Eng. Opt., 1995, Vol. 25, pp. 131–154
Reprints available directly from the publisher
Photocopying permitted by license only

GLOBAL OPTIMIZATION ALGORITHMS FOR CHIP LAYOUT AND COMPACTION

MICHAEL C. DORNEICH

Department of Electrical and Computer Engineering,

NIKOLAOS V. SAHINIDIS

Department of Mechanical and Industrial Engineering,
University of Illinois, 1206 W. Green Street, Urbana, IL 61801, U.S.A.

(Received 15 February 1995)

The package planning (chip layout and compaction) problem can be stated in terms of an optimization problem. The goal is to find the relative placement and shapes of the chips in a way that minimizes the total chip area subject to linear and nonlinear constraints. The constraints arise from geometric design rules, distance and connectivity requirements between various components, area and communication costs and other designer-specified requirements. The problem has been addressed in various settings. It is of unusual computational difficulty due to the nonconvexities involved. This paper presents a new mixed-integer nonlinear programming formulation for simultaneous chip layout and two-dimensional compaction. Global optimization algorithms are developed for this model as well as for an existing formulation for the chip compaction problem. These algorithms are implemented with the global optimization software BARON and illustrated by solving several example problems.

KEY WORDS: chip layout and compaction, package planning, branch-and-reduce, mixed-integer nonlinear programming

1 INTRODUCTION

When designing the layout of chips in an electronic package, several issues must be addressed. Issues such as wireability (wire routing), pin order, and package area all introduce constraints to the overall goal of optimizing the package plan. Often, when the design cycle is in its earliest stages, the designer is faced with an inadequate description of the constraints. The initial layout and partitioning decisions based on inadequate knowledge of the constraints can have expensive consequences at later stages of the design cycle. This situation necessitates the development of a package planner that the designer can employ to aid in assessing the consequences of choices made in the initial stage of the design cycle.

The chip layout and compaction problem can be stated in terms of an optimization problem. The goal is to find the arrangement of chips on a plane so that the chip area is minimized, subject to linear and nonlinear constraints. The constraints arise from several, often conflicting, design objectives that include issues of geometric design (chip aspect ratios, chip area, etc.), wireability (maximum wire distances,

connectivity between chips), and other design constraints identified by the designer. The compaction problem is a special case of the problem, obtained once the relative position (layout) of chips has been decided. The compaction problem arises in optimizing planar rectangular spaces, such as floor plans for buildings, as well as for electronic planar packages.

Problems related to chip layout and compaction have been addressed in the facilities location area (Love *et al.*[16]; Mirchandani and Francis[19]; Francis *et al.*[7]) and mostly in the circuit design area (Soukup[30]; Otten[22,23]; Ciesielski and Kinnen[3]; Heller *er al.*[8]; Maling *et al.*[17]; Stockmeyer[31]; Kedem and Watanabe[12]; Watanabe[33]; Hu and Kuh[11]; Preas and Lorenzetti[25]; Wolf and Dunlop[34]; Hill *et al.*[9]; Lengauer[15]; Onodera *et al.*[21]; Sutanthavibul *et al.*[32]; Bamji and Varadarajan[1]; Yao *et al.*[35]). The version of the problem that is addressed here has as an objective the minimization of the package area. The problem is of unusually high computational difficulty due to the number and kind of constraints required. A typical problem may have the following constraints: the chips on the perimeter of the package must supply enough space for the package input/outputs, neighboring chips must supply enough places on their common edge to accommodate any wiring between them, the wiring itself must be routed in such a way as to minimize wire length (delay) and area, and the aspect ratios of all chip areas must be within specified intervals. The problem differs substantially from the typical "close packing of rectangular blocks of fixed shape" problem as the dimensions of the package as well as of the package internals (such as chip and wiring space) are only constrained, not fixed. By casting the problem in terms of constraints, rather than fixed dimensions, the designer has greater freedom to manipulate the internals of the package. Therefore, the design problem is represented more accurately than in the close packing formulation.

When formulated as an optimization model, the problem does not easily fall into the domain of problems solved by linear or standard nonlinear programming techniques. Many nonconvexities are present in the optimization formulation. This is not surprising as even some special cases of the circuit layout and compaction problem are NP-complete (Lengauer[13,14]). Even the problem of one- or two-dimensional compaction of rectangular chips of fixed dimensions is NP-complete in the strong sense as it is equivalent to the 3-PARTITION problem (Doenhardt and Lengauer[5]). As a result of its difficulty, thus far only heuristics and optimization approaches to simplified versions of the problem have been developed for optimal package planning. With the exception of very small problems, VLSI layout and compaction is always performed in a hierarchical fashion. The package is partitioned into blocks and each block is laid out individually. Then placement and routing of the overall package are addressed (e.g. Wolf and Dunlop[34]; Bamji and Varadarajan[1]; Yao *et al.*[35]). Most approaches focus on planar package plans. Planar package plans are characterized by the property that a layout arrangement exists such that no two wires on the board will cross. However, nonplanar package plans can be planarized by the introduction of "wire macros" at the wire intersections (Heller *et al.*[8]). In an effort to obtain optimal planar package plans under the constraints described above, Maling *et al.*[17] developed a solution technique that is

divided into three phases. Each phase is more tractable and can be solved using more or less standard (convex) optimization techniques. First, the algorithm addresses the problem of package constraints by finding a large set of layouts that satisfy the perimeter and area constraints of the chip macros. Then, the remaining constraints are used to modify and select from this set of solutions. As the second phase of this approach requires the solution of a nonconvex bilinear program subject to bilinear constraints, no guarantees for global optimality were offered. This formulation is posed as an unsolved problem in a recent collection of global optimization problems (Floudas and Pardalos[6]).

Research in the global optimization area has intensified recently (e.g. Horst and Tuy[10]; Pardalos and Horst[24]) and the new algorithms seem more suitable to address difficult global optimization problems that arise from engineering applications. This paper describes the package planning problem, presents several illustrative example problems, and develops global optimization approaches for obtaining optimal planar package plans. The problem statement is presented in Section 2. A mixed-integer nonlinear programming (MINLP) formulation is developed in Section 3. This model solves the layout and compaction problem simultaneously and guarantees a global optimal solution as opposed to the decomposition approach of Maling *et al.*[17] which is reviewed in Section 4. As both approaches of Sections 3 and 4 require the solution of nonconvex optimization models, Section 5 develops global optimization algorithms for these models. These algorithms are shown to be convergent. In addition, for the problem of Maling *et al.*[17], a convexifying transormation is presented which allows for an efficient solution of problems from the literature by means of geometric programming techniques. The branch-and-reduce global optimization algorithm of Ryoo and Sahinidis[26,27] is used and the algorithms developed are implemented with the global optimization software BARON (Sahinidis[28]). Several examples are used to illustrate the problem and the proposed algorithms in Section 6, followed by the conclusions in Section 7.

2 PROBLEM STATEMENT

The problem studied in this paper involves a number of chips that are to be laid on a board. The objective is to minimize the overall dimensions of the package layout. There are several constraints that must be enforced when constructing the layout of the package plan. They are listed as follows:

- Contiguity between macros must be enforced (a macro is defined as a physical implementation of a high level function, *i.e.* a group of chips). This constraint requires that there be enough space for wire routes between chips on the package. Each chip has a certain number of Input/Outputs (I/Os) on each side of the chip, so contiguous macro I/Os must have matched orders. These constraints will be referred to as contiguity constraints.
- The aspect ratio of each macro must fall within a certain interval.
- Adjacency relations describing the relative placement of chips must be enforced. This is required to allow certain interconnections.

- Perimeter constraints must be enforced. This means that the perimeter must be able to accommodate the macro I/Os that lead off the board. These constraints will be referred to as perimeter constraints.
- In certain cases, it is important to be able to specify that the center of a certain chip should be at the same height as the center of another. For example, when one designs a cell that is to be replicated many times, the center of an input node should be at the same height as the center of an output node.

