A DUAL QUADRATIC PROGRAMMING ALGORITHM FOR PERFORMANCE-DRIVEN PLACEMENT

by

Arvind Srinivasan

Memorandum No. UCB/ERL M90/107

20 November 1990

COVER PAGE

A DUAL QUADRATIC PROGRAMMING ALGORITHM FOR PERFORMANCE-DRIVEN PLACEMENT

by

Arvind Srinivasan

Memorandum No. UCB/ERL M90/107

20 November 1990

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Dual Quadratic Programming Algorithm for Performance-Driven Placement

Arvind Srinivasan
Electronics Research Lab, University of California, Berkeley, CA 94720

Abstract

In an earlier paper [SJK90], we described a technique for the placement of small-cell ICs subject to performance constraints that employs a a nonlinear wirelength function and a timing model which uses a block-oriented representation of paths [NBHY89]. Given an initial feasible solution, it can efficiently find an optimal solution for the wirelength function subject to the timing constraints. However, finding an initial feasible solution by standard techniques is computationally expensive, taking more than 15 hours for a problem with about 1400 modules.

In this paper, an algorithm that the finds the optimal solution for the formulation *without requiring* an initial feasible solution is presented. As before, the timing constraints are implicitly represented using a network and nonlinear programming techniques are used to minimize wirelength subject to timing constraints. This allows critical paths to dynamically adjust while the placement changes to minimize wirelength. The solution of the nonlinear programming problem yields an initial placement of cells which is followed by hierarchical partitioning techniques to resolve the slot constraints.

Several new results are derived in this paper. In particular, the concept of a *reduced forest* of timing constraints is introduced. It represents the structure inherent in timing constraints and is the key to the efficiency of algorithms presented in the paper. Using reduced forests, the finiteness of the algorithm is proved under less stringent conditions than [SJK90].

The algorithm is shown to be effective on practical examples. For a circuit with 1418 modules, it is able to find an optimal solution in under 8 minutes of CPU time.

1 Introduction

As ICs are scaled to smaller dimensions, the performance of chips becomes dominated by wire delay [SM82]. This underscores the need to develop tools that explicitly optimize performance during physical design. Performance in physical design has been addressed by many researchers in the past and can be grouped into two categories: net-based and path-based approaches. Physical design is a net-based process, i.e., the physical design tools operate on nets and the objects to which they connect. Timing is inherently path-based, i.e., the timing constraints that guarantee certain performance criteria are imposed on paths (sequences of modules and nets) with well-defined starting points and ending points.

Net-based approaches are discussed by [NBHY89], [DAD+84], [TSS86], [BY85], [OIS+86], [MSL89], [WRA+78]. In some net-based algorithms, weights are assigned to nets to reflect the criticality of paths which may be determined by a timing verifier statically or dynamically. In other net-based approaches, a pre-timing analysis may be used to derive maximum bounds on the sizes of the nets. In path-based approaches [JK89], [PK89], the path nature of timing constraints and the physical representation of the IC are unified in a single formulation.

2 The Models

For the purpose of modeling timing behavior and physical layout, an IC may be viewed as a collection of *modules* (or *cells*) interconnected by *nets* that attach to the modules at *pins* (or *terminals*). Let $\mathcal{M} = \{m_1, m_2, \ldots, m_M\}$, $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$, and $\mathcal{P} = \{p_1, p_2, \ldots, p_P\}$ respectively denote the sets of modules, nets, and pins. The modules can be categorized by function as: combinational, synchronizing, primary input (PI), and primary output (PO). Let f represent the number of primary inputs, and g represent the number of primary outputs; thus, there are $M - f - g$ *internal* modules where an internal module is defined to be inside the periphery of the chip with freedom to move. The pins may be similarly categorized based on the classification of the cell to which they attach. Moreover, the pins may also be classified as being input or output, depending on whether they attach to the input or output of a cell. Nets have associated with them a set of modules and a set of pins that they connect.

2.1 The Wirelength Model

The cost function used to evaluate a placement is of great importance since the accuracy of the function in representing the true cost of the layout can have a dramatic impact on the effectiveness of optimization techniques. In order to obtain an exact estimate of the cost, the chip should be completely routed for each of the placements obtained during the sequence of optimizations. This is impractical at best, because of the computational expense involved. Hence, approximations have been proposed to estimate the cost of a layout.

2.1.1 Net length estimate

Let x_{p_i} and y_{p_i} denote the x and y coordinates of pin p_i on the chip. The following estimator for the length of net n is used:

$$L_n = \sum_{p_i, p_j \in n} ((x_{p_i} - x_{p_j})^2 + (y_{p_i} - y_{p_j})^2) \qquad (1)$$

The estimate L_n is the square of the Euclidean distance between the pins on net n. This approximation may be inaccurate for chips with large modules because it assumes that every pair of pins on a net is interconnected separately. However, for the type of ICs considered in this work, (small-cell ASICs), the estimate has been shown to be accurate by [Hal70], [CK84] and [TKH88] and has been widely used in practice.

It is assumed that the pins of a module are located at the center of the module and the module's location is represented by a single (x, y) coordinate that coincides with the center of the module on the chip. Again, these are reasonable assumptions for small-cell ICs because module orientation is not explicitly considered as part of the optimization process.

2.1.2 Total Wirelength

With the assumption that the pins on a module and the module share the same location, we can write an expression for the estimate of the cost of a placement as

$$L = \frac{1}{2} \sum_{m_i, m_j, i \neq j} c_{ij}((x_i - x_j)^2 + (y_i - y_j)^2) \tag{2}$$

where c_{ij} represents the number of nets that modules m_i and m_j share. (x_i, y_i) and (x_j, y_j) represent the locations of m_i and m_j.

The effectiveness of this cost function has been researched in the past and [Hal70], [CK84] and [TKH88] show that it is a reliable indicator of the final routed total wirelength of a placement.

2.1.3 Properties of the cost function

The modules are partitioned into two sets, *fixed* and *movable*. Fixed modules are IO pads or modules that have been assigned a location on the chip, for example, clock pads. Movable modules have variable x and y coordinates. The cost function can then be rewritten using matrix notation as

$$L(\mathbf{x}, \mathbf{y}) = \frac{1}{2}(\mathbf{x}^T \mathbf{B} \mathbf{x} + \mathbf{y}^T \mathbf{B} \mathbf{y}) + \mathbf{c}^T \mathbf{x} + \mathbf{d}^T \mathbf{y} \tag{3}$$

where $\mathbf{x}$ is a vector of the x-coordinates of the module locations and $\mathbf{y}$ is a vector of the y-coordinates. $\mathbf{c}$ and $\mathbf{d}$ are contributions from fixed modules. $\mathbf{B}$ is a symmetric matrix with

$$\mathbf{B} = \mathbf{D} - \mathbf{C} \tag{4}$$

where $\mathbf{C} = [c_{ij}]$ and $\mathbf{D}$ is a diagonal matrix with $d_{ii} = \sum_{j=1}^{n} c_{ij}$.

[Hal70] has shown that if the modules cannot be partitioned into disconnected subsets, then $\mathbf{B}$ is positive semi-definite. In addition, $\mathbf{B}$ is almost always sparse for practical gate-array and sea-of-gate circuits. This enables efficient numerical techniques to be applied to the matrix.

