QUADRATIC BOOLEAN PROGRAMMING FOR PERFORMANCE-DRIVEN SYSTEM PARTITIONING

by

Minshine Shih and Ernest S. Kuh

Memorandum No. UCB/ERL M93/19

8 March 1993
(Revised 23 November 1993)

QUADRATIC BOOLEAN PROGRAMMING FOR PERFORMANCE-DRIVEN SYSTEM PARTITIONING

by

Minshine Shih and Ernest S. Kuh

Memorandum No. UCB/ERL M93/19

8 March 1993
(Revised 23 November 1993)

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

QUADRATIC BOOLEAN PROGRAMMING FOR PERFORMANCE-DRIVEN SYSTEM PARTITIONING

by

Minshine Shih and Ernest S. Kuh

Memorandum No. UCB/ERL M93/19

8 March 1993
(Revised 23 November 1993)

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Quadratic Boolean Programming for Performance-Driven System Partitioning*

Minshine Shih and Ernest S. Kuh

Department of Electrical Engineering and Computer Sciences

University of California, Berkeley, California 94720

November 23, 1993

Abstract

We discovered and mathematically proved that a partitioning problem under timing and capacity constraints can be formulated *exactly* as a Quadratic Boolean Programming Problem. This new formulation allows arbitrary component sizes, arbitrary capacities of partitions, arbitrary interconnection costs and delay models between partitions. We then found a generalization/enhancement of Burkard's heuristic to efficiently solve the problem. Seven industrial circuits were used to compare our method against two other heuristics based on the traditional approach of component interchanges. Test results showed the superiority of our new method in terms of both solution quality and CPU usage, for problems under very tight Timing and Capacity Constraints.

*This work is supported by SRC Grant 93-DC-008

1 Introduction

Given a circuit or system consisting of interconnected components, there are two main types of partitioning problems. The first type of partitioning problem does not have a fixed underlying partition topology and therefore uses a "Ratio-Cut" cost function[9] as the objective. This is useful when we wish to determine the structure of the circuit and discover the so-called "natural clusters" of the circuit. The second type has a fixed, existing partition topology, which includes capacity for each partition, interconnection costs and delay models between partitions. This is the case for FPGA-type of partitioning and some MCM (Multi-Chip Module) types of partitioning problem, e.g. TCM (Thermal Conduction Module).

This paper will focus on the second type of partitioning problem. Each component in the circuit may have variable size, reflecting the silicon area demand for realizing the component. On the other hand each partition provides a fixed amount of Silicon area called the "capacity" of the partition. The Capacity Constraints state that the total size of all components assigned to the same partition must be no greater than the capacity of that partition. The Timing Constraints are stated as a set of maximum routing delay allowed between a pair of components. These constraints are driven by system cycle time and can be derived from the delay equations and intrinsic delay in combinational circuit components.

In this paper we will show that a partitioning problem under Timing and Capacity constraints can be formulated *exactly* as a Quadratic Boolean Programming Problem. Although in the past there were some attempts to formulate a partitioning problem as a Quadratic Assignment Problem[4][5][6] , these formulations allow arbitrary partition capacities but restrict each component to be of equally size. Furthermore, they cannot take Timing Constraints into considerations. Our new formulation allows arbitrary component sizes, arbitrary capacities of partitions, arbitrary interconnection costs and delay models between partitions.

To solve the Quadratic Boolean Programming problem, we generalized and enhanced an existing heuristic first proposed by Burkard[3], which he applied to solve the Quadratic Assignment Problem (QAP). We found that the heuristic can be applied to a much more general class of problems instead of just QAP's. In his original paper the method can only solve problems with up to 50 components or so due to the high computational complexity. We generalized his idea to handle additional Capacity Constraints and Timing Constraints. Furthermore, in order to handle real circuits with hundreds or thousands of components, we exploit the facts that (a); the number of partitions is very small compared to the number of components, and (b); the interconnections between the components are quite sparse. Thus we can achieve the speedup necessary to make it a practical method.

2 Problem Formulation and Applications

2.1 Input

The input to the problem includes the followings:

I. Descriptions of the Circuit:

1). J is a set of N circuit components. Let $j \in J$ be the index to the component.

2). s_j is the size of component j, representing the silicon area component j requires.

3). A is an $N \times N$ matrix, where $a_{j_1 j_2}$ is the number of interconnections from component j_1 to j_2.

4). D_C is an $N \times N$ matrix, where $D_C(j_1, j_2)$ is the maximum signal routing delay allowed from component j_1 to j_2.

II. Descriptions of Partitions:

1). I is a set of M partitions. Let $i \in I$ be the index to a partition.

2). c_i is the capacity of partition i.

3). B is an $M \times M$ matrix, where $b_{i_1 i_2}$ is the cost of wire routing from partition i_1 to i_2.

4). D is an $M \times M$ matrix, where $D(i_1, i_2)$ is the routing delay from partition i_1 to i_2. Notice that we don't assume any relationship between B and D in our formulation.

III. others:

1). P is an $M \times N$ matrix, where p_{ij} is the cost of assigning component j to partition i.

A solution to the problem is an assignment $\mathcal{A} : J \longrightarrow I$ satisfying the following two sets of constraints:

C1: (Capacity Constraints)

$$\sum_{\forall j, \mathcal{A}(j)=i} s_j \leq c_i \quad \forall i \in I$$

C2: (Timing Constraints)

$$D(\mathcal{A}(j_1), \mathcal{A}(j_2)) \leq D_C(j_1, j_2) \quad \forall j_1, j_2 \in J$$

The objective is to

$$minimize \quad \alpha \sum_{\forall i, j, \mathcal{A}(j)=i} p_{ij} \; + \; \beta \sum_{\forall j_1, j_2} a_{j_1 j_2} \; b_{\mathcal{A}(j_1)\mathcal{A}(j_2)} \quad subject\ to \ C1, \ C2. \tag{1}$$

If we introduce a matrix $[x_{ij}]_{M \times N}$ of binary variables and let $x_{ij} = 1$ if $\mathcal{A}(j) = i$, and $x_{ij} = 0$ otherwise, then each $\mathcal{A}$ corresponds to a unique $[x_{ij}]_{M \times N}$ such that $\sum_{i=1}^{M} x_{ij} = 1, \ \forall j \in J$ and vice versa. We say that $[x_{ij}]$ satisfies a certain constraint iff its corresponding $\mathcal{A}$ does. Now we can rewrite the problem into :

C1: (Capacity Constraints)

$$\sum_{j=1}^{N} s_j x_{ij} \leq c_i \quad \forall i \in I$$

C2: (Timing Constraints)

$$D(i_1, i_2) \leq D_C(j_1, j_2) \quad \forall (i_1, j_1) \ and \ (i_2, j_2) \ s.t. \ x_{i_1 j_1} = 1 = x_{i_2 j_2}$$

C3: (Generalized Upper Bound Constraints)

$$\sum_{i=1}^{M} x_{ij} = 1 \quad \forall j \in J$$

The transformation of $C2$ is obvious when we recognize the fact that $x_{ij} = 1$ means component j is assigned to partition i in the corresponding assignment $\mathcal{A}$.

