $l_{ii} > 0$, $i = 1, \ldots, n$) in the application of Section 4, because **C** is diagonally dominant and positive-definite; see 4.2 and Reference 5. In the full case, the complexity of the Cholesky decomposition is $O(n^3/6)$. In our application (18), the incomplete one needs $O(2n + 3m)$ operations when m is the number of non-zero entries in **C**. Hence, the method is very fast in this case.

The convergence of the method with preconditioning is guaranteed when we ensure that the preconditioner **D** has full rank, because the system matrix remains positive-definite. But until now there is no success in proving that the convergence speed is actually accelerated. Numerical experience indicates that we have a considerable speed-up factor when using the preconditioning method (ii) compared with the pure CG method for the original problem.

In Section 4.4 we give comparisons of the implemented methods and illustrate the numerical behaviour by three examples with the dimensions $n = 108, 445$ and 1133. We focused on the following two criteria, each with and without preconditioning.

- The *number of iterations* to attain a certain decrease of the residual
- The *overall computing time* including the calculation of the preconditioning matrix.

3.2. *Preconditioning for additional linear equality constraints*

We study again problem (1)

$$\min_{\mathbf{x}} \{\tfrac{1}{2}\mathbf{x}^T\mathbf{C}\mathbf{x} - \mathbf{d}^T\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

According to Section 2 with a feasible basic solution $\mathbf{x}_0$ ($\mathbf{A}\mathbf{x}_0 = \mathbf{b}$), this problem is equivalent to

$$\min_{\mathbf{z}} \{\tfrac{1}{2}\mathbf{z}^T\mathbf{W}^T\mathbf{C}\mathbf{W}\mathbf{z} + (\mathbf{x}_0^T\mathbf{C} - \mathbf{d}^T)\mathbf{W}\mathbf{z}\} \tag{10}$$

Preconditioning the transformed system matrix $\mathbf{W}^T\mathbf{C}\mathbf{W}$ is not appropriate because $\mathbf{W}^T\mathbf{C}\mathbf{W}$ is usually more dense than $\mathbf{C}$. The new aspect given in this paper is to *precondition* $\mathbf{C}$ *itself* and to *transform the constraints afterwards.*

We summarize the essential assumptions which have been made until now. After the preparation of some further quantities, our *projected* CG *algorithm* can be formulated.

Assumption 3.3.

(a) $\mathbf{D} \in \mathbb{R}^{n,n}$ is a non-singular matrix,
(b) $\mathbf{W} \in \mathbb{R}^{n,n-r}$ and $\mathbf{W}^T\mathbf{W} = \mathbf{I}_{n-r}$,
(c) for a given vector $\mathbf{u} \in \mathbb{R}^n$ the quantities $\mathbf{D}^{-T}\mathbf{D}^{-1}\mathbf{u}$ and $\mathbf{W}\mathbf{W}^T\mathbf{u}$ can be computed efficiently,
(d) $\mathbf{C} > \mathbf{0}$ (positive-definite).

Following (3) we have

$$\begin{aligned}
&\min_{\mathbf{z}} \{\tfrac{1}{2}(\mathbf{x}_0^T + \mathbf{z}^T\mathbf{W}^T)\mathbf{D}\mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}\mathbf{D}^T(\mathbf{x}_0 + \mathbf{W}\mathbf{z}) - \mathbf{d}^T(\mathbf{x}_0 + \mathbf{W}\mathbf{z})\} \\
&= \min_{\mathbf{z}} \{\tfrac{1}{2}\mathbf{x}_0^T\mathbf{D}\mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}\mathbf{D}^T\mathbf{x}_0 + \mathbf{x}_0^T\mathbf{D}\mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}\mathbf{D}^T\mathbf{W}\mathbf{z} \\
&\quad + \tfrac{1}{2}\mathbf{z}^T\mathbf{W}^T\mathbf{D}\mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}\mathbf{D}^T\mathbf{W}\mathbf{z} - \mathbf{d}^T(\mathbf{x}_0 + \mathbf{W}\mathbf{z})\} \\
&= \min_{\mathbf{z}} \{\gamma + \tfrac{1}{2}\mathbf{z}^T\mathbf{W}^T\mathbf{D}\mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}\mathbf{D}^T\mathbf{W}\mathbf{z} - (\mathbf{d}^T\mathbf{W} - \mathbf{x}_0^T\mathbf{D}\mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T}\mathbf{D}^T\mathbf{W})\mathbf{z}\} \\
&= \min_{\mathbf{z}} \{\gamma + \tfrac{1}{2}\mathbf{z}^T\mathbf{W}^T\mathbf{D}\bar{\mathbf{C}}\mathbf{D}^T\mathbf{W}\mathbf{z} - (\mathbf{d}^T\mathbf{W} - \mathbf{x}_0^T\mathbf{D}\bar{\mathbf{C}}\mathbf{D}^T\mathbf{W})\mathbf{z}\}
\end{aligned}$$

where

$$\bar{\mathbf{C}} = \mathbf{D}^{-1}\mathbf{C}\mathbf{D}^{-T} \quad \text{and} \quad \gamma = \tfrac{1}{2}\mathbf{x}_0^T\mathbf{C}\mathbf{x}_0 - \mathbf{d}^T\mathbf{x}_0$$

