



**CTU**

CZECH TECHNICAL  
UNIVERSITY  
IN PRAGUE

CZECH INSTITUTE OF INFORMATICS  
ROBOTICS AND CYBERNETICS

INDUSTRIAL INFORMATICS DEPARTMENT

# Matheuristic Local Search for the Placement of Analog Integrated Circuits


Josef Grus, Zdeněk Hanzálek

**DOI:** [https://doi.org/10.1007/978-3-031-49662-2\\_10](https://doi.org/10.1007/978-3-031-49662-2_10)

**Cite as:** J. Grus and Z. Hanzálek. Matheuristic local search for the placement of analog integrated circuits. In F. Liberatore, S. Wesolkowski, M. Demange, and G. H. Parlier, editors, *Operations Research and Enterprise Systems*, pages 178–200, Cham, 2024. Springer Nature Switzerland

© 2023. This manuscript version is made available under the CC-BY-NC-ND 4.0 license, see <http://creativecommons.org/licenses/by-nc-nd/4.0/>

# Matheuristic Local Search for the Placement of Analog Integrated Circuits

Josef Grus<sup>1,2</sup><sup>[0000-0002-1136-370X]</sup> and Zdeněk Hanzálek<sup>2</sup><sup>[0000-0002-8135-1296]</sup>

<sup>1</sup> DCE, FEE, Czech Technical University in Prague, Czech Republic  
`grusjose@fel.cvut.cz`

<sup>2</sup> IID, CIIRC, Czech Technical University in Prague, Czech Republic  
`zdenek.hanzalek@cvut.cz`

**Abstract.** The suboptimal physical design of the integrated circuits may not only increase the manufacturing costs due to the larger size of the chip but can also impact its performance by placing interconnected rectangular devices too far from each other. In the domain of Analog and Mixed-Signal Integrated Circuits (AMS ICs), placement automation is lacking behind its digital counterpart, mainly due to the variety of components and complex constraints the placement needs to satisfy. Integer Linear Programming (ILP) is a suitable approach to modeling the placement problem for AMS ICs. However, not even state-of-the-art solvers can create high-quality placements for large problem instances. In this paper, we study how to improve the results of our previous ILP model, first by introducing additional constraints and second by using matheuristics. Given the initial solution we obtain using our original ILP model, we use the solver to perform a local search. We try to improve the criterion by considering only a few spatially close rectangles while keeping the rest of the placement fixed. This local search approach enables us to significantly improve the quality of instances whose solution space we could not sufficiently explore before, even when the computation time reserved for the matheuristic is limited. Finally, we evaluate our revised approach on synthetically generated instances containing more than 200 independent rectangles and on real-life problems.

**Keywords:** Matheuristics · Placement Optimization · Analog Circuits.

## 1 Introduction

The importance of ICs for modern civilization is apparent. Advanced computing, Internet-of-Things devices, automotive, and consumer electronics rely on high-performance ICs. Such market pressure further motivates the companies to shorten the design time and lower the development costs to increase their profitability and strengthen their market position. One of the crucial steps in the design of the ICs is the physical design. During this step, the circuit diagram is converted into the geometrical representation of the final product - positions

and orientations of the rectangular devices (transistors, resistors, etc.) are determined during the placement phase, and the interconnections between them are planned during the routing phase. While these two steps are commonly solved one after another, the placement phase needs to consider the approximated interconnections to make the final product competitive and high-performant.

AMS components remain crucial nowadays, as operational amplifiers and analog-to-digital converters are required to convert signals from many sensors surrounding us. The placement phase for the digital ICs has already been successfully automated. Digital devices are in the form of standardized cells, each sharing the same height, and they are placed in rows rather than freely. These properties make the digital ICs' placement similar to the 1D bin packing problem and enable the automation tools to handle thousands of devices.

On the other hand, AMS ICs usually contain tens or hundreds of devices. However, the devices may appear in different sizes and aspect ratios and can be placed freely. They also have different voltage levels, which does not happen in a digital domain. Furthermore, the presence of noise and other negative effects inherent to the analog domain significantly influence the overall performance of the circuits. This is mitigated by additional constraints and rules the engineers must adhere to. Due to these complications, the placement of the AMS ICs has not been largely automated and still remains a time-consuming and error-prone manual process; its automation is pursued by projects funded both by DARPA [8] and EU [7]. It is further complicated by constraints specific to different technologies of the ICs. This paper specifically discusses BCD technology (technology combining analog, digital, and high-voltage components), which means the placer has to consider various minimum distances between devices and isolated pockets, among other features.

ILP offers a formalism to successfully model the placement problem of AMS ICs. Most constraints regarding the sizes of the devices and their mutual proximity or connectivity can be described using linear inequalities, while the non-linear criterion of the circuit's area might be approximated with its half-perimeter. Nevertheless, even the state-of-the-art ILP solvers, which improve every year, cannot sufficiently well explore the space of feasible placements of larger ICs.

In this paper, we build upon our previous work [13], where we used warm-started ILP to place devices of the AMS ICs. We discuss the effect of additional symmetry-breaking and redundant constraints on the model's performance. Finally, we develop a Matheuristic (MH) local search technique, which iteratively optimizes the initial solution obtained by solving the entire model, and which offers significant improvement, especially on the large synthetically generated instances with more than 200 devices to be placed. This paper is structured as follows. In Section 2, we mention the previous work done in domains of both placement and matheuristics. In Section 3, we formulate the placement problem for BCD technology. Section 4 describes our original ILP model, as well as additional redundant constraints we experimented with. Section 5 describes our MH approach. In Section 6, we describe the problem instances and present the experimental results, which show how well the MH approach performs. Also,

real-life instances are evaluated and compared with manual benchmarks. Finally, conclusions are drawn in Section 7.

## 2 Related Work

Even though the placement of the AMS ICs is not as automated as in the case of digital ICs, the problem has already been tackled in the past. Many methods use so-called topological representation - the solution is encoded using relative positions between the devices. Then, a packing procedure is used to convert the representation into the actual placement. Sequence pairs are one such representation. Proposed in [25], two permutations of the devices encode the relative positions between devices. Importantly, as was demonstrated in [21], this formulation can be extended to successfully model symmetry groups and other crucial features. Another example of the topological representation is B\*-trees, which use binary trees to determine the relative positions between the parent and child nodes. Used in [19,31], this representation offers a low level of redundancy in its search space.

Other methods consider the absolute coordinates of the devices. This makes encoding constraints such as symmetry groups easier; however, it also introduces infeasible solutions to search space. In the early work of [6], the simulated annealing was used to optimize the coordinates of the devices. The criterion contained both the area and wire length of the IC, as well as penalty terms for constraint violations. In [23], a similar approach, using a multi-objective constrained variant of simulated annealing, was also considered. Alternatively, methods described in papers [4,20] firstly use the global placement phase, where the approximate positions of the devices are determined using non-linear programming, and then the feasible placement without the overlaps is created using Linear Programming (LP). The mentioned core was extended to accommodate the different manufacturing layers of the ICs in [34]. In [17], the neural network was used to estimate the circuit's performance, and it was added to the differentiable criterion.

The force-directed approach was successfully applied to placement in [30], where the attractive and repulsive forces between the devices were derived from the connectivity of the IC and the devices' overlaps, respectively. Machine learning found its applications as well. An end-to-end pipeline of [24] was utilized as a placer of macros, while the learned model performed fine-optimization of the already-placed IC in [22].

While the methods outlined in the previous paragraphs successfully solved their associated placement problems, we cannot directly apply them to BCD technology ICs; these ICs rely on various minimum allowed distances between devices, isolated pockets, and other features that were rather omitted in the previous works. This was also a reason why we used the ILP, which allowed us to model these crucial features easily.

The ILP was applied to placement problems in the past. In [35], the authors used hierarchical decomposition to improve the solver's performance and created high-quality placements. In our previous work [13], we employed Force-Directed

Graph Drawing-based (FDGD) method to warm start the solver instead of relying on decomposition. Our proposed MH offers to improve the results produced by other methods even when the warm starting the solver or decomposing the problem is not sufficient or leads to low-quality solutions. Furthermore, ILP is often used to solve subproblems that arise within the problem of placement, such as the determination of the number of fingers of transistors [27].