The objective (equation (1)) becomes

$$minimize \quad \alpha \sum_{i=1}^{M}\sum_{j=1}^{N} p_{ij}x_{ij} \ + \ \beta \sum_{i_1=1}^{M}\sum_{j_1=1}^{N}\sum_{i_2=1}^{M}\sum_{j_2=1}^{N} a_{j_1 j_2} b_{i_1 i_2} x_{i_1 j_1} x_{i_2 j_2} \quad subject\ to \ C1, \ C2, \ and \ C3.$$

The first term in the cost function is associated with the (constant) cost of assigning a particular component to a particular partition and is called the *linear* term in this paper. It is useful in the MCM/TCM partitioning problem as described in the next section. The leading α is a scaling factor for this term.

The second term in the objective function is associated with the interconnection cost between components. It is called the *Quadratic* term and can be used to model any type of interconnection cost metrics. For example, when B is a matrix of all 1's except all 0's on the main diagonal, this term equals the total number of wire crossings for the given assignment $\mathcal{A}$. When b_{i_1,i_2} is the Manhattan distance from partition i_1 to i_2, this term equals the total Manhattan wire length. Similar arguments apply for quadratic wire length or other forms of cost metrics. The leading β is a scaling factor for this term.

We define the Partitioning Problem described above as $PP(\alpha, \beta)$.

2.2 Applications and Special Cases

2.2.1 MCM/TCM Partitioning

Here we briefly discuss some issues involved in the high level TCM design process. The partitioning process starts with an experienced designer manually assigning functional blocks (components) into TCM chip slots. Since the initial assignment is largely based on intuition and experience rather than calculations

(not much data is available at this stage), there will be lots of constraint violations in the later stage. It is desirable to reassign some components and remove the constraint violations in a way that causes minimum "deviation" from the initial assignment. In other words, given a "initial" component assignment which violates timing and capacity constraints, we want to find a "feasible (legal)" assignment that minimally deviates from the initial assignment.

In a previous work[2] we calculated the deviation of a component by Manhattan distance between the positions of the initial assignment and the final assignment, times the size of that component. This is due to the consideration that a larger component should be less desirable to move. The overall deviation is the sum of all individual component's deviations.

Now assuming we have an initial component assignment $\mathcal{A}_{initial} : J \longrightarrow I$ and s_j is the size of component j. we can compute cost matrix P as the following:

$$p_{ij} = s_j \times Manhattan\_distance(i, \mathcal{A}_{initial}(j))$$

As a result, the *linear* term is exactly the total deviation from an initial assignment. Therefore $PP(1,0)$ is exactly the MCM/TCM partitioning problem described above. This problem is solved in[2].

2.2.2 Generalized Assignment Problem and Linear Assignment Problem

For a Partitioning Problem $PP(1,0)$ with no C2 (Timing Constraints), it is called a Generalized Assignment Problem. This problem has been intensively studied in the past 10 years and efficient heuristics have been found to solve it[12].

For a Generalized Assignment Problem with $M = N$ and $s_j = c_i = \ constant, \ \forall i \in I \ and \ j \in J$, it becomes the well known Linear Assignment Problem. In this case the assignment $\mathcal{A}$ must be a permutation $\varphi : J \longrightarrow J$.

2.2.3 Quadratic Assignment Problem

For a Partitioning Problem $PP(\alpha, \beta)$ with no C2 (Timing Constraints), but with $M = N$ and $s_j = c_i = \ constant, \ \forall i \in I \ and \ j \in J$, it is called a Quadratic Assignment Problem. This is the special case where the assignment $\mathcal{A}$ must be a permutation $\varphi : J \longrightarrow J$.

Despite extensive research on this problem, there had not been effective algorithms or heuristics to solve it until recently[3]. Moreover, all existing methods can only handle problem sizes up to 50 components. Therefore it is not possible to solve a VLSI-layout problem (e.g. gate array placement problem) as a Quadratic Assignment Problem.

3 Transformation into Quadratic Form

Notice that any instance of $PP(\alpha,\beta)$ can be transformed into an instance of $PP'(1,1)$ in the following way: define $p'_{ij} = \alpha p_{ij},\ \forall i,j$ and define $a'_{j_1 j_2} = \beta a_{j_1 j_2},\ \forall j_1, j_2$. Now $PP'(1,1)$ with the new matrices $P' = [p'_{ij}]$ and $A' = [a'_{ij}]$ is equivalent to the original problem $PP(\alpha,\beta)$. Therefore for simplicity of notations, in the rest of this paper we only treat problems in the form of $PP(1,1)$.

3.1 Basic Transformation

Here we derive a relationship which has been used by Burkard[3] without justification.

We transform the 2-dimensional solution matrix $[x_{ij}]$ into a 1-dimensional column vector y of length MN by defining $y_r = x_{ij}$ for $r = i + (j-1) \times M$. We can imagine this transformation as a catenation of all the columns in $[x_{ij}]$ matrix into a tall column vector y. Notice that this transformation is uniquely determined, given a number $1 \le r \le MN$ there is a unique pair (i,j) satisfying $r = i + (j-1) \times M$ and vice versa. Thus y and $[x_{ij}]$ can be considered the same data but in different forms of "packaging". We say that y vector satisfies a certain constraint iff its corresponding $[x_{ij}]$ does.

We define $p'_{i_1 j_1 i_2 j_2} = p_{i_1 j_1}$ if $i_1 = i_2$ and $j_1 = j_2$, and $p'_{i_1 j_1 i_2 j_2} = 0$ otherwise. By this definition, for any fixed i_1 and j_1 we get (since $x_{i_1 j_1}$ is either 0 or 1)

$$p_{i_1 j_1} x_{i_1 j_1} = x_{i_1 j_1}(p_{i_1 j_1} x_{i_1 j_1}) = x_{i_1 j_1} \sum_{i_2=1}^{M} \sum_{j_2=1}^{N} p'_{i_1 j_1 i_2 j_2} x_{i_2 j_2}$$

