INTEGER PROGRAMMING TECHNIQUES FOR MULTIWAY SYSTEM PARTITIONING UNDER TIMING AND CAPACITY CONSTRAINTS

by

Minshine Shih and Ernest S. Kuh

Memorandum No. UCB/ERL M92/81

10 August 1992
(Revised 19 October 1992)

COVER PAGE

INTEGER PROGRAMMING TECHNIQUES FOR MULTIWAY SYSTEM PARTITIONING UNDER TIMING AND CAPACITY CONSTRAINTS

by

Minshine Shih and Ernest S. Kuh

Memorandum No. UCB/ERL M92/81

10 August 1992
(Revised 19 October 1992)

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

Integer Programming Techniques for Multiway System Partitioning Under Timing and Capacity Constraints

Minshine Shih
Ernest S. Kuh
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
California 94720

October 19, 1992

Abstract

We propose efficient and effective algorithms for MCM system partitioning under timing and capacity constraints. We take advantage of the small number of chip slots (compared to the large number of circuit modules) and formulate it as an Integer Programming problem. We propose a novel approach called **Constraints Decoupling**, which decomposes the original dual-constraint problem into two independent single-constraint problems, which are much easier to solve. Experimental results are encouraging, all timing and capacity violations in the initial designs can be eliminated.

1 Introduction

The emerging MCM (Multi-Chip Module) technology has greatly increased the need for a constrained multi-way partitioning algorithm. Clearly each chip in MCM corresponds to a partition and we wish to assign the set of circuit modules into the set of partitions (chips).

Due to the physical limitation of MCM technology, each chip has a limit (Capacity Constraint) on the maximum amount of circuitry it can implement. There is also a non-zero chip-to-chip delay introduced by traversing the chip I/O boundary and by MCM (interchip) wiring. Therefore the solution to the partitioning problem must also satisfy the Timing Constraint. Note that the number of partitioning (chips) in practice is usually small(2-100).

In this paper we discuss a general method for partitioning a network of N interconnected circuit modules into M subnetworks (i.e. M-way partitioning), where M is a fixed given number. Traditional methods for M-way partitioning have been based on applying recursive two-way partitionings, e.g. Min-Cut partitioning. The main drawback of this approach is the unnecessary restrictions imposed by higher level "cuts" on the freedom of assigning modules to partitions in the lower level.

It is possible to use 2-way ratio-cut[1][2] to identify natural clusters in the circuit and regroup these clusters into M-way partitions. However none of the existing M-way partitioning algorithms can take multiple "hard" constraints (e.g. capacity constraints of partitions and timing constraints between modules) into considerations. These two constraints are often conflicting and interacting with each another. For this problem we take advantage of the small number of partitions (relative to the number of modules) and formulate it as an integer programming problem. (Notice that an integer programming formulation can not be solved efficiently if the number of partitions is in the same order of number of modules.)

In this paper we propose a new method called "Constraint Decoupling", i.e., decoupling the original 2-constraint problem into two single-constraint subproblems A and B, which can be solved efficiently by exact algorithms or heuristics. The difference between two solutions from A and B can be used as a penalty term to update the cost functions of A and B in the next iteration. When solutions from A and B converge to the same solution, we obtain a solution which satisfies both capacity and timing constraints.

We use TCM (Thermal Conduction Module)[3] as a test case for this idea. All ideas discussed in this paper are applicable in a general M-way setting and TCM is one of the possible applications.

Section 2 describes our general M-way formulation and possible applications. Section 3 describes a chain of problem transformation. Sections 4 describes our main algorithms. Section 5 shows experimental results.

2 Problem Formulation for General M-way Partitioning and Applications

2.1 Formulation

The input to the problem includes the followings:

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

2). J is a set of N modules. Let $j \in J$ be the index to a module.

3). s_j is the size of module j, representing the module's area.

4). c_i is the capacity of partition i.

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

6). D_P is an $M \times M$ matrix, where $D_P(i_1, i_2)$ is the (abstract) routing distance between partitions i_1 and i_2.