When preconditioning, we are forced to work in the $\mathbf{x}$-space of dimension n whereas the problem itself is actually of dimension $n - r$. The substitution $\tilde{\mathbf{x}} \overset{\text{def}}{=} \mathbf{D}^T\mathbf{W}\mathbf{z} \in \mathbb{R}^n$ or, equivalently, $\mathbf{z} = \mathbf{W}^T\mathbf{D}^{-T}\tilde{\mathbf{x}} \in \mathbb{R}^{n-r}$ is only needed for formal purposes. With

$$\tilde{\mathbf{C}} \overset{\text{def}}{=} \mathbf{D}^{-1}\mathbf{W}\mathbf{W}^T\mathbf{D}\bar{\mathbf{C}}\mathbf{D}^T\mathbf{W}\mathbf{W}^T\mathbf{D}^{-T} \quad \text{and} \quad \tilde{\mathbf{d}}^T \overset{\text{def}}{=} \mathbf{d}^T\mathbf{W}\mathbf{W}^T\mathbf{D}^{-T} - \mathbf{x}_0^T\mathbf{D}\bar{\mathbf{C}}\mathbf{D}^T\mathbf{W}\mathbf{W}^T\mathbf{D}^{-T}$$

we obtain

$$\min_{\tilde{\mathbf{x}}} \{\tfrac{1}{2}\tilde{\mathbf{x}}^T\mathbf{D}^{-1}\mathbf{W}\mathbf{W}^T\mathbf{D}\bar{\mathbf{C}}\mathbf{D}^T\mathbf{W}\mathbf{W}^T\mathbf{D}^{-T}\tilde{\mathbf{x}} - (\mathbf{d}^T\mathbf{W}\mathbf{W}^T\mathbf{D}^{-T} - \mathbf{x}_0^T\mathbf{D}\bar{\mathbf{C}}\mathbf{D}^T\mathbf{W}\mathbf{W}^T\mathbf{D}^{-T})\tilde{\mathbf{x}}\}$$

$$= \min_{\tilde{\mathbf{x}}} \{\tfrac{1}{2}\tilde{\mathbf{x}}^T\tilde{\mathbf{C}}\tilde{\mathbf{x}} - \tilde{\mathbf{d}}^T\tilde{\mathbf{x}}\}$$

This problem has exactly the form of the standard minimization problem without constraints and, in principle, the algorithm of Section 2 is applicable when $\mathbf{D}$ and $\mathbf{D}^{-T}$ are given diagonal scaling matrices. For triangular $\mathbf{D}$, we avoid the explicit use of $\mathbf{D}^{-T}$ in the formulation of our CG algorithm. The derivation is explained below.

CG algorithm 3.4. (*Preconditioning in a linear subspace*)

```
(Initialization): Choose starting point $\mathbf{z}_0 \in \mathbb{R}^n$, $0 < \varepsilon < 1$ (accuracy of solution);
        Let $\mathbf{r}_0 := \mathbf{W}\mathbf{W}^T(\mathbf{d} - \mathbf{C}\mathbf{x}_0 - \mathbf{C}\mathbf{W}\mathbf{z}_0)$  (initial residual)
            $\delta := \|\mathbf{W}\mathbf{W}^T\mathbf{d}\|$
(Iteration): **For** $k = 0, 1, 2, \ldots$ **do**
        (Preconditioning)
          Solve $\mathbf{D}\mathbf{D}^T\mathbf{t}_k = \mathbf{r}_k$;
        $\rho_k := \mathbf{r}_k^T\mathbf{t}_k$  (weighted norm of residual);
        **if** $(\sqrt{\rho_k} \leqslant \varepsilon \max\{1, \delta\})$ **STOP**;
        (Projected search direction)
        $k = 0$:  $\mathbf{s}_0 := \mathbf{t}_0$;
        $k > 0$:  $\beta_k := \frac{\rho_k}{\rho_{k-1}} = \frac{\mathbf{t}_k^T\mathbf{r}_k}{\mathbf{t}_{k-1}^T\mathbf{r}_{k-1}}$
                  $\mathbf{s}_k := \mathbf{t}_k + \beta_k\mathbf{s}_{k-1}$;
        $\mathbf{p}_k := \mathbf{W}\mathbf{W}^T\mathbf{s}_k$;  $\mathbf{w}_k := \mathbf{W}\mathbf{W}^T\mathbf{C}\mathbf{p}_k$;
        (Optimal stepsize)
        $\alpha_k := \frac{\mathbf{t}_k^T\mathbf{r}_k}{\mathbf{s}_k^T\mathbf{w}_k}$;
        (Update)
        $\mathbf{z}_{k+1} := \mathbf{z}_k + \alpha_k\mathbf{p}_k$;
        $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{w}_k$  (new residual);
        $k := k + 1$;
    **end** (Iteration).
```

This algorithm can be derived as follows:

Given a starting point $\tilde{\mathbf{x}}_0 = \mathbf{D}^T\mathbf{W}\mathbf{z}_0$ ($\tilde{\mathbf{x}}_0 = \mathbf{0}$ ($\Leftrightarrow \mathbf{z}_0 = \mathbf{0}$) is a simple choice) the quantities of the preceding CG algorithm can be computed as follows:

Residual

$$\begin{aligned}\tilde{\mathbf{r}}_0 &= \mathbf{D}^{-1}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{d} - \mathbf{D}^{-1}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{C}\mathbf{x}_0 - \mathbf{D}^{-1}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{C}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{x}}_0\\ &= \mathbf{D}^{-1}\mathbf{W}\mathbf{W}^{\mathrm{T}}(\mathbf{d} - \mathbf{C}\mathbf{x}_0 - \mathbf{C}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{x}}_0)\end{aligned}$$

As the computation of $\tilde{\mathbf{r}}_0$ is expensive, we better work with

$$\mathbf{D}\tilde{\mathbf{r}}_0 = \mathbf{W}\mathbf{W}^{\mathrm{T}}(\mathbf{d} - \mathbf{C}\mathbf{x}_0 - \mathbf{C}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{x}}_0) = \mathbf{W}\mathbf{W}^{\mathrm{T}}(\mathbf{d} - \mathbf{C}\mathbf{x}_0 - \mathbf{C}\mathbf{W}\mathbf{z}_0)$$

For computation of $\mathbf{D}\tilde{\mathbf{r}}_k$, for $k > 0$, see below.