Now our cost function becomes:

$$\sum_{i_1=1}^{M}\sum_{j_1=1}^{N} p_{i_1 j_1} x_{i_1 j_1} + \sum_{i_1=1}^{M}\sum_{j_1=1}^{N}\sum_{i_2=1}^{M}\sum_{j_2=1}^{N} a_{j_1 j_2} b_{i_1 i_2} x_{i_1 j_1} x_{i_2 j_2} = \sum_{i_1=1}^{M}\sum_{j_1=1}^{N} x_{i_1 j_1}\Big(\sum_{i_2=1}^{M}\sum_{j_2=1}^{N} a_{j_1 j_2} b_{i_1 i_2} x_{i_2 j_2} + \sum_{i_2=1}^{M}\sum_{j_2=1}^{N} p'_{i_1 j_1 i_2 j_2} x_{i_2 j_2}\Big)$$

$$= \sum_{i_1=1}^{M}\sum_{j_1=1}^{N} x_{i_1 j_1}\Big(\sum_{i_2=1}^{M}\sum_{j_2=1}^{N}(a_{j_1 j_2} b_{i_1 i_2} + p'_{i_1 j_1 i_2 j_2}) x_{i_2 j_2}\Big) = \sum_{r_1=1}^{MN} y_{r_1}\Big(\sum_{r_2=1}^{MN}(a_{j_1 j_2} b_{i_1 i_2} + p'_{i_1 j_1 i_2 j_2}) y_{r_2}\Big) = y^T Q y$$

where $Q = [q_{r_1 r_2}]_{MN \times MN}$ is defined by $q_{r_1 r_2} = a_{j_1 j_2} b_{i_1 i_2} + p'_{i_1 j_1 i_2 j_2}$ where $r_1 = i_1 + (j_1 - 1) \times M$ and $r_2 = i_2 + (j_2 - 1) \times M$.

3.2 Transformation to Embed Timing Constraints

Now we have already transformed our original problem into:

$$minimize\ \ y^T Q y, \qquad subject\ to\ C1,\ C2,\ C3$$

where Q is our cost matrix.

Now we wish to embed the Timing Constraints $C2$ into cost matrix Q and solve a Quadratic Boolean Programming Problem without Timing Constraints. We prove in Appendix the following theorem which gives the construction method explicitly to obtain a new cost matrix Q' and that a solution to

$$minimize \;\; y^T Q' y, \qquad subject \; to \; C1, \; C3$$

is exactly a solution to our original problem. Notice in the new problem we do not have C2 (Timing Constraints).

Theorem 1 (Existence of Embedding) *If $\mathcal{F}_{\mathcal{R}}$ in nonempty, $QBP_{\mathcal{R}}(Q)$ is equivalent to $QBP(Q')$, where $Q' = [q'_{ij}]$ is defined by*

$$q'_{r_1 r_2} = q_{r_1 r_2}, \quad \forall (r_1, r_2) \in \mathcal{R}, \quad and \quad q'_{r_1 r_2} = U, \quad \forall (r_1, r_2) \notin \mathcal{R},$$

where

$$U \; is \; any \; number \; s.t. \; U > 2 \times \sum_{r_1=1}^{MN} \sum_{r_2=1}^{MN} |q_{r_1 r_2}|.$$

For a precise definition of the terms used in the theorem, please refer to our Appendix.

Although theoretically correct, a straightforward implementation of the construction for Q' may result in large values in the Q' matrix and introduce numerical inaccuracy for the optimization procedure following it. Therefore we wish to find a larger class of $\hat{Q}$ which will serve the same purpose, i.e., a minimum solution from the new problem ($\hat{Q}$) is a minimum solution of our original problem (Q). This condition is stated in the following theorem:

Theorem 2 (Sufficient Condition for Optimality of Solutions) *Let Q be coincident to $\hat{Q}$ over $\mathcal{R}$ and a minimum solution y^* of $QBP(\hat{Q})$ also satisfies $y^* \in \mathcal{F}_{\mathcal{R}}$. Then y^* is also a minimum solution of $QBP_{\mathcal{R}}(Q)$.*

The idea here is to raise the cost $\hat{q}_{r_1 r_2}$ to some large value for those candidate assignments (i_1, j_1) and (i_2, j_2) in which Timing Constraints are violated. Therefore Timing constraints can be discarded and we can solve the unconstrained (Timing) problem using the new cost matrix. The theorem says that no matter how slightly you raise the values, as long as no timing violation exists in the solution, this solution is guaranteed to be a minimum solution of the original problem. In experiments we set $\hat{q}_{r_1 r_2} = 50$ for those candidate assignments (i_1, j_1) and (i_2, j_2) in which Timing Constraints are violated. And this value is high enough for the optimization procedure to "reject" any timing violating assignments. We emphasize here that the new problem after transformation is *exactly* equivalent to the original problem.

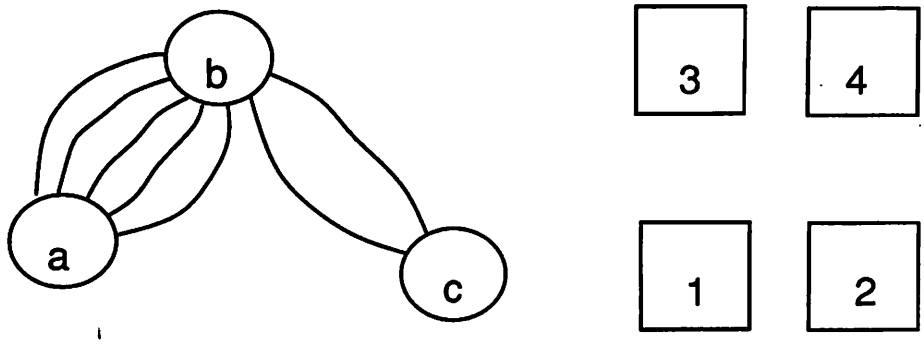

Figure 1:

3.3 An Example

The long proof of equivalence between the original problem and our final form is omitted here. Interested reader can refer to our Appendix for further reading. Here we use an example to illustrate the intuition behind our derivation.

Suppose we have 3 components a, b, c to be assigned into 4 partitions 1, 2, 3, 4, which are located as a 2×2 array as shown. There are five wires connecting a and b; 2 wires connecting b and c.

$$A = \begin{bmatrix} 0 & 5 & 0 \\ 5 & 0 & 2 \\ 0 & 2 & 0 \end{bmatrix}; \qquad D_C = \begin{bmatrix} 0 & 1 & \infty \\ 1 & 0 & 1 \\ \infty & 1 & 0 \end{bmatrix}; \qquad \textit{and} \quad B = D = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix}$$

$$\textit{and} \quad P = \begin{bmatrix} P_{1a} & P_{1b} & P_{1c} \\ P_{2a} & P_{2b} & P_{2c} \\ P_{3a} & P_{3b} & P_{3c} \\ P_{4a} & P_{4b} & P_{4c} \end{bmatrix}$$

Notice that B and D are just Manhattan distance matrices derived from the locations of the partitions assuming adjacent partitions are distance 1 apart.

