Automatic Placer for Analog Circuits Using Integer Linear Programming Warm Started by Graph Drawing

Josef Grus[1,2][a], Zdeněk Hanzálek[2][b], Dalibor Barri[3][c] and Patrik Vacula[3][d]

[1]*DCE, FEE, Czech Technical University in Prague, Czech Republic*

[2]*IID, CIIRC, Czech Technical University in Prague, Czech Republic*

[3]*STMicroelectronics, Prague, Czech Republic*

Keywords: Placement, Combinatorial Optimization, Analog Integrated Circuit, Rectangle Packing.

Abstract: Due to its diversity, the physical design of the Analog and Mixed-Signal Integrated Circuits is not as automated as the physical design of digital Integrated Circuits. The placement process is one of the critical steps of the physical design, and automating it would significantly shorten the design time. We formulate the placement process using an Integer Linear Programming approach, with features to support a specific semiconductor technology. We include an enumeration of possible variants of the circuit's topological structures, which are afterward considered during optimization. We use the Gurobi solver to minimize both the approximate wire length and the placement area. The results were evaluated by layout design experts and compared with manual designs. We also utilize a graph drawing-based method to generate an initial feasible solution to warm start the Integer Linear Programming solver, which noticeably improves the performance and shortens the computation time (5x to 15x), and makes the approach applicable even for larger problem instances containing 100 independent elements. Experiments performed on real-life industrial problem instances show that our graph drawing-enhanced approach can produce high-quality placement in a much shorter time than the designers need.

1 INTRODUCTION AND RELATED WORK

The physical design, or layout, is one of the most important steps in Integrated Circuit (IC) design. In the case of digital ICs, this process has already been successfully automated, placing thousands of elements in a short time. On the other hand, there is no such industry-accepted automation counterpart in the case of Analog and Mixed-Signal (AMS) ICs (Scheible and Lienig, 2015). With increasing demands and the need to shorten time-to-market for newer analog designs, automating the whole layout process is currently a highly discussed and important topic both in academia and in the industry (Scheible and Lienig, 2015). However, due to differences between various semiconductor technologies, some areas, like BCD (BIPOLAR-CMOS-DMOS) process, are less automated than others.

[a] https://orcid.org/0000-0002-1136-370X
[b] https://orcid.org/0000-0002-8135-1296
[c] https://orcid.org/0000-0002-8498-0135
[d] https://orcid.org/0000-0001-7288-7729

The placement process is a critical stage of the physical design. During this stage, the designer determines the positions and orientations of the circuit's devices, so there are no overlaps between devices that would render the entire placement infeasible, and other constraints are satisfied as well; this shows the link between the placement problem and combinatorial problems like rectangle packing. Our goal is to minimize both the final area of the ICs as well as the approximate wire length between devices. We use a point-to-point (P2P) metric based on the specific problem statement required by the industry partner STMicroelectronics to model wire length. The commonly used half perimeter wire length (HPWL) metric is also modeled and used for comparison with contemporary methods. Design constraints (minimum distance between devices, blockage areas, aspect ratio, topological structures, symmetries, and isolated pockets) are also considered to ensure that the resulting placement is valid.

Following our industry-specific problem statement, we only consider placement up to the so-called level 2 topology. Thus, as an input, we consider the circuit's netlist, consisting of level 0 topologies (sin-

Grus, J., Hanzálek, Z., Barri, D. and Vacula, P.
Automatic Placer for Analog Circuits Using Integer Linear Programming Warm Started by Graph Drawing.
DOI: 10.5220/0011789300003396
In *Proceedings of the 12th International Conference on Operations Research and Enterprise Systems (ICORES 2023)*, pages 106-116
ISBN: 978-989-758-627-9; ISSN: 2184-4372

gle devices, e.g., transistors or resistors) and level 1 topologies (sensitive topological structures, e.g., current mirrors, differential pairs). The industry experts provide the assignment of the circuit's devices to the specific topological structures in an additional file.

There are several research directions currently active in the field of analog placement automation. The first group of methods uses a topological representation of the placement - methods such as sequence pairs and B*-trees, often utilized in packing (Oliveira et al., 2016), encode relative positions between design elements. This means that each representation maps to a solution without infeasible overlaps between design elements. On the other hand, the more complex constraints, such as symmetries, are harder to encode, requiring intricate representation. The search over the solution space often uses simulated annealing or genetic algorithms. Topological representation was and still is widely used (Lourenco et al., 2006; Strasser et al., 2008; Ma et al., 2011; Dhar et al., 2021).

The other research direction considers absolute representation, where each element is described by its actual coordinates. While this approach makes it easy to describe the majority of the constraints, it can produce infeasible solutions due to the overlapping of design elements. These include work of (Chen et al., 2008; Xu et al., 2019a; Xu et al., 2019b; Lin et al., 2022), which initially determine the global placement using Nonlinear Programming (solved, e.g., by Nesterov's method (Nesterov, 1983)), and then produce feasible placement using Linear Programming (LP) or other methods. However, simulated annealing was also used with the absolute representation (Cohn et al., 1991; Martins et al., 2015).

Other approaches include using Integer Linear Programming (ILP). The recent hierarchical ILP approach (Xu et al., 2017) also considers multiple different configurations at each level of the optimization. Machine learning approaches for the placement problem are also investigated nowadays (Mirhoseini et al., 2021; Mallappa et al., 2022). Other methods, like the Forced-directed approach (Spindler et al., 2008), were also used for the placement problem.

Our proposed ILP solution derives its core ideas from approaches used for general rectangle packing problem (Korf et al., 2010; Berger et al., 2009). Algorithms for strip packing (Alvarez-Valdes et al., 2008) or other floorplanning problems could also be utilized. Due to its similarity with the proposed placement problem, methods for the Facility Layout Problem (FLP) also need to be mentioned. Methods for solving FLP (Kubalík et al., 2019; Xie and Sahinidis, 2008; Kanduč and Rodič, 2015) consider the minimization of travel distance between machines in the facility, which is very similar to the minimization of the connectivity between devices of the placement.

