Ann Oper Res (2010) 179: 243–260
DOI 10.1007/s10479-008-0462-7

A novel non-linear approach to minimal area rectangular packing

Volker Maag · Martin Berger · Anton Winterfeld · Karl-Heinz Küfer

Published online: 5 November 2008

Abstract This paper discusses the minimal area rectangular packing problem which is to pack a given set of rectangles into a rectangular container of minimal area such that no two rectangles overlap. Current approaches for this problem rely on metaheuristics like simulated annealing, on constraint programming or on non-linear models. Difficulties arise from the non-convexity and the combinatorial complexity. We investigate different mathematical programming approaches for this and introduce a novel approach based on non-linear optimization and the "tunneling effect" achieved by a relaxation of the non-overlapping constraints. We compare our optimization algorithm to a simulated annealing and a constraint programming approach and show that our approach is competitive. Additionally, since it is easy to extend, it is also applicable to a variety of related problems.

Keywords Rectangular packing · Non-overlapping constraints · Non-linear optimization · Regularization · Relaxation

1 Introduction

Packing problems of objects with arbitrary shapes arise in a multitude of important real world applications. In particular, packing problems of rectangular-shaped objects are intensively studied. Such rectangular packing problems for example arise in container loading, facility layout design, scheduling and many more.

V. Maag (✉) · M. Berger · A. Winterfeld · K.-H. Küfer
Fraunhofer Institute for Industrial Mathematics (Fraunhofer ITWM), Fraunhofer-Platz 1,
67663 Kaiserslautern, Germany
e-mail: volker.maag@itwm.fraunhofer.de

M. Berger
e-mail: martin.berger@itwm.fraunhofer.de

A. Winterfeld
e-mail: anton.winterfeld@itwm.fraunhofer.de

K.-H. Küfer
e-mail: karl-heinz.kuefer@itwm.fraunhofer.de

In microelectronics design, the layout of an electronic system includes the placement of its devices. Being part of the floorplanning, the *placement problem* is to position the interconnected devices on a rectangular board. As modern markets demand miniaturized systems the area minimization of the board is one of the main objectives of this rectangular packing problem.

Therefore, we focus on the following problem in rectangular packing:

Definition 1 A *minimal area rectangular packing problem* (MARPP) is to arrange a set of non-rotatable rectangles into a rectangular container of minimal area, such that the container includes all rectangles and no two rectangles overlap.

The optimization problem MARPP is $\mathcal{NP}$-hard (Murata et al. 1996). The rectangular container is also called *bounding box*. *Non-overlapping constraints* are that no two rectangles overlap and *containment constraints* are that all rectangles are in the container.

A large variety of models and optimization approaches have been developed and studied for rectangular packing problems. Approximation algorithms are mainly studied in the context of the theory of bin packing (Coffman et al. 1996; Bansal and Sviridenko 2004). They rely on the design of clever heuristics which are also used in the application to specific packing problems. Mixed integer programming (MIP) is another method to formulate such problems (Fasano 2004; Goetschalckx and Irohara 2007).

In particular for MARPP, also metaheuristics and constraint programming (CP) are applied. In microelectronics design, MARPP is often solved with simulated annealing (SA) and abstract solution representations (Wong and Liu 1986; Guo et al. 1999; Chang et al. 2000; Lin and Chang 2001; Pisinger 2007). Moffitt and Pollack (2006) apply CP and solve MARPP instances of up to 24 rectangles to optimality.

Formulating the MARPP as a non-linear problem may not seem like an obvious choice: Due to the non-convexity of the non-overlapping constraints standard gradient-based approaches likely stop in a local optimum and rarely find a good global solution (Horst and Tuy 1996). Nevertheless, there exist approaches to deal with such situations in general (Levy and Montalvo 1985; Ali et al. 1997; Wang and Zhang 2007) and in the context of rectangular packing problems (Alon and Ascher 1988; Dorneich and Sahinidis 1995; Ababei et al. 2005; Zhan et al. 2006).

For MARPP we propose a novel non-linear model, motivated by methods used for *general semi-infinite programming* (GSIP). The containment constraints are implemented as linear conditions of the form $Ax \leq b$. We formulate the non-overlapping constraints with a non-differentiable function and then regularize it, that is, we approximate it with a smooth function. The essential point is that the approximation is also a relaxation of the original problem in which the rectangles can change their relative positions more easily. We refer to this behaviour as the "*tunneling effect*". We use an extended version of the *penalty successive linear programming* approach (PSLP, Zhang and Kim 1985) to solve the resulting non-linear program (NLP).

While our non-linear approach allows to integrate arbitrary non-linear objective functions and non-linear constraints, a major drawback is its long runtime. However, the number of $\frac{n^2+n}{2}$ considered constraints, where n is the number of rectangles, can be significantly reduced by ignoring those constraints which are far from being active. This is known as "*active set method*" (Luenberger 1989). This method reduces the runtime of our algorithm considerably and renders it competitive to other approaches for MARPP.

The outline of this paper is as follows: In the second section we introduce our notation and apply it to MARPP. In the next section we give a broad survey of SA, CP and non-linear

approaches to rectangular packing. In the main section we present our novel non-linear model, propose an optimization algorithm for it and discuss properties of our approach. We show experiments in which we compare our method to two simulated annealing approaches and the optimal solutions given by a CP approach. We finish the paper with conclusions and perspectives for future research work.

2 Notation and standard problem formulation

Throughout this paper we use the following notation:

- $\mathcal{R} = \{r_1, \ldots, r_n\}$ denotes the set of rectangles.
- $l_1^{(i)}, l_2^{(i)}$ represent the width and the height of rectangle r_i.
- $c_1^{(i)}, c_2^{(i)}$ represent the center coordinate of rectangle r_i.
- b_1, b_2 represent the width and the height of the bounding box B.
- The area as objective function is denoted by $A = b_1 b_2$.

Now we can formulate MARPP as the following optimization problem:

$$\min\ b_1 b_2 \tag{$\mathcal{P}$}$$

subject to

$$\frac{1}{2}l_k^{(i)} \le c_k^{(i)} \le b_k - \frac{1}{2}l_k^{(i)} \quad \text{for } k \in \{1,2\} \text{ and } i \in \{1,\ldots,n\} \tag{1}$$

$$\left(c_1^{(i)} + \frac{1}{2}l_1^{(i)} \le c_1^{(j)} - \frac{1}{2}l_1^{(j)}\right) \vee \left(c_1^{(j)} + \frac{1}{2}l_1^{(j)} \le c_1^{(i)} - \frac{1}{2}l_1^{(i)}\right) \vee \tag{2}$$

$$\left(c_2^{(i)} + \frac{1}{2}l_2^{(i)} \le c_2^{(j)} - \frac{1}{2}l_2^{(j)}\right) \vee \left(c_2^{(j)} + \frac{1}{2}l_2^{(j)} \le c_2^{(i)} - \frac{1}{2}l_2^{(i)}\right)$$

$$\text{for } 0 < i < j \le n$$

We assume that the bounding box is anchored at the origin. Condition (1) guarantees that the rectangles are within the container B whereas (2) assures that no two rectangles overlap. The non-overlapping constraints express that any rectangle r_i is either left, right, below or on top of any other rectangle r_j.