The placement of AMS ICs much resembles other problems encountered within the domain of operations research. Rectangle packing can be viewed as a simplification of this paper’s topic due to the rectangular shape of the devices. Papers [2,16] used constraint programming to solve the rectangle packing problem. In [14], a genetic algorithm was used together with a Bottom-Left first packing heuristic. Later, the GRASP metaheuristic was applied to strip packing [1]. Even more closely related to our problem is Facility Layout Problem (FLP), where the task is to determine the positions of the facilities while minimizing the travel distances between them. This can be perceived as an analogy to the interconnectivity of the devices. ILP formulations of the FLP were investigated in [15,33]. The latter work optimized the paths between the departments simultaneously with the layout, which resembles the simultaneous optimization of placement and routing in the case of ICs.

MHs, heuristics based on mathematical programming, have been recently successfully applied to many combinatorial problems [12], especially with the ever-increasing performance of the black-box ILP solvers. While the solvers often cannot solve the industrial-size instances, their search capabilities when the model is smaller cannot be ignored. The MHs were used successfully in the domains of scheduling or routing, but the literature regarding their use for packing and cutting is rather sparse [28]. There are many ways how to build the heuristic around the ILP solver. The constructive MHs iteratively solve a series of simpler subproblems and construct the final solution by combining the intermediate results. This was used both for rostering problems [29], as well as for FLP [32]. In the latter, authors fix the relative positions between the already placed facilities and iteratively add the remaining ones until the layout is completed. Evolutionary MHs use mathematical programming to tackle the efficiently solvable subproblems encountered while using metaheuristics. In [26], parallel batch processing scheduling is tackled using a genetic algorithm, and LP is used to improve the solution by solving the minimum cost flow problem.

Finally, the MHs are often used to perform the local search. Given a starting solution to a problem, we try to improve it by solving the restricted variant of the original ILP model. There are several ways how to achieve such restriction. The first way, called local branching, limits how many variables can change its value. Assuming the ILP model only contains binary variables, then the following constraint can be introduced [12]:

$$\sum_{i \in B_0} x_i + \sum_{i \in B_1} (1 - x_i) \leq k \quad (1)$$

where variable  $x_i$  was originally assigned to 0, if  $i \in B_0$  and vice versa. The restrictiveness depends on the value of  $k$ . Local branching was successfully used

in the improvement phase of [29]. In [36], the flow-shop problem with time windows was tackled, and local branching was even used to construct the feasible solution from the initial infeasible one.

Another way to restrict the search space is to explicitly fix a subset of variables of the model. This application is very close to Large Neighborhood Search [11] or Ruin and Recreate heuristics [5]; the damaged solution (i.e., the free variables in the restricted ILP model) is repaired using the exact solver. Variable-fixing local search MHs were successfully applied to the scheduling domain, such as in the case of university timetabling [18], flow-shop scheduling [10], and evacuations scheduling [9]. In these works, the choice of free and fixed variables is crucial for the successful application of MHs and often depends on domain-specific information. In this paper, we decided to apply such variable-fixing MH to our placement problem.

### 3 Problem Formulation

During the placement phase of the physical design of the AMS ICs, the positions and orientations of the devices are determined. The input of the problem, the netlist, contains information about the sizes of the devices, their voltage level, and interconnectivity. The devices have a rectangular shape of fixed size and can be rotated. Furthermore, we need to consider topological structures. These are higher-level building blocks, such as differential pairs or current mirrors, and they consist of several devices that have to be placed in a regular pattern (see two columns of darker rectangles in Fig. 1). Thus, we enumerate all possible variants (with a varying number of rows and columns into which the devices are organized) of such topological structures beforehand, using algorithms based on list scheduling [13]. Afterward, we treat both the single devices and the topological structures as rectangles with multiple variants (in the case of single devices, the only alternative variant is rotation). Further in the text, we refer to both types of these building blocks as rectangles. Given a task to place  $n$  rectangles, we describe each one of them with the coordinates of its bottom-left corner  $(x_i, y_i)$  and its size  $(w_i, h_i)$ , which corresponds to one of its  $m_i$  variants.

Since we want to create as small a placement as possible, we would like to minimize its area  $W \cdot H$ . However, due to our use of ILP, we minimize the half perimeter of the placement's bounding box  $W + H$  instead.

The overall connectivity is modeled as Half Perimeter Wire Length (HPWL). The core concept of connectivity is a set of nets  $E$  - each net  $e \in E$  consists of a set of connected rectangles  $L_e$ . Each rectangle can be a member of multiple nets. The overall connectivity is formulated as follows:

$$\text{HPWL} = \sum_{\forall e \in E} c_e \cdot \left( \max_{i \in L_e} x_i^c - \min_{i \in L_e} x_i^c + \max_{i \in L_e} y_i^c - \min_{i \in L_e} y_i^c \right) \quad (2)$$

where the centroid coordinates are given by:

$$x_i^c = x_i + w_i/2 \quad (3)$$

$$y_i^c = y_i + h_i/2 \quad (4)$$

Multiplied by its cost  $c_e$ , each net contributes to the overall HPWL metric the half of the perimeter of the smallest bounding box that contains all of the net's rectangles' centroids [34]. Altogether, our task is to find a feasible placement that not only minimizes the area of its bounding box but minimizes the HPWL metric as well.

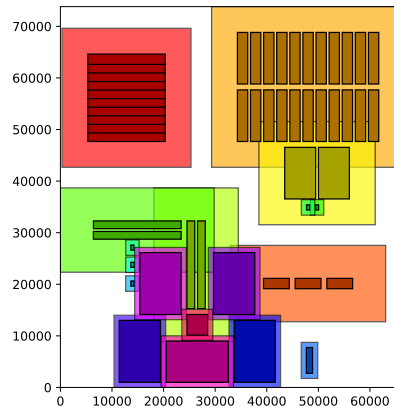


Fig. 1: Example placement with critical constraints of the BCD technology [13].

The physical devices (darker rectangles surrounded by lighter shells in Fig. 1), such as transistors, cannot overlap when they are manufactured in the same layer. Furthermore, an increased minimum distance can be imposed between some devices, e.g., to mitigate the effect of the noise on sensitive components. We also need to model additional empty space, or pocket, around the placed structures and devices (the lighter shells around packed devices in Fig. 1). Pockets are needed to isolate devices with different voltage levels, which is common for BCD technology. When the devices do not share their input voltage (BULK) net, and thus their voltage level may differ, we need to place them so their pockets do not overlap. Otherwise, their pockets can be merged as long as their internal devices do not overlap, as the yellow and orange rectangles in Fig. 1 demonstrate.

Additional constraints include the control of the aspect ratio of the final placement. Also, the engineer can restrict a subset of rectangles from a part of a canvas; we call this type of constraint a blockage area. An example is shown in the bottom-left corner of Fig. 1, which remained unoccupied due to the use of the blockage area. Furthermore, a group of rectangles may belong to a symmetry group, which shares a common axis of symmetry. An example is a group of darker rectangles with the vertical axis of symmetry located in the bottom part of Fig. 1.

## 4 ILP Model and Extensions

### 4.1 Baseline Model

We use our model proposed in [13], which was extended from rectangle packing formulation in [2]. Let  $\mathcal{I} = \{1, \dots, n\}$  be set of rectangles' indices. Four real variables represent each rectangle; 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  $m_i$  pre-defined variants  $(w_i^k, h_i^k)$ ,  $k \in \{1, \dots, m_i\}$ . Note that the sizes of rectangles' variants are increased to model the use of the pockets. The selection of variants is made using binary variables  $s_i^k$  for each rectangle  $i$  and variant  $k$ , as is shown in equations (6), (7).  $k$ -th variant is selected if  $s_i^k = 1$ . Placement's width  $W$  and height  $H$  are variables constrained by the positions of the placed rectangles.