Next we need for $k \geqslant 0$, an auxiliary vector $\mathbf{t}_k \overset{\mathrm{def}}{=} \mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{r}}_k$ which is given by solving

$$\mathbf{D}\mathbf{D}^{\mathrm{T}}\mathbf{t}_k = \mathbf{D}\tilde{\mathbf{r}}_k.$$

As

$$\tilde{\mathbf{r}}_k^{\mathrm{T}}\tilde{\mathbf{r}}_k = \tilde{\mathbf{r}}_k^{\mathrm{T}}\mathbf{D}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{r}}_k = (\tilde{\mathbf{r}}_k^{\mathrm{T}}\mathbf{D}^{\mathrm{T}})(\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{r}}_k) = (\tilde{\mathbf{r}}_k^{\mathrm{T}}\mathbf{D}^{\mathrm{T}})\mathbf{t}_k$$

all quantities for the computation of the update factor β_k are available, namely

$$\beta_k = \frac{\tilde{\mathbf{r}}_k^{\mathrm{T}}\tilde{\mathbf{r}}_k}{\tilde{\mathbf{r}}_{k-1}^{\mathrm{T}}\tilde{\mathbf{r}}_{k-1}}, \quad (k \geqslant 1).$$

Now we describe how to obtain the quantities corresponding to the search direction $\mathbf{s}_k$, the stepsize α_k and the projection $\mathbf{z}_k$ in the subspace.

$$\begin{aligned}&\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_0 = \mathbf{t}_0 && (k = 0)\\ &\tilde{\mathbf{s}}_k = \tilde{\mathbf{r}}_k + \beta_k\tilde{\mathbf{s}}_{k-1} \;\Rightarrow\; \mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k = \mathbf{t}_k + \beta_k\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_{k-1} && (k \geqslant 1)\\ &\alpha_k = \frac{\tilde{\mathbf{r}}_k^{\mathrm{T}}\tilde{\mathbf{r}}_k}{\tilde{\mathbf{s}}_k^{\mathrm{T}}\tilde{\mathbf{C}}\tilde{\mathbf{s}}_k} && (k \geqslant 1)\end{aligned}$$

The numerator was derived above; the denominator can be calculated as follows:

$$\tilde{\mathbf{s}}_k^{\mathrm{T}}\tilde{\mathbf{C}}\tilde{\mathbf{s}}_k = \tilde{\mathbf{s}}_k^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{C}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k = (\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k)^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{C}\mathbf{W}\mathbf{W}^{\mathrm{T}}(\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k)$$

$$\begin{aligned}\tilde{\mathbf{x}}_{k+1} = \tilde{\mathbf{x}}_k + \alpha_k\tilde{\mathbf{s}}_k \;&\Rightarrow\; \mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{x}}_{k+1} = \mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{x}}_k + \alpha_k\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k\\ &\Rightarrow\; \mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{x}}_{k+1} = \mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{x}}_k + \alpha_k\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k\\ &\Rightarrow\; \mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k \quad (\mathbf{z} = \mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{x}})\\ &\Rightarrow\; \mathbf{W}\mathbf{z}_{k+1} = \mathbf{W}\mathbf{z}_k + \alpha_k\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k\end{aligned}$$

Now it is clear that we really do not need $\tilde{\mathbf{x}}_k$ explicitly. It can always be substituted by $\mathbf{W}\mathbf{z}_k$. Finally, for the update step we have

$$\begin{aligned}\tilde{\mathbf{r}}_{k+1} &= \tilde{\mathbf{r}}_k - \alpha_k\tilde{\mathbf{C}}\tilde{\mathbf{s}}_k = \tilde{\mathbf{r}}_k - \alpha_k\mathbf{D}^{-1}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{C}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k\\ \Rightarrow \mathbf{D}\tilde{\mathbf{r}}_{k+1} &= \mathbf{D}\tilde{\mathbf{r}}_k - \alpha_k\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{C}\mathbf{W}\mathbf{W}^{\mathrm{T}}(\mathbf{D}^{-\mathrm{T}}\tilde{\mathbf{s}}_k)\end{aligned}$$

Remark. From Tables I–IV we observe that our preconditioning algorithm reduces the number of iterations in the considered non-trivial examples to less than 40%. During the iteration, however, there are two multiplications of the form $\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{x}$. In the original problem we only need one of this type. Since these multiplications become more expensive for an increasing number of constraints the CPU time cannot be reduced by the same factor as the iteration number (see also at the end of Section 4.3).

4. AN APPLICATION: CELL PLACEMENT IN CHIP LAYOUT

The CG method described above was developed for the following high-dimensional problem arising in chip design.

Recent progress in the area of large-scale integration was only possible by using automatic layout tools. These tools should enable a user to design the chip in such a way that the design gets wireable with shortest wire length and minimal area. The background of our approach can be found, for instance, in Reference 6.

The input to placement systems of this type consists of a pin list, that describes the connectivity of modules $\mu \in M$ and signals $\sigma \in S$, a cell library and a description of the chip geometry. A pin list for an introductory sample circuit with seven modules and three signals (a, b, c) is graphically presented in Figure 1. This pin list can be mapped into a bipartite graph $G = (V, E)$ (see Figure 2), where $V = S \cup M$ are the vertices. Signal σ and module μ are incident if and only if signal σ is connected with module μ.

A module μ represents a complex gate, a storage unit, etc. which can be located on the chip area as a rectangle with centre (x_μ, y_μ). Signals are certain wire nets. Following the mathematical model of the bipartite graph, we now view a signal σ as an object concentrated at a point with co-ordinates (x_σ, y_σ). The co-ordinates of the double-framed modulues 3 and 7 are fixed *a priori*. For instance, these may be pads or some very important macro cells with predetermined location. All other modules are movable.