3 Survey of other approaches to MARPP

3.1 Simulated annealing

In the following we briefly introduce the metaheuristic simulated annealing, show how to represent a rectangular packing with the *sequence pair* encoding scheme and sketch how one can solve MARPP in this way. We call the SA approach with the sequence pair in short SASP.

Many optimization problems appearing in real world applications are, in practice, not solvable with exhaustive search due to computation times exponentially growing with the size of the instances. Metaheuristics have successfully been applied to such optimization problems, especially to $\mathcal{NP}$-hard combinatorial problems. *Metaheuristics* are search methods which start from an initial solution and iteratively try to replace the current solution by

a better solution of the neighbourhood of the current solution. The class of metaheuristics includes—but is not restricted to—*Ant colony optimization*, *Evolutionary computation* including *Genetic algorithm*, *Iterated local search*, *SA* and *Tabu search* (Blum and Roli 2003).

We focus on SA as it is often used in placement problems (Wong and Liu 1986; Murata et al. 1996; Guo et al. 1999; Chang et al. 2000; Lin and Chang 2001; Pisinger 2007). SA is one of the oldest metaheuristics and originates from statistical mechanics (Kirkpatrick et al. 1983). The fundamental idea of SA applied to a minimization problem is to accept an intermediate solution z' to have a worse objective function $f(z')$ value than the current solution z. The probability $p(T, z', z)$ of such an acceptance decreases during search. It is generally computed following the Boltzmann distribution, i.e. $p(T, z', z) = \exp(-\frac{f(z')-f(z)}{T})$. The update of the temperature T usually follows a geometrical law, i.e. $T_{k+1} = \alpha T_k$ for $\alpha \in (0, 1)$. A pseudo-code of SA is given in Algorithm 1.

Algorithm 1 Pseudo code of SA

```
Initialize random starting solution z
Initialize temperature T
while termination condition not met do
  Pick neighbour z' ∈ N(z) through a random move
  if f(z') < f(z) then
    Replace z with z'
  else
    Accept z' as z with probability p(T, z', z)
  end if
  Update T
end while
```

The problem of encoding an arrangement of rectangles as a combinatorial object has been intensively studied. For placement problems, such an encoding of a packing is called a *floorplan representation*. Yao et al. (2003) give a broad overview of the multitude of different floorplan encoding schemes and their interrelationship. The floorplan representation *sequence pair* (SP) is an encoding scheme and was proposed in Murata et al. (1996). The following definition states the sequence pair for MARPP:

Definition 2 (Sequence Pair) Suppose the rectangles $r_i \in \mathcal{R}$ are to be packed. Then, a *sequence pair* $sp := (\Gamma_+, \Gamma_-)$ is a pair of rectangle sequences. Both sequences Γ_+ and Γ_- are permutations of $\mathcal{R}$.

Each non-overlapping constraint of (2) is a disjunction of linear inequalities. Depending on the linear order in Γ_+ and Γ_-, sp encodes exactly one topological relation $G \in$ {left of, right of, below, on top of} between each pair (r_i, r_j) of rectangles of $\mathcal{R}$, $i < j$. Therefore, at least one linear inequality holds. The consistent assignment of topological relations between the rectangles can be translated to a lower-left compacted packing. Each topological relation is encoded in one of two directed acyclic constraint graphs, one graph for horizontal relations and one for vertical relations. The vertices of the horizontal (vertical) constraint graph represent the rectangles and the edge weights represent the width (height) of the left (below) rectangle. The longest path to a vertex in the horizontal (vertical) constraint graph determines the horizontal (vertical) position of the corresponding rectangle of

a lower-left compacted packing. The longest paths to the rectangles are determined in $\mathcal{O}(n^2)$ time. For further properties on this translation we refer to Murata et al. (1996). This traditional translation was enhanced to other translations with $\mathcal{O}(n \log \log n)$ time by Tang et al. (2001); Pisinger (2007).

To complete the SASP approach for MARPP we define neighbourhood structures $\mathcal{N}$ which are explored during the annealing process. A *move* defines how to traverse randomly from a solution z to a neighbourhood solution $z' \in \mathcal{N}$. Moves for the sequence pair are based on randomly shifting or swapping rectangles in either one or both sequences Γ_+ and Γ_-. Typically, a rectangle is shifted in one sequence and pairs of rectangles are swapped in one or both sequences. To guarantee the diversification of SASP, moves should be chosen randomly out of several different move types. However, any sequence pair can be simply reached from any other sequence pair by consecutively applying any single move out of the described move types. More details on neighbourhood definitions and their properties can be found in Berger (2006).

3.2 Constraint programming

In the following we briefly survey constraint programming and how it can be applied to rectangular packing. represent the constraints of MARPP and sketch how to solve MARPP.

CP is a powerful paradigm for solving combinatorial search problems that draws on a wide range of techniques from artificial intelligence, computer science, databases, programming languages, and operations research (Rossi et al. 2006). From the CP viewpoint, the decision or optimization problem is to satisfy relations between variables stated in the form of constraints. In order to reduce the search effort CP develops strong inference and propagation methods for constraints.

Rectangular packing has also been a challenge for researchers from CP and several CP approaches are proposed for problems related to MARPP. Briefly, they differ in the way they model the non-overlapping constraint, how it is propagated and how search is branched. In general, branching is either done on the disjuncts of the non-overlapping constraint or done on the coordinates of the rectangles.

A constraint-based scheduling model for the *two-dimensional orthogonal packing problem* can be found in Clautiaux et al. (2008). The two-dimensional orthogonal packing problem consists in determining if a set of rectangles can be packed in a larger rectangle of fixed size. They use *energetic reasoning* together with a subset-sum propagation algorithm to effectively prune the search tree in a branch-and-bound framework.

Amossen and Pisinger (2006) proposed to solve multi-dimensional bin packing problems with guillotine constraints through a depth-first search with constructive assignment of the disjuncts of the non-overlapping constraints. During search, feasibility with respect to the guillotine constraints is maintained.