$$x_i + w_i \leq W, \quad y_i + h_i \leq H \quad \forall i \in \mathcal{I} \quad (5)$$

$$\sum_{k=1}^{m_i} s_i^k = 1 \quad \forall i \in \mathcal{I} \quad (6)$$

$$w_i = \sum_{k=1}^{m_i} w_i^k \cdot s_i^k, \quad h_i = \sum_{k=1}^{m_i} h_i^k \cdot s_i^k \quad \forall i \in \mathcal{I} \quad (7)$$

$$\sum_{k=1}^4 r_{i,j}^k \geq 1 \quad \forall i, j \in \mathcal{I} : i < j \quad (8)$$

$$x_i + w_i + a_{i,j} \leq x_j + M(1 - r_{i,j}^1) \quad \forall i, j \in \mathcal{I} : i < j \quad (9)$$

$$y_i + h_i + a_{i,j} \leq y_j + M(1 - r_{i,j}^2) \quad \forall i, j \in \mathcal{I} : i < j \quad (10)$$

$$x_j + w_j + a_{i,j} \leq x_i + M(1 - r_{i,j}^3) \quad \forall i, j \in \mathcal{I} : i < j \quad (11)$$

$$y_j + h_j + a_{i,j} \leq y_i + M(1 - r_{i,j}^4) \quad \forall i, j \in \mathcal{I} : i < j \quad (12)$$

$$x_i, y_i, w_i, h_i \geq 0 \quad \forall i \in \mathcal{I} \quad (13)$$

$$W, H \geq 0 \quad (14)$$

$$s_i^k \in \{0, 1\} \quad \forall i \in \mathcal{I} \quad \forall k \leq m_i \quad (15)$$

$$r_{i,j}^k \in \{0, 1\} \quad \forall i, j \in \mathcal{I} : i < j \quad (16)$$

$$\forall k \in \{1, 2, 3, 4\}$$

Non-overlapping of the devices is ensured by binary variables  $r_{i,j}^k$  and inequalities (8) - (12), which utilize the big-M approach [3]. At least one of the inequalities, which corresponds to the relationship (left/right/over/under) between rectangles, must be valid ( $r_{i,j}^k = 1$ ). 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 [34]. Ultimately, the ILP model for feasible placement of  $n$  rectangles uses  $\sum_{i=1}^n m_i$  binary variables to encode variant selection, and  $4 \cdot \binom{n}{2} = 2 \cdot n \cdot (n - 1)$  binary variables to encode the relative positions between rectangles.



Blockage areas are modeled as additional dummy rectangles. We fix their positions and sizes and define the minimum allowed distance parameters.  $a_{i,b} = 0$  if the rectangle  $i$  is blocked by the blockage area  $b$ ; if the rectangle is unaffected by the blockage area  $b$ , we simply omit the associated relative position constraints from the model.

We define the final aspect ratio as  $AR = \min\{W, H\} / \max\{W, H\}$ , and we want to ensure that  $l_R \leq AR \leq u_R$  holds for chosen aspect ratio parameters  $0 \leq l_R \leq u_R \leq 1$ . Then, the following additional constraints are needed. The binary variable  $r_R$  is used to handle the non-convex solution space that is induced when  $u_R \neq 1$ . When  $u_R = 1$ , we omit the associated inequalities entirely.

$$l_R \cdot W \leq H \leq u_R \cdot W + M \cdot (1 - r_R) \quad (17)$$

$$l_R \cdot H \leq W \leq u_R \cdot H + M \cdot r_R \quad (18)$$

$$r_R \in \{0; 1\} \quad (19)$$

To model the symmetry groups, we require another continuous variable per group to represent the axis of symmetry. Assume that  $G$  is the symmetry group with the vertical axis of symmetry, whose horizontal position is determined by the real variable  $x_G$ . The symmetry group consists of self-symmetric rectangles  $(i, -)$  and symmetric pairs  $(i, j)$ . Then the following equations constrain the symmetry group's rectangles to share the same axis of symmetry:

$$w_i = w_j \quad \forall (i, j) \in G \quad (20)$$

$$h_i = h_j \quad \forall (i, j) \in G \quad (21)$$

$$y_i = y_j \quad \forall (i, j) \in G \quad (22)$$

$$x_i + x_j + w_i = 2 \cdot x_G \quad \forall (i, j) \in G \quad (23)$$

$$2 \cdot x_i + w_i = 2 \cdot x_G \quad \forall (i, -) \in G \quad (24)$$

HPWL connectivity elements are formulated per net. Thanks to the minimization of the connectivity in the final criterion, no integer variables are needed. For each net  $e$ , we create four continuous variables  $X_e^M, X_e^m, Y_e^M, Y_e^m \in \mathbb{R}$ , which describe the net's bounding box. Then, we formulate the connectivity criterion  $\mathcal{L}_C$  using the following constraints for each net  $e \in E$ , given the set of the net's connected rectangles  $L_e$  and net cost  $c_e$ :

$$X_e^M \geq x_i + w_i/2 \quad \forall i \in L_e \quad (25)$$

$$X_e^m \leq x_i + w_i/2 \quad \forall i \in L_e \quad (26)$$

$$Y_e^M \geq y_i + h_i/2 \quad \forall i \in L_e \quad (27)$$

$$Y_e^m \leq y_i + h_i/2 \quad \forall i \in L_e \quad (28)$$

$$\mathcal{L}_C = \sum_{e \in E} c_e \cdot (X_e^M - X_e^m + Y_e^M - Y_e^m) \quad (29)$$

To minimize the area of the placement, which is a non-linear expression  $W \cdot H$ , we approximate it using the half perimeter of the placement's bounding box:

$$\mathcal{L}_A = W + H \quad (30)$$

We expect that thanks to the correlation between the perimeter and the area of the bounding rectangle, a solution minimizing  $\mathcal{L}_A$  will have a small area as well. Ultimately, the final criterion function is defined as:

$$\mathcal{L} = c_A \cdot \mathcal{L}_A + \frac{c_C}{\sum_{\forall e \in E} c_e} \cdot \mathcal{L}_C \quad (31)$$

where the  $c_A$ ,  $c_C$  are tunable costs; 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 cost. Furthermore, we divide  $\mathcal{L}_C$  by  $\sum_{\forall e \in E} c_e$ , so the effect of using a specific value of  $c_C$  is less sensitive to a number of nets present in the IC.

## 4.2 Improving the Performance of the Solver

As we have shown in [13], the presented formulation leads to feasible high-quality placements, but the performance of even the state-of-the-art ILP solvers is insufficient when the number of rectangles grows. We were able to mitigate this problem by providing a solver with an FDGD-based solution as a warm start. In this paper, we want to go even further, and we try to introduce redundant constraints to the original model that do not affect the optimal solutions but could potentially improve the performance of the solver.

**Symmetry Breaking** Firstly, we tried to remove the symmetric solutions from the search space. Since all of our constraints are rotation invariant (with the only exception being symmetry groups), we can prune the search space by fixing the orientations or positions of specific rectangles. Firstly, we select a suitable rectangle (the largest one as in [16]); let its index be  $K$ . Then, to remove the solutions symmetrical with respect to the  $y = x$  axis, we set all variant variables of rectangle  $K$ , which correspond to a rotated variant with index  $r$ , to zero.

The second approach is concerned with the solution symmetry achieved by swapping the quadrants of the bounding box. For example, from the current solution, another one can be created by simply mirroring it with respect to either  $x = \frac{W}{2}$  or  $y = \frac{H}{2}$  axes, or by reflecting it with respect to point  $(\frac{W}{2}; \frac{H}{2})$  point. To prune these parts of the search tree, we constrain the coordinates of rectangle  $K$  so its centroid lies within the first quadrant, closest to the origin:

$$2 \cdot x_K + w_K \leq W \quad (32)$$

$$2 \cdot y_K + h_K \leq H \quad (33)$$

**W+H Constraint** If we could predict how large the bounding box of the optimal solution would be, we could prune the search space using constraint:

$$W + H \leq P \quad (34)$$

where  $P$  is the upper bound on the half perimeter of the solution obtained from the prediction. There are two reasons why this could improve the performance of

the model. Firstly, such a hard constraint prunes some branches of the search tree that would otherwise be investigated, especially when the connectivity metric of the objective function is more emphasized and the LP relaxation does not offer a tight enough lower bound. Such restriction can also be beneficial by allowing the model to employ a much smaller big-M constant than previously possible, which can improve the LP relaxation and mitigate issues with numerical stability.

When the  $W + H$  constraint is introduced with a bound  $P$ , the big-M value can be set to  $M = P + a_M$  without making otherwise feasible solutions infeasible. We set  $a_M$  to the maximum of the minimum allowed distances between pairs of rectangles,  $a_M = \max_{(i,j)} a_{i,j}$ . This way, the constraints (9)-(12) hold even in the most extreme cases. In experiments regarding the W+H constraint, we set the  $P$  to half the perimeter of the previously found solution with additional slack to not restrict the solver too much. We discuss the obtained results in Section 6.2.

## 5 Matheuristic as a Local Search

Given the initial solution, which can be provided either by the ILP solver with limited computation time or a suitable heuristic, we try to improve it using the variable-fixing MH. We refer to this improvement phase as intensification.