2.2 Timing Model

This work restricts attention to one specific timing problem: the long path problem and ignores the related short path problem. It could be extended to deal with both problems if necessary. It is assumed that cell signal flow is unidirectional for every input-output conducting path in a cell. Similarly, each net has a signal direction associated with its output pin. Associated with every signal flow is a rising and falling delay that is a function of the corresponding cell and interconnect delay models. A single delay value is calculated for each signal flow that is based on the rising and falling transitions. The methods to be discussed are generalizable to the case of separate rising and falling delays [HSC83]. Each synchronizing cell is assumed to have a clock pin, data-input pins, and a data-output pin. In this paper, for simplicity of discussion it is assumed that edge-triggered synchronizing elements are used. The method described are generalizable to the case of level-sensitive latches.

The performance of a synchronous digital IC is inversely proportional to the circuit's cycle time or clock period. The clock waveforms that define the clocking methodology are ultimately determined by the circuit's *critical paths.* Informally, a *path* is defined to be a sequence of interconnected modules and nets with a well-defined starting point and ending point (the starting and ending points are represented by modules). Informally, a *critical path* is a path whose timing behavior constrains the performance of the design.

2.2.1 Graph Representation of Chip Timing

Let the digraph $D_T(V, A)$ represent the integrated circuit in the physical/timing domain. Let the vertex set V be in one-to-one correspondence with the sets of pins classified as: combinational, synchronizing , primary input, and primary output. Arc weights $d(v_i, v_j)$ denote the pin-to-pin signal propagation delays for all $(v_i, v_j) \in A$, and arc directedness represents the direction of signal flow in the circuit. Also, let A^I and A^E model the signal behavior *internal* and *external* to all cells respectively; thus, internal signal arcs represent cell signal flow while external arcs represent net signal flow.

$$A = A^I \cup A^E \tag{5}$$

Let $\{v_1, \ldots, v_{M-g}\}$ represent the cell output pins in the circuit (it is assumed that each net is driven by a single-output pin and that primary inputs have no input pin and primary outputs have no output pin) and $\{v_{M-g+1}, \ldots, v_p\}$ correspond to the cell-input pins. Assume that p_i is the output pin of m_i and connects to n_i. In the event that a cell has more than one output pin, the cell may be replicated for each output with identical nets feeding each replicated cell and the location of each copy of the cell is constrained to one location during physical design. A path Ψ, is defined by the sequence $(v_s, \ldots, v_e)$ of vertices that uniquely define the path.

Delay in an integrated circuit may be viewed as consisting of two components: cell delay and net delay. Let the delay of module m_i be characterized by

$$d(v_j, v_i) \ \forall \, (v_j, v_i) \in A^I \tag{6}$$

and let the delay of net n_i be characterized by

$$d(v_i, v_j) \ \forall \, (v_i, v_j) \in A^E \tag{7}$$

The greater flexibility of this multiple-arc cell and net model permits more accurate modeling than single cell and net delay models. This is particularly important when it becomes necessary to model different pin-to-pin net delays for aggressively scaled technologies where interconnect resistive contributions become significant.

Let E denote the set of vertices representing path end points that corresponds to the input pins of primary outputs and the data-input pins of the synchronizing modules. Associated with each path endpoint vertex is a *required arrival* time r_i. In a similar manner, let S denote the set of vertices representing path starting points that correspond to the primary inputs and data-output pins of the synchronizing modules. Associated with each path starting point vertex is an *actual arrival* time a_i.

Path delay in the integrated circuit is computed by a block-oriented search [HSC83], i.e., delay is determined in a breadth-first manner, beginning at the path starting points and terminating at the path ending points. The worst-case actual arrival time a_j is given by

$$a_j = \max\{a_i + d(v_i, v_j) \mid \forall (v_i, v_j) \in A\} \tag{8}$$

The required arrival times specified for the path end points may be propagated in a backward breadth-first manner through the circuit so that requirements on the required arrival times for vertices internal to the circuit may be determined. The required arrival time r_i is defined to be

$$r_i = \min\{r_j - d(v_i, v_j) \mid \forall (v_i, v_j) \in A\} \tag{9}$$

Based on the calculation of actual arrival and required arrival times for all v_i, a *slack* s_i may respectively be defined as

$$s_i = r_i - a_i \tag{10}$$

Slack values are useful in characterizing the timing behavior of a circuit. A negative value of s_i for v_i indicates that a violation of a timing constraint has occurred.

Definition 1 *The timing of the chip is said to be feasible if and only if* $s_i \geq 0,\ \forall v_i \in V$.

A critical long path is defined as follows

Definition 2 *A critical long path* Π *is a path* Ψ *in which the sequence of vertices* $(v_s, \ldots, v_e)$, $v_s \in S$ *and* $v_e \in E$ *comprising the path all have slack values less than or equal to zero.* $\Pi = \{v_i \mid s_i \leq 0\ \forall\ v_i \in \Psi\}$

Thus, a necessary and sufficient condition for the non-existence of long paths is $s_i > 0,\ \forall\ v_i \in\ V$. The arc weights $d(v_i, v_j)$ for all $(v_i, v_j) \in A^E$ are a function of the positions of the pins defining the cells. Let $X_i = (x_k)^T, m_k \in n_i$ be the vector of x locations of pins on net n_i. Y_i is similarly defined.

Proposition 1

Let $d(v_i, v_j) = f(X_i, Y_i),\ \forall m_k \in n_i$ *be any convex function corresponding to the arc* (v_i, v_j). *Then, the timing constraints form a convex set.*

Proof:

For each non-empty path $\Pi_{se} = v_s \rightarrow v_e, v_s \in S, v_e \in E$, let

$$d(\Pi_{se}) = \sum_{(v_i, v_j) \in \Pi_{se}} d(v_i, v_j)$$

If there is no path from v_s to v_e, let $d(\Pi_{se}) = -\infty$. The timing constraints are equivalent to the following constraints:

$$d(\Pi_{se}) \leq T_e, \forall v_s \in S, \forall v_e \in E$$

But $d(\Pi_{se})$ is the sum of convex functions and is therefore a convex function. So, $d(\Pi_{se}) \leq T_e$ is a convex set. ■

2.3 General Formulation

In this section, it is assumed that $f(X_i, Y_i)$ are convex delay functions associated with the arcs of D_T. The problem of minimizing wirelength subject to timing constraints can be stated as the following nonlinear program

$$\begin{array}{lll} \text{minimize} & \text{L} & \text{(NLP)} \\ \text{subject to} & & \end{array}$$

$$\begin{array}{llll} a_j & \geq & a_i + d(v_i, v_j) & \forall (v_i, v_j) \in A \\ a_j & \leq & T_e & \forall v_j \in E \\ a_j & \geq & T_s & \forall v_j \in S \\ d(v_i, v_j) & = & f(X_i, Y_i) & \forall n_i \in \mathcal{N} \end{array} \tag{11}$$

where T_e and T_s respectively represent the required arrival times at the path endpoints and the actual arrival times at the path starting points which are derived from the performance specifications and the clocking methodology. Although the ideas in the algorithm developed in this paper to solve the performance-driven placement problem can be generalized to any convex delay function, it is difficult to prove convergence and termination for arbitrary non-linear convex delay functions. The algorithms will be illustrated in the next section using a simple linear delay function for the arcs.