This paper has the following contributions:

1. ILP formulation of the placement problem. Our model considers all possible variants (pairs of width and height) of topological structures enumerated by the packing algorithm. The model offers features for the successful placement of BCD technology ICs, which was required by the industry partner due to its lack of automation.
2. Method based on force-directed graph drawing (FDGD) for finding partial initial solution used as a warm start, which significantly improves the performance of the utilized Gurobi (Gurobi Optimization, LLC, 2021) solver. Thus, we do not require the hierarchical decomposition (Xu et al., 2017) to speed up the optimization of the discussed problem instances.
3. Comparison of our approach with open-source solution ALIGN (Dhar et al., 2021), and evaluation on real-life industrial instances, provided and validated by the industry experts.

2 PROBLEM STATEMENT

The problem description is generated from the provided netlist of the IC, which describes the circuit's devices and their interconnections, a list of the topological structures, and a constraint database. Both single devices and topological structures of the netlist are modeled as rectangles with multiple shape variants. In the rest of the text, we refer to both the devices and structures as rectangles. This modeling is an extension of the rectangle packing formulation (Berger et al., 2009), a well-known combinatorial problem. Let n be the number of rectangles to be placed. Each rectangle i is associated with its set of m_i pre-defined variants $R_i = \{(w_i^1, h_i^1), \dots, (w_i^{m_i}, h_i^{m_i})\}$, which includes rotated variants as well. Each pair of rectangles (i, j) is associated with their **minimum allowed distance** $a_{i,j}$. Figure 1 is an illustrative example that shows the placement with 18 different independent rectangles distinguished by color.

Connectivity: between devices, which approximates the final wire length, is also derived from the netlist. We translate the net representation of the device interconnectivity to pairs of devices in a P2P manner. Let $G = (V, E)$ be a hypergraph with a set of vertices $V = \{v_1, \dots, v_n\}$ associated with rectangles, and a set of hyperedges $E = \{e_1, \dots, e_m\}$ associated with nets. Note that each hyperedge e corresponds to a list of its connected rectangles. Then we define connectivity

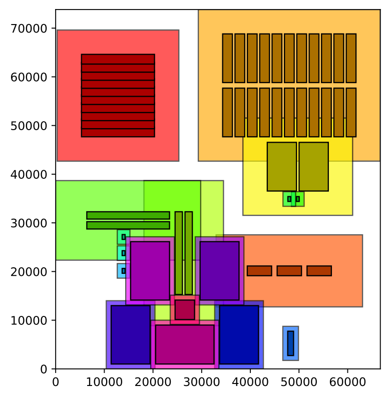


Figure 1: Example placement for constraint visualization.

pair weight between rectangles i, j by inspecting each net e:

$$c_{i,j} = \sum_{\forall e \in E} c^e_{i,j}, \tag{1}$$

where $c^e_{i,j}$ is 0 when rectangle i or j is not present in net e, and a positive integer otherwise. When we consider the topological structure, consisting of several devices prescribed by netlist, the corresponding weight accumulates partial weights over its internal devices. Overall P2P connectivity cost is defined, using the appropriate distance function d, as:

$$\mathcal{L}_C = \sum_{\forall i,j} c_{i,j} \cdot d(i,j). \tag{2}$$

When the HPWL metric is considered, we form the overall connectivity cost as follows:

$$\mathcal{L}_C = \sum_{\forall e \in E} c_e \cdot (\max_{i \in e} x^c_i - \min_{i \in e} x^c_i + \max_{i \in e} y^c_i - \min_{i \in e} y^c_i), \tag{3}$$

where c_e is the overall weight of the net e and x^c_i, y^c_i are coordinates of the center of the rectangle i.

The **aspect ratio** of the design can be constrained as well. Let W, H be the width and height of the produced layout. We define aspect ratio constraint using a pair of aspect ratio bounds l_R, u_R, such that:

$$0 \leq l_R \leq AR \leq u_R \leq 1, \tag{4}$$

where $AR = \frac{\min\{W,H\}}{\max\{W,H\}}$ is the aspect ratio of the design.

We can forbid rectangles to be placed within a specific subarea of the canvas; we refer to these subareas as **blockage areas**. This is shown in Figure 1, where a blockage area of size 10 000 x 20 000 was placed in the bottom left corner of the canvas.

Topological Structures: are easily modeled using multi-variant rectangles - they consist of a set of devices that need to be placed in a compact regular pattern, so the entire IC performs as intended. Given the list of internal devices of the structure, we need to generate all feasible variants of the structure for the final optimization. When a topological structure consists of devices with uniform dimensions, all possible variants can be enumerated by calculating the number of required columns for a given number of rows so all internal devices can fit in a regular tabular pattern. An advanced approach is needed when the structure's devices only have the same width, and efficient packing is required (see Figure 2). In this paper, the structures contain between 4 and 40 variants. Rectangles with multiple variants are also shown in Figure 1 (e.g., the orange rectangle's internal configuration uses two rows, internal devices shown as smaller darker rectangles enveloped by the lighter shell).

Furthermore, we model additional empty space, or **pocket**, around the placed structures and devices (shown as a lighter outer shell around packed devices in Figure 2). Pockets are critical for the successful design of BCD technology ICs. If two design elements do not share their BULK terminal net (which supplies power to the element), they need to be packed so that their pockets do not intersect and satisfy the minimum allowed distance. But when elements have the same BULK net, their pockets can be merged as long as the rules concerning the proximity between their internal devices are satisfied. In Figure 1, red and orange rectangles are not compatible for pocket merging, while the pockets of orange and yellow rectangles were merged. The dimensions of the rectangles' variants are appropriately enlarged to model the pockets.

Lastly, some devices may be required to form the **symmetry group** with a common axis of symmetry. Such a group contains pairs of symmetrical devices, or self-symmetrical ones that need to be placed directly on the axis. This was especially needed for comparison with the state-of-the-art tool ALIGN (Dhar et al., 2021). In Figure 1, there is a symmetry group with the vertical axis of symmetry, containing two symmetry pairs and two self-symmetric rectangles, shown in the bottom part of the figure.