Figure 1 provides an illustrative example from Maling *et al.*[17]. Figure 1a presents a block diagram of chips and their communication buses (or lines). The block diagram can be represented by an undirected planar graph, as shown in Figure 1b. There are six chips in this example denoted by A, B, C, D, E and F with minimal area requirements of 30, 20, 20, 25, 15, and 20 units, respectively. The links between rectangles of the block diagram (edges in the graph) represent connections between chips. Weights are assigned to these links and correspond to the number of wires between the two corresponding chips on the block diagram. These give rise to minimal requirements for shared edges between the corresponding chips. For example, chips A and B of this example must share an edge of length at least equal to 5 units. Edges incident to only one chip represent connections to the outside of the package and imply that the corresponding chip must be on the perimeter of the package.

Planar graphs such as the one of Figure 1b have the property that they have planar dual graphs (as in Figure 1c). A planar original graph (POG) is related to its dual in the following way:

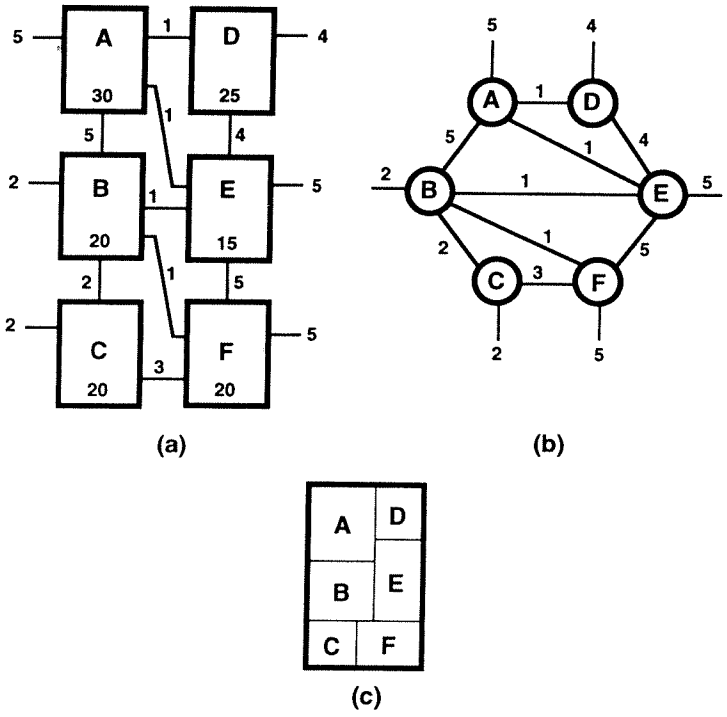

Figure 1 (a) Block diagram of 6 chips and communication lines. (b) The corresponding planar original graph. (c) One possible rectangular dual graph.

- An area (or mesh) inside the dual corresponds to a node of the POG.
- A node of the dual corresponds to a mesh of the POG.
- The area outside the dual corresponds to an explicit or implicit POG node at infinity.
- The edge separating two adjacent areas of the dual corresponds to the bus (wires) joining two corresponding nodes of the POG.

The discussion will be restricted to the geometric realizations of duals that are rectangular in shape. This is not a restrictive assumption as rectangles can serve as building blocks in the modelling of several objects:

- An L-shaped object can be decomposed into two perpendicular rectangular segments.
- Π-shaped and H-shaped objects can be decomposed into three appropriately interconnected rectangles.
- A frame can be decomposed into four appropriately interconnected rectangles.

Note that in all these cases, it is necessary to develop a model with the ability to enforce constraints that certain rectangles have a prerequired relative placement, share a certain edge and have certain of their corner points coincide.

From now on, it will be assumed that the problem information is provided through a description of the block diagram in terms of its undirected planar graph. Weights assigned to the edges correspond to the number of wires between the two corresponding chips on the block diagram while the weights of the nodes represent the size (area) of the chip itself. Given the planar original graph (POG), there exists one or more rectangular dual graphs (RDG). The only constraints enforced when producing these RDGs are the continuity constraints. There must exist an edge between two rectangles of the RDG that must have length equal to or greater than the bundle size (number of wires) between the two corresponding nodes (chips) of the POG. Hence, every set of two nodes that are connected by communication lines in the POG must have a shared edge between the two corresponding rectangles of the RDG. When the area and perimeter constraints are enforced, different RDGs will have different dimensions. The problem is then to generate the RDG morphology with minimum package area.

3 MINLP FORMULATION

The goal of the package planner is to locate the positions of all the rectangles in the plane, so that certain constraints are satisfied and the package has minimum overall dimensions. In developing the model, the indices i and j will be used to denote rectangles ($i, j = 1, \ldots, n$). Let x_i^c and y_i^c denote the x and y coordinates, respectively, of the center of rectangle i. Similarly, let x_i and y_i denote the x and y dimensions (width and height), respectively, of rectangle i. Finally, let x_p and y_p denote the x and y dimensions (width and height), respectively, of the overall package. The following model can be used:

$$\min x_p y_p \tag{1}$$

subject to

$$\begin{pmatrix} |x_i^c - x_j^c| \geq (x_i + x_j)/2 \\ \text{and/or } |y_i^c - y_j^c| \geq (y_i + y_j)/2 \end{pmatrix} \quad 1 \leq i < j \leq n \tag{2}$$

$$\left\{ \begin{pmatrix} |y_i^c - y_j^c| = (y_i + y_j)/2 \\ \text{and} \quad |x_i^c - x_j^c| \leq (x_i + x_j)/2 - R_{ij} \\ \text{and} \quad \min(x_i, x_j) \geq R_{ij} \end{pmatrix} \right\} \quad (i,j) \in R \tag{3}$$
$$\left\{ \text{or} \begin{pmatrix} |x_i^c - x_j^c| = (x_i + x_j)/2 \\ \text{and} \quad |y_i^c - y_j^c| \leq (y_i + y_j)/2 - R_{ij} \\ \text{and} \quad \min(y_i, y_j) \geq R_{ij} \end{pmatrix} \right\}$$

$$x_i y_i \geq A_i \qquad 1 \leq i \leq n \tag{4}$$

$$x_i^L \leq x_i \leq x_i^U \quad 1 \leq i \leq n \tag{5}$$

$$y_i^L \leq y_i \leq y_i^U \quad 1 \leq i \leq n \tag{6}$$

$$x_i^C - x_i/2 \geq 0 \quad 1 \leq i \leq n \tag{7}$$

$$y_i^C - y_i/2 \geq 0 \quad 1 \leq i \leq n \tag{8}$$

$$\mathbf{F(x, y, x^c, y^c) \leq 0} \tag{9}$$

$$x_p = \max_i \left(x_i^c + \frac{x_i}{2} \right) \tag{10}$$

$$y_p = \max_i \left(y_i^c + \frac{y_i}{2} \right) \tag{11}$$

In (1), the area of the overall package is minimized. Constraint (2) ensures that no two rectangles overlap. This is accomplished by requiring the rectangle centers to be at a distance no smaller than one half of the sum of the lengths of their sides. It is sufficient to enforce this requirement in either the x or the y direction. For those rectangles with a requirement that they share an edge, constraint (3) must be included. Here, R_{ij} is the minimum required length of the shared edge between rectangles i and j. Also, the set $R = \{(i,j): 1 \leq i < j \leq n; R_{ij} > 0\}$ includes all pairs of rectangles with shared edge requirements. The effect of the first three relationships in (3) is that the top (or bottom) edge of rectangle i touches the bottom (or top) edge of rectangle j while at the same time the shared edge is of length at least equal to R_{ij}. The last three relationships of (3) enforce a similar constraint on the side edges of rectangles i and j. Constraint (4) requires the area of rectangle i to be at least equal to its minimal requirement A_i. Simple bounds for the problem variables are given in (5–6) and constraints that force all rectangles to lie in the positive quadrant are provided in (7–8). The model also includes additional feasibility constraints (9). These can, for example, include perimeter constraints for each rectangles as well as constraints on the aspect ratio of each rectangle. Constraints required for the

modelling on L-, H-, Π- or other similar objects discussed in the previous section can also be included in (9). Even the requirement that a frame-like object be empty can be modelled by introducing a dummy rectangle and requiring its corner points to coincide with certain corner points of the four rectangles modelling the frame. All these constraints are linear. As long as (9) is linear or convex, the specific form of these constraints does not matter for the rest of the paper. Sometimes, but not always, it is known *a priori* which of the two possible edges (x or y) two rectangles must share. This implies that certain disjunctions can be simplified. Finally, (10) and (11) simply require all the rectangles to be within the package.