Theorem 1

If there exists at least one fixed module and the modules do not form disconnected subsets, then $\mathbf{x}^T\mathbf{B}\mathbf{x}$ *and* $\mathbf{y}^T\mathbf{B}\mathbf{y}$ *are positive definite.*

Proof:

Let there be a fixed module at location (a, b). Define $C(i) = \{m_j | c_{ij} \neq 0\}$. Consider a module m_i connected to the fixed module. A term of the form $\hat{c}(x_i - a)^2$ appears in the objective. The expansion of this term includes the term $\hat{c}x_i^2$. Now suppose there exists a non-zero vector $\mathbf{x}'$ such that $\mathbf{x}'^T\mathbf{B}\mathbf{x}' = 0$. x_i' must be zero in that vector. Consider a term of the form $c_{ij}(x_i' - x_j')^2, j \in C(i)$. This expression can be zero only if $x_i' = x_j'$. But this forces $x_j' = 0, \forall j \in C(i)$. Since the modules form a connected graph (by assumption), this in turn forces all modules connected to $C(i)$ to have a zero x location. Proceeding in this manner, we have $x_1 = x_2 = \ldots = x_M = 0$, which is a contradiction. That $\mathbf{y}^T\mathbf{B}\mathbf{y} > 0$ can be proved similarly. ■

Corollary 1.1

Any relative minimum of NLP is also a global minimum.

Corollary 1.2

The satisfaction of the Kuhn-Tucker first-order optimality conditions are sufficient for a point to be a global minimizer of NLP.

Active constraints at a point are defined to be those constraints that are satisfied with equality. Let $\mathbf{A}^*$ denote the vector function (possibly non-linear) of the active constraints at a global minimum $\mathbf{w}^*$ and $\nabla\mathbf{A}^*$ denote the associated Jacobian matrix. It is assumed that $\nabla\mathbf{A}$ is well-defined in the feasible region of the constraints. Corollary 1.2 states that there exist Lagrange multipliers λ satisfying

$$\begin{aligned} \nabla\mathbf{L}(\mathbf{w}^*) + \lambda^T\nabla\mathbf{A}^* &= \mathbf{0} \\ \lambda^T\nabla\mathbf{A}^* &= \mathbf{0} \\ \lambda &\geq \mathbf{0} \end{aligned} \tag{12}$$

provided $\mathbf{w}^*$ is a regular point of the constraints, i.e., at $\mathbf{w}^*$ the matrix $\nabla\mathbf{A}^*$ has full rank.

2.4 Reducing the Size of the Active Set

Proposition 2

The active timing constraints and delay equations at the optimal solution $\mathbf{w}^*$ *can be replaced by an equivalent set of active constraints which consists of equations for arcs on paths in* D_T *from vertices in* S *to* E*.*

Proof:

Let $E_c = \{v_i \in E | a_i = T_i\}$. In this discussion, consider external arcs. Internal arcs contribute only to the right-hand side of an equation. If $E_c \neq \emptyset$ there exists a forest of arcs τ_A such that

$$a_j = a_i + d(v_i, v_j), \ \forall(v_i, v_j) \in \tau_A$$

For each arc $(v_i, v_j) \in \tau_A$, if $v_j \notin E_c$ and v_j has no arcs in τ_A directed out of it, we can increase a_j by a finite positive value ϵ without affecting the feasibility of the current solution. The solution remains optimal because a_j does not appear in the objective function. Thus, the equation corresponding to (v_i, v_j) can be deleted from the active set. If v_i has no other arcs in τ_A directed out of it, or all the arcs directed out of v_i were deleted by the above process, we can delete the delay equation for the output net associated with vertex v_j.

Similarly, if there is an arc $(v_i, v_j) \notin S$ with no arcs in τ_A directed into v_i, we can decrease a_i by a finite positive constant ϵ and make it inactive without affecting the feasibility or optimality. Similar arguments can be made for the delay equations of the output net of a movable module with no arcs in τ_A directed into it. This process of deleting timing and delay equations can be repeated until the resulting forest τ'_A is rooted in S and all the arcs terminate in E_c. If E_c is empty, then all the constraints are removed by the process and the solution corresponds to the unconstrained optimal solution of the objective function. ■

The forest resulting from this reduction is called a *reduced active forest* (RAF). The significance of this proposition is that in searching for an optimal solution, it suffices to look for solutions such that the set of active arcs form trees rooted in S and terminating in E. This simplifies the algorithm used for solving the optimization problem.

3 Specific Formulation for Linear Delays

This section describes a formulation that uses linear delay functions for the arcs in D_T and an efficient representation for the resulting delay equations. The interconnect length estimate used to determine a net's timing behavior (i.e., the arc delays for arcs in A^E) is taken to be the net's bounding-box for two reasons: (1) techniques exist that bound the size of a *minimum rectilinear Steiner tree* (MRST) given the semi-perimeter of the net's enclosing rectangle and the number of pins on the net [CH79] and (2) the net's bounding-box is easy to calculate using linear inequalities.

Let $\overline{x}_i$, $\underline{x}_i$, $\overline{y}_i$, and $\underline{y}_i$ be the extents of the bounding-box of n_i. Define constraints on $\overline{x}_i$ as follows

$$\overline{x}_i \geq x_j, \forall m_j \in n_i \tag{13}$$

$\underline{x}_i$ is constrained as follows

$$\underline{x}_i \leq x_j, \forall m_j \in n_i \tag{14}$$

Define $\overline{y}_i$ as follows

$$\overline{y}_i \geq y_j, \forall m_j \in n_i \tag{15}$$

Define $\underline{y}_i$ as follows

$$\underline{y}_i \leq y_j, \forall m_j \in n_i \tag{16}$$

These equations, along with the timing constraints defined in Equations 8 and 9 are sufficient to ensure that the variables $\overline{x}_i$, $\underline{x}_i$, $\overline{y}_i$, and $\underline{y}_i$ always correctly define the bounding-box of a net.