In this paper, we are concerned with the relative placement of the modules on the chip area, see Figures 3 and 6. The result of the well-posed convex quadratic optimization problem defined below gives raw, but reliable information on the optimal module positions.

Remaining cell overlaps are eliminated by a postprocessor (Reference 6). The final geometric placement for our sample circuit is shown in Figure 4.

The main loop of this placement strategy consists of an iteration with quadratic optimization and partitioning steps. They aim at minimal wire length and uniform distribution of the modules over the available placement area, which is a rectangle defined by $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$; see Figure 5.

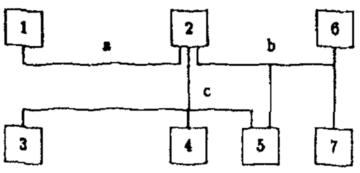

Figure 1. Electric circuit

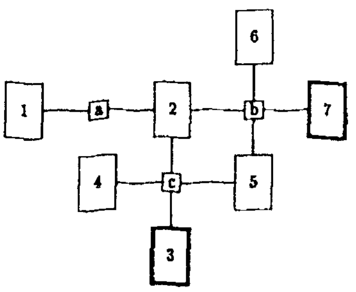

Figure 2. Bipartite graph

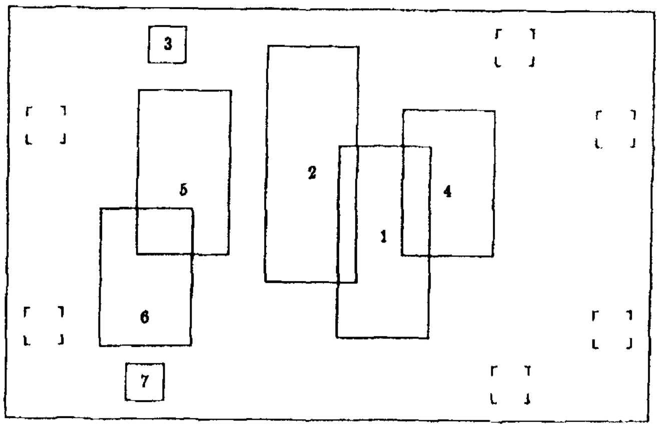

Figure 3. Relative placement

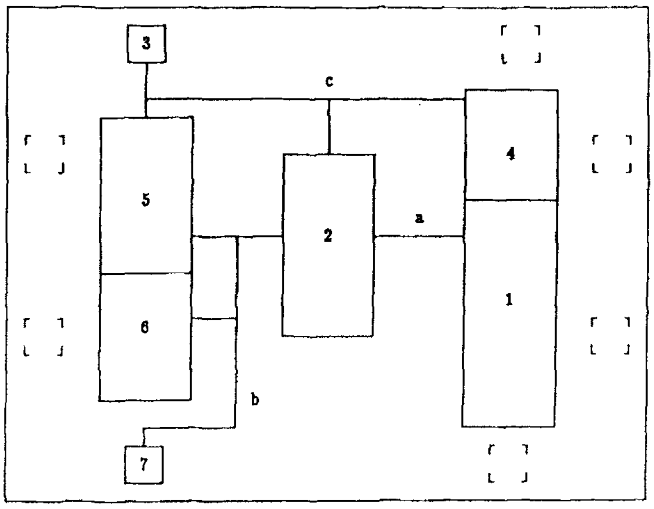

Figure 4. Final physical placement

For another small sample circuit, the results of a sequence of placement steps is shown in Figure 6.

Optimization starts at level 0 with an initial region that comprises the whole chip area and contains all modules to be placed. One constraint requires the centre of gravity of all these modules to be the centre of this region:

$$\begin{aligned} \sum_{\mu\in M} x_\mu/|M| &= x_{\min} + \tfrac{1}{2}(x_{\max} - x_{\min}) \\ \sum_{\mu\in M} y_\mu/|M| &= y_{\min} + \tfrac{1}{2}(y_{\max} - y_{\min}) \end{aligned} \tag{11}$$

where $|M|$ represents the number of entries of set M.

In each partitioning step (levels 1–4 in Figure 6) the module set is further divided and the placement regions are dissected into subregions accordingly, and new constraints are established for the next optimization step. This increasing number of constraints restricts the freedom of movement for the modules and forces the modules to drift away from each other and to move close to the positions of the final placement.

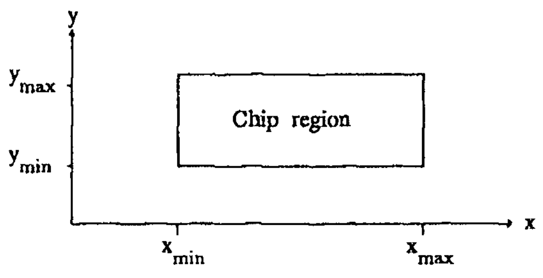

Figure 5.