In order to explicitly model the disjunctions in the above model and eliminate some of the nonlinearities involved in the absolute values, binary variables will be introduced. By convention, chip j is "to the right of or above" chip i (denoted by $j \succ i$) if $x_j^c - x_i^c \geqslant (x_i + x_j)/2$ and/or $y_j^c - y_i^c \geqslant (y_i + y_j)/2$. Two sets of binary variables can now be defined as follows:

$$\alpha_{ij} = \begin{cases} 1 & \text{if } j \succ i \\ 0 & \text{if } i \succ j \end{cases} \qquad 1 \leqslant i < j \leqslant n$$

$$\beta_{ij} = \begin{cases} 1 & \text{if } |y_i^c - y_j^c| \geqslant (y_i + y_j)/2 \\ 0 & \text{if } |x_i^c - x_j^c| \geqslant (x_i + x_j)/2 \end{cases} \qquad 1 \leqslant i < j \leqslant n$$

These binaries model the relative orientation of rectangles. In particular, rectangle j is "to the right of or above" rectangle i if $\alpha_{ij} = 1$, and "to the left of or below" if $\alpha_{ij} = 0$. More precisely, if $\alpha_{ij} = 1$, then $\beta_{ij} = 1$ means that j is "above" i while $\beta_{ij} = 0$ means that j is "to the right of" i. Similarly, if $\alpha_{ij} = 0$, then $\beta_{ij} = 1$ means that j is "below" i while $\beta_{ij} = 0$ means that j is "to the left of" i. For rectangle pairs with shared edge requirements, the binaries β_{ij} model whether two rectangles touch along the x ($\beta_{ij} = 1$) or y ($\beta_{ij} = 0$) direction. Let M be a sufficiently large positive constant. Using these binary variables, the package planning problem can be restated as a mixed-integer nonlinear programming model:

(P1) min $x_p y_p$

subject to

$$\left.\begin{aligned} & x_i^c - x_j^c \geqslant (x_i + x_j)/2 - M\alpha_{ij} - M\beta_{ij} \\ & x_j^c - x_i^c \geqslant (x_i + x_j)/2 - M(1 - \alpha_{ij}) - M\beta_{ij} \\ & y_i^c - y_j^c \geqslant (y_i + y_j)/2 - M\alpha_{ij} - M(1 - \beta_{ij}) \\ & y_j^c - y_i^c \geqslant (y_i + y_j)/2 - M(1 - \alpha_{ij}) - M(1 - \beta_{ij}) \end{aligned}\right\} \quad 1 \leqslant i < j \leqslant n \tag{12}$$

$$\left.\begin{aligned} & x_i^c - x_j^c \leqslant (x_i + x_j)/2 - R_{ij}\beta_{ij} \\ & x_j^c - x_i^c \leqslant (x_i + x_j)/2 - R_{ij}\beta_{ij} \\ & y_i^c - y_j^c \leqslant (y_i + y_j)/2 - R_{ij}(1 - \beta_{ij}) \\ & y_j^c - y_i^c \leqslant (y_i + y_j)/2 - R_{ij}(1 - \beta_{ij}) \\ & x_i \geqslant R_{ij}\beta_{ij} \\ & x_j \geqslant R_{ij}\beta_{ij} \\ & y_i \geqslant R_{ij}(1 - \beta_{ij}) \\ & y_j \geqslant R_{ij}(1 - \beta_{ij}) \end{aligned}\right\} \quad (i, j) \in R \tag{13}$$

$$x_p \geqslant x_i^c + x_i/2 \qquad 1 \leqslant i \leqslant n \tag{14}$$

$$y_p \geq y_i^c + y_i/2 \quad 1 \leq i \leq n \tag{15}$$
$$\alpha_{ij} \in \{0,1\} \quad 1 \leq i < j \leq n$$
$$\beta_{ij} \in \{0,1\} \quad 1 \leq i < j \leq n$$

and constraints (4–9).

It can be easily verified that (12) and (13) enforce (2) and (3), respectively, and that, due to (14) and (15), both (10) and (11) will be satisfied at the optimal solution of model P1. The model has the following additional features:

- It performs *simultaneous* chip layout and compaction while considering a large number of design constraints.
- It permits rectangle j to be both above (or below) and to the right (or left) of rectangle i, thereby, allowing all possible configurations among rectangles.
- It allows rotation of rectangles of fixed area. This is a consequence of the fact that chip dimensions (height and width) are not fixed but simply constrained by the areas.

In addition, it is a simple exercise to extend the model to handle the following situations:

- Upper bounds for the areas can be included. This, in combination with the lower bounds, allows the modelling of rectangles of fixed size. Rectangle dimensions can also be fixed by fixing the corresponding x and y variables. Therefore, the model can handle any combination of rigid and flexible rectangles.
- For those (usually few) rectangles required to be on the perimeter of the package, it is straightforward to introduce additional binary variables and linear constraints into the model in order to enforce that at least one of following constraints holds: $x_i^c \geq x_i/2$, $x_i^c + x_i/2 \geq x_p$, $y_i^c \geq y_i/2$, $y_i^c + y_i/2 \geq y_p$.

Several integer programming approaches to the package planning problem have been proposed in the literature, most notably the ones by Kedem and Watanabe[12] and Sutanthavibul *et al.*[32] As these approaches are linear, several simplifying assumptions had to be made for their derivation. For example, the areas (or even the dimensions) of the rectangles are fixed and, in general, nonlinear constraints were approximated by linear ones. Therefore, the proposed approach has several advantages from the modelling point of view. On the other hand, similar to previous integer programming approaches, the computational complexity of the present model is exponential. It is therefore best utilized when incorporated in a hierarchical design approach based on the sequential solution of small components of a large circuit.

The model consists of two parts: an integer part and a continuous part. This is not surprising as the chip layout problem also consists of two parts: relative placement of chips and chip dimensions. Each part of the problem carries its own contribution to the difficulties associated with solving this model. There are nonconvexities due to the integrality requirements as well as due to the presence of the bilinear terms in the objective and the constraints. The next section reviews a decomposition approach that tries to separate the discrete from the continuous parts of the problem.

4 SEQUENTIAL SOLUTION TECHNIQUE

A sequential solution technique that can be divided into three phases was developed by Maling *et al.*[17] Each phase is more tractable and can be solved using known techniques. First, their algorithm finds a large set of layouts that satisfy the continuity requirements for chip macros. Then, they attempt to satisfy perimeter and area constraints while they minimize the package area. Their algorithm can be interpreted as a sequence of the following steps:

Step 1. An assumption is made as to which rectangles define the corners of the chip (it can be two, three, or four rectangles). Note that the choice is arbitrary and that different choices will lead to different solutions.

Step 2. Let l denote the angles of the RDG and k denote the meshes of the POG (nodes of RDG). Use d_k to denote the number of angles of the RDG node k and define the following parameters:

$$a_{il} = \begin{cases} 1 & \text{if angle } l \text{ is in rectangle } i \\ 0 & \text{otherwise} \end{cases}$$

$$b_{kl} = \begin{cases} 1 & \text{if angle } l \text{ is in RDG node } k \\ 0 & \text{otherwise} \end{cases}$$

Observe that a mesh in the RDG (node in POG) must have exactly four right angles since only rectangular RDGs are considered here. Also, each of the m nodes in the RDG (mesh in POG) has 2 or 4 right angles depending on whether it corresponds to a "T" ($\top$) or a "plus" ($+$) configuration. Then, all possible layouts can be found by solving the following equations:

$$\sum_l a_{il} z_l = 4 \quad 1 \leqslant i \leqslant n$$

$$\sum_l b_{kl} z_l = d_k \quad 1 \leqslant i \leqslant m$$

where $z_l = 1$ or 0, represents an RDG angle of 90 or 180 degrees, respectively. All solutions to these equations must be obtained in terms of the z_l variables to provide all possible RDG morphologies. Note that although the above assignment-type of constraints are easy to solve, it is necessary to find all their possible solutions. Solving these equations amounts to fixing the binaries of the MINLP model P1 in a way that guarantees that the contiguity constraints are satisfied.