Let C_h and C_v denote the horizontal and vertical capacitance per unit length of the horizontal and vertical interconnect wires respectively. Let R_i denote the output resistance of module m_i. The delay of net n_i (and the arc weight $d(v_i, v_j), (v_i, v_j) \in A^E$) is determined by the following equation

$$d(v_i, v_j) = R_i[C_h(\overline{x}_i - \underline{x}_i) + C_v(\overline{y}_i - \underline{y}_i)] \tag{17}$$

In order to model the equations for bounds on a net, a digraph is created. The digraph is used in Section 3 of this paper to ensure linear independence in the specification of bounding-box constraints. Let the digraph $D_B(V_B, A_B)$ represent the relationship between the net bounding-box and the modules to which pins defining the net bounding-box are attached. Let the vertex set V_B be in one-to-one correspondence with the variables defining the maximum and minimum x and y extents of the net bounding-boxes and the x and y locations of the modules. The arc set A_B is

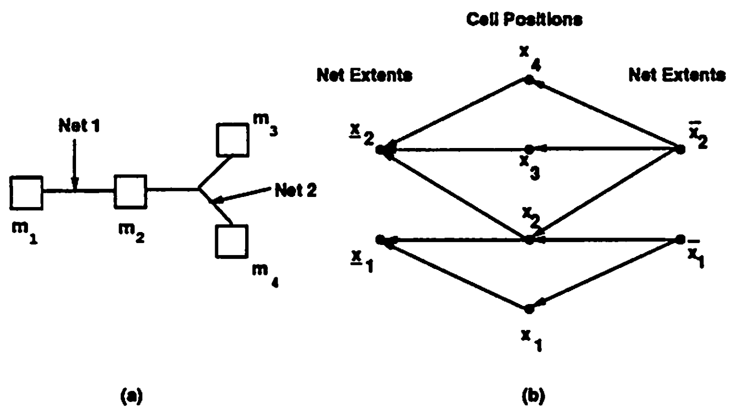

Figure 1: The bounding-box graph, D_B

in one-to-one correspondence with the constraints that define the extents of the bounding-boxes. Figure 1(a) shows connected cells and nets and the corresponding digraph D_B is shown in 1(b). Notice that the arcs are directed from those vertices with larger values.

To simplify further discussion, the following notation is introduced. Let

$$\mathbf{w}_{cell} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}$$

be the combined vector of x and y coordinates of cell positions. Let $\mathbf{w}_{net}$ denote the vector of net bounding-box variables and $\mathbf{w}_{pin}$ the vertex actual arrival time variables. Then

$$\mathbf{w} = \begin{bmatrix} \mathbf{w}_{cell} \\ \mathbf{w}_{net} \\ \mathbf{w}_{pin} \end{bmatrix}$$

is the $2M + 4N + P$ vector of all variables in the formulation of the problem. However, the extra $(4N + P)$ variables corresponding to the net bounding-boxes and arrival times do not enter into the cost function, so the value of the cost function at any point is unchanged and the sparsity of the matrix representing the cost function is retained. Let

$$\mathbf{Q} = \begin{bmatrix} \mathbf{B} & 0 & 0 & 0 \\ 0 & \mathbf{B} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

be the combined $(2M + 4N + P) \times (2M + 4N + P)$ matrix for the cost function and let

$$\mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

Then, the cost function can be rewritten as

$$L = \frac{1}{2}\mathbf{w}^T\mathbf{Q}\mathbf{w} + \mathbf{b}^T\mathbf{w} \tag{18}$$

The problem of minimizing wirelength subject to linear timing constraints be stated as:

$$\begin{array}{llll} \text{minimize} & \text{L} & & \text{(NLP)} \\ \text{subject to} & \overline{x}_i \geq x_j & & \forall m_j \in n_i,\ \forall n_i \in \mathcal{N} \\ & \underline{x}_i \leq x_j & & \forall m_j \in n_i,\ \forall n_i \in \mathcal{N} \\ & \overline{y}_i \geq y_j & & \forall m_j \in n_i,\ \forall n_i \in \mathcal{N} \\ & \underline{y}_i \leq y_j & & \forall m_j \in n_i,\ \forall n_i \in \mathcal{N} \\ & a_j \geq a_i + d(v_i, v_j) & & \forall (v_i, v_j) \in A \\ & a_j \leq T_e & & \forall v_j \in E \\ & a_j \geq T_s & & \forall v_j \in S \end{array} \tag{19}$$

<u>**Theorem 2**</u>

The system of equations represented by D_B is non-singular if and only if D_B has no cycles.

<u>**Corollary 2**</u>

Given a digraph D_B representing a set of bounding-box constraints, an independent set of equations forms a spanning tree on D_B.

4 The Primal Algorithm

In order to simplify the discussion of the Dual Algorithm, the Primal Algorithm for solving NLP is presented first. The algorithm follows a primal active set method [Lue84] but departs from conventional techniques in the representation, activation and deletion of constraints and the activity of variables.

<u>**Definition 3**</u>

An active *critical path is a path such that all of its vertices have zero slack.*

The algorithm proceeds as a sequence of major iterations. At major iteration k, a feasible point $\mathbf{w}^{(k)}$ is known, which satisifies the active constraints with equality. $\mathcal{A}$ denotes the set indices of constraints that are active. Let $\mathbf{A}^{(k)}$ denote the matrix of constraints that are active during iteration k, i.e., $\mathbf{A}^{(k)}$ denotes those constraints that are satisfied with equality. This includes both path constraints and bounding-box constraints for nets that lie on active critical paths. For linear constraints, $\nabla\mathbf{A} = \mathbf{A}$, so the Kuhn-Tucker conditions reduce to:

$$\begin{aligned} \nabla \mathbf{L}(\mathbf{w}^*) + \lambda^T \mathbf{A}^* &= \mathbf{0} \\ \lambda^T \mathbf{A}^* &= \mathbf{0} \\ \lambda &\geq \mathbf{0} \end{aligned} \tag{20}$$

Let $\Gamma^{(k)} = \{\Pi_1, \ldots, \Pi_\ell\}$ be the set of active critical paths during iteration k. Let $\mathcal{N}_\Gamma \subseteq \mathcal{N}$ denote the set of nets that form the critical paths. Let $A_\Gamma \subseteq A$ denote the set of arcs in D_T that lie on the active critical paths. Let E_Γ and S_Γ denote the starting and ending points of the active critical paths. Each major iteration attempts to locate the solution to an equality constrained problem formed by deleting the inactive constraints from NLP. This is done by shifting the origin to $\mathbf{w}^{(k)}$ and looking for a "correction vector" δ that solves the following problem.

$$\text{minimize} \quad \tfrac{1}{2}\delta^T \mathbf{Q} \delta + \delta^T \mathbf{g}^{(k)} \tag{EP}$$

$$\text{subject to} \quad \begin{array}{rcll} \overline{x}_i &=& x_j & \forall m_j \in n_i,\ \forall n_i \in \mathcal{N}_\Gamma \\ \underline{x}_i &=& x_j & \forall m_j \in n_i,\ \forall n_i \in \mathcal{N}_\Gamma \\ \overline{y}_i &=& y_j & \forall m_j \in n_i,\ \forall n_i \in \mathcal{N}_\Gamma \\ \underline{y}_i &=& y_j & \forall m_j \in n_i,\ \forall n_i \in \mathcal{N}_\Gamma \\ a_j &=& a_i + d(v_i, v_j) & \forall (v_i, v_j) \in A_\Gamma \\ a_j &=& T_e & \forall v_j \in E_\Gamma \\ a_j &=& T_s & \forall v_j \in S_\Gamma \end{array} \tag{21}$$

$\mathbf{g}^{(k)}$ is the gradient vector at the current point, defined as

$$\mathbf{g}^{(k)} = \nabla \mathbf{L}(\mathbf{w}^{(k)}) = \mathbf{Q}\mathbf{w}^{(k)} + \mathbf{b} \tag{22}$$

The solution to the above problem (EP) can be found by solving the Kuhn-Tucker conditions for EP, which form a linear system of equations, corresponding to the Hessian matrix of the augmented Lagrangian.

$$\begin{bmatrix} \mathbf{Q} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta^{(k)} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{g}^{(k)} \\ \mathbf{0} \end{bmatrix} \tag{23}$$

The step vector $\delta^{(k)}$ consists of three parts: δ_{cell}, the step vector corresponding to cell positions, δ_{net} corresponding to the net bound variables and δ_{pin} corresponding to the arrival time variables. The vector λ is the vector of Lagrange multipliers for the constraints in $\mathbf{A}$.