The cost matrix $\hat{Q}$ is: ("–" means zero entry)

		a 1	a 2	a 3	a 4	b 1	b 2	b 3	b 4	c 1	c 2	c 3	c 4
a	1	p_{1a}	–	–	–	–	5	5	50	–	–	–	–
a	2	–	p_{2a}	–	–	5	–	50	5	–	–	–	–
a	3	–	–	p_{3a}	–	5	50	–	5	–	–	–	–
a	4	–	–	–	p_{4a}	50	5	5	–	–	–	–	–
b	1	–	5	5	50	p_{1b}	–	–	–	–	2	2	50
b	2	5	–	50	5	–	p_{2b}	–	–	2	–	50	2
b	3	5	50	–	5	–	–	p_{3b}	–	2	50	–	2
b	4	50	5	5	–	–	–	–	p_{4b}	50	2	2	–
c	1	–	–	–	–	–	2	2	50	p_{1c}	–	–	–
c	2	–	–	–	–	2	–	50	2	–	p_{2c}	–	–
c	3	–	–	–	–	2	50	–	2	–	–	p_{3c}	–
c	4	–	–	–	–	50	2	2	–	–	–	–	p_{4c}

It is clear that the big matrix has a special structure: we can think of it as a 3×3 matrix, each element of this matrix is actually a 4×4 matrix. Each 4×4 matrix corresponds to an element in matrix A which is located at the same relative position in A as the 4×4 matrix is located in the big matrix. For example, $A(b, c) = 2$ and this element corresponds to the 4×4 matrix located at b row and c column in the big matrix. Discounting the effect of entries being assigned to 50 (for Timing violation), this 4×4 matrix is the B matrix multiplied by the scaler $A(b, c)$. This gives us a quick way of generating $\hat{Q}$ matrix directly from A and B matrices.

Now we explain the effect of embedding Timing Constraints as a high cost entry in $\hat{Q}$. Consider the entry at row $a, 2$ and column $b, 3$ which is 50. This entry corresponds to the assignment of assigning a to 2 and b to 3. It is clear that the the delay between a and b will be $D(2, 3) = 2$ which exceeds the Timing Constraint between a and b: $D_C(a, b) = 1$. Therefore we set it to a high cost 50 to prevent it from happening.

4 Quadratic Boolean Programming

Consider our last form of objective function:

$$minimize \quad y^T \hat{Q} y, \qquad subject\ to\ C1\ and\ C3$$

where y is a column vector of binary values.

If we define our solution space S to be

$$S = \{y |\ y\ satisfies\ C1\ and\ C3\}$$

we get our final form which is a Quadratic Boolean Program

$$\min_{y \in S} y^T \hat{Q} y$$

This is an unconstrained Quadratic Boolean Programming problem in the solution space S with cost matrix $\hat{Q}$ being embedded with Timing Constraints.

4.1 Linearization of Quadratic Boolean Programming

Notice that starting from this point we assume the cost matrix $\hat{Q}$ has non-negative elements. This is a valid assumption for our partitioning problem.

Now let us define ω to be a constant vector satisfying

$$\omega_r \geq \sum_{s=1}^{MN} \hat{q}_{rs} y_s \quad \forall y \in S \quad \forall\, 1 \leq r \leq MN \tag{2}$$

then we define for every solution vector $u^{(k)}$ at iteration k

$$\eta_s^{(k)} = \sum_{r=1}^{MN} \hat{q}_{rs} u_r^{(k)} + \omega_s u_s^{(k)} \quad and \quad \xi^{(k)} = \sum_{r=1}^{MN} \omega_r u_r^{(k)} \tag{3}$$

The following theorem is due to Balas and Mazzola[1].

Theorem 3 *Every optimal solution y^* of*

$$\min_{y \in S} y^T \hat{Q} y$$

corresponds uniquely to an optimal solution of

$$\min_{y \in S} z, \quad such \quad that\ z \geq \sum_{r=1}^{MN} \eta_r^{(k)} y_r - \xi^{(k)} \tag{4}$$

4.2 Burkard's Heuristic

Equation (4) provides a linearization of the Quadratic Programming Problem. However, directly solving equation (4) requires tremendous amount of storage and time. Therefore Burkard[3] proposed the following heuristic.

STEP 1. Initialize $k \leftarrow 1, h_r^{(0)} \leftarrow 0 \ \forall \ 1 \leq r \leq MN$

STEP 2. Compute bounds ω_r for $1 \leq r \leq MN$ according to equation (2)

start with a solution $u^{(1)} \in S$, set $u^* \leftarrow u^{(1)}$ and $z^* \leftarrow (u^*)^T \hat{Q} u^*$.

STEP 3. Compute

$$\eta_s^{(k)} = \sum_{r=1}^{MN} \hat{q}_{rs} u_r^{(k)} \quad and \quad \xi^{(k)} = \sum_{r=1}^{MN} \omega_r u_r^{(k)}$$

STEP 4. Solve

$$z = \min_{u \in S} (\sum_{r=1}^{MN} \eta_r^{(k)} u_r)$$

STEP 5. Compute

$$h_r^{(k)} = h_r^{(k-1)} + \frac{\eta_r^{(k)}}{\max(1, |z - \xi^{(k)}|)}$$

STEP 6. Solve

$$\min_{u \in S} \sum_{r=1}^{MN} h_r^{(k)} u_r^{(k+1)}$$

STEP 7. If $(u^{(k+1)})^T \hat{Q} u^{(k+1)} < z^*$ set $u^* \leftarrow u^{(k+1)}$ and $z^* \leftarrow (u^*)^T \hat{Q} u^*$.

STEP 8. If $k \geq N_{iterations}$ stop. Otherwise set $k \leftarrow k + 1$ go to STEP3.

The overall heuristic is similar to a line search procedure and the user can have precise control over the total runtime. The search stops after a predetermined number of iterations. The best result seen so far becomes the solution to this problem.

4.3 Generalization/Enhancement of Burkard's Heuristic

In Burkard's original paper he was solving for a Quadratic Assignment Problem, therefore the two minimization subproblems in STEP 4 and STEP 6 are in fact Linear Assignment Problems (his solution space S is the set consists of all possible permutations). In our problem, our solution space S consists of all possible assignments which satisfy Capacity Constraints, i.e. the corresponding $[x_{ij}]$ matrices satisfy C1 and C3. Therefore in STEP 4 and STEP 6 we are actually solving Generalized Assignment Problems, which are generalizations of Linear Assignment Problems. We use an existing heuristic due to Martello and Toth[12] to solve the Generalized Assignment Problems.