Step 3. The previous step provides all possible RDG morphologies under the contiguity requirements. It remains to determine the dimensions of the rectangles so that the package dimensions are minimized while the area

constraints for the rectangles are satisfied. Maling *et al.*[17] proposed a model using perimeter constraints for the package:

$$\text{(P2) min} \sum_{i=1}^{n} x_i y_i$$

subject to

$$\mathbf{L(x,y)} \leqslant \mathbf{0} \tag{16}$$

and constraints (4–6)

The variables of this model have the same interpretation as in model P1. Constraints (16) include equalities to enforce perimeter constraints for the package so that variables x_p and y_p do not have to be included in the model. Constraints (16) also include inequalities that require that there be sufficient overlap among certain rectangle edges. Model P2 is best illustrated by an example. For the POG of Figure 1b, one possible RDG layout is shown in Figure 1c. For this layout, the problem of finding the dimensions of the package is as follows:

$$\text{(P2-1) min } x_A y_A + x_B y_B + x_C y_C + x_D y_D + x_E y_E + x_F y_F$$

subject to

$$\left.\begin{aligned} &x_D = x_E \\ &x_A + x_D = x_B + x_E \Rightarrow x_A = x_B \\ &x_A + x_D = x_C + x_F \\ &y_C = y_F \\ &y_A + y_B + y_C = y_D + y_E + y_F \Rightarrow y_A + y_B = y_D + y_E \end{aligned}\right\} \tag{17}$$

$$\left.\begin{aligned} &x_B - x_C \geqslant 1 \\ &y_A - y_D \geqslant 1 \end{aligned}\right\} \tag{18}$$

$$\mathbf{xy} \geqslant (30,20,20,25,15,20) \tag{19}$$

$$\mathbf{x} \geqslant (5,5,2,4,4,5) \tag{20}$$

$$\mathbf{y} \geqslant (5,2,5,4,5,5) \tag{21}$$

The equalities (17) in model P2-1 enforce the requirement that all slices of the package in the x- or in the y- direction must have lengths equal to the package dimensions (width or height). These constraints were obtained by eliminating x_p and y_p from the following system of equations which follows from Figure 1c:

$$\begin{aligned} x_C + x_F &= x_p \\ x_E + x_B &= x_p \\ x_E + x_A &= x_p \\ x_D + x_A &= x_p \\ y_F + y_E + y_D &= y_p \\ y_F + y_B + y_A &= y_p \\ y_C + y_B + y_A &= y_p \end{aligned}$$

The inequalities in (18) require that there be sufficient overlap among certain rectangle edges. Although these constraints have been written only for two pairs of rectangles (A–C and A–D), it can be easily verified that the remaining requirements are implied by the rest of the problem constraints and the bounds. In general, (18) needs to include the requirements for shared edges only for those rectangles arranged in an "alternating-T" ($\top\!\!\!\!\perp$). Finally, inequality (19) for the product is to be interpreted component-wise and enforces the area constraints for rectangles.

Note that the constraint set of the bilinear model P2 is separable with the only exception of the area constraints (19). Furthermore, P2 can be reduced to an optimization problem with a bilinear objective if x_p and y_p are not eliminated from the problem. In either case, model P2 involves nonconvexities in the objective as well as in the constraints. For this reason, Maling *et al.*[17] developed a heuristic, Simplex-based approach for the solution of this optimization problem.

From the modelling point of view, the solution space of the sequential approach is more restricted than in model P1. This is because, in deriving P2, the solutions are restricted to the case for which the sums of the sizes (widths or heights) of individual rectangles along any slice (horizontal or vertical) of the total package are equal to the package dimensions. In this way, no uncovered areas are allowed within the package. From the computational point of view, there are two main difficulties associated with the sequential approach. First, Steps 1 and 2 must explicitly enumerate all of the possible chip layouts. Second, the last step of the sequential method requires the solution of a nonconvex optimization problem. Yet, the algorithm developed by Maling *et al.* provides a means of rapid computation of gross consequences on the optimal layout plan given the basic information that is usually available at an early stage of the design cycle. One can arbitrarily make an assignment in Step 1, obtain a few solutions in Step 2 and solve the model of Step 3 to derive a feasible solution to the problem. The sequential solution technique can therefore be used as an initialization heuristic to provide a good upper bound that will assist in the solution of the MINLP model P1. For this reason, the next section addresses the solution of model P2 to global optimality.

5 GLOBAL OPTIMIZATION ALGORITHMS

The two previous sections presented approaches to the package planning problem. Both approaches require the solution of nonconvex optimization models. Among several methods, given a nonconvex minimization problem, branch-and-bound methods can be used in order to develop lower and upper bounds of the optimal objective function value over subregions of the search space. Certain subregions are then dynamically refined while others are excluded from consideration based on optimality and feasibility criteria. Critical to the success of these methods is the tightness of the lower bounding procedures used. As the lower bounds are developed over a range of values of the variables involved, they become tighter as the search is confined to smaller subregions. The approach taken here is one that emphasizes the development of tighter lower bounds for the problem through the reduction of the range of the problem variables. As the algorithm will be used in the context of two

different models, it is stated here in its most generic form. Let P denote the global minimization problem to be solved: $\min f(\mathbf{x})$ subject to $\mathbf{g}(\mathbf{x}) \leqslant \mathbf{0}$, $\mathbf{x} \in \mathbf{X}$. Also, let P_0 be a relaxation of P, obtained by relaxing some of the constraints and/or underestimating the objective function of P. At each iteration, the algorithm updates a list, $\mathscr{P}$, of subregions (each denoted by the subscript i) of the feasible space and relaxations (P_i) of the nonconvex problem over these subregions. The algorithm can be represented by a search tree. Simple subproblems (relaxations) are solved at each node of the tree. Their solution provides lower bounds, L_i, which can be used to exclude nodes in the tree from further consideration. This is achieved by comparing these lower bounds to the best current upper bound, U. At all times the global minimum is bounded between the lowest lower bound, L, and the value, U, of the best found feasible solution:

Initialization

Put P_0 on the list $\mathscr{P}$ of active subproblems. Select the convergence parameter ε. Set the lower bound $L = -\infty$, and the upper bound $U = +\infty$.

Iterative Step

1. **Subproblem Selection** If the list $\mathscr{P}$ of active subproblems is empty, stop: the current best solution is optimal. Else, choose from $\mathscr{P}$ the subproblem P_i with the smallest lower bound and remove it from the list $\mathscr{P}$. Goto 2.
2. **Pre-Processing** Use optimality-based and feasibility-based tests to tighten variable bounds as much as possible for subproblem P_i. Goto 3.
3. **Lower Bounding** Solve P_i, or bound its solution from below. Let L_i be this lower bound. If the solution $\mathbf{x}^i$, found for P_i is feasible for P and $f(\mathbf{x}^i) < U$ update $U \leftarrow f(\mathbf{x}^i)$, make $\mathbf{x}^i$ the current best solution, and remove from $\mathscr{P}$ all those subproblems P_j for which $L_j \geqslant U - \varepsilon$. If $L_i \geqslant U - \varepsilon$, Goto 1. Else, let L be equal to the smallest of all L_i and Goto 4.
4. **Upper Bounding** Do local search or other heuristic(s) to find a feasible solution for problem P. If found, update U and the current best solution, and remove from $\mathscr{P}$ all those subproblems P_j for which $L_j \geqslant U - \varepsilon$. Goto 5.
5. **Post-Processing** Use optimality-based and feasibility-based tests to strengthen the bounds of as many variables as possible. If Steps 4 or 5 were successful in improving U, apply strengthening tests to all subproblems currently in $\mathscr{P}$, otherwise apply strengthening only to P_i. If this step was successful in reducing the variable bounds for P_i, reconstruct P_i using the new bounds and Goto 3. Else, Goto 6.
6. **Branching** Select one of the problem variables that appear in nonconvex terms that are not satisfied at the relaxed solution of P_i. Split the range of this variable into two subregions: one to the right and one to the left of the relaxed problem solution. Generate two corresponding subproblems, P_{i_1}, P_{i_2}, place them on the list $\mathscr{P}$, and Goto 1.