### 5.1 Intensification

**Rectangle Selection** The choice of which variables should be fixed and which should remain flexible during intensification is crucial. Inspired by the job-window approach of [10], we select a local group of rectangles  $\mathcal{G}$ . Given a position  $(x, y)$  within the placement and size of the group  $g$ , the set  $\mathcal{G}$  consists of  $g$  rectangles closest to the point  $(x, y)$ . We define the 'proximity' metric of rectangle  $i$  to point  $(x, y)$  as:

$$\text{proximity}(x, y, i) = \max \{|x_i - x|, |x_i + w_i - x|, |y_i - y|, |y_i + h_i - y|\} \quad (35)$$

This way, selected rectangles should be located spatially close to each other, and when removed from the placement, mostly unfragmented empty space should appear. This should enable the solver to locally improve the connectivity by modifying the spatially local part of the placement. However, the positions and variants of the selected rectangles are not constrained, giving the solver the freedom to move them significantly if necessary.

**ILP Intensification** After the rectangle selection, the solver tries to improve the solution. The used ILP model corresponds to the one shown in Section 4.1, so the feasibility of the solution is ensured. We fix the positions and variants of each rectangle  $i \notin \mathcal{G}$ ; thus, their respective relative position variables  $r_{i,j}^k$  or variant variables  $s_i^k$  are not necessary. The selected rectangles belonging to  $\mathcal{G}$  still

have all their associated variables free. Therefore, the number of binary variables associated with  $n$  rectangles decreases from:

$$\sum_{i=1}^n m_i + 2 \cdot n \cdot (n - 1) \quad (36)$$

to significantly smaller:

$$\sum_{i \in \mathcal{G}} m_i + 2 \cdot g \cdot (g - 1) + 4 \cdot g \cdot (n - g) \quad (37)$$

Before the optimization, the solver is warm-started with the current placement. For a sufficiently small value of  $g$ , the solver is able to solve the restricted model optimally or at least find an improvement in a short time. Since the growing number of rectangles  $n$  may slow intensification significantly, we impose a time limit on optimization.

**LP Fine Optimization** To account for gaps between rectangles that can emerge by the variable fixing approach, we follow the previous step with LP optimization, which can lead to a lower value of HPWL and make the placement more compact. For each pair of rectangles, we find the least violated relative position constraint (9)-(12), and its associated variable  $r_{i,j}^k$ . Then, we optimize the original model of Section 4.1, fixing the chosen relative position variables to 1 and the variant variables to select the variants present in the current solution. Thus, the model does not contain binary variables, and the optimization is done quickly, even for large instances.

**Overall Intensification** After each successful intensification iteration, the improved solution replaces the previous one. Then, a new selection point  $(x, y)$  is sampled, and the process repeats until the computation budget is exhausted. In this paper, we generate the selection points by sampling uniformly from interval  $\langle 0; W \rangle$ ,  $\langle 0; H \rangle$  respectively. While such a simplistic strategy performed well, a more informed approach could yield better results.

The process of ILP intensification is demonstrated in Figs. 2. The current placement is shown in Fig. 2a. The sampled position  $(x, y)$ , shown as a black dot, is located near the top side of the bounding box, and 5 rectangles were selected (red, purple, green, blue, and yellow). After the ILP intensification step, the new, improved placement is shown in Fig. 2b. We can see that the selected rectangles both moved and changed their variants. Both the half perimeter of the bounding box and the HPWL were decreased by this step, as is reported in the captions.

## 5.2 Diversification

While the ILP solver guarantees us that the local neighborhood of the current solution is thoroughly searched, the algorithm can get stuck in the local minimum. In that case, it is beneficial to divert from the current solution significantly and try to reach another potentially better local minimum.

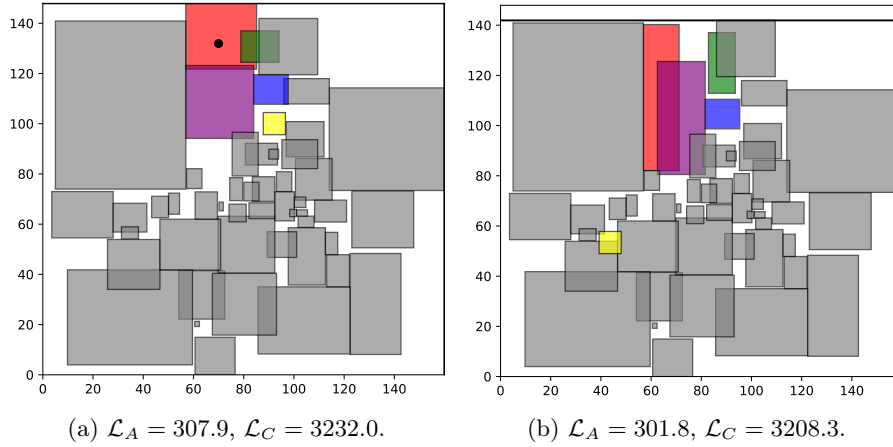


Fig. 2: Placement before and after ILP intensification. The black dot shows where the selection position  $(x, y)$  was sampled. Rectangles modified during the process are highlighted. See the decrease in height after intensification.

To perform a diversification step, we try to swap the positions of the rectangles so the overall placement changes, but we still try to keep the placement competitive. To do this, we create a swapping ILP model. In this model, each rectangle  $i$  is associated with its centroid coordinates  $(x_i^c, y_i^c)$ , as well as its area  $A_i = w_i \cdot h_i$ . Note that chosen variant and coordinates of the rectangles are retrieved from the current solution. Then, the ILP model is formed as follows:

$$\min \quad c_A \cdot \mathcal{L}_A + \frac{c_C}{\sum_{\forall e} c_e} \cdot \mathcal{L}_C + c_\xi \cdot \xi \quad (38)$$

$$\sum_{j=1}^n p_i^j = 1, \quad \sum_{j \in T_i} p_i^j = 1 \quad \forall i \in \mathcal{I} \quad (39)$$

$$\sum_{i=1}^n p_i^j = 1 \quad \forall j \in \mathcal{I} \quad (40)$$

$$x_i^s = \sum_{j=1}^n x_j^c \cdot p_i^j, \quad y_i^s = \sum_{j=1}^n y_j^c \cdot p_i^j \quad \forall i \in \mathcal{I} \quad (41)$$

$$\xi \geq N - (n - \sum_{i=1}^n p_i^i) \quad (42)$$

$$\xi \geq 0 \quad (43)$$

$$x_i^s, y_i^s \geq 0 \quad \forall i \in \mathcal{I} \quad (44)$$

$$p_i^j \in \{0, 1\} \quad \forall i, j \in \mathcal{I} \quad (45)$$

Binary variable  $p_i^j$  is equal to one if the rectangle  $i$  should be placed to the position of the rectangle  $j$  (thus,  $p_i^i = 1$  means the rectangle  $i$  does not move). To disallow the situation when a large rectangle would be placed in a position of the small one, we create a set of allowed swapping indices  $T_i$  for each rectangle  $i$ . Note that  $i \in T_i$  for each rectangle  $i$ .

$$T_i = \left\{ j \in \mathcal{I} \mid \frac{|A_i - A_j|}{\min\{A_i, A_j\}} \leq A_{\text{diff}} \right\} \quad (46)$$

Maximum relative difference ( $A_{\text{diff}} = 0.25$ ) limits the search space of the model significantly. Variables  $x_i^s, y_i^s$  track the new centroid positions of the swapped rectangles that are used to calculate the half perimeter and connectivity criteria, using additional constraints shown in Section 4.1. Finally,  $\xi$  is used to penalize the insufficient number of swaps performed, i.e., when  $p_i^i = 1$  for too many rectangles. If the less than the expected minimum number of swaps  $N$  is performed (we use  $N = n/3$ ), additional penalty  $c_\xi \cdot \xi$  is applied; we set the cost  $c_\xi$  to quite a large value  $\max\{W, H\}$ , so the solver is motivated to perform the swaps.

After determining which swaps should be performed, we modify the current solution so the centroids of the swapped rectangles are moved to their associated positions. However, this can make the solution infeasible due to possible overlaps. To make the solution feasible, we use the original ILP model of Section 4.1 again. As in LP fine optimization of the intensification phase, we find the least violated relative position constraint for each pair of rectangles, and we warm start the solver with the corresponding variables set to 1. The values of variant variables are also obtained from the previous solution. The feasible result of the diversification phase is obtained by solving the model for a limited time. Afterward, we continue with intensification.