When computing $\eta^{(k)}$ vector in STEP 3, each $\eta_s^{(k)}$ takes MN multiplications and the total computation for the whole vector is M^2N^2. If the number of partitions is close to the number of components, a single iteration will take N^4 multiplications, which is impractical even for medium sized problems. However, if the number of partitions is small, as it is always true for circuit partitioning problems, the computation can be greatly reduced. Furthermore, if the interconnection matrix A is sparse, the cost matrix $\hat{Q}$ will be sparse. Therefore the computations performed in STEP 3 are mostly multiplying zeros. We take advantage of this fact and avoid these null operations altogether by using a sparse matrix technique. We never explicitly generate $\hat{Q}$ matrix, instead, only the non-zero elements of $\hat{Q}$ are retrieved on demand from a sparse representation derived from connection matrix A. Therefore the summation only take place when both $\hat{q}_{rs}$ and $u_r^{(k)}$ are nonzeros. Also since $u_r^{(k)}$ is binary, we only need addition operations and multiplications can be avoided.

5 Experimental Results

Since there is no existing method for comparison, we developed two partitioning methods based on Kernighan & Lin[7] type of component interchange. The first one is a generalization of Fiduccia & Mattheyses'[8] approach - **GFM**, moving one component at a time. Associated with each component are $(M-1)$ gain entries, each entry representing the potential gain if that component is moved to the corresponding partition.

The second one is a generalization of Kernighan & Lin's heuristic - **GKL**, switching a pair of components at a time. Associated with each component are $(N-1)$ gain entries, each entry representing the potential gain if that component is switched with the corresponding component.

Both methods start with an initial solution with no timing or capacity violations, and the subsequent moves are allowed to take place only when they do not introduce timing or capacity violations. This will guarantee that the final solution will be violation-free. Besides the generalization into M-way partitioning, we also generalized the cost functions used in the gain computations. We allow arbitrary interconnection cost (e.g. Manhattan wire length, quadratic wire length, or just total number of wire crossings) for **GFM** and **GKL**.

A set of 7 industrial examples used in an earlier paper[2] were used to compare all 3 approaches. The sizes of the circuits are shown in the Table I. In each circuit, the components correspond to functional blocks in the high level design and have different sizes ranging about 2 orders of magnitude in the same circuit. The number of partitions is 16.

Strictly speaking, the total number of Timing Constraints should be N^2, where N is the number of components, since there is a delay constraint in any pair of components in our general formulation. However in reality a large number of these constraints are involved with components which do not have actual electrical connection or cycle time constraints between them. We discarded these constraints and only list the total number of critical constraints in the table to show the degree of timing-criticality of the problem.

In our Quadratic Boolean Programming approach - QBP, each circuit runs 100 iterations. (Notice that the solution quality is dependent on the number of iterations, the more CPU time spent, the better the results.) In GFM, each circuit runs till no more improvement is possible. In GKL, we have to force the algorithm to terminate after the first 6 outer loops due to excessive CPU runtime. Since any gain obtained beyond the first 6 outer loops is insignificant, this cutoff strategy provides speedup without sacrificing solution quality.

For GFM and GKL, an initial **feasible** solution is needed in order to guarantee the feasibility of the final solution. The fastest way to obtain a initial feasible solution is to use QBP algorithm with matrix B set to all zeros. This will generate an initial feasible solution in a few iterations. This same initial solution is used for all three approaches. In these tests we use total Manhattan wire length as our cost metric. Table II is results obtained by relaxing Timing Constraints and Table III is results with Timing Constraints. In both tables the second column is the cost (total Manhattan wire length) of the initial solution. The third column is the final cost from QBP, the fourth column is the percentage improvement, the fifth column is the CPU time in seconds on DECstation 5000/125.

Among three methods, GFM spent the least CPU but produced the worst results, especially when Timing Constraints are present. On the other hand, QBP obtained the best quality results within reasonable CPU time. The results from GKL are better than GFM because it explored a larger solution space at the cost of using much more CPU time. Notice that both GFM and GKL need to start with an initial feasible solution, (we actually obtained this initial solution by QBP), while QBP can start from any random solution. In our separate experiments we discovered that QBP maintained the same kind of good results from any arbitrary initial solution.

I. circuit descriptions:

ckt	# of components	# of wires	# of Timing Constraints
ckta	339	8200	3464
cktb	357	3017	1325
cktc	545	12141	11545
cktd	521	6309	6009
ckte	380	3831	3760
cktf	607	4809	4683
cktg	472	3376	3376

II. Without Timing Constraints:

		QBP			GFM			GKL		
circuits	start	final	(- %)	cpu	final	(- %)	cpu	final	(- %)	cpu
ckta	20756	17457	15.9	86.8	18894	9.0	12.2	17526	15.6	544.3
cktb	8239	5996	27.2	43.4	6966	15.5	18.5	6555	20.4	148.2
cktc	28210	20711	26.6	140.2	23185	17.8	37.1	20647	26.8	1192.0
cktd	14737	9724	34.0	97.1	12894	12.5	46.1	11780	20.1	608.4
ckte	8524	6293	26.2	58.3	6746	20.9	20.8	6329	25.8	298.3
cktf	10498	5887	44.0	93.4	7589	27.7	24.1	6643	36.7	514.1
cktg	8138	5170	36.5	64.1	5925	27.2	15.5	5951	26.9	354.7

III. With Timing Constraints:

		QBP			GFM			GKL		
circuits	start	final	(- %)	cpu	final	(- %)	cpu	final	(- %)	cpu
ckta	20756	18233	12.2	89.2	19341	6.8	9.4	18262	12.0	394.4
cktb	8239	6482	21.3	44.5	7054	14.4	9.0	7225	12.3	121.7
cktc	28210	22228	21.2	139.3	26195	7.1	51.9	21435	24.0	1887.5
cktd	14737	11278	23.5	100.7	13568	7.9	27.6	12866	12.7	558.6
ckte	8524	6758	21.0	58.0	7913	7.2	11.7	7218	15.3	230.0
cktf	10498	6916	34.1	94.4	8294	21.0	45.4	7627	27.3	492.5
cktg	8138	5721	30.1	65.9	6454	21.0	18.8	6014	26.1	313.6

6 Conclusion

We discovered and mathematically proved the exact problem formulation/translation of the partitioning problem under Timing and Capacity Constraints into a Quadratic Boolean Programming problem. We also found an effective solution and demonstrated it through industrial examples.