Implementation details as well as general purpose range reduction tests are discussed in Ryoo and Sahinidis[27,28]. The next subsection develops the problem

specific relaxations, branching rules and range reduction tests required for models P1 and P2.

5.1 *Relaxations*

With the exception of the integrality requirements in P1, the remaining nonconvexities in models P1 and P2 are due to the bilinear terms in the objective and the constraints. To develop the relaxations, use is made of the well known fact that using lower and upper bounds of the participating variables, bilinear terms of the form xy can be underestimated and overestimated by convex and concave functions, respectively, as follows (see, e.g. McCormick[18]):

$$\left.\begin{aligned} xy &\geqslant \max\{y^U x + x^U y - x^U y^U,\ y^L x + x^L y - x^L y^L\} \\ xy &\leqslant \min\{x^L y + y^U x - x^L y^U,\ y^L x + x^U y - y^L x^U\} \end{aligned}\right\} \tag{22}$$

To avoid the nondifferentiabilities introduced by the *min/max* operators, each product xy is replaced by a new variable w and the following constraint set is introduced:

$$\{w(x,y) \leqslant 0\} := \begin{Bmatrix} w \geqslant y^U x + x^U y - x^U y^U \\ w \geqslant y^L x + x^L y - x^L y^L \\ w \leqslant x^L y + y^U x - x^L y^U \\ w \leqslant y^L x + x^U y - y^L x^U \end{Bmatrix}$$

Then, the proposed relaxations $P1_0$ and $P2_0$ of P1 and P2, respectively, are as follows:

$(P1_0)$ min w_p

subject to

$$\begin{aligned} & w_p(x_p, y_p) \leqslant 0 \\ & w_i(x_i, y_i) \leqslant 0 && 1 \leqslant i \leqslant n \\ & w_i \geqslant A_i && 1 \leqslant i \leqslant n \\ & 0 \leqslant \alpha_{ij} \leqslant 1 && 1 \leqslant i \leqslant j \leqslant n \\ & 0 \leqslant \beta_{ij} \leqslant 1 && 1 \leqslant i < j \leqslant n \end{aligned}$$

and constraints (5–9), (12–15).

$(P2_0)$ min $\sum_{i=1}^{n} w_i$

subject to

$$\begin{aligned} & w_i(x_i, y_i) \leqslant 0 && 1 \leqslant i \leqslant n \\ & w_i \geqslant A_i && 1 \leqslant i \leqslant n \end{aligned}$$

and constraints (5), (6), (16).

In contrast to the original models, the relaxations are convex. In fact, $P2_0$ is always an LP and $P1_0$ will be linear as long as (9) involves only linear constraints. Linearity is a major advantage in the context of branch-and-bound-based

algorithms as it greatly facilitates the solution of the relaxed problems as the search proceeds down the tree. Note that the relaxations increase the size of the original nonconvex formulations. In particular, $n+1$ variables and $2(n+1)$ constraints must be introduced to model P1 while n variables and $2n$ constraints must be introduced to model P2. As the increase of problem size is modest, these relaxations are easy to implement. For this reason, the development of alternative relaxations was not explored, although several possibilities exist (e.g. McCormick[18]; Sherali and Alameddine[29]).

5.2 *Branching rules*

Branching is required whenever the approximators in the solution of the relaxed problem violate the original problem functions. Let α^*, β^*, $\mathbf{x}^*$, $\mathbf{y}^*$, $\mathbf{w}^*$ denote the solution of the relaxed problem at any iteration of the algorithm. In terms of the binary variables, α_{ij} and β_{ij}, there is a violation if their values are not integer. Define the violations (v_{ij}^{α} and v_{ij}^{β}) as decreasing functions of the difference of the corresponding binary variables from 0.5:

$$v_{ij}^{\alpha} = \min(\alpha_{ij}^*, 1-\alpha_{ij}^*) \quad 1 \leqslant i < j \leqslant n \tag{23}$$

$$v_{ij}^{\beta} = \min(\beta_{ij}^*, 1-\beta_{ij}^*) \quad 1 \leqslant i < j \leqslant n \tag{24}$$

In terms of the continuous problem variables, the violations can be defined as follows. If $w_i^* = x_i^* y_i^*$, there is no violation. Otherwise, assign a violation of $|w_i^* - x_i^* y_i^*|$ to both x_i and y_i:

$$v_i^x = |w_i^* - x_i^* y_i^*| \quad 1 \leqslant i \leqslant n$$

$$v_i^y = |w_i^* - x_i^* y_i^*| \quad 1 \leqslant i \leqslant n.$$

The last two equations represent a standard means of defining violations for bilinear terms and do not take into account the size of the interval of the corresponding variables. In order to create more balanced search trees, the above violations are weighted with the length of the smallest interval that will result after branching on the corresonding variable:

$$v_i^x = \min(x_i^U - x_i^*, x_i^* - x_i^L) \times |w_i^* - x_i^* y_i^*| \quad 1 \leqslant i \leqslant n \tag{25}$$

$$v_i^y = \min(y_i^U - y_i^*, y_i^* - y_i^L) \times |w_i^* - x_i^* y_i^*| \quad 1 \leqslant i \leqslant n \tag{26}$$

Once the violations are calculated, the variable with the largest violation is selected for branching. For this variable, its feasible region is partitioned into two subregions, one to the right and one to the left of the relaxed problem solution. Note that it is not necessary to consider both x_i and y_i for branching. Branching on only one of these variables is sufficient for convergence of the algorithm as, for a sufficiently detailed subdivision of the range of one of the two variables, the bilinear term $x_i y_i$ becomes linear and thus a convex function in the other variable (see also Sherali and Alameddine[29]). Nevertheless both types of variables are candidates for branching as it is not clear *a priori* which type of branching will have the largest impact on

the value of the relaxation. For the same reason, no preference is given to the integer variables over the continuous variables for branching.

5.3 *Range reduction tests*

If a feasible solution (upper bound) is available for the problem, the reduced costs of the relaxed problem can be used to fix the values of the integer variables of the problem. It is also possible to use marginal values to restrict the range (as opposed to fixing) of continuous variables. This is possible only for those variables that hit an upper or lower bounding constraint at the relaxed problem solution. For those that do not, it is possible to temporarily fix them at a bound and perform similar range reduction tests. All these tests are described in detail in Ryoo and Sahinidis[26,27] and can be applied to any nonconvex problem for which a convex (non)linear relaxation is at hand. They are therefore used with models P1 and P2.

In addition, the range of variables can be reduced using problem specific constraints. In particular, nonconvex constraints can be used to preprocess and postprocess a node. These relationships are easy to develop. For example, from the area requirement: $x_i y_i \geqslant A_i \Rightarrow x_i \geqslant A_i/y_i$ which implies that, if y_i^U is a valid upper bound for the height y_i, then a valid lower bound for the width x_i is A_i/y_i^U. Similarly, a known upper bound U for the objective function can be used to derive bounds:

$$\sum_i x_i y_i \leqslant U \Rightarrow x_k y_k \leqslant U - \sum_{i \neq k} x_i y_i \Rightarrow x_k \leqslant \left(U - \sum_{i \neq k} x_i^L y_i^L \right) \Big/ y_k^L$$

Although the above relationships are very simple, their importance is that they often capture the effect of nonconvexities of the problem that are ignored by the convex relaxations. Particularly useful is the use of these range reduction rules in order to *dynamically* update the value of the big-M in the MINLP model during the course of the algorithm. This greatly reduces the difference between the upper and lower bounds and expedites the search.