Moffitt and Pollack (2006) also applied backtracking search for constructively assigning disjuncts of the non-overlapping constraints of MARPP. They propose several new problem-specific as well as problem-independent pruning techniques to explore only feasible solutions of a reduced search tree. All-pair shortest path matrices are maintained during search and efficiently used to check in $\mathcal{O}(1)$ time if a topological relation between two rectangles is consistent to the current partial solution. They evaluate their approach by proving optimal solutions for packing squares of consecutive size into a container of minimal area. To our knowledge, their approach provides the fastest runtimes for the prove of optimality published at the time of submitting this paper. In Sect. 5 we compare the results of our approach to their optimal solutions.

3.3 Non-linear approaches

In the context of rectangular packing problems several approaches formulate the problem in different ways. We survey them in chronological order:

In Alon and Ascher (1988) the non-overlapping constraints are enforced by lower bounds on the Euclidean distance of the midpoints of the rectangles. This is an overestimation, however, it allows the rotation by any degree. The main objective is the minimization of the wire length. Constraints are added as a penalty term to the objective function.

The model proposed by Herrigel and Fichtner (1989) also allows 90° rotations and several other objectives. However, the resulting non-linear program has a structure which is hard to handle. The way the non-overlapping constraints are smoothed is similar to our approach, except that the regularization parameter is fixed. As a consequence, the error introduced by the regularization is not driven to zero which leaves a slight infeasibility after termination of the algorithm.

In Dorneich and Sahinidis (1995) a mixed integer non-linear programming approach is used. The shape of the rectangles can be changed to a certain amount and further constraints are considered, for example, that some pairs of rectangles have to share a common border. A combination of a non-linear solver and a branch-and-bound algorithm is proposed to solve the problem.

In Zhan et al. (2006) and Ababei et al. (2005) the main issue is a floorplanning algorithm. The size of the container is fixed but beside the positioning also the sizing of the rectangles is variable within a predefined range. The algorithm consists of two stages: In the first stage a uniform distribution of the rectangles is calculated which may violate feasibility slightly (that is, small overlaps may occur). In the second stage the overlapping is explicitly penalized to enforce feasibility. The overlapping is described by an approximation of maximum and minimum functions, therefore a post-processing step is necessary to eliminate remaining overlaps. The main objective here is to minimize the length of wires connecting the rectangles in a predefined way.

In Birgin et al. (2006) the container is assumed to be convex but need not be rectangular. The algorithm consists of an iterative loop where in each iteration the number of rectangles is increased and the violation of the containment and non-overlapping constraints is minimized. The algorithm terminates if the violation is not close to zero. For the containment constraints it is enough to check the four corners of a rectangle and for the non-overlapping constraints a smooth approximation of the maximum function is used.

A much more generic approach was recently stated by Winterfeld (2007) in the context of GSIP. It was shown that it is possible to fit several geometric objects O_i into a container C while optimizing the shape of both the objects and the container and preserving the non-overlapping constraints. In the context of MARPP, O_i correspond to the rectangles r_i and C to the bounding box B. Formulating the MARPP as a semi-infinite program leads to a more complex model than known non-linear models. Yet, in some numerical approaches for GSIP (Stein 2003) non-smooth functions (complementary conditions) appear which are regularized in a similar way as we do it in our approach. Our non-linear programming (NLP) solver is inspired by a solver for general semi-infinite programs and can easily integrate constraints of the above, very generic form.

4 A novel non-linear approach

We convert $\mathcal{P}$ to a non-linear problem in two steps. First we replace the disjunctions by maximum functions. Second, we approximate the maximum function by a parameterized

differentiable function. This step also introduces a relaxation of the problem which we study in more detail. Finally, we present our novel algorithm.

4.1 Reformulation of the problem

An equivalent formulation to $\mathcal{P}$ is the following:

$$\min\ b_1 b_2 \tag{$\mathcal{P}'$}$$

subject to

$$\frac{1}{2}l_k^{(i)} \le c_k^{(i)} \le b_k - \frac{1}{2}l_k^{(i)} \quad \text{for } k \in \{1,2\} \text{ and } i \in \{1,\dots,n\}$$

$$\max_{k\in\{1,2\}} \left(|c_k^{(i)} - c_k^{(j)}| - \frac{1}{2}(l_k^{(i)} + l_k^{(j)}) \right) \ge 0 \quad \text{for } 0 < i < j \le n \tag{3}$$

It is easy to see that (3) is just a reformulation of (2). The constraints (3) are still non-linear, non-convex and non-differentiable. Since differentiability is an essential assumption for most NLP solvers, we approximate the constraints by differentiable functions, a procedure which is known as *smoothing* or *regularization*.

4.2 Regularization of the problem

Our approach is based on a variant of the Chen-Harker-Kanzow-Smale function (Chen and Harker 1993) $f(a,b) = \frac{1}{2}(a+b-\sqrt{(a-b)^2})$ which is equivalent to the minimum function. The counterpart for the maximum function is $f(a,b) = \frac{1}{2}(a+b+\sqrt{(a-b)^2})$. A few similar functions (Sun and Qi 1999; Chen et al. 2000) are known as non-linear complementary problem (NCP) functions.[1] They are used to express the complementarity constraints appearing, for instance, in the Karush-Kuhn-Tucker optimality conditions (Bazaraa et al. 1993). For primal-dual and interior point methods these conditions arise explicitly and need to be regularized. This is usually done by inserting a regularization parameter τ such that the regularized function converges to the original function when τ goes to zero (Wright 1997; Ye 1997; Burke and Xu 2000).

The inequalities (3) are equivalent to

$$\max_{k\in\{1,2\}} \left((c_k^{(i)} - c_k^{(j)})^2 - \left(\frac{1}{2}(l_k^{(i)} + l_k^{(j)}) \right)^2 \right) \ge 0 \quad \text{for } 0 < i < j \le n$$

Therefore, by introducing the function $g_k(i,j) := (c_k^{(i)} - c_k^{(j)})^2 - \frac{1}{4}(l_k^{(i)} + l_k^{(j)})^2$, we get a new formulation of the non-overlapping constraints:

$$f(g_1(i,j), g_2(i,j)) \ge 0 \quad \text{for } 0 < i < j \le n \tag{4}$$

The regularized form[2] of the Chen-Harker-Kanzow-Smale function f is $f_\tau(a,b) := \frac{1}{2}(a+b+\sqrt{(a-b)^2+4\tau})$. For $\tau > 0$, f_τ is differentiable everywhere and the regularized problem is

$$\min\ b_1 b_2 \tag{$\mathcal{P}_\tau$}$$

[1] We do not have any evidence that one of the functions is preferable. The comparison of the numerical behaviour of different NCP functions in our context is subject to further research.