If $\delta^{(k)}$ is feasible with respect to the constraints not in $\mathbf{A}^{(k)}$, i.e., a timing verification on the graph yields a feasible timing graph, then the step is accepted and $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \delta^{(k)}$. If not, then a line search is made in the direction of $\delta^{(k)}$to find the best feasible point. The line search procedures are explained in detail in following sections. At this point, it suffices to note that these procedures return a step length parameter $\alpha^{(k)}$ such that

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \alpha^{(k)}\delta^{(k)} \tag{24}$$

minimizes L along the direction $\delta^{(k)}$ and $\mathbf{w}^{(k+1)}$ is feasible with respect to all the constraints. If $\alpha^{(k)} < 1.0$, a new constraint becomes active and this is added to the current active set $\mathcal{A}$. If $\delta^{(k)} = \mathbf{0}$, and $\lambda_i^{(k)} \geq 0$, $i = 1, \ldots, |\mathcal{A}|$, then by Theorem 1, we are at the optimal solution.

If

$$\lambda_q^{(k)} < 0 \tag{25}$$

for some constraint $q \in \mathcal{A}$, it is possible to drop constraint q from $\mathcal{A}$. After removing constraint q, the algorithm continues as before. If more than one constraint satisfies Equation 25, then select

$$q = \arg \min_{\mathrm{i}} \lambda_i^{(k)} \tag{26}$$

The flow of the algorithm may be summarized as follows. The algorithm starts with a feasible point $\mathbf{w}^{(1)}$ and an initial active set of constraints whose matrix is $\mathbf{A}^{(1)}$. (Finding a feasible point is discussed in section 4.5.)

Algorithm

1. Given $\mathbf{w}^{(1)}$ and an active set $\mathcal{A}$, set the iteration index k to 1.
2. Solve (EP) for $\delta^{(k)}$.
3. If $\delta^{(k)} = \mathbf{0}$ and $\lambda_i^{(k)} \geq 0$, $\forall\, i \in \mathcal{A}$, stop. The optimal solution has been reached.
4. Find $\alpha^{(k)}$, a step length parameter and set $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \alpha^{(k)}\delta^{(k)}$
5. If $\alpha^{(k)} < 1$ add some constraint(s) to $\mathcal{A}$ according to section 4.3
6. If $\lambda_q^{(k)} < 0$ for some q, delete a constraint from $\mathcal{A}$
7. Set $k = k + 1$ and go to step 2

Definition 4

The unique arcs *of a path are those arcs that are not part of any other active critical path.*

Proposition 3

When a path becomes active, the equations for the unique arcs of the path can be added to the active set simultaneously, i.e., the step vector need not be computed after adding each unique arc on the path.

Proof:

Let the path be Ω. Let there be q unique arcs in the path. Suppose we have added timing and bounding-box constraints for $t < q$ arcs to the current active forest τ_A, obtaining a new active forest τ'_A. Then, at least one of the following are present in τ'_A: (1) an arc $(v_i, v_j) \in \Omega$ such that $v_i \notin S$ and v_i has no arcs in τ'_A directed into it, (2) an arc $(v'_i, v'_j) \in \Omega$ such that such that $v'_j \notin E$ and v'_j has no arcs in τ'_A directed outward from it. Informally, when we have activated $t < q$ unique constraints, Ω is disconnected because of the uniqueness of the q arcs. By Proposition 2, the forest τ'_A can be reduced to τ_A, so the optimal solution for τ_A is also optimal for τ'_A. ∎

Proposition 4

If the conditions of Theorem 1 are satisfied and at each non-terminal step $\alpha^{(k)} \neq 0$*, the algorithm given above terminates in a finite number of steps at the optimal solution* $\mathbf{w}^*$.

Proof: The proof of this proposition follows the proof of the active set theorem in [Lue84]. After the solution corresponding to an active set is found, since the step length is positive, the step results in a strict decrease in the objective function. Thus, once the algorithm leaves an active set, it never returns to it. There are only a finite number of *reduced active forests*. Associated with each RAF, there are only a finite number of active sets corresponding to different active bounding-box constraints. ∎

4.1 Line Search Procedure (1): Bounding-Box Constraints

Given δ, and the current active set of bounding box constraints, this procedure computes α_1, the maximum step length such that the cells defining the bounding-box for a net on an active critical path change. Let N_Γ denote the set of nets that lie on some active critical path. Let δx_j denote the computed step in the x direction for cell j and δy_j the step in the y direction. Let $\delta\overline{x}_n$ denote the step for the maximum x bound for net n and $\delta\underline{x}_n$ the step for the minimum x bound. $\delta\overline{y}_n$ and $\delta\underline{x}_n$ are similarly defined for the y direction.

For each net $n \in N_\Gamma$, the procedure involves computing the parameters

$$\alpha_n^{\overline{x}} = \min_{m_j \in n} \left| \frac{\overline{x}_n - x_j}{\delta x_j - \delta\overline{x}_n} \right| \tag{27}$$

$$\alpha_n^{\underline{x}} = \min_{m_j \in n} \left| \frac{\underline{x}_n - x_j}{\delta x_j - \delta\underline{x}_n} \right| \tag{28}$$

$$\alpha_n^{\overline{y}} = \min_{m_j \in n} \left| \frac{\overline{y}_n - y_j}{\delta y_j - \delta\overline{y}_n} \right| \tag{29}$$

$$\alpha_n^{\underline{y}} = \min_{m_j \in n} \left| \frac{\underline{y}_n - y_j}{\delta y_j - \delta\underline{y}_n} \right| \tag{30}$$

$$\alpha_1 = \min_{n \in N_\Gamma} \{\min(1, \alpha_n^{\overline{x}}, \alpha_n^{\underline{x}}, \alpha_n^{\overline{y}}, \alpha_n^{\underline{x}})\} \tag{31}$$

4.2 Line Search Procedure (2): Timing Constraints

Given δ and α_1, this procedure computes the maximum step length $\alpha^{(k)}$ such that the chip's timing remains feasible. $\alpha^{(k)}$ is determined by performing a bisection timing verification (BTV). The bisection timing verification is a combination of bisection line search and timing verification and departs from conventional techniques in that the graph representation of the timing constraints is used to compute a maximum feasible step.

```
1. $\alpha = \alpha_1$
2. Update the cell positions based on $\alpha$
3. Update the net bounding-box positions
4. Update the delays for all external arcs
5. Calculate the minimum slack in $D_T$
6. If the minimum slack $\leq 0$ (the step length is too long)
   (a) $\sigma = \alpha \ / \ 2$
   (b) $\alpha = \alpha - \sigma$
   (c) While the absolute value of the minimum slack in $D_T > 0$, do
       i. Update the cell positions based on $\alpha$
       ii. Update the net bounding-box positions
       iii. Update the delays for all external arcs
       iv. Calculate the minimum slack in $D_T$
       v. $\sigma = \sigma \ / \ 2$
       vi. If the minimum slack $> 1.0e16$
           A. $\alpha = \alpha + \sigma$
       vii. If the minimum slack $< -1.0e-16$
           A. $\alpha = \alpha - \sigma$
7. STOP
```

4.3 Activating a constraint

There are several conditions under which new constraints are added to the active set. These may be listed as follows:

1. $\alpha^{(k)} = 1$. No new constraint is added since the full step is feasible. The algorithm proceeds to the next major iteration.