5.4 *Correctness of the Algorithm*

To prove correctness of the algorithm, two results must be established: validity of the lower bounding operation and convergence of the algorithm. First, it must be shown that the relaxations provide lower bounds which are nondecreasing functions of the size of the feasible region over which the relaxation is defined (this is the requirement of step k.4 of page 115 of the prototype branch-and-bound algorithm of Horst and Tuy[10]). These results have already been established (McCormick[18]; Sherali and Alameddine[29]) for the inequalities (22) used here to approximate the bilinear terms and they are well known in the integer programming literature for relaxing a 0–1 variable in the continuous range from 0 to 1. Therefore, it suffices to prove convergence of the algorithm. Definition 1, Definition 2 and Theorem 1 correspond, respectively, to Definition IV.4, Definition IV.6 and Theorem IV.3 from Horst and Tuy[10] and will be used in the proof:

Definition 1. A bounding operation is called *consistent* if at every step any unfathomed partition element can be further refined, and if, whenever an infinitely decreasing sequence of partition sets emanating out of a parent set converges to a certain limit set, the upper bound over this limit set also converges to the lower bound of the objective over the parent set.

Definition 2. A selection operation is said to be *bound improving* if, at least each time after a finite number of steps, at least one partition element where the actual lower bound is attained is selected for further partition in the algorithm.

Theorem 1: *In the infinite branch-and-bound procedure, suppose that the bounding operation is consistent and the selection operation is bound improving. Then the procedure is convergent.* □

Further refinement in Definition 1 above is to be interpreted as meaning that a subproblem in the list $\mathscr{P}$ of currently active problems can be either fathomed or else further partitioned using the branching rules of Section 5.2. The limit behavior of the bounds in the second part of Definition 1 is implied if, whenever a decreasing sequence of partition sets converges to a certain limit set, the lower bounds *over this limit set* also converge to the exact minimum of the objective over this limit set.

Lemma 1. Let v be the maximum of all violations calculated in Eqs. (23–26). Then, the solution to the current subproblem is feasible to the original nonconvex problem if only if $v = 0$.

Proof: If $v = 0$, then all the violations are zero. This means that all binary variables have assumed 0–1 values in the relaxation. It also means that $w_i^* = x_i^* y_i^*$ for all possible i. Therefore, the relaxed problem solution satisfies the nonconvex problem constraint set. Conversely, if there is a violation, then at least one of the right hand sides of Eqs. (23)–(26) will be positive and, therefore, their maximum will also be positive. □

Lemma 2. Consider an infinite sequence of nested partitions of the feasible region. Then, the algorithm develops lower and upper bounds for this region whose difference tends to zero.

Proof: This is a consequence of the rectangular subdivisions used by the algorithm. Recall that the interval of the variable corresponding to the maximum violation is split into two parts. Therefore, in the limit, the interval of a variable becomes a point. Now observe that, if both the lower and upper bound of a variable coincide, the underestimators in (22) become an exact representation of the product. Therefore, the current subproblem solution becomes feasible to the nonconvex problem and the lower and upper bounds for this partition converge. In the case that binary variables are present in the model (model P1), the argument is simpler as branching leads to complete removal of nonconvexities and there is only a finite number of possible partitions. □

Theorem 2: *The bounding operation of the proposed algorithm is consistent.*

Proof: Lemma 1 guarantees that, if an unfathomed subproblem cannot be fathomed in the current iteration, there is a positive violation in Eqs. (23–26). If the

maximum violation arises from Eq. (23), this means there is a fractional binary variable and branching will further refine the feasible space by considering the cases of this binary being equal to 0 or 1. On the other hand, if the maximum violation occurs in Eqs. (24–26), then there is a corresponding x or y variable that is not at one of its bounds. Otherwise, (22) provides an exact represention of the bilinear term and the corresponding violation would have to be zero. Without loss of generality, assume the maximum violation corresponds to variable x whose relaxed problem value is x^*. Then, $x^L < x^* < x^U$. Therefore, branching will again create two feasible regions (one to the left and one to the right of x^*) that are strictly smaller than the feasible region of the parent node. It can be concluded that the bounding operation satisfies the first part of Definition 1. The second part follows directly from Lemma 2. □

Theorem 3: *The proposed algorithm is convergent.*

Proof: According to the selection operation used by the algorithm (best bound first), in every iteration, a partition element where the actual lower bound is attained is selected for further partition. Therefore, the selection operation is bound improving. As, in addition, the bounding operation is consistent from Theorem 2, convergence of the algorithm follows directly from Theorem 1. □

Remark 1. The argument in the proof of Theorem 2 is still valid for any x^* which is such that $x^L < x^* < x^U$. For example, the midpoint can be used, or the value corresponding to a known local minimum which is suspected to be the global solution. In the latter case, this branching rule makes the underestimators exact at the candidate solution and is, therefore, expected to expedite the search.

Remark 2. Theorem 3 establishes convergence of the algorithm at the limit. This implies that, since there are nonconvexities in the constraint set, in the worst case only a near-feasible, near-optimal solution can be guaranteed. Of course, if the algorithm terminates finitely, the solution is guaranteed to be a global minimum, since the lower bounding procedure is valid. In practice, it has been found that the algorithm always converges to the global solution finitely. This is demonstrated in Section 6. In addition, for special cases of model P2, the following transformation can be used and leads to a very efficient solution approach by means of convex optimization techniques.

5.5 *A Convexifying Transformation*

Assume that the overlap constraints (of the type (18)) of model P2 are of the following form:

$$x_i - x_j \geq R_{ij} \quad (i,j) \in R_x \subseteq R \tag{27}$$

$$y_i - y_j \geq R_{ij} \quad (i,j) \in R_y \subseteq R \tag{28}$$

Denote by S_x and S_y the number of all different slices of the RDG in x and y direction, respectively. In this case, instead of model P2, the following more general formulation can be used:

(P3) min $x_p y_p$
Subject to

$$\sum_{i \in I_k^x} x_i \leqslant x_p \qquad 1 \leqslant k \leqslant S_x$$

$$\sum_{i \in I_k^y} y_i \leqslant y_p \qquad 1 \leqslant k \leqslant S_y$$

$$x_i y_i \geqslant A_i \qquad 1 \leqslant i \leqslant n$$

$$x_i^L \leqslant x_i \leqslant x_i^U \qquad 1 \leqslant i \leqslant n$$

$$y_i^L \leqslant y_i \leqslant y_i^U \qquad 1 \leqslant i \leqslant n$$

and constraints (27–28).

As all rectangle dimensions have to be strictly positive, P3 can be restated as a geometric program:

(P4) min $x_p y_p$
Subject to

$$\sum_{i \in I_k^x} (x_i/x_p) \leqslant 1 \qquad 1 \leqslant k \leqslant S_x$$

$$\sum_{i \in I_k^y} (y_i/y_p) \leqslant 1 \qquad 1 \leqslant k \leqslant S_y$$

$$A_i/(x_i y_i) \leqslant 1 \qquad 1 \leqslant i \leqslant n$$

$$x_i^L/x_i \leqslant 1 \qquad 1 \leqslant i \leqslant n$$

$$x_i/x_i^U \leqslant 1 \qquad 1 \leqslant i \leqslant n$$

$$y_i^L/y_i \leqslant 1 \qquad 1 \leqslant i \leqslant n$$

$$y_i/y_i^U \leqslant 1 \qquad 1 \leqslant i \leqslant n$$

$$x_j/x_i + R_{ij}/x_i \leqslant 1 \qquad (i,j) \in R_x \subseteq R$$

$$y_j/y_i + R_{ij}/y_i \leqslant 1 \qquad (i,j) \in R_y \subseteq R$$

Since all the coefficients in the left-hand-sides of the constraints of P4 as well as in its objective are positive, this geometric program can be convexified by the well known exponential transformation of variables. In this way, the convexified problem can be solved by standard NLP techniques to yield the global optimum of P2. In general, though, the overlap constraints (27–28) will include a sum of variables added to x_i and y_i in the left-hand-side, thereby rendering the transformation non applicable.