[2] Often also stated as $f_\tau(a,b) := \frac{1}{2}(a+b+\sqrt{(a-b)^2+4\tau^2})$.

subject to

$$\frac{1}{2}l_k^{(i)} \le c_k^{(i)} \le b_k - \frac{1}{2}l_k^{(i)} \quad \text{for } k \in \{1,2\} \text{ and } i \in \{1,\dots,n\} \tag{5}$$

$$f_\tau(g_1(i,j), g_2(i,j)) \ge 0 \quad \text{for } 0 < i < j \le n \tag{6}$$

Note that $f_0 \equiv f \equiv \max$ and $f_\tau(a,b) \ge f(a,b)$. Therefore, the set of feasible solutions of $(\mathcal{P})$ is contained in the one of $(\mathcal{P}_\tau)$. That means that replacing the condition (4) by (6) causes not only a smoothing but also a relaxation of the problem. The relaxation has a specific interpretation: Depending on the size of τ, condition (6) allows partial overlapping or even containment of the rectangles. In the context of the global optimization problem this can be used to get away from local minima. The effect of this mechanism is illustrated in Fig. 3 in Sect. 5.3. Winterfeld (2007) describes an analog observation in the context of GSIP.

4.3 Analysis of the tunneling effect

In this section we properly analyze the effect caused by the relaxation and derive information on the dependency between the maximal overlapping and the regularization parameter τ. We restrict ourselves to squares here, because the fact that the width is equal the height simplifies the presentation. However, the results can be transferred to rectangles.

We assume in the following $l_1^{(i)} = l_2^{(i)}$ and omit the subscript. First we need a stricter notion of the overlapping.

Definition 3 Consider two squares with midpoints $c^{(1)},\ c^{(2)} \in \mathbb{R}^2$ and side lengths $l^{(1)},\ l^{(2)}$, respectively. The *degree of overlapping* is given by $d(c^{(1)}, c^{(2)}) = \max\{0, \frac{1}{2}(l^{(1)} + l^{(2)}) - \max_{k\in\{1,2\}}\{|c_k^{(1)} - c_k^{(2)}|\}\}$.

Note that $d(c^{(1)}, c^{(2)}) > 0$ if and only if the corresponding squares overlap, that means, the interior of their intersection is non-empty. Furthermore, $d(c^{(1)}, c^{(2)}) \le \frac{1}{2}(l^{(1)} + l^{(2)})$ and equality holds when the midpoints coincide. In Fig. 1 the degree of overlapping is indicated by o.

Extending the definition of the degree of non-overlapping to rectangles would require to consider subcases depending whether the larger overlap occurs in vertical or horizontal direction. This is because vertical and horizontal side length differ in case of non-squares.

Lemma 1 *Given two squares with side lengths $l^{(1)}$ and $l^{(2)}$. For $r := \frac{1}{2}(l^{(1)} + l^{(2)})$, any $o \in [0, r]$ and $\tau := (2ro - o^2)r^2$ equation (6) guarantees a degree of overlapping smaller than or equal to o.*

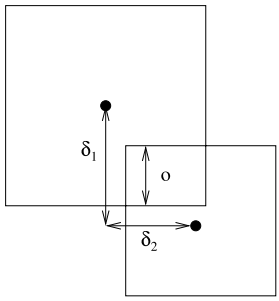

Fig. 1 Degree of overlapping for two squares

Proof Assume that the midpoints $c^{(1)}$ and $c^{(2)}$ of the two squares have a distance of δ_1 and δ_2 in the corresponding dimension and $d(c^{(1)}, c^{(2)}) > o$. Furthermore without loss of generality assume $\delta_1 \geq \delta_2$. Then $\delta_1 < r - o$ (the assumption implies $o < r$), $g_k(1,2) = \delta_k^2 - r^2$ for $k \in \{1,2\}$ and we have

$$\begin{aligned} f_\tau(g_1(1,2), g_2(1,2)) &= \frac{1}{2}\left(\delta_1^2 - r^2 + \delta_2^2 - r^2 + \sqrt{(\delta_1^2 - \delta_2^2)^2 + 4(2ro - o^2)r^2}\right) \\ &\leq -r^2 + \frac{1}{2}\left(\delta_1^2 + \sqrt{\delta_2^4 + 4(2ro - o^2)r^2}\right) \\ &< -r^2 + \frac{1}{2}\left((r-o)^2 + \sqrt{(r-o)^4 + 8r^3o - 4r^2o^2}\right) \\ &= -r^2 + \frac{1}{2}\left(r^2 - 2ro + o^2 + \sqrt{(-r^2 - 2ro + o^2)^2}\right) \\ &= 0 \text{ using that } -r^2 - 2ro + o^2 \leq 0 \end{aligned}$$

which contradicts (6). □

Corollary 1 *Given two squares with side lengths $l^{(1)}$ and $l^{(2)}$, $r := \frac{1}{2}(l^{(1)} + l^{(2)})$ and $\tau \in [0, r^4]$, the equation (6) guarantees a degree of overlapping smaller than or equal to $r - \sqrt{r^2 - \frac{\tau}{r^2}}$.*

Corollary 2 *There exists a τ such that the constraints of the relaxed problem $\mathcal{P}_\tau$ hold if and only if the containment constraints are fulfilled.*

The above statements show how to explicitly control the maximal overlapping for a given pair of squares. However, since the overlapping depends also on the size of the squares, the same τ will lead to different maximal overlappings for different pairs of squares.

Corollaries 1 and 2 are also relevant for rectangles, since for any pair of them one can take the smallest enclosing squares, that is, the side length of each square is the maximum of width and height of the corresponding rectangle. The degree of overlapping of the squares overestimates the overlap of the rectangles and, therefore, upper bounds on the overlap of the smallest enclosing squares are also valid for the enclosed rectangles.

4.4 Our novel algorithm

The algorithm consists of three nested loops as shown in Algorithm 2. In the outer loop we determine n_1 starting solutions and in the middle loop an initial $\tau = \tau_1^{(j)}$ is fixed. The maximal number of iterations in this loop is restricted by n_2. The inner loop is within the regularized NLP solver, which iteratively calculates a locally optimal solution of $\mathcal{P}'$. In the following we describe the NLP solver in more detail, the integration of the active set method, the way we generated starting solutions and the most influential parameters of the algorithm.

The regularized NLP solver. The non-linear solver we used is based on PSLP extended by a strategy to reduce the regularization parameter τ to zero.[3] The essential ingredients of PSLP are:

[3] In our implementation this means $\tau < 10^{-6}$.