References

[1] E. Balas and J.B. Mazzola, "Quadratic 0-1 Programming by a New Linearization," Presented at the joint ORSA/TIMS National Meeting, Washington, D.C. 1980,

[2] M. Shih, E.S. Kuh and R.-S. Tsay, "Integer Programming Techniques for Multiway System Partitioning under Timing and Capacity Constraints" *Memorandum No. UCB/ERL M92/81, University of California at Berkeley, also to appear in EDAC 1993*

[3] R.E. Burkard and T. Bonniger, "A Heuristic for Quadratic Boolean Programs with Applications to Quadratic Assignment Problems," *European Journal of Operational Research,* 1983, 13, 372-386,

[4] E.R. Barnes, "An Algorithm for Partitioing the Nodes of a Graph," *SIAM Journal of Algebraic and Discrete Methods,* 1982, 3(4):541-550,

[5] E.R. Barnes, A. Vannelli, and J.Q. Walker, "An New Heuristic for Partitioing the Nodes of a Graph," *SIAM Journal of Discrete Mathematics,* 1988, 1(3):299-305,

[6] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout,* 1990, Chap 6, pp:285-286,

[7] B.W. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *The Bell System Technical Journal,* 49(2). Feb. 1970. pp. 291-307

[8] C.M. Fiduccia and R.M. Mattheyses, "An Linear Time Heuristic for Improving Network Partitions," *Proc. 19th Design Automation Conference,* 1982, pp. 175-181

[9] Y. Wei and C. Cheng, "Toward Efficient Hierarchical Designs by Ratio Cut Partitioning," *Proc. IEEE Int. Conf. on Computer-Aided Design,* 1989, pp. 298-301

[10] Y. Wei and C. Cheng, "A Two-Level Two-Way Partitioning Algorithm," *Proc. IEEE Int. Conf. on Computer-Aided Design,* 1990, pp. 516-519

[11] R.R. Tummala and E.J. Rymaszewski, "Microelectronics Packaging Handbook, Chap. 16" Van Nostrand Reinhold, 1989

[12] S. Martello and P. Toth, "Knapsack Problems" 1990, Chap 7, pp. 189-220

[13] M. Shih, E. Kuh and R. Tsay, "Performance-Driven System Partitioning on Multi-Chip Modules," *Proc. Design Automation Conference,* 1992, pp. 53-56

[14] K. Jornsten, M. Nasberg, "A new Lagrangian relaxation approach to the generalized assignment problem," *European Journal of Operational Research 27,* 1986, pp. 313-323

APPENDIX

This appendix includes all our original theoretical work on the transformation technique from a **constrained Quadratic Boolean Programming Problem** into an **unconstrained Quadratic Boolean Programming Problem**. This technique can be applied to a general class of constraints as long as they can be written in the form of a *Region of Feasible Pairs* (defined later).

We transform a constrained Quadratic Boolean Programming Problem into an unconstrained Quadratic Boolean Programming Problem by embedding the constraints into the quadratic cost matrix Q to form a new constraint-embedded cost matrix Q'. In our first theorem we will explicitly give the construction method and prove the equivalence between the two problems. Although the existence of such an embedding can be proven, the construction method in the first theorem may or may not produce values too large and introduce numerical inaccuracy in the optimization procedure that follows it. Therefore we extended our embedding method to a larger, more general set of constraint-embedded cost matrix $\hat{Q}$. In the second theorem we will prove the sufficient condition for a minimum solution from the unconstrained problem to be a minimum solution of our original constrained problem. The third theorem is a generalization of the second, we discovered it after we gained some insight into the nature of the problem. It can be applied to other constrained optimization problem as well. Although it resembles some existing penalty or barrier methods, the underlying concept is quite different.

To make the appendix self-contained, we start with the general settings to be used in the rest of the appendix.

1 General Settings

1.1 Basic Definitions

Let J and I be two non-empty sets and $|I| = M$ and $|J| = N$.

Let $[x_{ij}]_{M\times N}$ be a matrix of binary values. Now define a binary column vector y of length MN by $y_r = x_{ij}$ where $r = i + (j-1) \times M,\ \ 1 \leq r \leq MN$. Notice that given r there is a unique (i,j) corresponding to it and vice versa. The vector y is just a rearrangement of the elements of $[x_{ij}]$. We refer to these two different representations "$[x_{ij}]$" and "y" with the understanding that they correspond to the same solution. We say y satisfies certain constraints iff its corresponding $[x_{ij}]$ does.

1.2 The Partitioning Problem

In our application to the partitioning problem, we use $[x_{ij}]$ to represent an assignment $\mathcal{A} : J \longrightarrow I$ by defining

$$x_{ij} = 1, \ if \mathcal{A}(j) = i \quad and \quad x_{ij} = 0, \quad otherwise.$$

As a consequence there is an additional constraint which is usually called Generalized Upper Bound Constraints in related literature

$$\sum_{i=1}^{M} x_{ij} = 1, \quad \forall j \in J$$

Notice that our general theorems do not require such a constraint on $[x_{ij}]$.

The "physical meaning" of any pairs $(i_1, j_1), (i_2, j_2)$ such that $x_{i_1 j_1} = x_{i_2 j_2} = 1$ is that component j_1 is assigned to partition i_1 and component j_2 is assigned to partition i_2. Therefore the Timing Constraints for the partitioning problem becomes

$$D(i_1, i_2) \leq D_C(j_1, j_2), \quad \forall \ (i_1, j_1) \ \ and \ \ (i_2, j_2) \ \ s.t. \ \ x_{i_1 j_1} = 1 = x_{i_2 j_2}$$

As a conclusion, we have the following 3 sets of constraints for $[x_{ij}]$

C1: (Capacity Constraints)

$$\sum_{j=1}^{N} s_j x_{ij} \leq c_i, \quad \forall i \in I$$

C2: (Timing Constraints)

$$D(i_1, i_2) \leq D_C(j_1, j_2), \quad \forall \ (i_1, j_1) \ \ and \ \ (i_2, j_2) \ \ s.t. \ \ x_{i_1 j_1} = 1 = x_{i_2 j_2}$$

C3: (Generalized Upper Bound Constraints)

$$\sum_{i=1}^{M} x_{ij} = 1, \quad \forall j \in J$$

1.3 Definitions

Definition 1 (Region of Feasible Pairs) *A* **Region of Feasible Pairs** $\mathcal{R}$ *is an arbitrary subset of* $\{(r_1, r_2) \mid \forall r_1, r$ $r_1 \leq MN \ \ and \ \ 1 \leq r_2 \leq MN \ \}$.

A set $\mathcal{R}$ defines a relation. We can imagine any ordered pair $(r_1, r_2) \in \mathcal{R}$ means r_1 and r_2 is in this relationship, which means that r_1 is "Constraint Feasible" to r_2. This relation is not necessarily reflexive,

symmetric or transitive. Therefore r_1 being "Constraint Feasible" to r_2 does not have to imply r_2 being "Constraint Feasible" to r_1.

In the special case of Timing Feasibility relationship, we define $(r_1, r_2) \in \mathcal{R}$ if and only if

$$D(i_1, i_2) \leq D_C(j_1, j_2),$$

where $r_1 = i_1 + (j_1 - 1) \times M$ and $r_2 = i_2 + (j_2 - 1) \times M$. Therefore the "physical meaning" of r_1 "Constraint Feasible" to r_2 is that component j_1 assigned to partition i_1 and component j_2 assigned to partition i_2 satisfies the timing constraint from j_1 to j_2.