2. $\alpha^{(k)} = \alpha_1 < 1$. In this case, a new bounding box constraint becomes active. Without loss of generality, assume that $\alpha_1 = \mid \frac{\overline{x}_n - x_j}{\delta x_j - \delta \overline{x}_n} \mid$, for some net n. The constraint that becomes active is $\overline{x}_n \geq x_j$. This constraint is added as an equality to $\mathcal{A}$. If more than one net constraint gives $\alpha_1 = \mid \frac{\overline{x}_n - x_j}{\delta x_j - \delta \overline{x}_n} \mid$, all of them are added to the active set.

3. $\alpha^{(k)} < \alpha_1$. In this case, a new critical path becomes active. Several constraints corresponding to timing constraints for unique arcs of the critical path and bounding-box equations for each unique arc become active. By Proposition 3, we can add all the equations simultaneously to the active set.

Let $\Omega = (v_s, \ldots, v_e)$ denote the new critical path. Let $\mathcal{N}_\Omega \subseteq \mathcal{N}$ denote the set of nets that form the new critical path. Let $A_\Omega \subseteq A$ denote the set of unique arcs in D_T that lie on the new critical path. For the critical path Ω, the constraints that become active are

$$\begin{array}{rcll} \overline{x}_i & = & \max\{x_j | m_j \in n_i\}, & \forall n_i \in \mathcal{N}_\Omega \\ \underline{x}_i & = & \min\{x_j | m_j \in n_i\}, & \forall n_i \in \mathcal{N}_\Omega \\ \overline{y}_i & = & \max\{y_j | m_j \in n_i\}, & \forall n_i \in \mathcal{N}_\Omega \\ \underline{y}_i & = & \min\{x_j | m_j \in n_i\}, & \forall n_i \in \mathcal{N}_\Omega \\ a_j & = & a_i + d(v_i, v_j) & \forall (v_i, v_j) \in A_\Omega \\ a_e & = & T_e & v_e \in E_\Omega \\ a_s & = & T_s & v_s \in S_\Omega \end{array} \quad (32)$$

4.4 Deleting a constraint

The strategy used to delete a constraint is to select the constraint with the most negative Lagrange multiplier and remove it from the active set $\mathcal{A}$. This selection is not invariant to scaling of the variables in the problem, but since the Quadratic Program (QP) for placement is naturally well scaled, the strategy works well. When deleting a bounding-box constraint from the active set, the algorithm follows the usual active set method. Since the algorithm adds only the unique arcs, when one timing constraint is deleted, following Proposition 2, equations for all the unique arcs associated with a path are deleted.

5 The Dual Algorithm

One of the problems with the primal algorithm is that it requires an initial feasible solution. For many practical examples, it is observed that finding an initial feasible solution using conventional techniques like Phase-I of the Simplex method, a Big-M method [Mur83] or even an interior point method [Kar84] takes a long time.

The version of the dual algorithm described in this paper does not require a feasible solution, and can be implemented more efficiently [Idn80] than the primal algorithm. In tests on real examples, the dual algorithm when started from an arbitrary *infeasible* point, found the optimal solution faster than the corresponding primal algorithm started from a *feasible* point.

The dual algorithm is a modification of the procedure described by [Idn80]. Like the primal algorithm, the key differences are in the addition and deletion of equations. The general outline of a dual active set method is as follows [GI83]. The method always maintains a "solved subproblem" which is *dual feasible* i.e., primal optimal with respect to a subset of the constraints. The active set for the solved problem is iteratively updated according to the following strategy.

1. Let the current solved subproblem be $SP = \emptyset$.

2. Select a violated constraint, if any, say p.

3. Let $VP = SP \cup \{p\}$.

4. Solve VP.

5. If VP has no solution, terminate because the feasible region is empty; othewise we have a new solved subproblem. Let $SP = VP$, go to step 2.

Let n^+ denote an added (violated) constraint. Let N denote the matrix of currently active constraints. The constraints are of the form:

$$Nx \leq b \tag{33}$$

$$n^+x \leq b^+ \tag{34}$$

As shown in [Idn80], VP has no solution if the added constraint is linearly dependent on the constraints in the active set and the following condition is satisfied:
Condition 1: There exists an r such that $N^T r = {n^+}^T$ and $r \leq 0$.

Suppose z is a allowable step direction for the above constraints. It follows that z must satisfy the following condition.
Condition 2: $Nz \leq 0; n^+z < 0$.

But $n^+ = r^T N$. Therefore,

$$n^+z = r^T N z$$

But $r \leq 0$ and $Nz \leq 0$ imply that $n^+z \geq 0$ which is a contradiction. Therefore, there is no allowable direction such that $Nz \leq 0$ and $n^+z < 0$ and therefore, the subproblem is infeasible. By maintaining *infeasibility multipliers* as shown in [GI83] the above conditions can be checked efficiently whenever a new constraint is added to the active set.

Note that unlike the primal method, there is a choice of which constraint to add in the dual method. By modifying the above procedure, carefully selecting the added constraints and modifying the definition of a subproblem, it is possible to prove finite termination (without the degeneracy assumption) and solve the problem efficiently.

<u>Definition 5</u>
A subproblem corresponding to a **reduced forest** τ_A *(RFP_A) is defined as the subproblem of NLP obtained by considering the timing and bounding-box constraints corresponding to* τ_A *and ignoring all the other constraints.*

<u>Definition 6</u>
A solved subproblem corresponding to a **reduced forest** τ_A *($RFSP_A$) is defined as the* **reduced active forest** τ_{SA} *and solution vector* $\overline{\mathbf{w}}$ *obtained by finding the optimal solution for NLP subject to the timing and bounding-box constraints corresponding to* τ_A*, i.e., by solving* RFP_A.

<u>Definition 7</u>

In the following discussion, define a critical path Ω to be the the set of timing and bounding-box constraints for some path with nonpositive slack.

We will informally use τ_A to refer to both - a forest and the timing and bounding-box equations for arcs in the forest. The notation is unambiguous for the purposes of explanation. The modified dual active set algorithm proceeds as follows:

1. Solve the unconstrained problem corresponding to NLP. The current active forest is $\tau_A = \emptyset$.

2. Select a path Ω with negative slack, if any.

3. Let $\tau'_A = \tau_A \cup \Omega$.

4. Solve the subproblem corresponding to τ'_A (RFP_A).

5. If RFP_A has no solution, terminate because the feasible region is empty; othewise we have a new solved subproblem. Let $\tau_A = \tau_{SA}$, go to step 2.