6 COMPUTATIONAL EXPERIENCE

6.1 *Implementation*

The proposed global optimization algorithms were implemented using BARON (Ryoo and Sahinidis[26,27]). BARON is a general purpose global optimization software for solving nonlinear and mixed-integer nonlinear programs. Its optimization

strategy integrates conventional branch-and-bound with a wide variety of range reduction tools and branching rules. The user provides only the problem-specific subroutines for lower and upper bounding. The implementation was done on a SUN SPARC Station 2 with 32 MB of RAM using the GAMS (Brooke *et al.*[2]) version of BARON. The commercial LP software CPLEX[4] was used to solve the relaxed LPs and MINOS (Murtagh and Saunders[20]) was used to locally minimize the nonconvex NLPs. In solving all problems, the algorithms were terminated when the absolute difference between the upper and lower bounds for the optimal objective was within 10^{-5}.

6.2 *Sequential Approach Example Problems*

Consider the POG of Figure 1. There are five RDG morphologies corresponding to this circuit as shown in Figure 2. These were obtained by solving the layout equations of Step 2 of the sequential approach of Section 4. Each of these RDGs gives rise to one example of model P2. The first example corresponds to Figure 1c and gives rise to model P2-1 which was detailed in Section 4. The remaining RDGs of Figure 2 give rise to four additional examples. All examples share the same objective function but have different constraint sets corresponding to the different package layout:

Example 1:

(P2-1) min $x_A y_A + x_B y_B + x_C y_C + x_D y_D + x_E y_E + x_F y_F$

Subject to constraints (17–21).

Example 2:

(P2-2) min $x_A y_A + x_B y_B + x_C y_C + x_D y_D + x_E y_E + x_F y_F$

Subject to

$$x_B = x_C$$
$$x_B + x_E = x_C + x_F \Rightarrow x_E = x_F$$
$$x_A + x_D = x_C + x_F$$
$$y_A = y_D$$

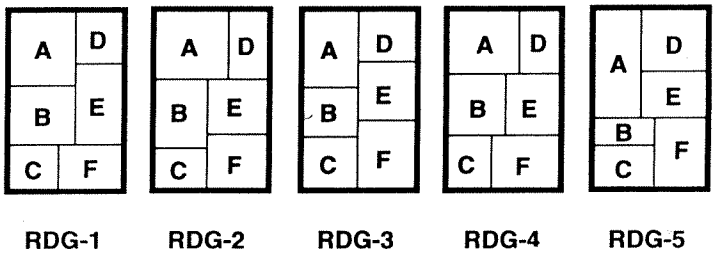

Figure 2 Five graphical RDG solutions to the layout equations.

$$y_A + y_B + y_C = y_D + y_E + y_F \Rightarrow y_B + y_C = y_E + y_F$$
$$x_A - x_B \geq 1$$
$$y_F - y_C \geq 1$$

and constraints (19–21).

Example 3:

(P2-3) min $x_A y_A + x_B y_B + x_C y_C + x_D y_D + x_E y_E + x_F y_F$

Subject to

$$x_B = x_C$$
$$x_B + x_E = x_C + x_F \Rightarrow x_E = x_F$$
$$x_A + x_E = x_C + x_F \Rightarrow x_A = x_C$$
$$x_A + x_D = x_C + x_F \Rightarrow x_D = x_F$$
$$y_A + y_B + y_C = y_D + y_E + y_F$$
$$y_F - y_C \geq 1$$
$$y_C + y_B - y_F \geq 1$$
$$y_A - y_D \geq 1$$

and constraints (19–21).

Example 4:

(P2-4) min $x_A y_A + x_B y_B + x_C y_C + x_D y_D + x_E y_E + x_F y_F$

Subject to

$$x_B + x_E = x_C + x_F$$
$$x_A + x_D = x_C + x_F$$
$$y_A = y_D$$
$$y_A + y_B = y_D + y_E \Rightarrow y_B = y_E$$
$$y_A + y_B + y_C = y_D + y_E + y_F \Rightarrow y_C = y_F$$
$$x_B - x_C \geq 1$$
$$x_A - x_B \geq 1$$

and constraints (19–21).

Example 5:

(P2-5) min $x_A y_A + x_B y_B + x_C y_C + x_D y_D + x_E y_E + x_F y_F$

Subject to

$$x_B = x_C$$
$$x_A + x_E = x_C + x_F$$
$$x_A + x_D = x_C + x_F$$

$$y_B + y_C = y_F$$

$$y_A + y_B + y_C = y_D + y_E + y_F \Rightarrow y_A = y_D + y_E$$

$$x_B - x_A \geqslant 1$$

and constraints (19–21).

6.3 *Computational Results with Bilinear Model*

Computational results with the example problems of the previous section are presented in Table 1. For each problem, the table presents the total number of nodes in the search tree, the node in which the optimal solution was found and the maximum number of nodes that had to be stored in memory at any point during the search. In all cases, using the solution of the root node relaxation as a starting point, local minimization led to the global minimum at the root node. Also, the CPU time averages about 1 sec. The row labeled "Objective" denotes the value of the global optimum while "Initial Bound" indicates the value of the relaxed problem at the root node. The last row of the table is defined as:

$$\text{"\% gap"} = 100(\text{"Objective"} - \text{"Initial Bound"})\ /\text{"Objective"}$$

This gap averages about 2.5% indicating that the relaxations are fairly accurate descriptions of model P2. However, this is not indicative of the difficulty of the package planning problem as, in deriving P2, the relative orientations of the chips have already been decided.

In addition to solving the above examples globally, local minima have been sought. In particular, GAMS/MINOS (Brooke *et al.*[2]) was used with 100 different randomly generated starting points for each example. In all cases, MINOS converged to the global solution. This is not surprising in light of the discussion of Section 5.5. It can be easily verified that the overlap constraints for these examples are of the form or can be brought into the form assumed in Section 5.5. Therefore, all these problems can be solved globally through the convexifying transformation and possess unique local minima. This result indicates that solving the bilinear models P2 in practice with a standard NLP code might be easy due to the absence of multiple local minima. Recall, however, that the convexifying transformation does

Table 1 Computational requirements of bilinear model P2.

	Example Problem				
	1	2	3	4	5
Nodes total	3	9	5	9	15
Nodes Optimal	1	1	1	1	1
Nodes memory	2	4	3	4	7
CPU sec	0.5	0.8	0.8	1	1.5
Objective	146.25	172.50	160.00	162.50	192.50
Initial Bound	143.56	168.04	156.89	158.00	186.13
% gap	1.84	2.59	1.94	2.77	3.31

not apply to all cases. In addition, P2 models only the continuous part of the problem and the relative positions of the chips were fixed in deriving P2. The effect of different chip orientations is shown in Table 2 where the solutions for the five examples are presented. It can be seen that the total package areas range from 146.25 for RDG-1 to 192.50 for RDG-5. The difference between the worse and best RDG is more than 30% indicating the importance of sampling all possible RDGs. In other words, it seems that different layouts may have a dramatic impact on the package area.

6.4 *Computational Results with MINLP Model*

In addition to generating all possible RDGs and solving model P2 for each of them, the MINLP model P1 has also been solved directly for the example problem of Figure 1. This approach alleviates the need for assuming the corner rectangles of the package. The MINLP model involved 30 binary variables, 26 continuous variables and 155 constraints. The LP relaxation contained 63 variables and 183 constraints. By noting that the edge-sharing requirements between rectangles define the relative positions of most of them, most of the binaries were fixed thus reducing the number of binary variables down to only 6 (two for each of the rectangle pairs A–E, B–E and B–F). The process of identifying the binary variables that can be fixed was completely automated. The solution of this model required 13 CPU sec. The root node relaxation had a value of 95 indicating a large gap of 35% when compared to the optimal solution of 146.25 (the same as with the sequential method of the Section 6.3). However, the range reduction tests were able to quickly reduce the gap. After the solution of the two descendant nodes of the root problem (*i.e.* after the

Table 2 Global solutions of example problems.