Since our intensification implementation does not exhaustively search all possible neighborhoods, we need a mechanism to decide when to perform the diversification and when to keep searching locally. Whenever the local search does not improve the solution’s quality, we increment the counter. When the counter reaches 10, we perform the diversification and reset the counter. The counter is also reset when the improvement is achieved during the intensification.

## 6 Experiments

### 6.1 Methodology and Data

We utilized the Gurobi ILP solver v9.5.1, using four threads in each experiment. The project was implemented using Python 3.7. Experiments were performed on an Intel Xeon E5-2690.

We generated several sets of instances inspired by the structure of real-life ones. Sets  $S_{50}$  and  $S_{100}$  were already discussed in our previous work [13]. Additional sets  $S_{200}$  and  $S_{200}^{\text{sym}}$  contain a larger number of rectangles, and the latter also contains several symmetry groups as a part of each instance. Each instance

contains both the smaller rectangles, which only allow rotation, and larger ones with multiple variants. In total, 120 instances were evaluated. The computation time was fixed for each instance, depending on its set (shown in Table 1). When the MH was used, the initial solution was obtained by optimizing the original ILP model for a third of the computation time, and the rest was reserved for MH. The time required for warm starting the original model with the FDGD method was included in the total computation time. The costs in the criterion function were set to  $c_A = 1.0$ , and  $c_C \in \{0.1, 1.0, 8.0\}$  respectively.

As baseline results, the methods proposed in [13] were used. The baseline model without any improvement, denoted as **ILP**, was run only on the instance set  $S_{50}$  and  $S_{100}$ , as it could not recover any solution for larger instances within the given runtime. FDGD warm-started variant **FDGD-ILP** solved all the instances.

Table 1: Description of synthetically generated instances.

instance set	# instances	# rectangles	symmetry	comp. time
$S_{50}$	60	20, 30, 50	No	10 min
$S_{100}$	20	100	No	20 min
$S_{200}$	20	200	No	40 min
$S_{200}^{\text{sym}}$	20	200+	Yes	40 min

To compare the results obtained on the synthetically generated instances, we use the average relative difference (aRD) of the criterion, calculated for method  $m$  and instance set  $S$  as:

$$\text{aRD}_S^m = \frac{1}{|S|} \cdot \sum_{i \in S} \frac{\mathcal{L}^{i,m} - \mathcal{L}^{i,best}}{\mathcal{L}^{i,best}} \cdot 100 \text{ [\%]} \quad (47)$$

where  $\mathcal{L}^{i,m}$  is the value of criterion achieved on instance  $i$  by method  $m$ , and  $\mathcal{L}^{i,best}$  is the lowest value of criterion of among studied methods. Therefore, aRD refers to the ratio of the method’s and best-known solution’s criterion values averaged over the entire instance set. The best hits metric (BH) tells us how many times a specific method achieved the best-known value of the criterion.

## 6.2 Performance with Redundant Constraints

To study how the additional constraints affect the performance of the ILP solver, we performed experiments on instance sets  $S_{50}$  and  $S_{100}$ . In the case of set  $S_{100}$ , only results for  $c_C \in \{0.1, 1.0\}$  are reported, as for  $c_C = 8.0$ , not all methods found a feasible solution for each instance. The baseline **ILP** model is compared with symmetry-breaking one **SB-ILP** from Section 4.2, and the model **WH-ILP** using the W+H constraint from Section 4.2. Note that parameter  $P$  used to define the W+H constraint was derived from the half perimeter of the feasible solution obtained using **FDGD-ILP**, which we increased by 20 %. Furthermore,

the studied instances did not contain symmetry groups; thus, utilizing symmetry breaking did not cause any problems.

Table 2: Comparison of solutions obtained using baseline **ILP** model and the models with additional constraints, with reported values of aRD (BH) for each instance set and connectivity cost  $c_C$ .

method	$S_{50}$			$S_{100}$	
	$c_C = 0.1$	$c_C = 1.0$	$c_C = 8.0$	$c_C = 0.1$	$c_C = 1.0$
<b>ILP</b>	2.01 (19)	1.69 (20)	4.41 (22)	<b>1.81 (13)</b>	<b>2.29 (11)</b>
<b>SB-ILP</b>	1.90 (21)	3.16 (21)	5.55 (21)	4.56 (12)	6.07 (12)
<b>WH-ILP</b>	<b>0.92 (31)</b>	<b>1.56 (33)</b>	<b>2.81 (29)</b>	-	-

As shown in Table 2, the results are rather inconclusive. The symmetry-breaking constraints help a little for  $c_C = 0.1$  on  $S_{50}$  scenario, but lead to worse solutions on average. The W+H constraint leads to better solutions, but we were not able to find a feasible solution consistently for  $S_{100}$  instances. In the case of the  $c_C = 0.1$  experiment on  $S_{100}$  instance set, the feasible solution was found only for 10 of 20 instances. Furthermore, the average time needed to find the first feasible solution was 356 seconds. We concluded that imposing the upper bound on the half perimeter of the bounding box, and thus also on the big-M value, can improve the results. However, without passing the initial solution to a solver, the solver has a problem finding any feasible solution.

### 6.3 Matheuristics on Synthetic Data

Our MH approaches rely on several parameters which may significantly influence the outcome of the local search. We fixed several parameters beforehand. When the diversification is used, we apply it after 10 non-improving intensification attempts. The maximum time reserved for each intensification and diversification optimization step was set to 10 seconds.

We performed experiments with four different MH settings. The settings **MH-5**, **MH-10**, and **MH-10D** used **FDGD-ILP** to find the initial solution for local search. Settings **MH-5** and **MH-10** relied only on intensification and differed in the number of rectangles  $g$  selected to be optimized in each iteration (see Section 5.1). The first setting **MH-5** used  $g = 5$ , and the second setting **MH-10** used  $g = 10$ . The larger value of  $g$  was not used, as the complexity of the larger model decreased the performance of the ILP solver significantly. The third setting **MH-10D** also used  $g = 10$  and employed diversification.

Finally, the remaining setting **MH-10B** used the baseline **ILP** method instead of the warm-started one to generate the initial solution. Then, it only uses intensification with  $g = 10$ , thus being comparable to **MH-10**.

**Choice of suitable setting** Firstly, we tried to determine how the value of  $g$  and the use of diversification affects the results. We ran the experiments on



all instance sets for all three values of the  $c_C$ . The experiments were performed with **MH-5**, **MH-10**, and **MH-10D** settings, and with **FDGD-ILP** serving as a baseline. The results are reported in Table 3.

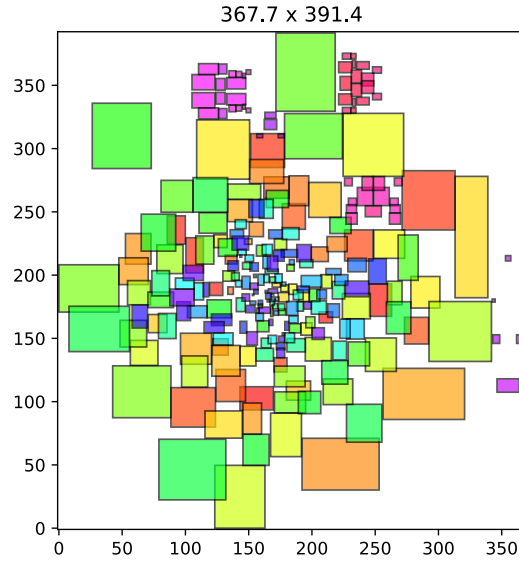
We can see the baseline **FDGD-ILP** was outperformed on all instance sets. Furthermore, the improvement provided by MHs seems to be much more significant when the connectivity cost is high. This corresponds to the expected behavior of the intensification phase. Since we only free up to 10 rectangles in each iteration, and they are selected locally close to each other, there often remains a fixed rectangle that keeps the half perimeter of the bounding box unchanged. On the other hand, the connectivity of a single net can be significantly changed by moving even a single rectangle.

The improvements provided by the MHs are especially important in the case of instance set  $S_{200}^{\text{sym}}$ , where the differences between the baseline results and the proposed methods are the largest - 30 % on average. We believe that the main reason is the rigid handling of the symmetry groups our FDGD warm start uses. To create a feasible initial solution, each symmetry group is handled as a single entity, which, however, may lead to low-quality placement shown in Fig. 3a (note, that we do not show internal devices inside the rectangles). Then, the solver cannot sufficiently improve the solution within the provided computation time due to the complexity of the model. On the other hand, the MH approach is able to decrease the value of the criterion significantly, and the overall placement looks more compact, see Fig. 3b. We also demonstrate this in Fig. 4, which shows how the criterion value changes as the computation progresses. We can see that both shown MH settings, after their initialization phase, lower the criterion rapidly, while the solver optimizing the entire model struggles. This holds true even from the area-wise point of view; MH transforms the FDGD-produced circular placement to a more compact rectangular one.