The details of the algorithm are described below. Let $\mathbf{A}^{(k)}$ denote the matrix of constraints that are active during iteration k, i.e., $\mathbf{A}^{(k)}$ denotes those constraints that are satisfied with equality. This includes both path constraints and bounding-box constraints for nets that lie on active critical paths. Let $\Gamma^{(k)} = \{\Pi_1, \ldots, \Pi_\ell\}$ be the set of active critical paths during iteration k, i.e., the *reduced active forest.* Let $\mathcal{N}_\Gamma \subseteq \mathcal{N}$ denote the set of nets that form the active critical paths. Let $A_\Gamma \subseteq A$ denote the set of arcs in D_T that lie on active critical paths. Let E_Γ and S_Γ denote the starting and ending points of active critical paths.

Let $\lambda^{(k)}$ denote the Lagrange multipliers at iteration k. Let $\delta\lambda$ denote the computed step direction for the Lagrange multipliers.

1. Compute the unconstrained minimum.

$$\mathbf{w}^{(1)} = -\mathbf{Q}^{-1}\mathbf{b}$$

   $\mathbf{w}^1$ is a *dual feasible* point for NLP. Set the initial active set $\mathbf{A}^{(1)} = \emptyset$ and the initial reduced forest $\Gamma^{(1)} = \emptyset$. Set the initial Lagrange multipliers $\lambda^{(1)} = 0$. Let the current subproblem $SP = \emptyset$.

2. Solve the RFP.
   Compute s_i the slack for each vertex of the reduced forest. Define

$$s_\tau(\mathbf{w}) = \min_{v_i \in \tau_A} \{s_i\}$$

   Define

$$r(\mathbf{w}) = \max_j \{r_{ij} | r_{ij} = x_i - \overline{x_j}, \forall i \in n_j, \forall n_j \in N_\Gamma\}$$

$$l(\mathbf{w}) = \max_j \{l_{ij} | l_{ij} = \underline{x_j} - x_i, \forall i \in n_j, \forall n_j \in N_\Gamma\}$$
$$u(\mathbf{w}) = \max_j \{u_{ij} | u_{ij} = y_i - \overline{y_j}, \forall i \in n_j, \forall n_j \in N_\Gamma\}$$
$$b(\mathbf{w}) = \max_j \{b_{ij} | b_{ij} = \underline{y_j} - y_i, \forall i \in n_j, \forall n_j \in N_\Gamma\}$$

(a) If $s_\tau(\mathbf{w}) < 0$ then timing constraints in τ_A are violated. The constraints to be added to the active set are the timing and bounding-box constraints for unique arcs of the violated path.

(b) If $s_\tau(\mathbf{w}) \geq 0$ and $\max(r(\mathbf{w}), l(\mathbf{w}), u(\mathbf{w}), b(\mathbf{w})) > 0$ then a bounding-box constraint for some net in τ_A is violated. The constraint to be added is the violated bounding-box constraint.

(c) If $s_\tau(\mathbf{w}) \geq 0$ and $\max(r(\mathbf{w}), l(\mathbf{w}), u(\mathbf{w}), b(\mathbf{w})) \leq 0$ go to step 3.

(d) Compute the step directions $\delta\mathbf{w}$ and $\delta\boldsymbol{\lambda}$ assuming the violated constraint(s) is(are) active.

(e) Compute t_1 the maximum step before a Lagrange multiplier turns negative. Compute t_2, the maximum step such that the added constraint becomes active. Let $t = \min(t_1, t_2)$.

(f) Compute the infeasibility multipliers for the added constraint(s). For a bounding-box constraint the procedure is straightforward. For timing constraints, by Proposition 3, only one of the timing equations added to the forest needs to be checked for infeasibility. If timing constraints are added, we can compute the infeasibility multipliers for the last timing constraint added, check Condition 2 and stop if the constraints are infeasible.

(g) If $t = t_1$ drop the constraint corresponding to t_1. Such a step is called a *partial step* because the violated constraints still remain violated. The algorithm will try to add them again during the next iteration.

(h) If $t = t_2$ a constraint(s) is(are) added to the active set.

(i) Update $\mathbf{w}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$, $\Gamma^{(k)}$ and $\mathbf{A}^{(k)}$, set $k = k + 1$ and repeat step 2.

3. Define a new RFP.

(a) If $\max(r(\mathbf{w}), l(\mathbf{w}), u(\mathbf{w}), b(\mathbf{w})) \leq 0$, and $s_\tau(\mathbf{w}) \geq 0$ the current *RFP* is solved. Find a path Ω such that

$$s_\Omega(\mathbf{w}) = \min_{v_i \in V} \{s_i\}$$

for every vertex $v_k \in \Omega$. If $s_k \geq 0$, the optimal solution has been reached, terminate.

(b) If $s_k < 0$, add the timing and bounding-box equations for unique arcs of Ω to the reduced forest τ_A and call the new forest τ'_A. ($\tau'_A = \tau_A \cup \Omega$).

(c) Compute the step direction assuming τ'_A is active.

(d) Only one of the timing equations added to the forest needs to be checked for infeasibility. Therefore, at this step, we can compute the infeasilility multipliers for the last timing constraint added, check Condition 2 and stop if the constraints are infeasible.

(e) If the constraints are feasible, compute t_1 the maximum step before a Lagrange multiplier turns negative. Compute t_2, the maximum step such that the added timing constraints become active. Let $t = \min(t_1, t_2)$.

(f) If $t = t_1$ and t_1 corresponds to a bounding-box constraint, drop a bounding-box constraint

(g) if $t = t_1$ and t_1 corresponds to some active critical path equation, drop the timing constraints for that path.

(h) If $t = t_2$, add the timing and bounding-box equations corresponding to Ω to the active set.

(i) Update $\mathbf{w}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$, $\Gamma^{(k)}$ and $\mathbf{A}^{(k)}$, set $k = k + 1$ and go to step 2.

Proposition 5

The dual active set algorithm defined above will find the optimum point in a finite number of iterations or terminate when there is no feasible solution.

Proof:

Just before step 3 of the algorithm is executed, some subproblem of the original problem is solved because the algorithm always maintains dual feasibility. Every time that the step is executed, the objective value increases, since it corresponds to a RFP containing a violated critical path not in the previous RFP. Therefore, after solving an RFP we never return to the RFP again. There are only a finite number of RFPs. The proof that solving an RFP is a finite process is identical to the proof of finiteness of the dual algorithm in [GI83]. ■

6 Experimental Results and Practical Considerations

Most of the computational effort in the algorithms described is in solving a linear system every iteration. In a practical implementation of the primal and dual algorithms, fast techniques for maintaining and updating matrix factorizations could be applied [GMW89].