7). D_C is an $N \times N$ matrix, where $D_C(j_1, j_2)$ is the maximum (abstract) distance allowed between modules j_1 and j_2.

A solution to the problem is a mapping $F : J \longrightarrow I$ satisfying the following two constraints:

C1: (Capacity Constraints) $\sum_{\forall j, F(j)=i} s_j \leq c_i, \quad \forall i \in I$

C2: (Timing Constraints) $D_P(F(j_1), F(j_2)) \leq D_C(j_1, j_2), \quad \forall j_1, j_2 \in J$

The objective is to

$$minimize \sum_{\forall i,j,F(j)=i} p_{ij}, \quad subject\ to\ C1,\ C2.$$

2.2 Application to TCM Partitioning

Our objective is to minimize the overall cost of assigning modules to partitions. Different value settings of the P matrix have direct influence on the solutions. Therefore this formulation is quite flexible in terms of adapting to different applications. For example, if the designer wishes to fix module 3 in partition 5, he can set $p_{5,3}$ to a sufficiently low value. Or if he wishes module 3 *not* to be in partition 5, he can set $p_{5,3}$ to a sufficiently high value. If he has equal preference for module 3 to go to partitions 2 and 4, he can set $p_{2,3}$ and $p_{4,3}$ to the same value.

We also notice that D_P can be used to specify any variety of delay models, since it explicitly specifies (abstract) routing distance between any pair of two partitions. In other words, the delay is specified in an *abstract* sense, not in a physical sense, since it assumes no physical configuration.

In the case of TCM partitioning, we immediately observe that each TCM chip slot corresponds to a partition. The matrix D_P can be easily derived from the physical or electrical characteristics of the TCM. The only remaining question is how to derive the cost matrix P.

To answer the question we must first examine the TCM partitioning process. The process starts with an experienced designer manually assigning functional blocks (modules) into TCM chip slots. Since the initial assignment is largely based on intuitions and experiences rather than calculations (not much data available at this stage), there will be lots of constraint violations in the later stage. As the high level design evolves, more accurate estimates on the sizes and delays of the modules become available. The designer is forced to move some modules across chip slots in order to avoid constraint violations. Since these movements cause a ripple effect to other modules, the whole process becomes quite complex and labor intensive. Furthermore, the cut/move procedure is needed throughout the whole high level design phase and contributes greatly to the overall design time.

It is clear that we need an automatic tool to move the modules to satisfy constraints. It is desirable to do so in a way that causes minimum "disturbance" to the initial assignment. In other words, given a "initial" modules assignment which violates timing and capacity constraints, we want to find a "feasible(legal)" assignment that deviates from initial assignment minimally. There are many ways to calculate the amount of deviation. For example, module 3 is initially assigned to slot 1 and gets reassigned to slot 6 in the feasible assignment, then the deviation of module 3 is the distance (Manhattan or Euclidean) between slot 6 and slot 1. In this work we calculate the deviation of a module by Manhattan distance times the size of the module. This is due to the consideration that a larger module should be less prone to move. The overall deviation is the sum of all individual module's deviation.

Now assuming we already have a (possibly illegal) module assignment $F_{initial} : J \longrightarrow I$ and s_j is the size of module j. we can compute cost matrix P as the following:

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

As a result, we are *exactly* minimizing the overall weighted deviation from a initial assignment.

3 Problem Transformations

3.1 Transformation into Integer Programming Problem

The problem formulation proposed in section 2 can be transformed into an integer programming problem by introducing an $M \times N$ matrix X of binary variables which prescribes the mapping. We define $x_{ij} = 1$ if module j is assigned to partition i and $x_{ij} = 0$ otherwise.

We note that C1, C2 and the objective function can be rewritten using this transformation (rewriting C2 is avoided here for simplicity). We also added implicit constraints C3 for X matrix.