Algorithm 2 Pseudo code of our novel algorithm

```
for i := 1 to n_1 do
    Initialize random starting solution
    for j := 1 to n_2 do
        Initialize τ_1^(j)
        Run regularized NLP solver
        if no significant improvement was achieved then
            leave middle loop
        end if
    end for
end for
return best of the n_1 obtained solutions
```

- The constraints are added as a penalty term to the objective function multiplied with a penalty factor μ.
- In each iteration k the current solution $x^{(k)} = \{b^{(k)}, c^{(1,k)}, \ldots, c^{(n,k)}\}$ is calculated by solving a linearization of the problem at the previous solution $x^{(k-1)}$.
- As an additional constraint it is required that $x^{(k)}$ must be within a *trust region* $T(x^{(k-1)}) = \{y \in \mathbb{R}^{2n+2} : ||y - x^{(k-1)}||_\infty \leq \delta_k\}$ where $\delta_k > 0$ determines the size of the trust region in step k.
- The trust region is adapted depending on the ratio of the improvement of the objectives of the linearized model and of the non-linear model. If the ratio is close to one or larger, the trust region size is increased, if it does not exceed an upper bound δ_{UB}. If it is not too far from zero the trust region size is decreased. If it is nearly zero or negative, $x^{(k)}$ is rejected and the current iteration is repeated with a smaller trust region.
- The stopping criterion is that the gradient of the penalized objective is close to zero and there is no change in the objective value.

If the initial solution is feasible and μ is chosen large enough,[4] this algorithm terminates with a Karush-Kuhn-Tucker point which is usually a local optimum.

In order to handle the regularization, τ is considered as another variable with a separate kind of trust region. τ also appears as an additional term in the extended objective weighted by a factor. In this way, it is automatically driven from its initial value $\tau_1^{(j)}$ to zero during the iterations of the inner loop.

Using active set method to improve the runtime. The part of the algorithm which is most time-consuming is solving the linearized problem in each PSLP iteration. Hence, decreasing the size of it is the most efficient way to speed up the algorithm. We do that by removing constraints which cannot be violated within an iteration of PSLP even if they are temporarily disregarded by the solver. This holds for constraints of the form $h(x) \geq 0$ at the k-th iteration if

$$h(y) \geq 0 \quad \text{for all } y \in T(x^{(k-1)}) \tag{7}$$

since $x^{(k)} \in T(x^{(k-1)})$. For the constraints of $\mathcal{P}'$ we can check (7) directly. Since $\mathcal{P}_\tau$ is a relaxation, we can disregard a regularized constraint if (7) holds for the corresponding non-regularized constraint.

[4] We use $\mu = 10^5$.

This approach reduces the number of constraints to a fraction of the original number, since now the non-overlapping constraints need only be checked for pairs of rectangles which are in a neighbourhood of each other. This is much smaller than the overall number of rectangles, provided that the trust region size δ is not too large.

In the standard PSLP method, the trust region size depends on the quality of the linear approximation. The larger the size is, the larger each step may be and the fewer iterations are necessary. For the active set method, the trust region size also determines the number of constraints to be considered. The smaller the size is, the faster each iteration is. Therefore, the adjustment of the trust region described above was extended such that $\delta_{k+1} = \min\{\delta_k, \sigma\|x^{(k)} - x^{(k+1)}\|_\infty\}$ for some $\sigma \geq 1$. That means that the trust region size is decreased if the change in the current solutions of successive iterations is small anyway.[5]

The starting solution. The quality of the final solution significantly depends on the starting solution. Yet, the dependency seems to be arbitrary. We cannot expect to find starting solutions in a general way such that our algorithm always converges to a final solution close to a global optimum.

Therefore, we did not use sophisticated heuristics, but rather arranged the rectangles in such a way that the lower right corner of the i-th rectangle touches the upper right corner of the $i+1$-th rectangle. The order of the rectangles is subject to randomization. In the first iterations of the inner loop the rectangles are pushed together without any bias to a particular arrangement, which is a necessary requirement for good starting solutions.

It is worth noting that the starting solution need not be feasible for $\mathcal{P}$ but only for $\mathcal{P}_{\tau_1^{(1)}}$. If $\tau_1^{(1)}$ is chosen large enough, one could even put all rectangles on top of each other.

5 Numerical results

In this section we first present our observations concerning the dependency of the algorithm on its parameters. Then, we compare our results with two SASP implementations, one of Pisinger (2007) and our own using the SA of Algorithm 1 with SP translation as in Murata et al. (1996). Furthermore, for small problem instances, we show the relation to optimal results from Moffitt and Pollack (2006). Finally we have a closer look to an exemplary single run of the novel algorithm and discuss similarities of it to SA.

To enable the comparison with Moffitt and Pollack (2006) we use the same problem setup as they did, that is, n squares with side lengths $1, 2, \ldots, n$, respectively. For solving the linear programs arising in PSLP we used ILOG CPLEX. The results of the non-linear approach were calculated on an Intel Quad-Core Xeon E5420 with 2.5 GHz. For licensing reasons of the ILOG CPLEX solver we could not parallelize our program. When using the four processors of the machine in parallel, our runtimes can roughly be quartered. Our SASP approach was tested on a Intel Core Duo processor T2500 with 2.0 GHz and the SASP approach of Pisinger (2007) was run on an AMD64 processor with 2.4 GHz. Since in all cases only one CPU core was used, the processor performances are comparable and do not delimit our comparison. All presented runtimes are given in seconds.

[5] We used $\sigma = 2$.

Table 1 The *second row* shows the best area calculated by the algorithm for different numbers of squares (*column*) over $n_1 = 512$ random starting solutions. The following rows present the deviation of the best solution in percent for smaller numbers of starting solutions

	10	50	100	200
512	408	47443	388128	3.2373e+6
16	5.88	3.32	1.19	6530.83
32	4.17	1.96	1.19	6416.85
64	0.00	1.93	1.19	1.92
128	0.00	1.07	1.19	0.43
256	0.00	0.40	0.00	0.00

Table 2 Average values, taken over a sample of 512 runs

Squares	10	50	100	200
Average runtime	0.075	0.48	1.3	6.1
Average absolute deviation	0.024	0.073	0.070	1.1

5.1 Dependency of the novel algorithm on the parameters

The PSLP algorithm with the extensions for regularization and the active set method has a number of parameters whose values we chose empirically. From our studies we conclude that the exact choice of most parameters is not essential for the performance of the algorithm. For the results presented here we used the following values:

- initial value $\tau_1^{(1)} = 370$
- adaptation factor $\alpha = \frac{\tau_1^{(j+1)}}{\tau_1^{(j)}} = 0.6$
- upper bound for the trust region size $\delta_{\mathrm{UB}} = 50$
- initial trust region size $\delta_1 = 1.7$

The number of starting solutions n_1 has a large impact on the quality of the final solution. As we take the best of the n_1 corresponding outcomes, an increase of n_1 improves the likelihood to find a good solution. In order to determine a reasonable value for n_1 we ran the algorithm for different values up to 512 and for different problem classes from 10 to 200 squares as shown in Table 1. The improvements for more than 128 runs were small, hence we used $n_1 = 128$ for the following comparisons in Fig. 2 and Table 3. Table 1 also indicates, that for another set of 128 random starting solutions the outcome of the algorithm is not much different therefore, we do not present different results from runs in the following.