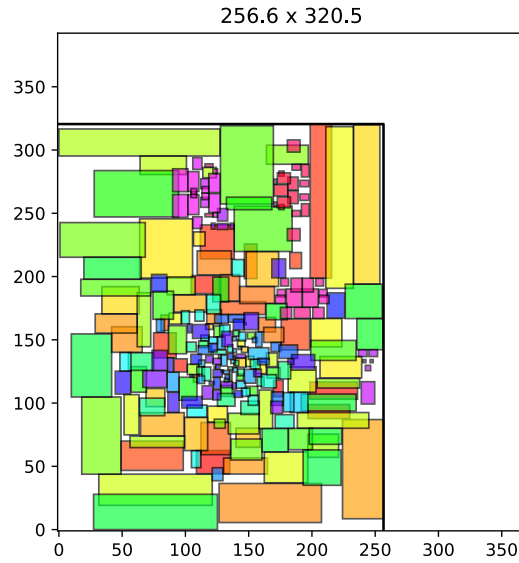
Table 3: Comparison of different MH settings and **FDGD-ILP** baseline, with reported values of aRD (BH) for each instance set and connectivity cost  $c_C$ .

method	$S_{50}$			$S_{100}$		
	$c_C = 0.1$	$c_C = 1.0$	$c_C = 8.0$	$c_C = 0.1$	$c_C = 1.0$	$c_C = 8.0$
<b>FDGD-ILP</b>	2.06 (5)	3.51 (0)	10.35 (1)	2.08 (0)	5.50 (0)	12.51 (0)
<b>MH-5</b>	1.71 (6)	2.34 (4)	5.77 (2)	0.85 (9)	1.17 (7)	1.81 (6)
<b>MH-10</b>	<b>0.14 (44)</b>	<b>0.26 (43)</b>	2.48 (15)	<b>0.48 (11)</b>	<b>0.20 (13)</b>	<b>0.26 (14)</b>
<b>MH-10D</b>	2.12 (5)	1.15 (13)	<b>0.70 (42)</b>	3.80 (0)	3.81 (0)	3.82 (0)
method	$S_{200}$			$S_{200}^{\text{sym}}$		
	$c_C = 0.1$	$c_C = 1.0$	$c_C = 8.0$	$c_C = 0.1$	$c_C = 1.0$	$c_C = 8.0$
<b>FDGD-ILP</b>	3.79 (3)	8.02 (1)	15.50 (0)	27.10 (0)	28.72 (0)	31.61 (0)
<b>MH-10</b>	<b>0.33 (16)</b>	<b>0.03 (19)</b>	<b>0.68 (19)</b>	<b>0.68 (15)</b>	<b>0.75 (11)</b>	2.53 (1)
<b>MH-10D</b>	5.38 (1)	5.83 (0)	5.38 (1)	1.51 (5)	0.85 (9)	<b>0.04 (19)</b>

From the experiments on sets with less complex instances  $S_{50}, S_{100}$ , we found out that while freeing only 5 rectangles leads to significant improvements and



(a) **FDGD-ILP** result,  $\mathcal{L} = 1190.09$ , area = 143894, HPWL = 128885.



(b) **MH-10** result,  $\mathcal{L} = 911.54$ , area = 82235, HPWL = 100000.

Fig. 3: Comparison of final placements obtained by **FDGD-ILP** and **MH-10** respectively, on instance from set  $S_{200}^{\text{sym}}$  with  $c_C = 1.0$ . Both experiments' computation time was set to 2400 s.

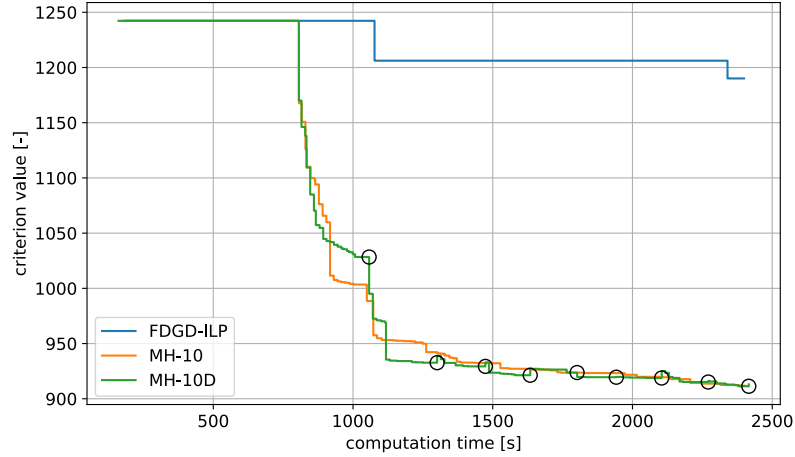


Fig. 4: Value of criterion during optimization, for instance shown in Fig. 3. Black circles show when **MH-10D** performed diversification.

shorter optimization time per iteration, using  $g = 10$  yields better results on average. Therefore, we omitted the **MH-5** from the experiments on larger instances. Then we studied the effect of diversification. **MH-10** without diversification worked well in all cases, while the **MH-10D** was less predictable. However, for two instance sets with  $c_C = 8.0$ , the **MH-10D** offered the best results, as is shown in Table 3. Also, the larger diversification step can lead to significant improvements, as is illustrated in Fig. 4, where the first time the diversification step is used (computation time 1100), the criterion drops significantly. We concluded that diversification offers advantages that could be more thoroughly exploited. However, due to the consistency of its results, we used the **MH-10** setting in the rest of the paper instead.

**Importance of the FDGD warm start** After the previous experiments, we wanted to study whether it is still necessary to use the FDGD warm start to find the initial solution for MH. To do so, we evaluated the original ILP model without warm start **ILP**, as well as its MH variant **MH-10B** on instances from  $S_{50}$  and  $S_{100}$ . The results are reported in Table 4.

From the provided table, we can see that the MH local search significantly improves the ILP baseline; it is even able to outperform the **FDGD-ILP** setting. When we compare the **MH-10B** with our main setting **MH-10**, we see that the FDGD warm start still provides some benefits. The warm-started variant of MH outperforms its non-warm-started counterpart, and this becomes more prevalent for more complex instances (where the **ILP** may not even find any solution).

Table 4: Comparison of FDGD-warm started and non-warm started MHs and ILP baselines, with reported values of aRD (BH) for each instance set and connectivity cost  $c_C$ .

method	$S_{50}$			$S_{100}$		
	$c_C = 0.1$	$c_C = 1.0$	$c_C = 8.0$	$c_C = 0.1$	$c_C = 1.0$	$c_C = 8.0$
<b>ILP</b>	5.62 (2)	10.29 (1)	19.49 (1)	19.76 (0)	45.16 (0)	46.98 (0)
<b>FDGD-ILP</b>	2.74 (1)	3.78 (0)	9.41 (1)	2.21 (2)	5.36 (0)	12.33 (0)
<b>MH-10B</b>	1.93 (26)	2.43 (15)	2.95 (24)	7.46 (4)	5.60 (4)	7.08 (2)
<b>MH-10</b>	<b>0.82 (31)</b>	<b>0.52 (44)</b>	<b>1.62 (34)</b>	<b>0.62 (14)</b>	<b>0.07 (16)</b>	<b>0.10 (18)</b>

#### 6.4 Improvement on Real Life Instances

Afterward, we studied how well MH works on real-life instances that were provided by industry partner STMicroelectronics and which we used previously in [13]. 17 instances were provided, each consisting of up to 60 independent rectangles, and we ran two different experimental settings for each instance, either allowing or forbidding the use of pocket merging. Thus, the total number of experiments was 34. As in our previous work, the optimization was limited to 8 minutes. Three connectivity costs  $c_C \in \{0.1, 1.0, 8.0\}$  were considered for each experiment.

In Table 5, we report the metrics of manual designs and our solutions (the use of pocket merging depended on the manual design). The shown metrics are the half perimeter of the bounding box  $W+H$ , the placement area, and the connectivity metric HPWL. We found a solution dominating the metrics of the manual one for 12 out of 17 instances, matching our previous results. However, when we focus on the average ratios between automated and manual designs and compare them to results generated by **FDGD-ILP** in [13], we can see that we were able to quite significantly lower the connectivity while keeping the area and half-perimeter competitive.