Definition 2 (Feasible Solution Set Over $\mathcal{R}$) *A* **Feasible Solution Set** $\mathcal{F}$ *over* $\mathcal{R}$ *is defined as*

$$\mathcal{F}_{\mathcal{R}} = \{y \in S \mid y_{r_1} = y_{r_2} = 1 \; implies \; (r_1, r_2) \in \mathcal{R}\},$$

where S *is our underlying solution space.*

A solution vector $y \in \mathcal{F}_{\mathcal{R}}$ has the property that the 1-valued coordinates of y are all Constraint Feasible to each other.

Again in the special case of Timing-feasibility, the solution space S is defined by

$$S = \{y \in \{0,1\}^{MN} \mid y \; satisfies \; C1 \; and \; C3\}.$$

And the "physical meaning" of $y \in \mathcal{F}_{\mathcal{R}}$ is that y satisfies all Timing Constraints. Therefore any solution $y \in \mathcal{F}_{\mathcal{R}}$ is a feasible solution which satisfies $C1$, $C2$ and $C3$.

2 Problem Definitions

Let Q be an $MN \times MN$ matrix with real values, we define a regular **Quadratic Boolean Programming Problem**

$$\mathbf{QBP(Q)}: \quad \min_{y \in S} y^T Q y$$

We define a **Quadratic Boolean Programming Problem with a Region of Feasible Pairs** $\mathcal{R}$

$$\mathbf{QBP_{\mathcal{R}}(Q)}: \quad \min_{y \in \mathcal{F}_{\mathcal{R}}} y^T Q y$$

2.1 Coincident Matrices

Let $Q = [q_{r_1r_2}]_{M\times N}$ and $\hat{Q} = [\hat{q}_{r_1r_2}]_{M\times N}$ be two real matrices.

Definition 3 (Coincident Matrices Over $\mathcal{R}$) *Q is* **coincident** *to $\hat{Q}$ over $\mathcal{R}$ if $q_{r_1r_2} = \hat{q}_{r_1r_2}$ $\forall(r_1, r_2) \in \mathcal{R}$*

Lemma 1 *Let Q be coincident to $\hat{Q}$ over $\mathcal{R}$, then $y^TQy = y^T\hat{Q}y$ $\forall y \in \mathcal{F}_{\mathcal{R}}$.*

Proof.

$$y^TQy = \sum_{r_1=1}^{MN} y_{r_1}(\sum_{r_2=1}^{MN} q_{r_1r_2}y_{r_2}) = \sum_{\forall(r_1,r_2)\ s.t.\ y_{r_1}=y_{r_2}=1} q_{r_1r_2},$$

but since $y \in \mathcal{F}_{\mathcal{R}}$, $y_{r_1} = y_{r_2} = 1$ implies $(r_1, r_2) \in \mathcal{R}$, therefore $q_{r_1r_2} = \hat{q}_{r_1r_2}$ because of coincidency and the equation becomes

$$\sum_{\forall r_1,r_2,\ s.t.\ y_{r_1}=y_{r_2}=1} q_{r_1r_2} = \sum_{\forall r_1,r_2,\ s.t.\ y_{r_1}=y_{r_2}=1} \hat{q}_{r_1r_2} = \sum_{r_1=1}^{MN} y_{r_1}(\sum_{r_2=1}^{MN} \hat{q}_{r_1r_2}y_{r_2}) = y^T\hat{Q}y.$$

□

3 Embedding Theorems

Theorem 2 (Existence of Embedding) *If $\mathcal{F}_{\mathcal{R}}$ in nonempty, $QBP_{\mathcal{R}}(Q)$ is equivalent to $QBP(Q')$, where $Q' = [q'_{ij}]$ is defined by*

$$q'_{r_1r_2} = q_{r_1r_2}, \quad \forall(r_1, r_2) \in \mathcal{R}, \quad \textit{and} \quad q'_{r_1r_2} = U, \quad \forall(r_1, r_2) \notin \mathcal{R},$$

where

$$U \textit{ is any number s.t. } U > 2 \times \sum_{r_1=1}^{MN}\sum_{r_2=1}^{MN} |q_{r_1r_2}|.$$

Proof.

We notice that by definition, Q is coincident to Q' over $\mathcal{R}$.

PART I: Let $y^*_{\mathcal{R}}$ be a minimum solution of $QBP_{\mathcal{R}}(Q)$, see that $y^*_{\mathcal{R}}$ is a minimum solution of $QBP(Q')$.

Since $y^*_{\mathcal{R}}$ is a solution of $QBP_{\mathcal{R}}(Q)$, $y^*_{\mathcal{R}} \in \mathcal{F}_{\mathcal{R}} \subseteq S$. Suppose on the contrary that there exists $y \in S$ such that

$$y^TQ'y < (y^*_{\mathcal{R}})^TQ'y^*_{\mathcal{R}} \tag{1}$$

Case 1: If $y \in \mathcal{F}_{\mathcal{R}}$, by Lemma we have $y^TQy = y^TQ'y$. Since $y^*_{\mathcal{R}} \in \mathcal{F}_{\mathcal{R}}$ (by definition of $y^*_{\mathcal{R}}$) we also have $(y^*_{\mathcal{R}})^TQy^*_{\mathcal{R}} = (y^*_{\mathcal{R}})^TQ'y^*_{\mathcal{R}}$, now by inequality (1) we have

$$y^TQy = y^TQ'y < (y^*_{\mathcal{R}})^TQ'y^*_{\mathcal{R}} = (y^*_{\mathcal{R}})^TQy^*_{\mathcal{R}},$$

the relationship between the first and last term and the fact that $y \in \mathcal{F}_{\mathcal{R}}$ contradicts that $y^*_{\mathcal{R}}$ is a minimum solution of $QBP_{\mathcal{R}}(Q)$.

Case 2: If $y \notin \mathcal{F}_{\mathcal{R}}$, by definition of $\mathcal{F}_{\mathcal{R}}$ we can find $(r_1, r_2) \notin \mathcal{R}$ $s.t.$ $y_{r_1} = y_{r_2} = 1$. Now by definition of Q',

$$y^TQ'y = \sum_{\forall(t_1,t_2),\ s.t.\ y_{t_1}=y_{t_2}=1} q'_{t_1t_2} = q'_{r_1r_2} + \sum_{\forall(t_1,t_2)\neq(r_1,r_2),\ s.t.\ y_{t_1}=y_{t_2}=1} q'_{t_1t_2}$$

By definition of Q' the last expression is actually

$$U + \sum_{\forall(t_1,t_2)\neq(r_1,r_2),\ s.t.\ y_{t_1}=y_{t_2}=1} q'_{t_1t_2} > \sum_{r_1=1}^{MN}\sum_{r_2=1}^{MN} |q_{r_1r_2}| \geq v^TQv, \quad \forall v \in \{0,1\}^{MN}$$

Let $y^*_{\mathcal{R}} = v \in \{0,1\}^{MN}$, we get

$$y^TQ'y > (y^*_{\mathcal{R}})^TQy^*_{\mathcal{R}} = (y^*_{\mathcal{R}})^TQ'y^*_{\mathcal{R}}$$

(The last equality is due to $y^*_{\mathcal{R}} \in \mathcal{F}_{\mathcal{R}}$.) Now this contradicts inequality (1).