The runtime of our algorithm is proportional to n_1. Table 2 shows the average runtimes for single runs and the average absolute deviation, that is, the average of the absolute value of the deviation from the average runtimes. Since this deviation is small, the overall runtime over 128 starting solutions hardly depends on the choice of the starting solutions. Therefore, one run of the algorithm gives an accurate value for the runtime in average.

5.2 Comparisons with other approaches

For up to 24 squares Moffitt and Pollack (2006) provide optimal results. Their results are depicted in Fig. 2, together with our results from the non-linear and SASP approach. Our

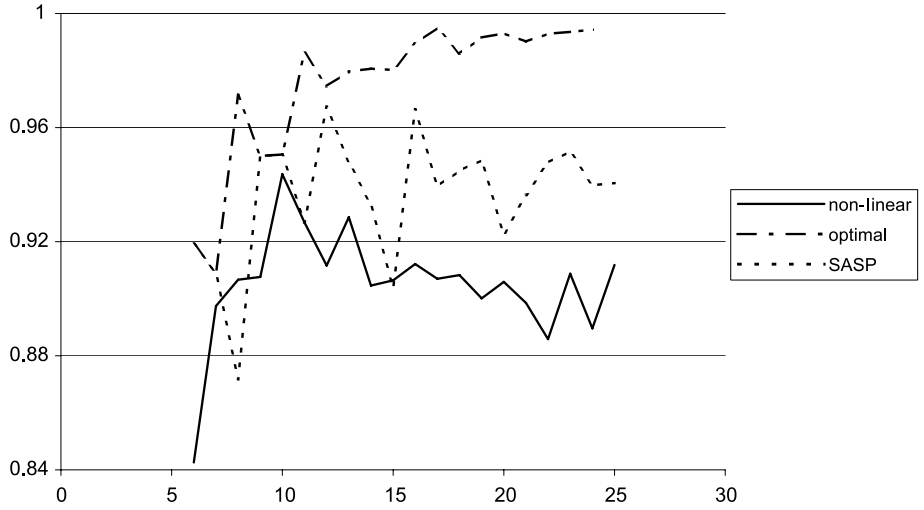

Fig. 2 Comparison of the area usage (y-axis) for the three different methods for different numbers of squares (x-axis)

Table 3 Results for larger number of squares

	Non-linear			Our SASP			Pisinger (2007)		
Squares	Area	Usage	Time	Area	Usage	Time	Area	Usage	Time
10	408	0.944	6.29	405	0.951	2.56	405	0.951	6.95
25	6059	0.912	19.4	5874	0.941	18.5	5655	0.977	28.88
50	47952	0.895	56.9	44928	0.955	210.0	43575	0.985	97.80
75	162000	0.885	123.4	150577	0.953	448.8	–	–	–
100	392764	0.861	156.6	355568	0.952	1016	343371	0.985	372.79
150	1301340	0.873	339.2	1192202	0.953	3386	–	–	–

results are worse than the optimal results, but the time needed to calculate the optimal solutions increases much faster for the CP approach of Moffitt and Pollack (2006). It needs one and a half minute for 15 squares and 146 hours for 24 squares, while our non-exhaustive approaches need less than 20 seconds in that case.

Table 3 depicts the outcomes for larger problem sizes. The results of the non-linear approach are compared with the SASP approaches. From Pisinger (2007) we took the results for the long runs of the algorithm with one million iterations. For each problem instance, Pisinger presented the area and the runtime for the best of ten runs. Our non-linear approach takes the best out of 128 runs but the sum of the runtimes of all runs. In the table we refer to *usage* as the area usage given by the sum of the area of the squares divided by the area of the minimal bounding box calculated by the algorithms.

Our SASP implementation proved to be less sensitive regarding the choice of the starting solution and of the moves. Different runs did not yield significant differences in the quality of the solutions. Therefore, we run SASP only once from a randomly chosen starting solution.

The algorithm from Pisinger (2007) yields the best area usage. Yet, their runtimes are in the same order of magnitude.

We used straightforward heuristics for the choices of parameters and its adaptation during the algorithm. We are convinced, that improvements of these will lead to better results and better runtimes. Yet, SASP is more efficient in representing and searching the solution space. The advantage of our novel algorithm is, that it is not restricted to the characteristics of MARPP and can be easily extended by further constraints and other objectives. So, the strengths of our approach are its expressiveness and extensibility.

5.3 Detailed study of the behaviour of our novel algorithm

Figure 3 demonstrates how the outer loop of the algorithm works. Here, we used rectangles with side length uniformly distributed between 1 and 9 units of length. Starting from the initial solution with relaxed non-overlapping constraints the rectangles move quickly together and overlap for some iterations (Fig. 3(b)) until the first local optimum in Fig. 3(c) is reached. When the NLP solver is restarted and the non-overlapping constraints are relaxed