3 ILP MODELING

3.1 Core of the Model

The core of the proposed ILP model, extended from (Berger et al., 2009), is shown in equations (5) - (16). Let $I = \{1, \ldots, n\}$ be set of rectangles' indices. Each rectangle is represented by four continuous variables; coordinates of its bottom-left corner (x_i, y_i) and width and height (w_i, h_i), which has to correspond to one of the pre-defined variants $(w^k_i, h^k_i) \in R_i$. The choice of exactly one variant is made using binary variables s^k_i for each rectangle i and variant k, as shown in equations (6), (7). Placement's width W and height H are variables constrained to be the upper bounds for the most distant part of any rectangle from origin $(0,0)$.

Non-overlapping of the devices is ensured by binary variables $r^k_{i,j}$ and inequalities (8) - (12). At least one of the inequalities, which corresponds to the relationship (left/right/over/under) between rectangles, must be valid ($r^K_{i,j} = 1$) without the big-M element (Camm et al., 1990). Parameter $a_{i,j}$ defines the minimum allowed distance between rectangles. By setting the parameter $a_{i,j}$ to the negative value, the solver can place associated rectangles with their pockets merged, similarly to device layer-aware placements (Xu et al., 2019a).

$$x_i + w_i \leq W, \;\; y_i + h_i \leq H \qquad \forall i \in I \tag{5}$$

$$\sum_{k=1}^{m_i} s^k_i = 1 \qquad \forall i \in I \tag{6}$$

$$w_i = \sum_{k=1}^{m_i} w^k_i \cdot s^k_i, \;\; h_i = \sum_{k=1}^{m_i} h^k_i \cdot s^k_i \qquad \forall i \in I \tag{7}$$

$$\sum_{k=1}^{4} r^k_{i,j} \geq 1 \qquad \forall i,j \in I : \; i < j \tag{8}$$

$$x_i + w_i + a_{i,j} \leq x_j + M(1 - r^1_{i,j}) \qquad \forall i,j \in I : \; i < j \tag{9}$$

$$y_i + h_i + a_{i,j} \leq y_j + M(1 - r^2_{i,j}) \qquad \forall i,j \in I : \; i < j \tag{10}$$

$$x_j + w_j + a_{i,j} \leq x_i + M(1 - r^3_{i,j}) \qquad \forall i,j \in I : \; i < j \tag{11}$$

$$y_j + h_j + a_{i,j} \leq y_i + M(1 - r^4_{i,j}) \qquad \forall i,j \in I : \; i < j \tag{12}$$

$$x_i, y_i, w_i, h_i \geq 0 \qquad \forall i \in I \tag{13}$$

$$W, \; H \geq 0 \tag{14}$$

$$s^k_i \in \{0,1\} \qquad \forall i \in I \; \forall k \leq m_i \tag{15}$$

$$r^k_{i,j} \in \{0,1\} \qquad \forall i,j \in I : \; i < j \quad \forall k \in \{1,2,3,4\} \tag{16}$$

A good estimate of M has a significant effect on the efficiency of the computation. We set M to a value that would satisfy even the most extreme placement - when all rectangles, using their widest variant, would be placed next to each other in a single row with a gap equal to the largest minimum allowed distance. However, if we would impose upper bounds on W and H (e.g., based on the user's input), we could easily scale down the value of M as well.

3.2 Blockage Areas

Blockage areas enable us to restrict specific rectangles from a subsection of the canvas. This requirement is handled by introducing the dummy rectangles. Since each blockage area is defined with fixed bottom-left corner coordinates x_b, y_b and dimensions w_b, h_b, relative position constraints need to be added for each blockage area. Let b be the label of the blockage area, and S_b be a set of indices of rectangles restricted by blockage area b.

Model is extended by $4 \cdot |S_b|$ binary variables $r^k_{b,j}$ for each blockage area b, as is shown in equations (17) - (22). In case when the reference point of the blockage area corresponds to the origin (0,0) or when it lies on one of the axes, some variables and constraints can be omitted to simplify the model.

$$\sum_{k=1}^{4} r^k_{b,j} \geq 1 \qquad \forall j \in S_b \tag{17}$$

$$x_b + w_b \leq x_j + M(1 - r^1_{b,j}) \qquad \forall j \in S_b \tag{18}$$

$$y_b + h_b \leq y_j + M(1 - r^2_{b,j}) \qquad \forall j \in S_b \tag{19}$$

$$x_j + w_j \leq x_b + M(1 - r^3_{b,j}) \qquad \forall j \in S_b \tag{20}$$

$$y_j + h_j \leq y_b + M(1 - r^4_{b,j}) \qquad \forall j \in S_b \tag{21}$$

$$r^k_{b,j} \in \{0,1\} \qquad \forall j \in S_b \; \forall k \leq 4 \tag{22}$$

3.3 Aspect Ratio

In order to comply with the aspect ratio requirements from Section 2 given the parameters of equation (4), additional constraints are required. Binary variable r_R is needed since this formulation of ratio constraint induces non-convex variable space - when $r_R = 0$, we expect that the inequality $u_R \cdot H \geq W$ holds.

$$l_R \cdot W \leq H \leq u_R \cdot W + M \cdot (1 - r_R) \tag{23}$$

$$l_R \cdot H \leq W \leq u_R \cdot H + M \cdot r_R \tag{24}$$

$$r_R \in \{0;1\} \tag{25}$$

The second approach uses soft constraint, propagated into the criterion function. Aspect ratio criterion element $\mathcal{L}_R$ is defined using the following pair of constraints:

$$\mathcal{L}_R \geq W - H \tag{26}$$

$$\mathcal{L}_R \geq H - W \tag{27}$$

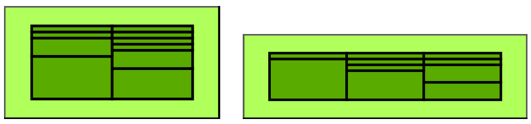


Figure 2: Two different variants of the same topological structure.

Therefore, the expression $\mathcal{L}_R$ is zero whenever both dimensions of the boundary box are equal, which is generally more appreciated by designers. Nevertheless, we omit this criterion element in the rest of the paper and experiments.

3.4 Topological Structures

As mentioned in Section 2, when topological structures consist of devices with uniform dimensions, all their possible variants can be easily enumerated. Starting from a configuration with a single row, the minimum number of columns needed to fit all member devices into the structure is calculated, and the actual size of the structure, including minimum internal distances, can be determined. Afterward, the number of rows is increased, and an additional variant is calculated until the number of rows exceeds the number of devices in the structure.