To highlight the differences between solutions found by **FDGD-ILP** and **MH-10**, we show Table 6. We can see that with the exception of the  $c_C = 0.1$  scenario, the MH approach, on average, reduced the criterion of the final solution and found the better solution in a majority of the cases. This again corresponds to the observations we made in Section 6.3. The real-life instances also contain only up to 60 rectangles, and as we have shown, the effect of the MH shows off when the instances are more complex.

Furthermore, note the values reported in columns DOM. These correspond to a number of occurrences when the method found a solution that had both a smaller area and HPWL than the solution found by the other method; such a solution is objectively better given our two main metrics. We can see that **MH-10** was able to do so in more cases, which again highlights the power of local search performed by the ILP solver.

We illustrate the mentioned results with Figs.5, which show instance number 14. Three figures correspond to the manual solution, the solution obtained using **FDGD-ILP**, and finally using **MH-10**. Note that the manual design does

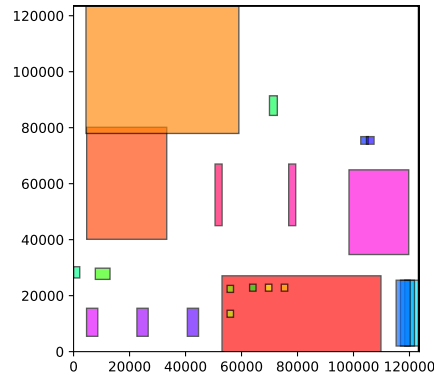
not show the positions of the physical devices within the rectangles. We can see that both automatically generated solutions dominated the manual design. The differences between both automated solutions are subtle; their areas are actually equal. However, even these subtle changes in the positions of smaller rectangles are enough to dramatically decrease the HPWL in the case of the solution generated by **MH-10**.

Table 5: Values of  $W+H$  in  $\mu\text{m}$ , area in  $\mu\text{m}^2$  and HPWL in  $\mu\text{m}$  for each instance, and average ratios of automated and manual metrics, obtained using **MH-10**. The average ratios obtained by **FDGD-ILP** in [13] are shown in the last row for comparison. Solutions dominating the manual one, given all three metrics, are highlighted.

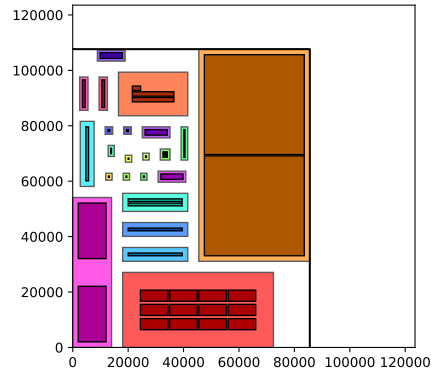
instance	manual			<b>MH-10</b>								
	W+H	area	HPWL	$c_{conn} = 0.1$			$c_{conn} = 1.0$			$c_{conn} = 8.0$		
				W+H	area	HPWL	W+H	area	HPWL	W+H	area	HPWL
1	158	6118	1850	157	6172	1636	157	6183	1562	166	6889	1478
2	116	2710	1784	<b>88</b>	<b>1936</b>	<b>1024</b>	<b>91</b>	<b>2070</b>	<b>928</b>	106	2757	797
3	106	2650	906	<b>85</b>	<b>1779</b>	<b>660</b>	<b>89</b>	<b>1968</b>	<b>654</b>	<b>92</b>	<b>2119</b>	<b>547</b>
4	129	4096	812	<b>112</b>	<b>3117</b>	<b>782</b>	<b>114</b>	<b>3256</b>	<b>717</b>	131	4064	662
5	207	8972	13797	<b>159</b>	<b>6351</b>	<b>9955</b>	<b>165</b>	<b>6789</b>	<b>8141</b>	<b>169</b>	<b>7120</b>	<b>7863</b>
6	178	7698	4039	<b>169</b>	<b>7167</b>	<b>3666</b>	<b>167</b>	<b>7009</b>	<b>3647</b>	<b>174</b>	<b>7224</b>	<b>3615</b>
7	168	6580	2908	164	6756	2633	168	7093	2314	173	7466	2307
8	173	7294	1501	<b>160</b>	<b>6399</b>	<b>1224</b>	<b>169</b>	<b>6973</b>	<b>1068</b>	<b>173</b>	<b>7139</b>	<b>1093</b>
9	243	14129	4705	<b>225</b>	<b>12647</b>	<b>4205</b>	<b>234</b>	<b>13664</b>	<b>4003</b>	241	14487	3882
10	205	10214	28386	191	9093	38626	194	9446	32363	236	13714	24930
11	225	9922	28527	197	9356	29074	205	10313	17864	241	13717	13210
12	155	5953	3824	<b>123</b>	<b>3803</b>	<b>2315</b>	<b>126</b>	<b>3937</b>	<b>2162</b>	159	6298	1597
13	162	6511	2061	<b>153</b>	<b>5855</b>	<b>1822</b>	<b>155</b>	<b>6002</b>	<b>1665</b>	<b>155</b>	<b>6008</b>	<b>1693</b>
14	247	15235	2399	<b>193</b>	<b>9212</b>	<b>1720</b>	<b>193</b>	<b>9263</b>	<b>1557</b>	<b>211</b>	<b>10657</b>	<b>1363</b>
15	123	3758	1619	115	3309	1817	113	3178	1852	116	3385	1712
16	232	12397	2676	<b>215</b>	<b>11551</b>	<b>1973</b>	<b>223</b>	<b>12318</b>	<b>1792</b>	<b>221</b>	<b>12143</b>	<b>1944</b>
17	247	12525	4586	<b>225</b>	<b>12172</b>	<b>3313</b>	235	13708	3008	252	15790	2964
avg ratio <b>MH-10</b>	1.00	1.00	1.00	0.89	0.84	0.86	0.91	0.89	0.77	0.98	1.02	0.71
avg ratio <b>FDGD-ILP</b> [13]	1.00	1.00	1.00	0.88	0.84	0.93	0.91	0.89	0.82	0.99	1.04	0.74

Table 6: Comparison of **FDGD-ILP** and **MH-10** for all 34 experiments performed on real-life instances. DOM shows in how many cases the method dominated the other one with respect to both the area and HPWL.

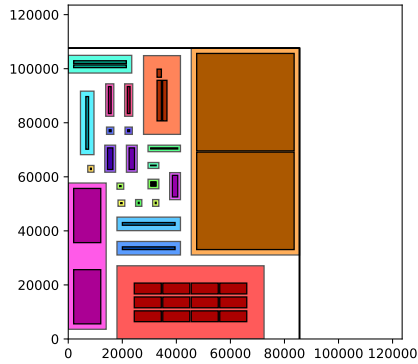
	$c_C = 0.1$		$c_C = 1.0$		$c_C = 8.0$	
	aRD (BH)	DOM	aRD (BH)	DOM	aRD (BH)	DOM
<b>FDGD-ILP</b>	<b>0.46 (22)</b>	1	1.82 (11)	1	3.84 (14)	0
<b>MH-10</b>	0.60 (12)	8	<b>0.11 (23)</b>	10	<b>0.49 (20)</b>	11



(a) Manual design, area =  $15235\mu\text{m}^2$ , HPWL =  $2399\mu\text{m}$ .



(b) **FDGD-ILP**, area =  $9212\mu\text{m}^2$ , HPWL =  $1898\mu\text{m}$ .



(c) **MH-10**, area =  $9212\mu\text{m}^2$ , HPWL =  $1720\mu\text{m}$ .

Fig. 5: Comparison of manual and automated placements, obtained for  $c_C = 0.1$ . Shown instance corresponds to the 14th row in Table 5.

## 7 Conclusion

In this paper, we extended our previous work on the automation of the placement of AMS ICs. We studied the effect of additional redundant constraints on the performance of the state-of-the-art ILP solver. While the symmetry-breaking constraints did not enhance the solver’s performance, imposing an additional constraint on the maximum value of half perimeter of the placement led to improvement on the smaller instances. However, for larger instances, such constraint made the solver unable to find any feasible solution in a given computation time, even though the bound was derived from a known feasible solution. Therefore, we need to provide a solver with an initial solution if we would like to exploit the half-perimeter constraint in the future.