A program that implements the primal and dual algorithms has been developed. It uses complete LU factorization at every iteration to solve the linear system rather than complex schemes for updating the factors. To illustrate the speed-up in computation time obtained even with a simplified implementation, the program was tested on a 210 cell industrial example that is part of the control logic for a 4-bit microprocessor and a 1418 cell gate-array. For the primal algorithm, an initial feasible point was found using the Simplex method[1]. The results are compared to a standard

[1]For the Primal Algorthm, initial feasible solution took 400 sec for ex1 and 9 hours for ex2 using Phase I of the Simplex method

Example	cells	nets	nets on CP	Primal (sec) (see footnote)	Dual (sec)	QP (sec)
ex1	210	170	59	30	8	850
ex2	1418	1161	133	8min	7min	>10hrs

Figure 2: Results obtained on DECStation 3100

Example	%improvement in wire delay	%increase in Quad. wirelen
ex1	50%	3%
ex2	5.5%	0.07%

Figure 3: Improvement in wire delay

efficient Quadratic Programming (QP) package. In the 210 cell example the speed-up obtained by the dual algorithm over standard QP is a factor of 10, while in the 1418 cell example the speed-up is substantial. After 10 hours of running, the standard QP package had not found the optimal solution for the 1418 cell array and it was terminated, while the dual algorithm found the optimal solution in about 7 minutes of CPU time. The improvement in wire delay is 50% for the 210 cell example with only 3% increase in quadratic wirelength over the unconstrained placement while for the 1418 cell example, the dual and primal methods were able to obtain a 5.5% improvement in wire delay with an increase of only 0.07% in quadratic wirelength. The results are shown in Figure 2.

One of the reasons for the failure of standard QP techniques is that the problem is *highly degenerate*. In tests on standard QP, it was observed that there are at least four active bounding-box constraints for each net and the Lagrange multipliers for those constraints not on critical paths have zero value. Similarly, for every pin, there is one active timing constraint - the one that corresponds to the maximum in equation 8. Again, the multipliers for the timing constraints not on critical paths are zero. The number of active variables in conventional techniques can be enormous. For a typical problem with 1000 cells and 3000 nets, the number of active variables could be upto 18,000 and the active constraints could number 15,000. Using techniques described in the paper, for the same typical example it is possible to reduce the number of active variables to about 2200 and the number of active constraints to a few hundreds.

Standard QP techniques activate arcs from the graph in some arbitrary sequence, leading to many degenerate steps. Also, standard methods fail to recognize that only one timing constraint for each primary output cell on a critical path leads to a non-degenerate step. On the other hand, the algorithms presented above are able to recognize the path nature of the constraints through the notion of a reduced forest, thereby avoiding degenerate steps.

7 Conclusions

The ideas in this paper unify the physical and temporal aspects of IC design in a new theoretical framework and appear to hold much promise for solving large scale problems efficiently. The speed-up is obtained by operating on graphs and by recognizing the fact that we need only restrict attention to those timing constraints and variables that are associated with a *reduced forest.* The dual algorithm is to be preferred over the primal algorithm for solving large problems because it does not require feasibility[2]. and can be implemented more efficiently. The algorithm provides a "global placement" which can then be refined by successive partitioning. Each partition generates constraints on cell positions. These constraints can be dealt with easily using the non-linear programming approach outlined.

Acknowledgements The author would like to acknowledge the support received for this research from SRC Grant 90-DC-008.

References

[BY85] Michael Burstein and Mary N. Youssef. Timing influenced layout design. In *IEEE Proceedings of the 22nd Design Automation Conference*, pages 124–130, 1985.

[CH79] F. K. Chung and F. K. Hwang. The largest minimal rectilinear steiner trees for a set of *n* points enclosed in a rectangle with given perimeter. *Networks*, 9(1):19–36, Spring 1979.

[CK84] C. K. Cheng and E. S. Kuh. Module placement based on resistive network optimization. *IEEE Trans. Computer-Aided Design*, CAD-3:218–225, July 1984.

[DAD+84] A. E. Dunlop, V. D. Agrawal, D. N. Deutsch, M. F. Jukl, P. Kozak, and M. Wiesel. Chip layout optimization using critical path weighting. In *IEEE Proceedings of the 21st Design Automation Conference*, pages 133–136, 1984.

[GI83] D. Goldfarb and A. Idnani. A numerically stable dual method for strictly convex quadratic programs. *Mathematical Programming*, 27:1–33, 1983.

[GMW89] P. Gill, W. Murray, and M. Wright. *Practical Optimization.* Academic Press, New York, New York, 1989.

[Hal70] K. M. Hall. An r-dimensional quadratic placement program. *Management Science*, 17(3):219–229, November 1970.

[2]For the Primal Algorthm, initial feasible solution took 400 sec for ex1 and 9 hours for ex2 using Phase I of the Simplex method

[HSC83] R. B. Hitchcock, G.L. Smith, and D.D. Cheng. Timing analysis of computer hardware. *IBM Journal of Research and Development*, 26(1):100–105, 1983.

[Idn80] Ashok U. Idnani. *Numerically Stable Dual Projection Methods for Solving Positive Definite Quadratic Programs.* PhD thesis, City University of New York, November 1980.

[JK89] M. A. B. Jackson and E. S. Kuh. Performance-driven placement of cell-based ic's. In *IEEE Proceedings of the 26th Design Automation Conference*, pages 370–375, 1989.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proc. 16th Annual Symposium on the Theory of Computing*, 1984.

[Lue84] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison Wesley, Reading, Massachusetts, 2nd edition, 1984. Ch. 14.

[MSL89] M. Marek-Sadowska and S. P. Lin. Timing-driven placement. *IEEE International Conference on Computer-Aided Design ICCAD-89*, pages 94–97, 1989.

[Mur83] Katta G. Murty. *Linear Programming.* John Wiley and Sons, 1983.

[NBHY89] R. Nair, C. L. Berman, P.S. Hauge, and E. J. Yoffa. Generation of performance constraints for layout. *IEEE Trans. Computer-Aided Design*, CAD-8:860–874, August 1989.

[OIS+86] Yasushi Ogawa, Tatsuki Ishii, Yoichi Shiraishi, Hidekazu Terai, Tokinori Kozawa, Kyoji Yuyama, and Kyoji Chiba. Efficient placement algorithms optimizing delay for high-speed ecl masterslice lsi's. In *IEEE Proceedings of the 23rd Design Automation Conference*, pages 404–410, 1986.

[PK89] S. Prasitjutrakul and W. J. Kubitz. Path-delay constrained floorplanning: A mathematical programming approach for initial placement. In *IEEE Proceedings of the 26th Design Automation Conference*, pages 364–369, 1989.

[SJK90] A. Srinivasan, M. Jackson, and E.S. Kuh. A fast algorithm for performance driven placement. In *Int. Conf. on Computer-Aided Design*, volume 8, page to appear, 1990.

[SM82] K. C. Saraswat and F. Mohammadi. Effect of scaling of interconnections on the time delay of vlsi circuits. *IEEE Transactions on Electron Devices*, ED-29:645–650, April 1982.

[TKH88] R. S. Tsay, E. S. Kuh, and C. P. Hsu. Proud: A sea-of-gates placement algorithm. *IEEE Design and Test of Computers*, pages 318–323, December 1988.

[TSS86] S. Teig, R. L. Smith, and J. Seaton. Timing-driven layout of cell-based ic's. *VLSI System's Design*, pages 63–73, May 1986.

[WRA+78] P.K. Wolff, A. E. Ruehli, B. J. Agule, J. D. Lesser, and G. Goertzel. Power/timing: Optimization and layout techniques for lsi chips. *Journal of Design Automation and Fault-Tolerant Computing*, pages 145–164, 1978.