When devices only share a single dimension, a more advanced approach needs to be used to efficiently pack the internal devices together. A solution inspired by the Longest processing time list scheduling (LPT) algorithm (Della Croce and Scatamacchia, 2018) can be utilized. We substitute processing time with our devices' non-shared dimension. Even though the scheduling algorithm is only approximate, high-quality packing for a given number of rows, which corresponds to the number of parallel machines in the scheduling case, can be obtained. All possible variants are again generated by iterating over all possible numbers of rows. Examples of two variants (2 and 3 rows, rotated) produced by LPT are shown in Figure 2.

3.5 Device Connectivity

Due to the use of the ILP, which requires both the constraints and criterion function to be linear, Euclidean (L_2) norm between rectangles cannot be used as an appropriate distance in the P2P connectivity metric. Instead, two alternative approaches for calculating the distance between pairs of rectangles are shown. Let $d^x_{i,j}, d^y_{i,j}$ be the distance between a pair of rectangles i, j in x and y dimensions, respectively. We can represent the actual distance between a pair of rectangles either by the Manhattan (L_1) norm, defined as:

$$L_1(i,j) = d^x_{i,j} + d^y_{i,j}, \tag{28}$$

or Quasi-Euclidean (L_*) norm (Devgan et al., 2019), which more closely matches L_2 norm:

$$L_*(i,j) = \max\{d^x_{i,j}, d^y_{i,j}\} + (\sqrt{2}-1)\cdot \min\{d^x_{i,j}, d^y_{i,j}\}. \tag{29}$$

Also, distance can be calculated between the two closest points of the pair of rectangles or between their centroids. To model these phenomena, the following inequalities are used for the x dimension. The equations are identical for the y dimension. We define x-offset $x^{\text{off}}_{i,j}$ as 0, if centroid distance is considered and as $(w_i + w_j)/2$ otherwise. Then inequalities:

$$d^x_{i,j} \geq x_i + \frac{w_i}{2} - x_j - \frac{w_j}{2} - x^{\text{off}}_{i,j}, \tag{30}$$

$$d^x_{i,j} \geq x_j + \frac{w_j}{2} - x_i - \frac{w_i}{2} - x^{\text{off}}_{i,j}, \tag{31}$$

$$d^x_{i,j} \geq 0, \tag{32}$$

define dimension elements of the distance between rectangle pair i, j. In order to combine these elements into the final accumulated criterion expression, let C be a set of pairs of rectangles corresponding to non-zero connectivity weight.

$$C = \{(i,j) \mid i < j,\ c_{i,j} > 0\} \tag{33}$$

Constant t is set to 1, if L_1 norm is used, and to $(\sqrt{2}-1)$ in case of L_*. The following inequalities and final equality are sufficient to define the total weighted P2P connectivity cost expression $\mathcal{L}^{\text{P2P}}_C$. Thanks to the minimization of the final criterion function, there is no need for additional binary variables.

$$d_{i,j} \geq d^x_{i,j} + t \cdot d^y_{i,j} \tag{34}$$

$$d_{i,j} \geq d^y_{i,j} + t \cdot d^x_{i,j} \tag{35}$$

$$\mathcal{L}^{\text{P2P}}_C = \sum_{\forall (i,j) \in C} c_{i,j} \cdot d_{i,j} \tag{36}$$

We formulate the overall HPWL connectivity using the same distance elements $d^x_{i,j}$, $d^y_{i,j}$, but instead of summation, the maximum distance between pair of rectangles in the given net would be determined using continuous variables d^x_e, d^y_e for each net e. Then, $\mathcal{L}^{\text{HPWL}}_C$ is defined as a weighted sum of each net's cost element.

$$d^x_e \geq d^x_{i,j} \qquad \forall i,j \in e \tag{37}$$

$$d^y_e \geq d^y_{i,j} \qquad \forall i,j \in e \tag{38}$$

$$\mathcal{L}^{\text{HPWL}}_C = \sum_{\forall e \in E} c_e \cdot (d^x_e + d^y_e). \tag{39}$$

3.6 Symmetry Groups

To successfully model the symmetry groups, we require the additional continuous variable to represent the axis of symmetry. Assume that Sym^y is the symmetry group with the vertical axis of symmetry, whose position is determined by the continuous variable x_{sym}. The symmetry group consists of indices of self-symmetric rectangles $(s_0, -)$ and pairs of indices associated with symmetric pairs (s_1, s_2). Then the following equations constrain the symmetry group's rectangles to share the same axis of symmetry:

$$w_{s_1} = w_{s_2} \qquad \forall (s_1, s_2) \in \mathrm{Sym}^y \tag{40}$$
$$h_{s_1} = h_{s_2} \qquad \forall (s_1, s_2) \in \mathrm{Sym}^y \tag{41}$$
$$y_{s_1} = y_{s_2} \qquad \forall (s_1, s_2) \in \mathrm{Sym}^y \tag{42}$$
$$x_{s_1} + x_{s_2} + w_{s_1} = 2 \cdot x_{\mathrm{sym}} \qquad \forall (s_1, s_2) \in \mathrm{Sym}^y \tag{43}$$
$$2 \cdot x_{s_0} + w_{s_0} = 2 \cdot x_{\mathrm{sym}} \qquad \forall (s_0, -) \in \mathrm{Sym}^y \tag{44}$$

Constraints for a group with the horizontal axis of symmetry would be constructed analogously.

3.7 Criterion

In order to minimize the area of the placement, which is a nonlinear expression $W \cdot H$, we approximate it using the half perimeter of the resulting placement:

$$\mathcal{L}_A = W + H. \tag{45}$$

We expect that thanks to the correlation between half perimeter and the area of the bounding rectangle, a solution minimizing $\mathcal{L}_A$ will have a small area as well. This correlation assumption can also be improved by using suitable aspect ratio constraints, forcing the solver to produce square-like designs.