	Example Problem				
	1	2	3	4	5
x_A	5	6	5	6	5
x_B	5	5	5	5	6
x_C	4	5	5	4	6
x_D	4	4	5	4	6
x_E	4	5	5	5	6
x_F	5	5	5	6	5
y_A	7.25	6.25	6	6.25	9.1667
y_B	4	6	5	5	3.333
y_C	5	5	5	5	5
y_D	6.25	6.25	5	6.25	4.1667
y_E	5	5	5	5	5
y_F	5	6	6	5	8.333
Area of chip A	36.25	37.5	30	37.5	45.83
Area of chip B	20	30	25	25	20
Area of chip C	20	25	25	20	30
Area of chip D	25	25	25	25	25
Area of chip E	20	25	25	25	30
Area of chip F	25	30	30	30	41.667
Package Area	146.25	172.50	160.00	162.50	192.50

third iteration of the algorithm), the lower bound was improved to 137. The search tree took only 47 nodes, the optimum was found at iteration 9 and at most 9 nodes had to be stored in memory at any point during the search.

7 CONCLUSIONS

This paper addressed the chip layout and compaction problem. A MINLP formulation was proposed for simultaneous layout and two-dimensional compaction. This formulation allows for the dimensions of the chips to be merely constrained and not fixed. The model also incorporates a great variety of design constraints that arise from geometric design rules, distance and connectivity requirements between various components of the circuit, area and communication costs and other designer-specified requirements. A convergent global optimization algorithm was developed for the MINLP model as well as for a related bilinear programming formulation. The bilinear formulation arises as a subproblem in a sequential approach to the package planning problem that decomposes the layout from the packing decisions. Despite the theoretical difficulties associated with the problem, promising computational results were obtained using the global optimization software BARON to find the optimal layout and shape of a small circuit from the literature. It would be interesting to test the proposed algorithm on larger circuits. This will require the development of a careful computational implementation for handling the data structures of the algorithm, the development of specialized algorithms for efficiently solving all the subproblems, and also the interfacing of the algorithm with wire routing algorithms from the literature. Also, the extension of the MINLP formulation to the three dimensional case seems to be of interest for plant design and layout applications.

References

1. Bamji, C. S. and Varadarajan, R. (1992) Hierarchical pitchmatching compaction using minimum design. *Proc. 29th ACM/IEEE Design Automation Conference*, 311–317.
2. Brooke, A., Kendrick, D. and Meeraus, A. (1988) *GAMS-A User's Guide*. The Scientific Press, Redwood City.
3. Cieslielski, M. J. and Kinnen, E. (1982) An analytic method for compacting routing area in integrated circuits. *Proc. 19th ACM/IEEE Design Automation Conference*, **19**, 30–37.
4. CPLEX Optimization, Inc. (1993) *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*. Incline Village, NV.
5. Doenhardt, J. and Lengauer, T. (1987) Algorithmic aspects of one-dimensional layout compaction. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, **CAD-6**, 863–878.
6. Floudas, C. A. and Pardalos, P. M. (1990) *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Lecture Notes in Computer Science, **455**, Springer-Verlag, Berlin, Heidelberg.
7. Francis, R. L., McGinnies, L. F., Jr. and White, J. A. (1992) *Facility Layout and Location: An Analytical Approach*, 2nd Ed. Prentice-Hall, Englewood Cliffs, New Jersey.
8. Heller, W. R., Sorkin, G. and Maling, K. (1982) The planar package planner for system designers. *Proc. 19th ACM/IEEE Design Automation Conference*, **19**, 253–660.

The global optimization software BARON and related technical reports can be obtained by anonymous ftp from aristotle.me.uiuc.edu (128.174.124.10) or by sending an electronic mail to nikos@uiuc.edu.

9. Hill, D., Shugard, D., Fishburn, J. and Keutzer, K. (1989) *Algorithms and Techniques for VLSI Layout Synthesis.* Kluwer Academic Publishers, Norwell (Massachusetts).
10. Horst, R. and Tuy, H. (1993) *Global Optimization: Deterministic Approaches*, 2nd. Ed. Springer-Verlag, Berlin.
11. Hu, T. C. and Kuh, E. S. (eds.) (1985) *VLSI Circuit Layout: Theory and Design.* IEEE Press Selected Reprint Series, New York.
12. Kedem, G. and Watanabe, H. (1984) Graph-optimization techniques for IC layout and compaction. *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, **CAD-3**, 12–20.
13. Lengauer, T. (1982) The complexity of compacting hierarchically specified layouts of integrated circuits. *Proceedings, 23rd Annual Symp. on Foundations of Computer Science*, 358–368.
14. Lengauer, T. (1988) The combinatorial complexity of layout problems. Chapter 10 in Preas, B. T. and Lorenzetti, M. J. (eds.) (1988) *Physical Design Automation of VLSI Systems.* The Benjamin/Cummings Publishing Company, Menlo Park, California.
15. Lengauer, T. (1990) *Combinatorial Algorithms for Integrated Circuit Layout.* Wiley, New York.
16. Love, R. F., Morris, J. G. and Wesolowsky, G. O. (1988) *Facilities Location. Models & Methods.* North-Holland, New York.
17. Maling, K., Mueller, S. H. and Heller, W. R. (1982) On finding most optimal package plans. *Proceedings 19th ACM/IEEE Design Automation Conference*, **19**, 663–670.
18. McCormick, G. P. (1983) *Nonlinear Programming. Theory, Algorithms, and Applications.* Wiley Interscience, New York.
19. Mirchandani, P. B. and Francis, R. L. (eds.) (1990) *Discrete Location Theory.* Wiley, New York.
20. Murtagh, B. A. and Saunders, M. A. (1986) *MINOS 5.0 User's Guide.* Technical Report SOL 83-20, Systems Optimization Laboratory, Dept. of Operations Research, Stanford University, CA, USA.
21. Onodera, H., Taniguchi, Y. and Tamaru, K. (1991) Branch-and-bound placement for building block layout. *Proc. 28th ACM/IEEE Des. Aut. Conf.*, 433–439.
22. Otten, R. H. J. M. (1981) Automatic floorplan design. *Proc. 18th ACM/IEEE Design Automation Conference*, 261–267.
23. Otten, R. H. J. M. (1983) Efficient floorplan optimization. *Proc. Int. Conf. on Computer Design*, 499–503.
24. Pardalos, P. M. and Horst, R. (eds.) (1994) *Handbook of Global Optimization.* Kluwer Academic Publishers, Norwell, Massachusetts.
25. Preas, B. T. and Lorenzetti, M. J. (eds.) (1988) *Physical Design Automation of VLSI Systems.* The Benjamin/Cummings Publishing Company, Menlo Park, California.
26. Ryoo, H. S. and Sahinidis, N. V. (1995) Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, **19**, 5, 551–566.
27. Ryoo, H. S. and Sahinidis, N. V. (1994) A branch-and-reduce approach to global optimization, *Journal of Global Optimization*, submitted.
28. Sahinidis, N. V. (1994) *The Branch And Reduce Optimization Navigator (BARON)*, Working Paper, University of Illinois at Urbana-Champaign, Department of Mechanical and Industrial Engineering.
29. Sherali, H. D. and Alameddine, A. (1992) A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization*, **2**, (4), 379–410.
30. Soukup, J. (1981) Circuit layout. *Proceedins of IEEE*, **69**, (10), 1281–1304.
31. Stockmeyer, L. (1983) Optimal orientation of cells in slicing floorplan designs. *Information and Control*, **57**, (2), 91–101.
32. Sutanthavibul, S., Shragowitz, E. and Rosen, J. B. (1991) An analytical approach to floorplan design and optimization. *IEEE Transaction on Computer-Aided Design*, **10**, (6), 761–769.
33. Watanabe, H. (1984) *IC Layout Generation and Compaction Using Mathematical Optimization*, Ph. D. Thesis, Computer Science Dept., The University of Rochester.
34. Wolf, W. H. and Dunlop, A. E. (1988) Symbolic layout and compaction. Chapter 6 in Preas, B. T. and Lorenzetti, M. J. (eds.) (1988) *Physical Design Automation of VLSI Systems*, The Benjamin/Cummings Publishing Company, Menlo Park, California.
35. Yao, S. -Z., Cheng, C. -K., Dutt, D., Nahar, S. and Lo, C. -Y. (1993) Cells-based hierarchical pitch-matching compaction using minimal LP, *Proc. 30th ACM/IEEE Design Automation Conference*, 395–400.