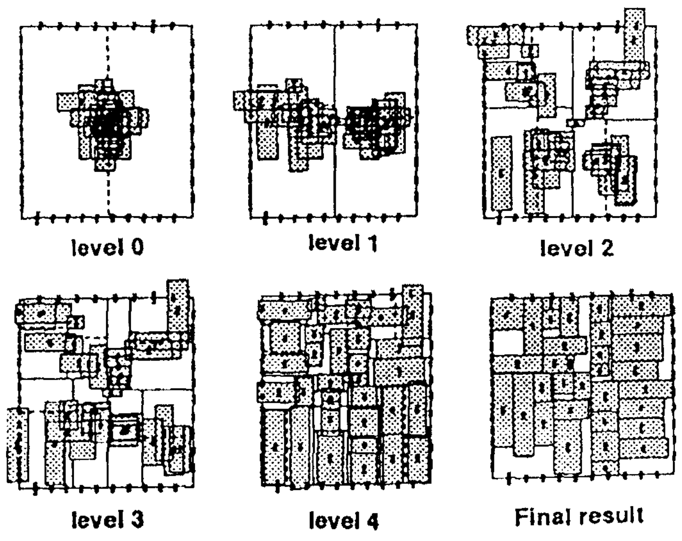

Figure 6. Stepwise placement refinement

The module set M is partitioned into left and right subsets M_l and M_r according to the modules x-co-ordinates of level 0. The single centre of gravity at level 0 is now replaced by two:

$$
\begin{aligned}
\sum_{\mu \in M_l} x_\mu / |M_l| &= x_{\min} + \tfrac{1}{4}(x_{\max} - x_{\min}) \\
\sum_{\mu \in M_l} y_\mu / |M_l| &= y_{\min} + \tfrac{1}{2}(y_{\max} - y_{\min}) \\
\sum_{\mu \in M_r} x_\mu / |M_r| &= x_{\max} - \tfrac{1}{4}(x_{\max} - x_{\min}) \\
\sum_{\mu \in M_r} y_\mu / |M_r| &= y_{\min} + \tfrac{1}{2}(y_{\max} - y_{\min})
\end{aligned}
\tag{12}
$$

Optimization of level 1 results in two module clusters. Each of the two subsets is bipartitioned again with respect to the modules' y-co-ordinates. Thus, at level 2 a placement with four centres of gravity results. This process is continued until each region contains only one module.

4.1. Model formulation

The objective function to be minimized is based on the Euclidean lengths of the signals. The length L_σ of a signal σ is measured by the sum of squared distances from the centres of the modules μ which are connected with signal σ, to the co-ordinate of signal σ (cf. adjacency relation E of graph G; Figure 2):

$$L_\sigma(\mathbf{x}, \mathbf{y}) \overset{\text{def}}{=} \sum_\mu [(x_\mu - x_\sigma)^2 + (y_\mu - y_\sigma)^2] \tag{13}$$

Summation over all signals gives the objective function

$$\phi(\mathbf{x}, \mathbf{y}) \overset{\text{def}}{=} \tfrac{1}{2} \sum_\sigma L_\sigma(\mathbf{x}, \mathbf{y}) = \phi_x(\mathbf{x}) + \phi_y(\mathbf{y}) \tag{14}$$

consisting of two separable positive-semidefinite quadratic forms. So far we have disregarded fixed modules and pads. However, in a realistic design, besides the $n_s = |S|$ signal co-ordinates (x_σ, y_σ) only $n_m < |M|$ module co-ordinates (x_μ, y_μ) are free.

Focusing only on x-variables and introducing known pin co-ordinates $r_{\sigma\mu}$ (relative to the module centre) which indirectly describe the size of the modules, the following objective function is obtained

$$\phi_x(\mathbf{x}) \overset{\text{def}}{=} \tfrac{1}{2} \sum_{\sigma \in S} \sum_{\mu \in M} (x_\mu + r_{\sigma\mu} - x_\sigma)^2 t_{\sigma\mu} \omega_\sigma = \tfrac{1}{2} \mathbf{x}^T \mathbf{C} \mathbf{x} - \mathbf{d}_x^T \mathbf{x} \tag{15}$$

which is quite close to reality. $t_{\sigma\mu}$ are the appropriate 0–1 entries of the binary adjacency matrix for the bipartite graph G (see also Figure 2), and ω_σ are positive weights to privilege the most important signals. The ω_σ can be chosen by the designer. Under weak and realistic requirements on the topology of the fixed modules, $\mathbf{C}$ is positive-definite and diagonally dominant. The bipartite graph G transmits its structure to the symmetric matrix $\mathbf{C}$ which has the so-called red/black partitioning (cf. Reference 7). This means that there are two groups of variables (modules and signals) and these variables are independent of each other within either group. Hence, $\mathbf{C}$ is of the form

$$\mathbf{C} = \left[\begin{array}{c|c} \mathrm{diag} & \mathbf{V}^T \\ \hline \mathbf{V} & \mathrm{diag} \end{array}\right] \begin{array}{c} \\ \end{array} \quad \text{with column blocks } \underbrace{\quad}_{n_s} \; \underbrace{\quad}_{n_m} \tag{16}$$

The diagonal matrix in the upper left is of dimension n_s; that in the bottom on the right is diagonal of dimension n_m. $\mathbf{V}$ has the sparsity structure of the adjacency matrix of G. $\mathbf{C}$ itself is quadratic and has the same dimension

$$n \overset{\text{def}}{=} n_s + n_m < |S| + |M| \tag{17}$$

as the vector $\mathbf{x}$ of all free x-variables.

For the minimization of the quadratic objective function, the broad palette of conjugate gradient techniques with convergence acceleration is available. This approach is sometimes preferred to other competing relative placement algorithms like Mincut and algebraic multigrid (AMG) methods because the designer can easily influence the positioning of specific modules or module groups. An experienced designer will manage this by a skilful tuning of the parameters $r_{\sigma\mu}$ and the signal weights ω_σ. This work must be possible interactively on commonly used CAD workstations. Therefore, a complete relative placement for at least thousand modules must be fast enough, say at most few minutes, on such a workstation.

The red/black partitioning leads to a significant simplification in the computation of the incomplete Cholesky factorization, which was introduced in Section 3.