Ultimately, the final criterion function is defined as:

$$\mathcal{L} = c_A \cdot \mathcal{L}_A + \frac{c_C}{S} \cdot \mathcal{L}_C, \tag{46}$$

where S is normalization constant and c_A, c_C are weights; by tuning them, we can achieve a suitable trade-off between both $\mathcal{L}_A$ and $\mathcal{L}_C$. However, since there are only two criterion elements, we fix $c_A = 1$ and tune only the connectivity weight. We use the normalization constant S to make the weight c_C less sensitive to a number of nets of the instance, which may vary significantly; therefore, we can re-use the same value of c_C and expect a similar outcome in the sense of connectivity importance. In the case of P2P connectivity, we define the normalization constant using each pair of rectangles as:

$$S^{\mathrm{P2P}} = \sum_{(i,j) \in C} c_{i,j}. \tag{47}$$

In the case of HPWL connectivity, we only combine each net's weights together:

$$S^{\mathrm{HPWL}} = \sum_{e \in E} c_e. \tag{48}$$

4 FORCE-DIRECTED GRAPH DRAWING

The computational complexity of the model from Section 3 is directly connected to the use of binary variables, specifically relative position variables $r^k_{i,j}$ and rectangle variant variables s^k_i. The number of $r^k_{i,j}$ variables grows quadratically with respect to a number of independent rectangles in the instance; even the state-of-the-art ILP solvers cannot keep up with such an increase in the number of decision variables. However, when a subset of these binary variables is set to the specific values, leading to a potentially high-quality initial solution, it will allow the solver to prune parts of the search space more efficiently.

One way to obtain the partial assignment of values to relative position variables is by using algorithms originally dedicated to graph drawing, specifically, FDGD algorithms (Fruchterman and Reingold, 1991). The so-called spring embedding algorithms distribute the graph vertices so that highly connected vertices are close to each other while minimizing overlaps. In (Kanduč and Rodič, 2015), the author uses the FDGD for the factory floor layout problem. This problem concerns the placement of machines on the factory floor to minimize the travel distances between the machines and the total area as well. Thanks to the similarity with our placement problem, we utilized the proposed solution. Note that other force-directed approaches were proposed to solve placement in the past (Spindler et al., 2008).

Only the best aspect ratio-wise variant is selected per rectangle. With a probability of 0.5, each rectangle is introduced in a rotated variant. This selection of variants is fixed before the main phase of the algorithm. Then, rectangles are randomly distributed on the canvas, defined by their centroids $\mathbf{p}_i$. The box, inside which the rectangles are distributed and outside of which the boundary forces apply, was defined as a square with an area 125 % larger than the total area of rectangles and blockage areas.

The paper's (Kanduč and Rodič, 2015) main contributions used in our heuristic are definitions of attractive and repulsive forces. $G_{i,j}$ refers to an attractive force element applied to the rectangle i due to the rectangle j. Similarly, $F_{i,j}$ refers to the repulsive force element. Boundary repulsive force B_i pushes the rectangles back into the initial bounding box. Lastly, to

```
initialize centroids of rectangles
fix symmetry groups
$i \leftarrow 0$
while $i <$ iterations do
  for all $c \in$ rectangle indices do
    calculate $O_c$ and $B_c$
    set $F_c$ and $G_c$ to $\mathbf{0}$
    for all $j \in$ rectangles $\cup$ blockage areas do
      $F_c \leftarrow F_c + F_{c,j}$
    end for
    for all $j \in$ rectangles connected to $i$ do
      $G_c \leftarrow G_c + G_{c,j}$
    end for
    $Q_c \leftarrow f \cdot F_c + g \cdot G_c + b \cdot B_c + o \cdot O_c$
    $\mathbf{p}_c \leftarrow \mathbf{p}_c + \delta \cdot Q_c$
  end for
  $i \leftarrow i + 1$
end while
return rectangles' positions
```

Algorithm 1: FDGD algorithm for warm start heuristic.

explicitly consider our goal of minimizing the area, the origin force O_i that attracts each rectangle to the origin of the coordinate system is introduced.

To accommodate the existence of the symmetry groups, we employ the level-based placement heuristics (Coffman et al., 1980) to find suitable packing of the group. We only pack one rectangle from each symmetry pair (the position of the other is determined by its partner), and we ensure that the self-symmetric rectangles' centers lie on the axis of symmetry. The symmetry group is considered a single structure within the algorithm, the forces affecting the group's members are aggregated, and the group is afterward moved as a single entity.

The pseudo-code of the used algorithm is shown in Algorithm 1. All mentioned forces are calculated for each rectangle (or symmetry group), and the rectangle's position is asynchronously updated. Hyper-parameters of the algorithm are coefficients of the forces, f, g, b, o, that control the relative effect of each applied force. Parameter δ describes the simulation step; a too-large step will lead to numerical instability due to the large changes of positions, while an extremely small step would require too many iterations to reach the local minimum. How the algorithm redistributes the rectangles is shown in Figure 3.

Several runs can be performed to avoid the dependence on good initial position distribution. After the run of the Algorithm 1, relative positions for the ILP model are extracted. For each pair of rectangles, relative position constraints of Section 3 with proximity bounds from Section 2 are evaluated. Similarly to the placement legalization (Lin et al., 2022), the relation

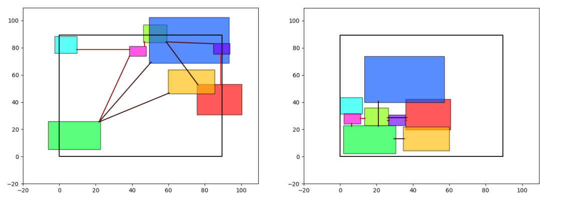


Figure 3: Initial and final distribution of rectangles by FDGD method. Attractive forces between rectangles are highlighted.

whose constraint is least violated is selected, and the corresponding relative position variable $r_{i,j}^k$ is set to 1. We ensure that this assignment of relative position variables creates no cycles. Together with the fixed variant variables s_i^k, these assignments are used as a partial initial solution to warm start the solver.

5 EVALUATION

5.1 Methodology

We utilized the Gurobi ILP solver v9.1.2, using four threads in each experiment. The project was implemented in Python 3.7 and C. Experiments were performed on an Intel Xeon Silver 4110 2.10 GHz.