C1: (Capacity Constraints) $\sum_{j=1}^{N} s_j x_{ij} \leq c_i, \quad \forall i \in I$

C2: (Timing Constraints) $D_P(F(j_1), F(j_2)) \leq D_C(j_1, j_2), \quad \forall j_1, j_2 \in J$

C3: (Generalized Upper Bound Constraints) $\sum_{i=1}^{M} x_{ij} = 1, \quad \forall j \in J$

The objective is to

$$minimize \sum_{i=1}^{M}\sum_{j=1}^{N} p_{ij}x_{ij}, \quad subject\ to\ C1,\ C2,\ C3.$$

3.2 Transformation by Variables Splitting and Constraints Decoupling

Our idea of Constraints Decoupling came from the Variable Splitting method introduced by Jornsten and Nasberg[6], who used this method to derive a tight upper bound in Branch and Bound algorithm which solved Generalized Assignment Problems (only capacity constraints were considered). Our main contribution here is to derive a nontrivial generalized formulation which can handle Timing Constraints and extend their basic Variable Splitting method to decouple timing and capacity constraints.

We introduce A and B ($M \times N$ binary) matrices and transform the original integer programming problem into the followings:

C1: (Capacity Constraints for A) $\sum_{j=1}^{N} s_j a_{ij} \le c_i, \quad \forall i \in I$

C2: (Timing Constraints for B) $D_P(F_B(j_1), F_B(j_2)) \le D_C(j_1, j_2), \quad \forall j_1, j_2 \in J$

where F_B is an assignment corresponding to binary matrix B.

C3: (Generalized Upper Bound Constraints for A) $\sum_{i=1}^{M} a_{ij} = 1, \quad \forall j \in J$

C4: (Generalized Upper Bound Constraints for B) $\sum_{i=1}^{M} b_{ij} = 1, \quad \forall j \in J$

C5: (A/B Equivalence Constraint) $A = B$

The objective is to

$$minimize\ (\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} p_{ij}a_{ij} + \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} p_{ij}b_{ij}), \quad subject\ to\ C1,\ C2,\ C3,\ C4,\ C5.$$

The optimum solution $A = B$ from this new formulation is equal to the optimum solution X for the original integer programming problem. We omit the proof of equivalence here but give some intuition for the reason of doing so. Notice in this new formulation A matrix is subject to constraints C1, C3, C5 and B matrix is subject to constraints C2, C4 and C5. The only interaction between A and B is through C5. If we relax C5 and put a penalty term corresponding to C5 into the cost function, we can decouple the problem into two independent subproblems and easily solve them individually. This is discussed in the next section.

3.3 Lagrange Relaxation

As a standard technique we dualize C5 and add a penalty term into the cost function to get:

$$\max_{\forall U} \min_{\forall A,B}(\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} p_{ij}a_{ij} + \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{N} p_{ij}b_{ij} + \sum_{i=1}^{M}\sum_{j=1}^{N} u_{ij}(a_{ij} - b_{ij})), \quad subject\ to\ C1,\ C2,\ C3,\ C4.$$

where U is an $M \times N$ binary matrix corresponding to Lagrange Multipliers.

We can further simplify the expression into:

$$\max_{\forall U} \min_{\forall A,B}(\sum_{i=1}^{M}\sum_{j=1}^{N}(\frac{1}{2}p_{ij} + u_{ij})a_{ij} + \sum_{i=1}^{M}\sum_{j=1}^{N}(\frac{1}{2}p_{ij} - u_{ij})b_{ij}) \quad subject\ to\ C1,\ C2,\ C3,\ C4.$$

Note that the first term in this expression involves only matrices A and U, and the second term involves only B and U. Moreover A is only involved in C1 and C3 and B is only involved in C2 and C4. For any given U, we can solve

$$\min_{\forall A,B}(\sum_{i=1}^{M}\sum_{j=1}^{N}(\frac{1}{2}p_{ij} + u_{ij})a_{ij} + \sum_{i=1}^{M}\sum_{j=1}^{N}(\frac{1}{2}p_{ij} - u_{ij})b_{ij})$$

by solving

Subproblem A:

$$\min_{\forall A}(\sum_{i=1}^{M}\sum_{j=1}^{N}(\frac{1}{2}p_{ij} + u_{ij})a_{ij}) \quad subject\ to\ C1,\ C3.$$

and Subproblem B:

$$\min_{\forall B}(\sum_{i=1}^{M}\sum_{j=1}^{N}(\frac{1}{2}p_{ij} - u_{ij})b_{ij}) \quad subject\ to\ C2,\ C4.$$

Subproblem A is known as the Generalized Assignment Problem (GAP) and has been intensively studied in the past[4]. Exact algorithms and fast heuristics have been developed. In this work we use the heuristic proposed by Martello and Toth[4]. Interested readers are referred to their original text.