In the following well-known algorithm

```
For $i = 1, \ldots, n$
    For $j = 1, \ldots, i-1$: $l_{ij} := \frac{1}{l_{jj}} \cdot \left( c_{ij} - \sum_{\nu=1}^{j-1} l_{i\nu} l_{j\nu} \right)$
    $l_{ii} := \left( c_{ii} - \sum_{\nu=1}^{i-1} |l_{i\nu}|^2 \right)^{1/2}$                    (18)
```

the sum $\Sigma\, l_{i\nu} l_{j\nu}$ in the j loop vanishes completely. Note that we let $l_{ij} = 0$ whenever $c_{ij} = 0$. In total, there are $3m$ multiplications and additions, n subtractions and n square roots, where m denotes the number of non-zero off-diagonal entries in $\mathbf{C}$.

4.2. *Structure of linear equality constraints*

At the lth level of partitioning, the placement area is divided into $r = 2^l$ subregions. The centre of gravity constraints on this level is given by

$$\tilde{\mathbf{A}}\mathbf{x} = \mathbf{b}_x \quad \text{and} \quad \tilde{\mathbf{A}}\mathbf{y} = \mathbf{b}_y \tag{19}$$

The matrix $\tilde{\mathbf{A}}$ has r rows and n_m columns. Each row corresponds to a subregion and an entry $a_{ij} = 1$ means that module j belongs to subregion i. Since a module can be attached only to one subregion, there is exactly one non-zero entry in each column. We take advantage of this feature and save an immense amount of numerical computation when transforming the constraint matrix for our algorithm.

The simple example below illustrates the structure for a circuit with $r = 4$ subregions and $n_m = 10$ modules.

$$\tilde{\mathbf{A}} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

4.3. *Projection of constraints*

Now, in each of the described steps one of the following problems is to be solved:

$$\min \{ \tfrac{1}{2}\, \mathbf{x}^T \mathbf{C} \mathbf{x} - \mathbf{d}_x^T \mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}_x \} \tag{21}$$

$$\min \{ \tfrac{1}{2}\, \mathbf{y}^T \mathbf{C} \mathbf{y} - \mathbf{d}_y^T \mathbf{y} \mid \mathbf{A}\mathbf{y} = \mathbf{b}_y \} \tag{22}$$

The feasible region can be designed as follows (it suffices to give the description for the x-co-ordinates; y analogously). The constraints have the special structure

$$\mathbf{A} = r\{[\underbrace{\mathbf{0}}_{n_s} \mid \underbrace{\tilde{\mathbf{A}}}_{n_m}]$$

and are only referring to the module co-ordinates. Hence, decomposing

$$\mathbf{x} = \begin{Bmatrix} \mathbf{x}_s \\ \mathbf{x}_m \end{Bmatrix}$$

we can view the signal variables $\mathbf{x}_s$ as free whereas the module variables $\mathbf{x}_m$ are subject to linear equality constraints. Consequently, the feasible points range in the subspace

$$S \overset{\text{def}}{=} \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} = \mathbf{x}_0 + \mathbf{W}\mathbf{z}\}$$

where

$$\mathbf{x}_0 = \begin{Bmatrix} \mathbf{0} \\ \hline \tilde{\mathbf{x}}_0 \end{Bmatrix}; \quad \tilde{\mathbf{x}}_0 \in \mathbb{R}^{n_m}$$

is a feasible solution of

$$\tilde{\mathbf{A}}\mathbf{x}_m = \mathbf{b}_x \overset{\text{def}}{=} \mathbf{b} \tag{23}$$

Denote the entries of $\tilde{\mathbf{A}}$ by $\tilde{a}_{ij}$. As $\tilde{a}_{ij} \in \{0, 1\}$ and each column of $\tilde{\mathbf{A}}$ contains exactly one 1 a feasible solution $\tilde{\mathbf{x}}_0$ and, hence, $\mathbf{x}_0$ can be obtained readily as

$$\tilde{x}_{0j} = \sum_{i=1}^{r} \tilde{a}_{ij} b_i, \quad j = 1, \ldots, n_m \tag{24}$$

Or, the starting point $\tilde{x}_{0j}$ is the centre of gravity of region i the module j belongs to.

In the CG algorithm (3.4) we need orthogonal projections $\mathbf{W}\mathbf{W}^T\mathbf{x}$ which are computed according to (4) by products $\mathbf{P}^T\mathbf{x}$ and $\mathbf{P}\mathbf{v}$.

The Householder vectors $\mathbf{p}_i$ characterizing the factors $\mathbf{P}_i \overset{\text{def}}{=} \mathbf{I} - 2\mathbf{p}_i\mathbf{p}_i^T$ of $\mathbf{P}$ are of the form

$$\mathbf{p}_i = \frac{\mathbf{a}_i - \|\mathbf{a}_i\|\mathbf{e}_i}{\|\mathbf{a}_i - \|\mathbf{a}_i\|\mathbf{e}_i\|}$$

where $\mathbf{a}_i$ and $\mathbf{e}_i$ $(i = 1, \ldots, r)$ denote the rows of $\mathbf{A}$ and canonical unit vectors, respectively. Apart from one element, the sparsity structure of the Householder vectors is that of $\mathbf{a}_i$. Hence, $\mathbf{P}$ is completely characterized by $O(n_m)$ essential entries and the time for establishing the vectors $\mathbf{p}_i$ $(i = 1, \ldots, r)$ depends not heavily on the constraint number r.