5.2 Effect of the FDGD

Eighty synthetic instances containing either 20, 30, 50, or 100 independent rectangles (both single and multiple variant ones) were randomly generated, with the character of instances based on real-life designs. Twenty instances were generated for each problem size. The subset of instances contained blockage areas, or their aspect ratio was restricted to test the ability of the solver to handle these constraints. Instance set S_1 contained instances with 20 and 30 rectangles, and harder sets S_2 and S_3 contained instances with 50 and 100 independent rectangles, respectively. Both the stand-alone ILP model and ILP model warm started by FDGD heuristics (FDGD ILP) were evaluated, with the solver running for up to 600 seconds. The closest point L_* metric was used to model P2P connectivity. We used values $c_C \in \{0.5, 25\}$ to study how two widely different scenarios with respect to connectivity importance affect the computation.

As shown in Figure 4, the main advantage of the proposed warm start heuristic is that the solver finds a high-quality feasible solution almost immediately. We can observe this phenomenon in Table 1. FDGD ILP produced better results (lower criterion value) in the majority of the 60 studied cases of sets S_1, S_2

Table 1: Average relative percent gap of method's criterion (at given computation time) and best-known result. The last line shows a number of instances for which the method achieved a better result than the other one, for sets S_1, S_2.

S_1	$c_C = 0.5$		$c_C = 25$	
t $[s]$	ILP	FDGD ILP	ILP	FDGD ILP
30	13.36	3.31	45.58	8.50
120	7.23	1.65	21.86	4.51
600	2.75	0.40	7.97	1.28
n = 40	10	30	9	31

S_2	$c_C = 0.5$		$c_C = 25$	
t $[s]$	ILP	FDGD ILP	ILP	FDGD ILP
30	81.00	8.02	163.21	9.94
120	35.26	2.84	84.40	3.89
600	12.06	0.00	44.00	0.00
n = 20	0	20	0	20

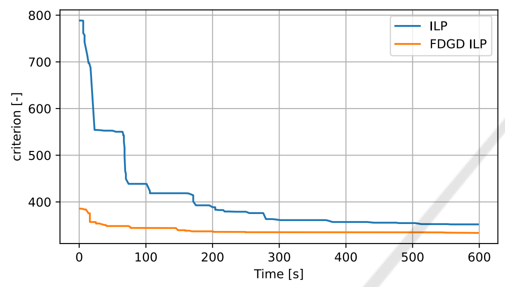


Figure 4: Example of the changes in criterion value during computation time.

for both studied weights. The performance gap between FDGD ILP and stand-alone ILP models increases with larger values of c_C and growing instance size, which makes the FDGD ILP more suitable for connectivity-oriented and larger problems. We also found that stand-alone ILP, on average, needs between 5x to 15x more computation time to find the same-quality solution that FDGD ILP finds after 30 seconds, depending on an instance size and value of c_C. A further experiment, which included problem instances from the set S_3 with 100 independent rectangles, showed that the stand-alone ILP fails to recover any feasible solution within its 600 s time limit, while the FDGD ILP's warm start solution can be extended to the complete solution almost immediately; making comparison for set S_3 unnecessary. Thus, in the rest of the paper, we only consider FDGD ILP.

5.3 Performance Comparison

To provide an explicit comparison with the state-of-the-art tools, we used the placer of the open-source framework ALIGN (Dhar et al., 2021). ALIGN's annotation tool extracted information about grouped elements and symmetries in the design. Test instances were two Operational Transconductance Amplifiers

Table 2: Comparison of proposed FDGD ILP method with ALIGN placer (Dhar et al., 2021).

	ALIGN placer		FDGD ILP		
instance	area [μm^2]	HPWL [μm]	area [μm^2]	HPWL [μm]	time [s]
CC-OTA	73.2	132.2	58.3	141.4	6.0
T-OTA	16.9	28.7	18.6	28.5	0.3
DTS-A	52.8	69.4	44.7	90.0	0.5
SCF	1995.6	478.4	1963.4	485.7	13.8
LE	58.2	47.0	56.5	56.2	6.7

(OTA), a Double Tail Sense Amplifier (DTS-A), a Switched Capacitor Filter (SCF), and a Linear Equalizer (LE). A FinFET 14nm Process design kit (PDK), part of ALIGN's repository, was also utilized. Results containing the area and HPWL of the final design both for ALIGN and our approach FDGD ILP, together with our solution's computation time, are shown in Table 2, with a value of c_C tuned for each instance.

Even though our solution was tuned to a slightly different problem formulation (including the definition of HPWL with respect to topological structures), and we needed to manually sanitize the data due to the differences in input data description, making the comparison mainly illustrative, our solver found solutions whose evaluation metrics were comparable with ALIGN's. This further demonstrates the extensibility and performance of the FDGD ILP. The computation time of our ILP solver was limited to approximately match the computation time of methods presented in a recent paper (Lin et al., 2022), which were evaluated on similar instances, to document the relatively short time our method needed to produce a comparable design.

5.4 Manual Design Comparison

The industry partner STMicroelectronics provided 17 real-life industrial designs. The BCD technology was applied, and thus our solution's capabilities were necessary. Provided designs contained both the input data (netlist, constraints, and structure list) as well as the data describing the positions of the devices in the manual placement created by the experts. Provided instances contained up to 60 independent rectangles. The manual and automated placements could be compared with their respective values of $\mathcal{L}_C$ and $\mathcal{L}_A$, and the placement area. Several runs of the FDGD ILP were performed, each running for 8 minutes, with $c_C \in \{0.1, 15, 50\}$. We have chosen these values to find both the placements prioritizing area and connectivity; however, using even larger values eventually led to placements that would not be applicable in the industry setting (unsuitable aspect ratio, excessive empty space). Closest-point L_* P2P connectivity metric was used.