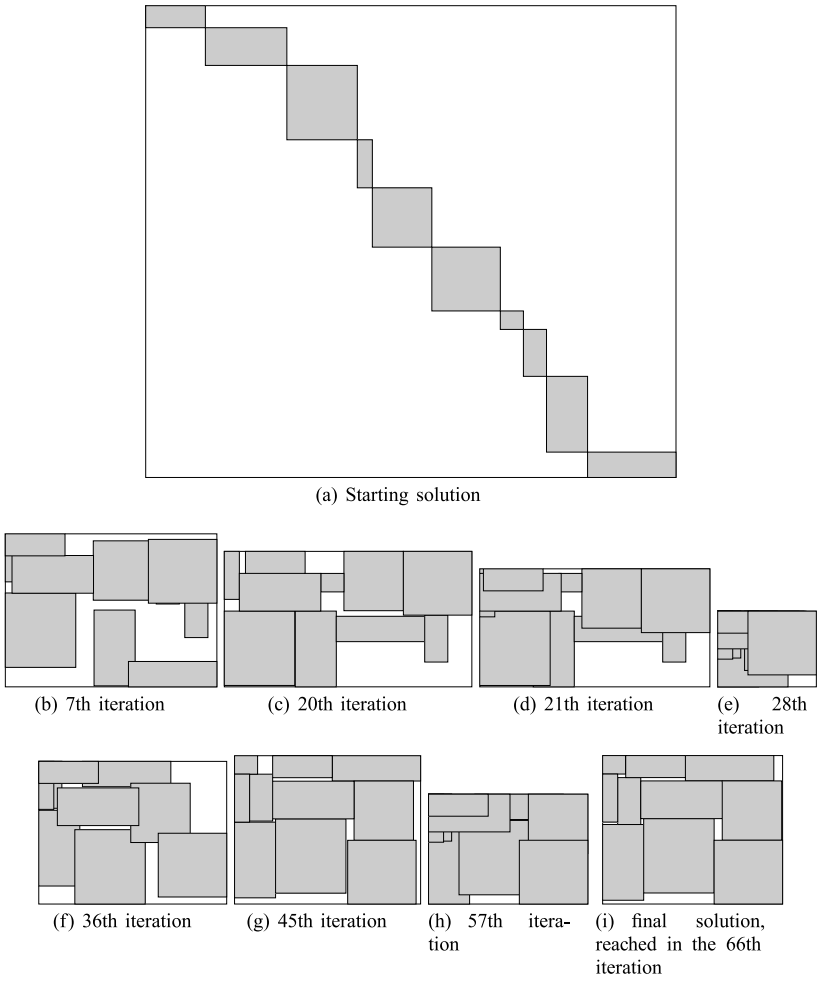

Fig. 3 Selected iterations of one optimization run with ten rectangles

again, the bounding box is further compressed and the rectangles shift on top of each other. Figure 3(e) depicts the state in which the overlap is largest. When the allowed relaxation is reduced, overlaps of rectangles disappear and the rectangle positions converge to another locally optimal, feasible solution shown in Fig. 3(g). For the next restart of the NLP solver the maximal overlap is smaller, therefore the next state with smallest bounding box shown in Fig. 3(h) is not as compact anymore as in Fig. 3(e). In iteration 66 another locally optimal solution (Fig. 3(i)) is reached which still provides an improvement compared to Fig. 3(g). But the next NLP solver restart ends up with the same local optimum and the algorithm terminates.

The figures indicate that the rectangles tend to move to the left upper corner and the overlap close to that corner is largest. This phenomenon is caused by the underlying non-linear solver and cannot be controlled directly. It does not contribute to an improvement of the objective and is undesired since it gives the solution some bias. One way to eliminate this property is to change the non-linear solver. Alternatively, a bound on the maximal degree of overlapping can prevent this phenomenon.

5.4 Similarities between the our novel algorithm and SA

Using SA for global optimization of a continuous function is not a new idea (Ali et al. 1997). Recently, Wang and Zhang (2007) explicitly combined SA with a gradient-based optimization method. By interpreting the inner and the middle loop in a particular manner, we can see that our approach has similarities to SA.

For the inner loop we take the regularization parameter τ to be the temperature and the infeasibility as the energy configuration for a moment. A low energy configuration is achieved when no rectangles overlap and is enforced for $\tau = 0$. By corollary 1 we can interpret the reduction of τ as cooling the system, since we reduce the allowed degree of overlapping. The difference to SA is that a worse state is not accepted according to a probability function. Instead, any improvement of the actual objective, the area of the bounding box, is accepted which does not violate the limit of overlapping determined by τ. In practice it turns out that in each iteration the current solutions achieve the maximal degree of non-overlapping allowed by the current value of τ.

Also the middle loop has a similar interpretation. Again we can consider τ as the temperature. Now, the inner loop can be seen as a move which changes the current solution. The degree of the change is determined by τ, with which the regularized NLP solver is initialized. However, this analogy is not carried out completely so far. The middle loop does not stop when τ is small enough but when no further improvements were achieved. Also, deteriorations are not accepted in any case. However, the algorithm can easily be adapted to represent this strategy.

Putting both loops together one can consider the outer loop as a kind of reheating, which is an idea well known for SA (Kolonko 1999; Anagnostopoulos et al. 2006).

6 Conclusion and future research

We presented a novel approach to solve the MARPP based on a continuous model and on a regularization of the maximum function. We compared our approach to SASP approaches and it turned out that it is competitive. The special features of this algorithm are that it always provides a feasible solution and the tunneling effect. This technique uses the relaxation of the non-overlapping constraints to escape from local optima and shows similarities to meta-heuristic concepts. The active set method reduces the number of constraints considerably

and speeds up the algorithm. Finally, the major strength of our model is that it can be easily extended with other smooth objectives and constraints.

Such extensions may especially benefit from the tunneling effect. For instance, when minimizing the length of wires the gradient of the objective yields more information. These are useful in particular when the relaxation causes a large degree of freedom. Furthermore, one could make use of the possibilities offered by the non-linear, continuous formulation of the problem. For example, in microelectronics, the rectangles correspond to modules which are interconnected by wires in a predefined way and one important goal is to keep the length of the wires as short as possible. Common wire-length models lead to a continuous objective function.

Extending this approach to three (or higher) dimensions may be interesting. The main issue here is that instead of smoothing $\max\{a(x), b(x)\}$ one has to consider $\max\{a(x), b(x), c(x)\}$. To do so, other regularization functions are needed. However, the underlying algorithm stays the same, whereas SA or CP approaches have to deal with a significantly higher combinatorial complexity.

Hybridization of our approach with other approaches like SA or CP might yield improvements. They could complement each other in a framework which unifies the robust sampling of the solution space from metaheuristics, the strong propagation mechanisms from CP and the flexible relaxation from global non-linear optimization. For example, one could switch between SA moves and iterations of the NLP solver as described in (Wang and Zhang 2007). Also, it should be possible to use CP with its strong methods to investigate arrangements of a subset of the rectangles with additional constraints.

Acknowledgements We are indebted to three anonymous referees for their insightful comments on drafts of this paper. We thank Conor John Fitzsimons for his constructive and helpful feedback on our work. Also, we are grateful to Michael Schröder for his great support and guidance throughout our research. For the first and second author the research originated from current PhD projects. It was funded by Fraunhofer ITWM.

References

Ababei, C., Feng, Y., Goplen, B., Mogal, H., Bazargan, K., Sapatnekar, S. S., & Zhang, T. (2005). Placement and routing in 3D integrated circuits. *IEEE Design and Test of Computers*, *22*(6), 520–531.

Ali, M., Törn, A., & Viitanen, S. (1997). *A direct search simulated annealing algorithm for optimization involving continuous variables* (Technical Report TUCS-TR-97).

Alon, A., & Ascher, U. (1988). Model and solution strategy for placement of rectangular blocks in the euclidean plane. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *7*(3), 378–386.