There is no existing work on Subproblem B as far as we know. We call Subproblem B a "Distance-Constrained Simple Assignment (DCSA) Problem" and propose a simple heuristic to solve it.

4 Lagrange Heuristic

4.1 Primal-Dual Iterations

In the previous sections we showed that for a given U matrix, we can solve for A and B matrices independently. The only remaining question is how to find the dual variable matrix U that corresponds to the optimum solution of our original problem. This is a very common and familiar question for discrete optimization. It is generally solved by the "Primal-Dual iteration" process. In our case, the Primal problem is to solve for A and B matrices in

Subproblems A and B, and the Dual problem is to solved for Lagrange Multiplier matrix U. Primal-Dual iterations starts with an initial guess of U matrix. For this fixed U matrix we can solve Subproblems A and B. Then we can update U according to the solutions from Subproblem A and B. This new U is used in the next iteration to solve Subproblems A and B. This process can continue until the optimal solution is reached.

4.2 Subgradient Optimization

The most popular way to update Lagrange Multiplier matrix U is Subgradient Optimization. In our case, this translates to

$$u_{ij}^{(k+1)} = u_{ij}^{(k)} + t^{(k)}(a_{ij} - b_{ij})$$

where k is the iteration count, $t^{(k)} = \frac{p_{average}}{k}$ is the step size at iteration k and $p_{average}$ is a scaler constant defined as the average value of all p_{ij} in the P matrix.

4.3 Main Algorithm

The following is our main algorithm:

step 1: (Initialization)

set $k = 1$ and $u_{ij}^{(1)} = p_{average}$;

step 2: (Primal problem)

Solve

$$\min_{\forall A}\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\frac{1}{2}p_{ij} + u_{ij}^{(k)}\right)a_{ij}\right) \quad subject\,to\ C1,\ C3.$$

and

$$\min_{\forall B}\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\left(\frac{1}{2}p_{ij} - u_{ij}^{(k)}\right)b_{ij}\right) \quad subject\ to\ C2,\ C4.$$

step 3: (Dual problem)

update U matrix

$$u_{ij}^{(k+1)} = u_{ij}^{(k)} + \frac{p_{average}}{k}(a_{ij} - b_{ij})$$

step 4: (Check for convergence)

If $A \neq B$ goto step 2, otherwise terminate.

4.4 Terminating Criteria

The main algorithm will terminate only when $A = B$ is reached. However in practical situation this condition may take too long to satisfy. A careful examination of the constraints reveals two additional stopping criteria:

1). If solution matrix A from Subproblem A also satisfies C2, stop.

2). If solution matrix B from Subproblem B also satisfies C1, stop.

In either case the solution matrix is already a legal solution to our original problem and therefore the algorithm can terminate.

4.5 Distance-Constrained Simple Assignment (DCSA) Problem

We propose the following simple heuristic for solving subproblem B.

```
step 1: (Sorting)
  Sort all modules according to their sizes into non-increasing order.
step 2: (Initialization)
  set all $b_{ij}$ to 0;
step 3: (Sequential Simple Assignment)
  for each module $j \in J$ in that orer
          set $i^* = \arg\min\{p_{ij} : \forall i \in I\}$;
          set $b_{i^*j} = 1$ (assign $j$ to $i^*$);
          for each module $j'$ not assigned
                  for each partition $i$
                          if $D_P(i, i^*) > D_C(j', j)$
                                  set $p_{ij'} = \infty$;
                          endif
                  endfor
          endfor
  endfor
```

5 Testing the Ideas

Due to the current trend of adopting high density CMOS technology, the chip level of integration has been greatly increased. This results in a reduction in the number of chips per module used in TCM/MCM. However, the ever-increasing system complexity and performance still drives the need to use multiple-chip configuration. We expect the number of partitions used in each packaging level will remain steady in the range of 10-50. In this work we use a 16-chip (4×4) TCM for our testing purpose. Several industrial circuits are used to illustrate the effectiveness of our approach.