Table 3 contains calculated metrics for each problem instance, with an average ratio of the automated

Table 3: Values of half perimeter $\mathcal{L}_A$ in μm, area in μm^2 and weighted P2P connectivity $\mathcal{L}_C$ in μm and an average ratio of placer-produced and manual designs' metrics aR for real-life instances.

	manual			placer-produced								
				$c_C = 0.1$			$c_C = 15$			$c_C = 50$		
instance	$\mathcal{L}_A$	area	$\mathcal{L}_C$	$\mathcal{L}_A$	area	$\mathcal{L}_C$	$\mathcal{L}_A$	area	$\mathcal{L}_C$	$\mathcal{L}_A$	area	$\mathcal{L}_C$
1	158	6118	8572	160	6420	4781	163	6583	4585	172	7347	4595
2	116	2710	1829	88	1911	1193	92	2091	968	96	2247	942
3	106	2650	1336	86	1800	1251	88	1898	448	94	2157	388
4	129	4096	555	112	3107	423	115	3289	285	116	3360	283
5	207	8972	36492	159	6305	13196	170	7013	6078	173	7375	5971
6	178	7698	12756	169	7148	9739	186	8172	6017	197	9013	5852
7	168	6580	14646	162	6558	13677	170	7104	9200	169	7105	8766
8	173	7294	2512	160	6435	2052	169	7082	1740	178	7896	1496
9	243	14129	49076	229	13054	22544	269	18056	17849	254	16151	16543
10	205	10214	41730	192	9192	41542	198	9835	25555	203	10325	24107
11	225	9922	4571	197	9356	1629	200	10000	481	200	10000	481
12	155	5953	5277	133	4440	3105	139	4811	2425	142	5025	2379
13	162	6511	6265	151	5676	5512	157	6161	5055	165	6749	5058
14	247	15235	7676	188	8831	6537	209	10905	2940	209	10872	2625
15	123	3758	1772	115	3286	3698	121	3666	2171	133	3936	2150
16	232	12397	14687	218	11817	7687	229	13037	7524	238	13976	7320
17	247	12525	42532	237	13645	30436	241	14369	26509	262	16474	27778
aR	1.00	1.00	1.00	0.89	0.86	0.78	0.94	0.95	0.53	0.97	1.01	0.51

and manual placement's metrics:

$$aR = \frac{1}{17} \cdot \sum_{i=1}^{17} \frac{\text{metric}^{(i)}_{\text{method}}}{\text{metric}^{(i)}_{\text{manual}}}, \qquad (49)$$

shown in the last row. On average, we may see that all three placer settings outperformed the manual results in both optimized metrics $\mathcal{L}_A$ and $\mathcal{L}_C$. Even though area $W \cdot H$ is not part of the criterion function, we were still able to find area-wise favorable solutions in the majority of the cases. Ultimately, for 14 out of 17 instances, the placer was able to find a solution outperforming manual design in all metrics discussed in Table 3. Figure 5 demonstrates the differences between placer-produced solutions, depending on the value of c_C, and includes the distribution of the devices within topological structures and allowed pocket merging. Notice the use of differing rectangle variants in the two designs.

Even though the solver optimized each problem for up to 480 seconds, we could easily limit the computation time. If the computation was aborted after 60 or 30 seconds, respectively, the found solution would be worse, on average, by 6 %, or 7 % than the final result. This relatively small gap is made possible due to the use of FDGD warm start (see Figure 4).

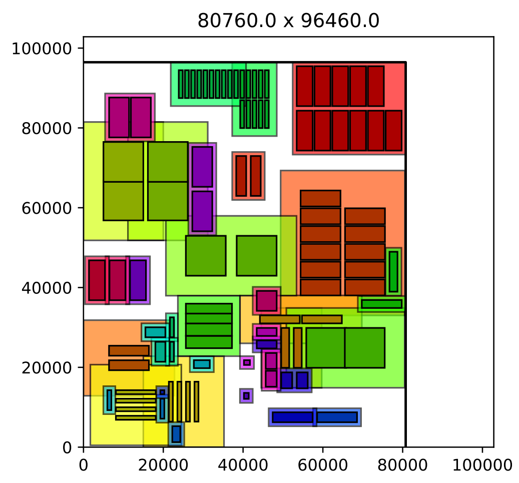
Industry experts validated produced results and provided qualitative feedback. Experts positively commend the relatively short time needed to obtain a high-quality solution; therefore, a solver can be called several times with different criteria to obtain a portfolio of solutions, from which experts can easily select, while hours of work are needed to create a single placement manually. However, experts pointed out several suspiciously low values of the P2P connectivity metric of the produced solutions in comparison with manual designs in Table 3. Thus, we concluded that the chosen P2P metric does not always capture all the aspects of the connectivity, and we should focus on HPWL or minimum spanning tree-based metrics in the future.

Table 4: Average ratio of placer-produced and manual designs' area and HPWL for real-life instances, depending on the type of connectivity metric optimized by the solver.

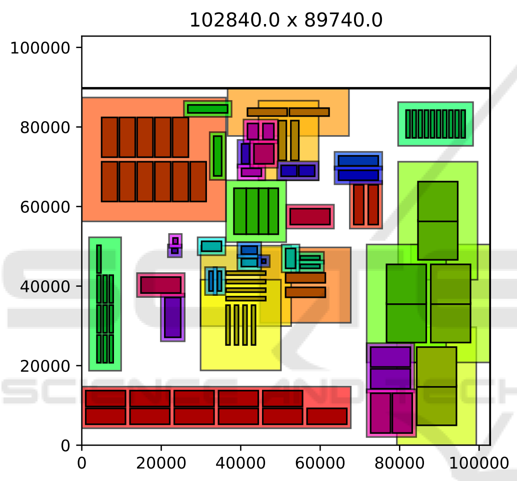
	P2P optimizing			HPWL optimizing	
c_C	0.1	15	50	0.1	5
$\mathcal{L}_A$ *aR*	0.89	0.94	0.97	0.88	0.96
area *aR*	0.86	0.95	1.01	0.83	0.98
HPWL *aR*	1.08	0.98	0.98	0.92	0.77

To validate the previous statement, we calculated the value of HPWL connectivity for each problem instance and each value of c_C from Table 3. We also performed a brief experiment with ILP solver optimizing the HPWL connectivity directly (choosing connectivity weights suitable for this scenario), and we report the average ratio of placer-produced and manual design's $\mathcal{L}_A$, area and HPWL metrics in Table 4.

We can see that in the case of the solver optimizing the P2P connectivity, the ratio for HPWL is much worse than its P2P counterpart reported in Table 3. However, we could still find HPWL-wise competitive designs since the solver found an overall better solution for 8 instances, making P2P formulation applicable even in this indirect scenario. Furthermore, when



(a) $c_C = 0.1,$ area $= 7790\ \mu\text{m}^2$.