Using the partitioning $\mathbf{P} = (\mathbf{Q}|\mathbf{W})$ and equations (4)–(7), a projection $\mathbf{W}\mathbf{W}^T\mathbf{x}$ in the orthogonal space of span$(\mathbf{A}^T)$ is computed as follows:

$$\mathbf{P}^T\mathbf{x} = \begin{Bmatrix} \mathbf{Q}^T\mathbf{x} \\ \hline \mathbf{W}^T\mathbf{x} \end{Bmatrix} = \begin{Bmatrix} \mathbf{v}_Q \\ \mathbf{v}_W \end{Bmatrix} \tag{25}$$

$$\mathbf{P}\begin{Bmatrix} \mathbf{0} \\ \mathbf{v}_W \end{Bmatrix} = [\mathbf{Q}, \mathbf{W}]\begin{Bmatrix} \mathbf{0} \\ \hline \mathbf{W}^T\mathbf{x} \end{Bmatrix} = \mathbf{W}\mathbf{W}^T\mathbf{x} \tag{26}$$

The CPU times of Section 4.4 depend on the number of constraints. There are two reasons:

(1) The reported times refer to the entire layout algorithm and include a lot of organization amount due to an increasing number of module clusters and, hence, on the number of linear equality constraints.

(2) The projection $\mathbf{WW}^T\mathbf{x}$ called at each CG iteration consists of $2r$ Householder multiplications involving the vectors $\mathbf{p}_i$. As the sparsity structure is exploited completely, the computational amount is less than $O(n_m r)$, but we cannot do the complete projection in a time independent of r (see also a remark at the end of Section 3).

4.4. *Numerical results*

The algorithms were implemented in FORTRAN 77. The CPU times of Tables I–IV refer to a CDC CYBER 995 with machine precision epsmach $= 5 \times 10^{-15}$. The iteration was stopped, when $\sqrt{\rho_k} \leqslant \varepsilon \max\{1, \sqrt{\rho_0}\}$, $\varepsilon = 10^{-14}$ (see algorithm 3.4).

For practical problems, ε between 10^{-7} and 10^{-10} would suffice. Hence, the CPU times reported in this paper will reduce. For instance, in a problem of dimension $n = 1133$ with incomplete Cholesky preconditioning CPU times are 3·38 s for $\varepsilon = 10^{-14}$ and ≈ 2 s for $\varepsilon = 10^{-7}$; see also Table I and the graphs in Figure 7. The three examples of dimension $n = 108$, 445 and 1133 were run at the Leibniz-Rechenzentrum, München.

Table I. Iterations/CPU time (s)

n	Without preconditioning	With diagonal matrix	With incomplete Cholesky fact
108	67/0·167	44/0·132	21/0·092
445	108/0·981	82/0·918	41/0·685
1133	209/4·798	161/4·408	81/3·384

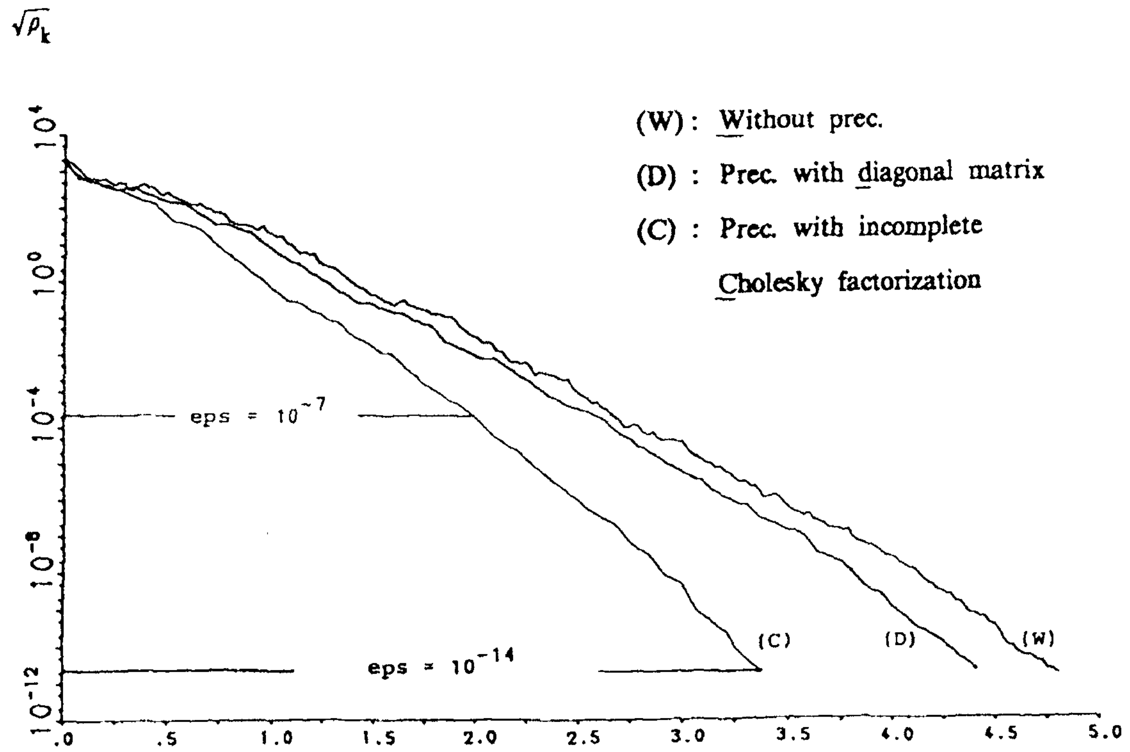

Figure 7. CPU time (s) versus norm of residual. Dimension: $n = 1133$; initial residual: $\sqrt{\rho_0} \approx 2107$; current residual: $\sqrt{\rho_k}$; k increases according to CPU time

4.4.1. The unconstrained problem. Figure 7 shows the CPU time of the CG algorithms depending upon the norm of the residual. We observe that diagonal preconditioning behaves not essentially better than the simple CG algorithm. Incomplete Cholesky preconditioning performs superior throughout (see also Table I).

