**Project Report-EECS 527 (Akanksha Jain)**

**Topic:** Implementation and Comparison of different numerical techniques for solving linear system of equations.

Background & Introduction: Solving linear system of equations is common in many optimization problems. More particularly these arise often in force-directed placement algorithms where the objective is to minimize a quadratic function. This reduces to solving a large system of linear equations. Typically, even other non-quadratic optimization objectives for placement, such as the Half Perimeter Wire Length (HPWL) can be approximated by quadratic optimization goals by using the Bound2Bound (B2B) net model. Many different numerical techniques exist which solve such systems. Two commonly used algorithms are the Conjugate Gradient method and the SOR. Due to the large number of iterations required to solve the B2B objective, solving of the linear system may dominate the entire placement algorithm and thus it is highly imperative that the runtime of the linear solver be optimized. This project aims to implement and compare the effectiveness of existing techniques like the CG and SOR, in terms of its runtime.

**Description:**

**Implementation and comparison of CG and SOR method for solving linear equations:**

Overview:

Any placement with a given netlist can be represented with a graph G with nodes V and edges E, with edge weights wij from node I to node j. The aim of global initial placement is to assign x and y co-ordinates to movable nodes and reducing HPWL. The quadratic objective for HPWL can be represented in matrix form which can be by numerical techniques used for linear system of equations.

The linear system of equations for placement can be written as:

Ax = b, where matrix A represents weighted connections between movable and fixed vertices, b represents weighted connections between movable and fixed vertices.

Bound2Bound Model for Wire length estimation:

Each p pin net is decomposed into 2 pin nets where the extreme nodes are connected to each other and to each internal node by following weight: wx,ij = 1/(p − 1)|xi − xj|

Bound2Bound model changes with placement and requires multiple updates for every cycle for placement algorithm.

Conjugate Gradient Method for solving linear system of equations:

The conjugate gradient method is an [algorithm](http://en.wikipedia.org/wiki/Algorithm) for the [numerical solution](http://en.wikipedia.org/wiki/Numerical_solution) of particular [systems of linear equations](http://en.wikipedia.org/wiki/System_of_linear_equations), whose matrix is [symmetric](http://en.wikipedia.org/wiki/Symmetric_matrix) and [positive-definite](http://en.wikipedia.org/wiki/Positive-definite_matrix). The conjugate gradient method is an [iterative method](http://en.wikipedia.org/wiki/Iterative_method), so it can be applied to [sparse](http://en.wikipedia.org/wiki/Sparse_matrix) systems that are large.

The algorithm for Conjugate Gradient method is:

Input: A, b and x0

Output: x

A description...

A description...

A description...

A description...

A description...

A description...

A description...

A description...

A description...

A description...

A description...

A description...

Preconditioners can be used to enhance the spectral properties and convergence rate of CG.

**Preconditioned Conjugate Gradient:**

When system Is large, preconditioners are used to enhance the spectral properties of CG and ensure the fast convergence of the system. There are many types of preconditioer which can be used for this purpose and system reduces to solving: \mathbf{M}^{-1}(\mathbf{A x}-\mathbf{b}) = 0

Where M is preconditioner matrix.

In this project, Jacobi preconditioner is used which is a diagonal matrix consisting of diagonal of Matrix A as its diagonal elements. Inverse of Jacobi Preconditioner is easy to calculate and consists of reciprocal of its diagonal elements.

Algorithm for implementation of Preconditioned Conjugate Gradient:

Input: A, M, b and x0

Output: x

\mathbf{r}_0 := \mathbf{b} - \mathbf{A x}_0

\mathbf{z}_0 := \mathbf{M}^{-1} \mathbf{r}_0

\mathbf{p}_0 := \mathbf{z}_0

k := 0 \, 

**repeat**

\alpha_k := \frac{\mathbf{r}_k^\mathrm{T} \mathbf{z}_k}{\mathbf{p}_k^\mathrm{T} \mathbf{A p}_k}

\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k

\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A p}_k

**if** **r***k*+1 is sufficiently small **then** exit loop **end if**

\mathbf{z}_{k+1} := \mathbf{M}^{-1} \mathbf{r}_{k+1}

\beta_k := \frac{\mathbf{z}_{k+1}^\mathrm{T} \mathbf{r}_{k+1}}{\mathbf{z}_k^\mathrm{T} \mathbf{r}_k}

\mathbf{p}_{k+1} := \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k

k := k + 1 \, 

**end repeat**

**Successive Over Relaxation Method:**

Successive over-relaxation (SOR) is a variant of the [Gauss–Seidel method](http://en.wikipedia.org/wiki/Gauss–Seidel_method) for solving a [linear system of equations](http://en.wikipedia.org/wiki/Linear_system_of_equations), resulting in faster convergence. For a given matrix A, it can be written as: A = D + U + L, where D is a diagonal matrix, U is upper triangular matrix and L is lower triangular matrix. Then , the system of linear equation Ax = b may be written as:

A description...

Since, (*D*+*ωL*) is a triangular matrix, elements of x can be calculated as:

A description...

Choosing 0<ω<1, leads to convergence of the system.

The algorithm for SOR is as follows:

Inputs: A, b, ω  
Output: A description...  
**repeat** until convergence

**for***i***from** 1 **until***n***do**

A description...

**for***j***from** 1 **until***n***do**

**if***j* ≠ *i***then**

A description...

**end if**

**end** (*j*-loop)

A description...

**end** (*i*-loop)

check if convergence is reached

**end** (repeat)

**Results for SOR and Conjugate Gradient Implemented in C++**

**Bechmark: ICCAD 2012 Superblue18**

**Source : Natarajan Viswanathan, Charles Alpert, Cliff Sze, Zhuo Li, Yaoguang Wei, "ICCAD-2012 CAD Contest in Design Hierarchy Aware Routability-Driven Placement and Benchmark Suite," In Proc. ICCAD, pp. 345-348, 2012**

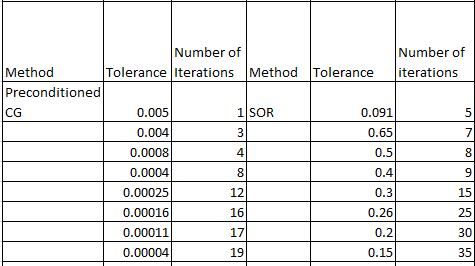
**Benchmark Size: 442k (#)**

**Total Vertices (nodes) of graph: 483452**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Numerical Method** | **HPWL** | **Time(minutes) (for one co-ordinate)** | **Number of iterations** |
| **SOR** | **1.15 \* 108 after going for 5 iterations of improvement** | **22.5 minutes ( for 5 iterations for HPWL)** | **50(max)** |
| **CG(without preconditioner)** | **1.44 \* 108 after going for 5 iterations of improvement** | **14.5 minutes ( for 5 iterations for HPWL)** | **35** |
| **CG(with Jacobi preconditioner)** | **1.44 \* 108 after going for 5 iterations of improvement** | **8 minutes ( for 5 iterations for HPWL)** | **17** |

**Number of iterations versus convergence**

****

**Summary:**

I have implemented, tested and compared the SOR, CG and CG-preconditioned method for solving linear system of equations. I have also implemented Prim’s algorithm and Kruskal’s algorithm for construction of minimum spanning tree, however the stretch for them is not calculated.