5.1 Preparing Data

In this section we describe how we obtained various data from real circuits and prepare input data to our general M-way system partitioning algorithm.

Each of the industrial circuit we obtained contains a cycle time value and a System Graph[5] $G(R \bigcup C, E)$, where R is the set of all register nodes and C is the set of all combinational nodes. Each node in this graph is labeled as either register or combinational block. Each node has a delay attribute and a size attribute. We apply a Super-Node Merging procedure as described in a previous paper[5] to the circuits. This merging procedure essentially merges all nodes that *cannot afford* to be assigned to different chip slots without violating cycle time constraints.

The resulting graph is called a Super-Node Graph. Each node in this graph has a size attribute, which is the sum of all the size attributes from the constituent register or combinational nodes. The set of all Super-Nodes is the set of modules J mentioned in section 2.1. s_j is the size attributes of the Super-Node j. The set of partitions is simply the set of 16 chip slots. The capacity of each slot is assumed to be the same and calculated as:

$$c_i = 1.10 \times \frac{\sum_{j=1}^{N} s_j}{16}, \qquad \forall i \in I$$

The 10% extra capacity is a common allowance given to the designer.

We estimate the wiring delay between adjacent chip slot (d_{unit_length}) to be 7-14% of the system cycle time.

We assume the overall TCM wiring delay has two components: a constant component d' corresponding to total delay introduced when

1). signal travels out of the chip, through the TCM distribution layers, to the signal routing layers and *plus*

2). signal travels from the signal routing layers, through the TCM distribution layers, then into the chip.

The variable delay component is the delay introduced by actual TCM signal layer wiring. This component is assumed to be linearly proportional to the manhattan distance. Based on these assumptions we can derive:

$$D_C(j_1, j_2) = \frac{(cycle\_time - d' - d'')}{d_{unit\_length}}$$

where d'' is the maximum internal delay of any combinational block that lies in a data path connecting j_1 and j_2. In other words, $D_C(j_1, j_2)$ is the maximum distance allowed for TCM signal layer wiring.

If there is no connection between j_1 and j_2, $D_C(j_1, j_2)$ is set to infinity.