Our experiments with MHs were more successful. We proposed applying the ILP solver to perform a local search in the created placement. The intensification phase of the MH relied on freeing variables associated with a few spatially close rectangles while fixing the other. We showed an additional ILP model that we used to perform the diversification step, to diverge further from the current solution when the local minimum is reached. We evaluated several different MH settings on synthetically generated instances. We concluded that using intensification only and freeing 10 rectangles in each iteration led to the best results overall. However, the potential benefits of diversification cannot be overlooked, but its application would probably require a more advanced control mechanism than presented in our paper. Ultimately, we significantly improved our previous results, obtained using FDGD-warm started ILP, on large instances with 200 and more rectangles, especially when symmetry groups are present in the instance.

Finally, we created automatically generated placements for real-life instances provided by our industry partner STMicroelectronics. We could compare our results with the manually created benchmarks and our previous results. We were again able to outperform both the area and the HPWL in the case of 12 instances; furthermore, we were able to reduce the average value of HPWL even further while keeping the area metric unaffected. When we analyzed the improvement against our previous results more closely, we found that the MH approach dominated its ILP-only counterpart regarding both the HPWL and area in one-third of the experiments performed on real-life instances. This again suggests that the use of MH could be beneficial not only in the specific domain of AMS IC placement but in the domains of packing and cutting as well, where only the area is minimized.

**Acknowledgements** This work was supported by the Grant Agency of the Czech Republic under the Project GACR 22-31670S. This work was co-funded by the European Union under the project ROBOPROX - Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22\_008/0004590). We would like to thank the STMicroelectronics company, namely Dalibor Barri and Patrik Vacula, for providing real-life instances and helpful discussions about a problem.

## References

1. Alvarez-Valdes, R., Parreño, F., Tamarit, J.: Reactive GRASP for the strip-packing problem. *Computers & Operations Research* **35**(4), 1065–1083 (2008)
2. Berger, M., Schröder, M., Küfer, K.H.: A constraint-based approach for the two-dimensional rectangular packing problem with orthogonal orientations. In: *Operations Research Proceedings 2008*. pp. 427–432 (2009)
3. Camm, J.D., Raturi, A.S., Tsubakitani, S.: Cutting Big M down to Size. *Interfaces* **20**(5), 61–66 (1990)
4. Chen, T.C., Jiang, Z.W., Hsu, T.C., et al.: NTUplace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **27**(7) (2008)
5. Christiaens, J., Vanden Berghe, G.: Slack induction by string removals for vehicle routing problems. *Transportation Science* **54** (2020)
6. Cohn, J., Garrod, D., Rutenbar, R., Carley, L.: KOAN/ANAGRAM II: new tools for device-level analog placement and routing. *IEEE Journal of Solid-State Circuits* **26**(3), 330–342 (1991)
7. CORDIS: Analog/Mixed Signal Back End Design Automation based on Machine Learning and Artificial Intelligence Techniques (AMBEATion), <https://cordis.europa.eu/project/id/101007730>, Last accessed 16 May 2023
8. DARPA: Intelligent Design of Electronic Assets (IDEA), <https://www.darpa.mil/program/intelligent-design-of-electronic-assets>, Last accessed 16 May 2023
9. Deghdak, K., T'kindt, V., Bouquard, J.L.: Scheduling evacuation operations. *Journal of Scheduling* **19**(4), 467–478 (2016)
10. Della Croce, F., Grosso, A., Salassa, F.: A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research* **213**(1), 67–78 (2014)
11. Dumez, D., Lehuédé, F., Péton, O.: A large neighborhood search approach to the vehicle routing problem with delivery options. *Transportation Research Part B: Methodological* **144**, 103–132 (2021)
12. Fischetti, M., Fischetti, M.: *Matheuristics*, vol. 1-2, pp. 121–153. Springer (2018). [https://doi.org/10.1007/978-3-319-07124-4\\_14](https://doi.org/10.1007/978-3-319-07124-4_14)
13. Grus., J., Hanzálek., Z., Barri., D., Vacula., P.: Automatic placer for analog circuits using integer linear programming warm started by graph drawing. In: *Proceedings of the 12th International Conference on Operations Research and Enterprise Systems*. pp. 106–116 (2023)
14. Hopper, E., Turton, B.: A genetic algorithm for a 2D industrial packing problem. *Computers & Industrial Engineering* **37**(1), 375–378 (1999)
15. Klausnitzer, A., Lasch, R.: Optimal facility layout and material handling network design. *Computers & Operations Research* **103**, 237–251 (2019)
16. Korf, R., Moffitt, M., Pollack, M.: Optimal rectangle packing. *Annals of Operations Research* **179**(1), 261–295 (2010)
17. Lin, Y., Li, Y., Fang, D., et al.: Are analytical techniques worthwhile for analog IC placement? In: *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. pp. 154–159 (2022)
18. Lindahl, M., Sørensen, M., Stidsen, T.: A fix-and-optimize matheuristic for university timetabling. *Journal of Heuristics* **24**(4), 645–665 (2018)
19. Lourenco, N., Vianello, M., Guilherme, J., Horta, N.: LAYGEN - automatic layout generation of analog ICs from hierarchical template descriptions. In: *2006 Ph.D. Research in Microelectronics and Electronics*. pp. 213–216 (2006)



20. Lu, J., Chen, P., Chang, C.C., et al.: EPlace: Electrostatics-based placement using fast fourier transform and nesterov's method. *ACM Trans. Des. Autom. Electron. Syst.* **20**(2) (2015)
21. Ma, Q., Xiao, L., Tam, Y.C., Young, E.F.Y.: 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 (2011)
22. Mallappa, U., Pratty, S., Brown, D.: RLPlace: Deep RL guided heuristics for detailed placement optimization. In: *Proceedings of the 2022 Conference & Exhibition on Design, Automation & Test in Europe*. p. 120–123 (2022)
23. Martins, R., Lourenço, N., Horta, N.: Multi-objective optimization of analog integrated circuit placement hierarchy in absolute coordinates. *Expert Systems with Applications* **42**(23), 9137–9151 (2015)
24. Mirhoseini, A., Goldie, A., Yazgan, M., et al.: A graph placement methodology for fast chip design. *Nature* **594**(7862), 207–212 (Jun 2021)
25. Murata, H., Fujiyoshi, K., Nakatake, S., Kajitani, Y.: VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **15**, 1518–1524 (1996)
26. Mönch, L., Roob, S.: A matheuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective. *Applied Soft Computing* **68**, 835–846 (2018)
27. Ou, H.C., Tseng, K.H., Liu, J.Y., Wu, I.P., Chang, Y.W.: Layout-dependent effects-aware analytical analog placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **35**(8), 1243–1254 (2016)
28. Polyakovskiy, S., M'Hallah, R.: A lookahead matheuristic for the unweighed variable-sized two-dimensional bin packing problem. *European Journal of Operational Research* **299**(1), 104–117 (2022)
29. Smet, P., Wauters, T., Mihaylov, M., Vanden Berghe, G.: The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights. *Omega* **46**, 64–73 (2014)
30. Spindler, P., Schlichtmann, U., Johannes, F.M.: Kraftwerk2—a fast force-directed quadratic placement approach using an accurate net model. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* **27**(8) (2008)
31. Strasser, M., Eick, M., Grab, H., Schlichtmann, U., Johannes, F.M.: Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions. In: *2008 IEEE/ACM International Conference on Computer-Aided Design*. pp. 306–313 (2008)
32. Xiao, Y., Xie, Y., Kulturel-Konak, S., Konak, A.: A problem evolution algorithm with linear programming for the dynamic facility layout problem—a general layout formulation. *Computers & Operations Research* **88**, 187–207 (2017)
33. Xie, W., Sahinidis, N.V.: A branch-and-bound algorithm for the continuous facility layout problem. *Computers & Chemical Engineering* **32**(4), 1016–1028 (2008)
34. Xu, B., Li, S., Pui, C.W., et al.: Device layer-aware analytical placement for analog circuits. In: *Proceedings of the 2019 International Symposium on Physical Design*. p. 19–26. *ISPD '19* (2019)
35. Xu, B., Li, S., Xu, X., et al.: Hierarchical and analytical placement techniques for high-performance analog circuits. In: *Proceedings of the 2017 ACM on International Symposium on Physical Design*. p. 55–62. *ISPD '17* (2017)
36. Yang, F., Leus, R.: Scheduling hybrid flow shops with time windows. *Journal of Heuristics* **27**(1), 133–158 (2021)