Amossen, R. R., & Pisinger, D. (2006). Multi-dimensional bin packing problems with guillotine constraints. *Computers and Operations Research*.

Anagnostopoulos, A., Michel, L., Hentenryck, P., & Vergados, Y. (2006). A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling*, *9*(2), 177–193.

Bansal, N., & Sviridenko, M. (2004). New approximability and inapproximability results for 2-dimensional bin packing. In *Proceedings of the fifteenth annual SIAM symposium on discrete algorithms* (pp. 196–203). Philadelphia, PA, USA. Philadelphia: Society for Industrial and Applied Mathematics.

Bazaraa, M., Sherali, H. D., & Shetty, C. (1993). *Nonlinear programming: theory and algorithms* (2nd ed.). New York: Wiley.

Berger, M. (2006). *Module placement in 2.5D system in package design automation*. Master's thesis, University of Applied Sciences, Mittweida.

Birgin, E. G., Martìnez, J. M., Nishihara, F. H., & Ronconi, D. P. (2006). Orthogonal packing of rectangular items within arbitrary convex regions by nonlinear optimization. *Computers and Operations Research*, *33*(12), 3535–3548.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, *35*(3), 268–308.

Burke, J. V., & Xu, S. (2000). A non-interior predictor-corrector path-following algorithm for the monotone linear complementarity problem. *Mathematical Programming*, *A87*, 113–130.

Chang, Y. C., Chang, Y.-W., Wu, G. M., & Wu, S. W. (2000). B*-trees: A new representation for non-slicing floorplans. In *Proceedings of the 37th conference on design automation* (pp. 458–463).

Chen, B., Chen, X., & Kanzow, C. (2000). A penalized Fischer-Burmeister NCP-function. *Mathematical Programming*, *88*(1), 211–216.

Chen, B., & Harker, P. T. (1993). A non-interior-point continuation method for linear complementarity problems. *SIAM Journal on Matrix Analysis and Applications*, *14*(4), 1168–1190.

Clautiaux, F., Jouglet, A., Carlier, J., & Moukrim, A. (2008). A new constraint programming approach for the orthogonal packing problem. *Computers and Operations Research*, *35*(3), 944–959.

Coffman, E. G., Garey, M. R., & Johnson, D. S. (1996). Approximation algorithms for bin packing: A survey. In D. Hochbaum (Ed.), *Approximation algorithms for NP-hard problems* (pp. 46–93). Boston: PWS Publishing.

Dorneich, M. C., & Sahinidis, N. V. (1995). Global optimization algorithms for chip layout and compaction. *Engineering Optimization*, *25*(2), 131–154.

Fasano, G. (2004). A MIP approach for some practical packing problems: Balancing constraints and tetris-like items. *4OR: A Quarterly Journal of Operations Research*, *2*(2), 161–174.

Goetschalckx, M., & Irohara, T. (2007). Efficient formulations for the multi-floor facility layout problem with elevators. *Optimization Online*.

Guo, P.-N., Cheng, C.-K., & Yoshimura, T. (1999). An O-Tree representation of non-slicing floorplan and its applications. In *Proceedings of the 1999 design automation conference* (pp. 268–273).

Herrigel, A., & Fichtner, W. (1989). An analytic optimization technique for placement of macro-cells. In *DAC '89: Proceedings of the 26th ACM/IEEE conference on design automation* (pp. 376–381). New York, NY, USA. New York: ACM Press.

Horst, R., & Tuy, H. (1996). *Global optimization: deterministic approaches* (3rd ed.). Heidelberg: Springer.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598).

Kolonko, M. (1999). Some new results on simulated annealing applied to the job shop scheduling problem. *European Journal of Operational Research*, *113*(1), 123–136.

Levy, A. V., & Montalvo, A. (1985). The tunneling algorithm for the global minimization of functions. *SIAM Journal on Scientific and Statistical Computing*, *6*(1), 15–29.

Lin, J. M., & Chang, Y.-W. (2001). TCG: A transitive closure graph-based representation for non-slicing floorplans. In *Proceedings of the 2001 design automation conference* (pp. 764–769).

Luenberger, D. G. (1989). *Linear and nonlinear programming*. Reading: Addison-Wesley.

Moffitt, M. D., & Pollack, M. E. (2006). Optimal rectangle packing: A Meta-CSP approach. In *Proceedings of the 16th international conference on automated planning and scheduling*.

Murata, H., Fujiyoshi, K., Nakatake, S., & Kajitani, Y. (1996). VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *15*(12), 1518–1524.

Pisinger, D. (2007). Denser packings obtained in O(n log log n) time. *INFORMS Journal of Computing*, *19*(3), 395–405.

Rossi, F., van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming*. Amsterdam: Elsevier B.V.

Stein, O. (2003). *Bi-level strategies in semi-infinite programming*. Boston: Kluwer.

Sun, D., & Qi, L. (1999). On NCP-functions. *Computational Optimization and Applications*, *13*, 201–220.

Tang, X., Tian, R., & Wong, D. F. (2001). FAST-SP: A fast algorithm for block placement based on sequence pair. In *Proceedings of the 2001 Asia and South Pacific design automation conference* (pp. 521–526). New York: ACM Press.

Wang, Y.-J., & Zhang, J.-S. (2007). An efficient algorithm for large scale global optimization of continuous functions. *Journal of Computational and Applied Mathematics*, *206*(2), 1015–1026.

Winterfeld, A. (2007). *Large-scale semi-infinite optimization applied to industrial gemstone cutting*. PhD thesis, University of Kaiserslautern.

Wong, D. F., & Liu, C. L. (1986). A new algorithm for floorplan design. In *DAC '86: Proceedings of the 23rd ACM/IEEE conference on design automation* (pp. 101–107). Piscataway, NJ, USA. New York: IEEE Press.

Wright, S. J. (1997). *Primal-dual interior-point methods*. Philadelphia: SIAM.

Yao, B., Chen, H., Cheng, C.-K., & Graham, R. L. (2003). Floorplan representations: Complexity and connections. *ACM Transactions on Design of Automated Electronic Systems*, *8*(1), 55–80.

Ye, Y. (1997). *Interior point algorithms, theory and analysis*. New York: Wiley.
Zhan, Y., Feng, Y., & Sapatnekar, S. S. (2006). A fixed-die floorplanning algorithm using an analytical approach. In *Proceedings of the 2006 conference on Asia South Pacific design automation* (pp. 771–776). New York, NY, USA. New York: ACM Press.
Zhang, J. Z., & Kim, N. H. (1985). An improved successive linear programming algorithm. *Management Science*, *31*(10), 1312–1331.