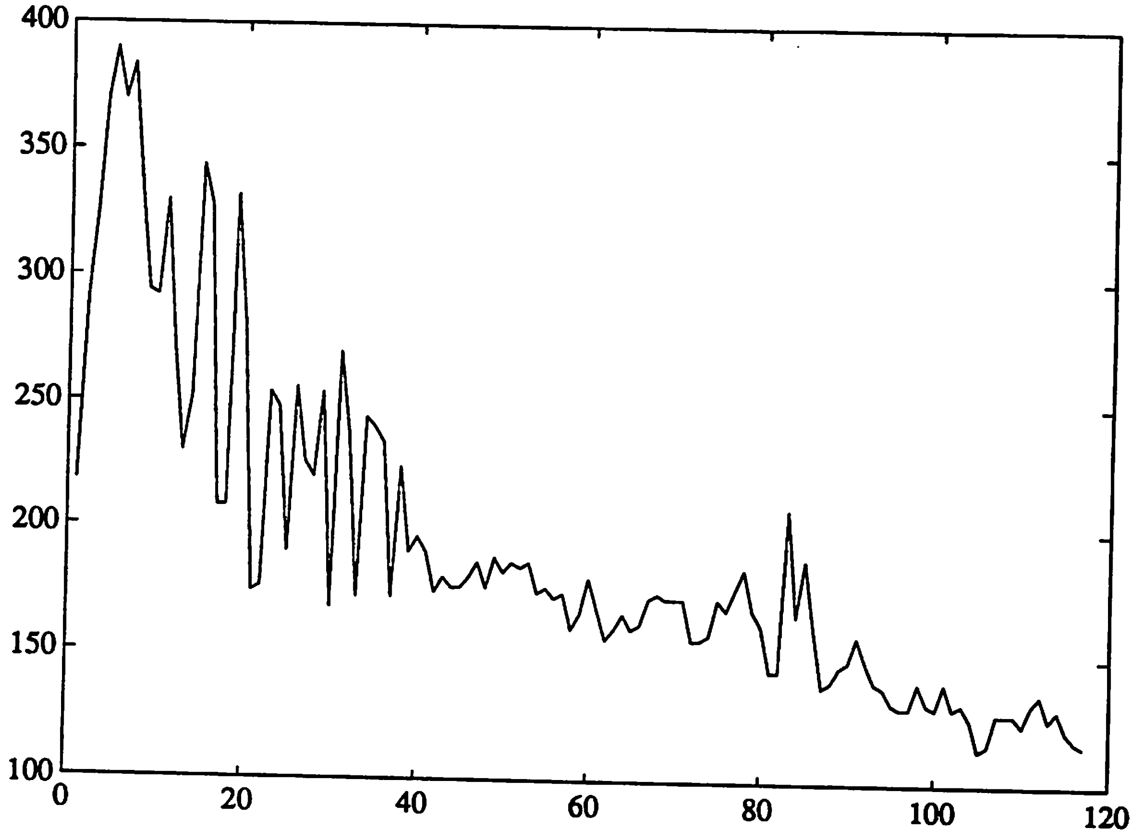
The D_C matrix constructed this way is not suitable for our purpose because it is not transitively closed. For example, if $D_C(1,2) = 10$ and $D_C(2,3) = 14$, from triangular inequality we can conclude that the maximum distance allowed between module 1 and 3 must be smaller or equal to 24. However this *implied* inequality may be missing from the D_C matrix originally constructed. Therefore we need to compute the transitive closure of the inequality relationships. We found that the problem of finding transitive closure of triangular inequality is equivalent to the all-pairs shortest path problem. This problem can be solved using existing techniques. Currently we implemented Floyd-Warshall algorithm, which can be replaced by Johnson's algorithm in order to exploit sparsity.

We first make a random initial assignment of all the Super-Nodes into the 16 slots. We then apply the method described in section 2.2 to this (possibly illegal) initial assignment to obtain the P matrix.

5.2 Test Results

A typical curve of convergence is shown below. The horizontal coordinate is iteration count and the vertical coordinate is (2 times) the number of disagreeing module assignments from the solutions of subproblems A and B.

The program terminated at iteration 117 since solution from subproblem A satisfies Timing Constraints C2 and therefore obtained a solution satisfying all constraints.



The results are summarized in the following table.

For all testcases we eliminated all timing violations and capacity overflows within the number of iterations indicated in the table.

circuit name	# of Reg.	# of Comb.	# of Super-Nodes	# of crit. edges	# of timing violations	max. cap. overflow	# of iterations	CPU min:scnd
ckt1	545	12172	545	2022	70	20%	435	56:06
ckt2	342	8280	339	1962	90	57%	117	9:56
ckt3	357	3026	357	1180	62	73%	45	3:52
ckt4	521	6325	521	3924	184	64%	21	3:18
ckt5	380	3850	380	1094	34	53%	5	0:25
ckt6	607	4990	607	1392	46	54%	4	0:35
ckt7	472	3378	472	916	30	27%	2	0:12

The respective columns in the table (from left to right) are: 1). Circuit name, 2). Number of register nodes, 3). Number of combinational nodes, 4). Number of Super-Nodes, 5). Number of timing critical edges in the Super-Node Graph, 6). Number of timing violations in the initial assignment, 7). Percentage of capacity overflow in the worst slot in the initial assignment, 8). Number of iterations needed for a feasible solution, 9). CPU in minutes:seconds.

References

[1] 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

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

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

[4] S. Martello and P. Toth, "Knapsack Problems" 1990

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

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