PART II: Let y^* be a minimum solution of $QBP(Q')$, see that y^* is a minimum solution of $QBP_{\mathcal{R}}(Q)$.

We first show that $y^* \in \mathcal{F}_{\mathcal{R}}$. Suppose contrary that $y^* \notin \mathcal{F}_{\mathcal{R}}$, then by a previous argument in **Case 2, PART I** we have

$$(y^*)^TQ'y^* > v^TQv, \quad \forall v \in \{0,1\}^{MN} \tag{2}$$

Since $\mathcal{F}_{\mathcal{R}}$ is nonempty, there exists $y_{\mathcal{R}} \in \mathcal{F}_{\mathcal{R}}$, by Lemma we have $y^T_{\mathcal{R}}Qy_{\mathcal{R}} = y^T_{\mathcal{R}}Q'y_{\mathcal{R}}$. But from inequality (2) and letting $y_{\mathcal{R}} = v$ we have

$$(y^*)^TQ'y^* > y^T_{\mathcal{R}}Qy_{\mathcal{R}} = y^T_{\mathcal{R}}Q'y_{\mathcal{R}}$$

This contradicts that y^* is a minimum of $QBP(Q')$.

We then show that y^* is indeed a minimum of $QBP_{\mathcal{R}}(Q)$. Since $y^* \in \mathcal{F}_{\mathcal{R}}$ we have

$$(y^*)^TQy^* = (y^*)^TQ'y^* \leq y^TQ'y, \quad \forall y \in S$$

(the last inequality holds because y^* is a minimum of $QBP(Q')$.)

But for all $y \in \mathcal{F}_{\mathcal{R}}$, by Lemma $y^T Q y = y^T Q' y$. Therefore we get

$$(y^*)^T Q y^* = (y^*)^T Q' y^* \leq y^T Q' y = y^T Q y, \quad \forall y \in \mathcal{F}_{\mathcal{R}}$$

Therefore y^* is a minimum of $QBP_{\mathcal{R}}(Q)$. □

Theorem 3 (Sufficient Condition for Optimality of Solutions) *Let Q be coincident to $\hat{Q}$ over $\mathcal{R}$ and a minimum solution y^* of $QBP(\hat{Q})$ also satisfies $y^* \in \mathcal{F}_{\mathcal{R}}$. Then y^* is also a minimum solution of $QBP_{\mathcal{R}}(Q)$.*

Proof. Proof by Contradiction. Suppose on the contrary $y_{\mathcal{R}} \in \mathcal{F}_{\mathcal{R}}$ is a solution of $QBP_{\mathcal{R}}(Q)$ and that

$$y_{\mathcal{R}}^T Q y_{\mathcal{R}} < (y^*)^T Q y^* \tag{3}$$

Since $y_{\mathcal{R}} \in \mathcal{F}_{\mathcal{R}}$, we have

$$y_{\mathcal{R}}^T Q y_{\mathcal{R}} = y_{\mathcal{R}}^T \hat{Q} y_{\mathcal{R}} \tag{4}$$

By the given condition $y^* \in \mathcal{F}_{\mathcal{R}}$ we also have

$$(y^*)^T Q y^* = (y^*)^T \hat{Q} y^* \tag{5}$$

Combining (3),(4) and (5) we get

$$y_{\mathcal{R}}^T \hat{Q} y_{\mathcal{R}} = y_{\mathcal{R}}^T Q y_{\mathcal{R}} < (y^*)^T Q y^* = (y^*)^T \hat{Q} y^*$$

The inequality between the first and last term contradicts that y^* is a minimum of $QBP(\hat{Q})$. □

Theorem 4 (General Sufficient Condition for Optimality of Solutions) *Let $f(y) = \hat{f}(y), \forall y \in \mathcal{F}_{\mathcal{R}} \subseteq S$ and y^* minimize $\hat{f}$ over S. Suppose $y^* \in \mathcal{F}_{\mathcal{R}}$, then y^* also minimizes f over $\mathcal{F}_{\mathcal{R}}$.*

Proof. Suppose contrary that $\exists y \in \mathcal{F}_{\mathcal{R}}$ and $f(y) < f(y^*) \quad \Rightarrow \hat{f}(y) = f(y) < f(y^*) = \hat{f}(y^*)$
$\Rightarrow y^*$ does not minimize $\hat{f}$ over $\mathcal{F}_{\mathcal{R}} \quad \Rightarrow y^*$ does not minimize $\hat{f}$ over S (contradiction)
□

Our original problem is in the form of:

$$minimize \;\; y^T Q y, \qquad subject \; to \; C1, \; C2, \; C3$$

where $f(y) = y^T Q y$ and Q is our cost matrix.

Let $\mathcal{F}_{\mathcal{R}} = \{y | y \ satisfies \ C1, C2, C3\}$ and $S = \{y | y \ satisfies \ C1, C3\}$, where S is the (larger) solution space without timing constraints. Now we construct a new cost matrix $\hat{Q}$ such that $y^T Q y = y^T \hat{Q} y \ \ \forall y \in \mathcal{F}_{\mathcal{R}}$ and obtain a minimizer y^* of this new function $y^T \hat{Q} y$ in S. If y^* also satisfies $C2$, by the theorem we know that y^* is also a minimizer of $y^T Q y$ over $\mathcal{F}_{\mathcal{R}}$. One intuitive way to construct $\hat{Q}$ is to raise the cost $\hat{q}_{r_1 r_2}$ to some large value for those candidate assignments (i_1, j_1) and (i_2, j_2) in which Timing Constraints are violated. Then Timing constraints can be discarded and we can solve the unconstrained problem using the new cost matrix $\hat{Q}$:

$$minimize \ \ y^T \hat{Q} y, \qquad subject \ to \ C1, \ C3$$

The theorem tells us that no matter how much we augment the values in the cost matrix, as long as no timing violation exists in the solution, this solution is guaranteed to be a minimum solution of the original problem. In experiments we set this value to be 50. This value is high enough for the optimization procedure to "reject" any timing violating assignments. We emphasize here that the new problem after transformation is *exactly* equivalent to the original problem.