4.4.2. The constrained problem. First we recall the most important features of the layout algorithm for the constrained problem:

(i) The preconditioning matrix **D** needs to be computed only once at the beginning of the layout algorithm.
(ii) Additional amount in the preconditioning algorithm:
- Two multiplications with projection matrix $\mathbf{W}\mathbf{W}^T$ during one iteration instead of one without preconditioning.
- The auxiliary problem $\mathbf{D}\mathbf{D}^T\mathbf{t} = \mathbf{y}$ to be solved.

Since the computational effort for the multiplication with the projection matrix increases with the number of constraints and finally becomes dominant, it is important to decrease the number of iterations. This happens only in the case of preconditioning with incomplete Cholesky factorization. Therefore, this method remains better for many constraints, too. Concerning CPU time, the method without preconditioning is worse than incomplete Cholesky, but better than preconditioning with a diagonal matrix because here less iterations had been saved (see Tables II–IV).

The numbers in Tables II–IV confirm these assertions. For three methods and various constraint sets we listed number of iterations and CPU seconds

Table II. Iterations/CPU time (s) for $n = 108$, $n_m = 29$

Constraints	Without preconditioning	With diagonal matrix	With incomplete Cholesky factorization
1	63/0·22	42/0·186	21/0·125
2	64/0·234	42/0·2	22/0·135
4	59/0·235	39/0·213	22/0·149
8	57/0·26	37/0·251	22/0·177
11	53/0·267	35/0·27	23/0·208

Table III. Iterations/CPU time (s) for $n = 445$, $n_m = 202$

Constraints	Without preconditioning	With diagonal matrix	With incomplete Cholesky factorization
1	93/1·318	69/1·28	35/0·856
2	92/1·387	66/1·372	34/0·902
4	87/1·481	62/1·539	34/1·038
8	86/1·811	61/2·009	34/1·319
16	86/2·504	63/3·094	35/1·932
32	84/3·755	57/4·639	33/2·88
61	82/6·006	55/7·725	33/4·847
91	72/7·357	49/9·977	31/6·509

Table IV. Iterations/CPU time (s) for $n = 1133$, $n_m = 546$

Constraints	Without preconditioning	With diagonal matrix	With incomplete Cholesky factorization
1	173/ 6·163	134/ 6·463	71/ 4·479
2	177/ 6·808	135/ 7·259	72/ 4·97
4	166/ 7·35	125/ 8·136	67/ 5·403
8	168/ 9·38	124/10·715	70/ 7·249
16	166/13·012	121/15·747	70/10·392
32	166/20·483	119/25·633	71/16·866
64	149/31·606	105/40·661	65/27·023
118	120/43·503	84/57·275	53/37·914
219	96/61·592	66/80·641	42/53·924

5. CONCLUSION

Relative placement problems as described here have already been solved with over-relaxation techniques. The CG algorithm without preconditioning competes with this approach as it does for all large-scale quadratic programs.

For additional linear equality constraints, the behaviour of our projected CG method is superior, because we do not destroy the sparsity structure of the large system matrix. Hence, convergence acceleration is possible by incomplete Cholesky preconditioning and the numerical results document an encouraging speed-up factor (for further details see References 8 and 9). The efficiency of the projection depends on the simple structure of the linear equality constraints which can be expected in many high-dimensional applications.

The approach outlined is well-suited for many distance geometry problems which also arise in chemistry, molecular biology or physics, for instance. But the proposed technique of preconditioning and separate projection extends to other fields as well. As a further example, we mention sequential quadratic programming (SQP) a standard tool for solving (large) non-linear optimization problems. There, we have to solve a sequence of quadratic subproblems, all being of type (1).

ACKNOWLEDGEMENTS

We thank the referees for many valuable suggestions and constructive comments. Christian Zillober was supported by 'Deutsche Forschungsgemeinschaft' under 'Schwerpunkt Anwendungsbezogene Optimierung und Steuerung'.

REFERENCES

1. G. H. Golub and C. F. van Loan, '*Matrix Computations*,' 2nd edn, The Johns Hopkins University Press, Baltimore, 1989.
2. P. Concus, G. H. Golub and G. Meurant, 'Block preconditioning for the conjugate gradient method', *SIAM J. Sci. Stat. Comp.*, **6**, 220–252 (1985).
3. D. R. Fulkerson and P. Wolfe, 'An algorithm for scaling matrices', *SIAM Rev.*, **4**, 142–147 (1962).
4. A. van der Sluis, 'Condition numbers and equilibration of matrices', *Numer. Math.*, **14**, 14–23 (1969).
5. T. A. Manteuffel, 'Shifted incomplete Cholesky factorization', in I. S. Duff and G. W. Stewart (eds.), *Sparse Matrix Proceedings*, SIAM, Philadelphia, 1978, pp. 41–61.
6. J. M. Kleinhans, G. Sigl and F. M. Johannes, 'GORDIAN: A new global optimization/rectangle dissection method for cell placement', in *Proceedings of IEEE International Conference on Computer-Aided Design ICCAD-88*, IEEE Computer Society Press, Washington D.C. 1988, pp. 506–509.

7. K. M. Just and J. M. Kleinhans, 'Zur simultanen Plazierung von Moduln integrierter Schaltungen', *Archiv für Elektronik and Übertragungstechnik*, **39**, 217–224 (1985).
8. F. Johannes, Ch. Kredler, G. Sigl, H. Warsitz and Ch. Zillober, 'Relative placement in chip design by preconditioned quadratic optimization subject to linear equality constraints', DFG-Schwerpunkt; 'Anwendungsbezogene Optimierung und Steuerung', *Report Nr. 158*, Institut für Angewandte Mathematik und Statistik, Technische Universität München, 1989.
9. Ch. Zillober, 'Lösung von großen Plazierungsproblemen mittels präkonditionierter konjugierter Gradienten bei Gleichungsnebenbedingungen', *Diplomarbeit*, Institut für Angewandte Mathematik und Statistik, TU München, 1988.