(b) $c_C = 50,$ area $= 9229\ \mu\text{m}^2$.

Figure 5: Examples of FDGD ILP-produced designs with different values of c_C.

the solver optimized the HPWL connectivity directly, we were again able to find the solution of overall better quality (dominating manual designs in 12 cases), with an average ratio of HPWL metric equal to 0.92 and 0.77, respectively.

6 CONCLUSION

In this paper, we present the ILP formulation of the placement process of the physical design of AMS ICs. We successfully formalized the required constraints in cooperation with our industry partner STMicroelectronics to support the BCD technology. We also provided an FDGD-based warm start heuristic, which significantly improved the performance of the ILP solver. Ultimately, we evaluated our solution on both synthetically generated and real-life industrial problem instances, and we compared our solution with the open-source ALIGN framework. Both quantitative results and experts' feedback regarding the industrial problem instances showed that our proposed solution would be beneficial for solving the placement problem formulated by the industry partner.

Even though our proposed warm start heuristic significantly improves the performance of the ILP solver, the problem of scalability persists; a number of decision variables grow quadratically with an increasing number of independent rectangles to be placed. Therefore, we currently focus on developing a constructive heuristic combined with a genetic algorithm to tackle the placement problem outlined in this paper, which would not require a state-of-the-art commercial solver for competitive results. We believe that this approach, based on methods developed for strip packing and facility layout problems, could offer competitive results with the ability to scale.

ACKNOWLEDGMENTS

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS22/167/OHK3/3T/13.

This work was partially supported by the EU ICT-48 2020 project TAILOR no. 952215.

REFERENCES

Alvarez-Valdes, R., Parreño, F., and Tamarit, J. (2008). Reactive GRASP for the strip-packing problem. *Computers & Operations Research*, 35(4):1065–1083.

Berger, M., Schröder, M., and Küfer, K.-H. (2009). A constraint-based approach for the two-dimensional rectangular packing problem with orthogonal orientations. In *Operations Research Proceedings 2008*, pages 427–432. Springer Berlin Heidelberg.

Camm, J. D., Raturi, A. S., and Tsubakitani, S. (1990). Cutting Big M down to Size. *Interfaces*, 20(5):61–66.

Chen, T.-C., Jiang, Z.-W., et al. (2008). NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(7):1228–1240.

Coffman, Jr., E. G., Garey, M. R., et al. (1980). Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826.

Cohn, J., Garrod, D., et al. (1991). KOAN/ANAGRAM II: new tools for device-level analog placement and routing. *IEEE Journal of Solid-State Circuits*, 26(3):330–342.

Della Croce, F. and Scatamacchia, R. (2018). Longest processing time rule for identical parallel machines revisited. *Journal of Scheduling*, 23/2:163–176.

Devgan, V., Singh, V., et al. (2019). Using a novel image analysis metric to calculate similarity of input image and images generated by WAE. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 953–957.

Dhar, T., Kunal, K., et al. (2021). ALIGN: A system for automating analog layout. *IEEE Design and Test of Computers*, 38(2):8–18.

Fruchterman, T. M. J. and Reingold, E. M. (1991). Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164.

Gurobi Optimization, LLC (2021). Gurobi Optimizer Reference Manual. https://www.gurobi.com.

Kanduč, T. and Rodič, B. (2015). Optimisation of factory floor layout using force-directed graph drawing algorithm. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1087–1092.

Korf, R., Moffitt, M., and Pollack, M. (2010). Optimal rectangle packing. *Annals of Operations Research*, 179(1):261–295.

Kubalík, J., Kadera, P., et al. (2019). Plant layout optimization using evolutionary algorithms. In *Industrial Applications of Holonic and Multi-Agent Systems*, pages 173–188, Cham. Springer International Publishing.

Lin, Y., Li, Y., et al. (2022). Are analytical techniques worthwhile for analog IC placement? In *Proceedings of the 2022 Conference & Exhibition on Design, Automation & Test in Europe*, DATE '22, page 154–159. European Design and Automation Association.

Lourenco, N., Vianello, M., et al. (2006). LAYGEN - automatic layout generation of analog ICs from hierarchical template descriptions. In *2006 Ph.D. Research in Microelectronics and Electronics*, pages 213–216.

Ma, Q., Xiao, L., et al. (2011). Simultaneous handling of symmetry, common centroid, and general placement constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 30(1):85–95.

Mallappa, U., Pratty, S., and Brown, D. (2022). RLPlace: Deep RL guided heuristics for detailed placement optimization. In *Proceedings of the 2022 Conference & Exhibition on Design, Automation & Test in Europe*, DATE '22, page 120–123. European Design and Automation Association.

Martins, R., Lourenço, N., and Horta, N. (2015). Multi-objective optimization of analog integrated circuit placement hierarchy in absolute coordinates. *Expert Systems with Applications*, 42(23):9137–9151.

Mirhoseini, A., Goldie, A., et al. (2021). A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212.

Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547.

Oliveira, J. F., Júnior, A. N., et al. (2016). A survey on heuristics for the two-dimensional rectangular strip packing problem. *Pesquisa Operacional*, 36:197–226.

Scheible, J. and Lienig, J. (2015). Automation of analog IC layout: Challenges and solutions. In *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, ISPD '15, page 33–40.

Spindler, P., Schlichtmann, U., and Johannes, F. M. (2008). Kraftwerk2—A fast force-directed quadratic placement approach using an accurate net model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(8):1398–1411.

Strasser, M., Eick, M., et al. (2008). Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions. In *2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 306–313.

Xie, W. and Sahinidis, N. V. (2008). A branch-and-bound algorithm for the continuous facility layout problem. *Computers & Chemical Engineering*, 32(4):1016–1028.

Xu, B., Li, S., et al. (2017). Hierarchical and analytical placement techniques for high-performance analog circuits. In *Proceedings of the 2017 ACM on International Symposium on Physical Design*, ISPD '17, page 55–62.

Xu, B., Li, S., et al. (2019a). Device layer-aware analytical placement for analog circuits. In *Proceedings of the 2019 International Symposium on Physical Design*, ISPD '19, page 19–26.

Xu, B., Zhu, K., et al. (2019b). MAGICAL: Toward fully automated analog IC layout leveraging human and machine intelligence: Invited paper. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8.