Analysis and design of algorithms for the manufacturing process of integrated circuits

Sonia Fleytas
Universidad Nacional de Asuncion, Facultad Politecnica,
San Lorenzo, Paraguay, 2160,
beatrizfleytas@fpuna.edu.py

and

Diego P. Pinto-Roa
Universidad Nacional de Asuncion, Facultad Politecnica,
San Lorenzo, Paraguay, 2160,
dpinto@pol.una.py

and

Jose Colbes
Universidad Nacional de Asuncion, Facultad Politecnica,
San Lorenzo, Paraguay, 2160,
jcolbes@pol.una.py

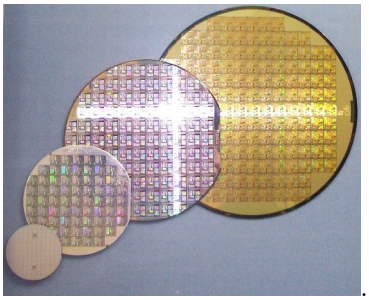
Abstract

The stage of transporting semiconductor chips from the wafer to the support strip is crucial in the integrated circuit manufacturing process. This process can be modeled as a combinatorial optimization problem where the objective is to reduce the total distance the robotic arm must travel to pick up each chip and place it in its corresponding position within the support structure. This problem is of the *pick-and-place* type and is NP-hard. The (approximate) solution proposals of state-of-the-art methods include rule-based approaches, genetic algorithms, and reinforcement learning. There is a binary integer programming model for this problem in the literature, from which its authors proposed a genetic algorithm to obtain approximate solutions. Our work analyzes this model and identifies its limitations. From this, we propose: (i) a new ILP model, and (ii) a new solution representation, which, unlike the reference work, guarantees that feasible solutions are obtained throughout the generation of new individuals. Based on this new representation, we proposed and evaluated other approximate methods, including a greedy algorithm and a genetic algorithm that improve the state-of-the-art results for test cases usually used in the literature. Additionally, the results obtained from our new ILP model indicate that our genetic algorithm results are very close to the optimal values.

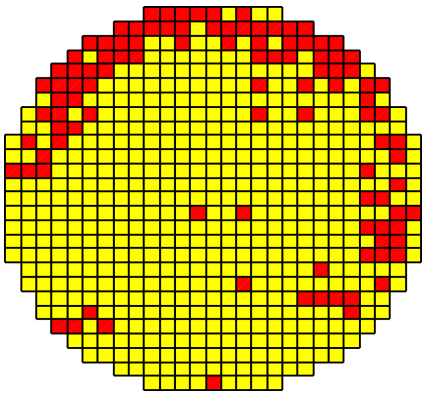
Keywords: integrated circuit manufacturing, back-end production, chip placement process, greedy algorithm, genetic algorithm, pick-and-place, integer linear programming.

1 Introduction

The integrated circuit (IC) is one of the most significant technological achievements of the last 60 years. The demand for ICs has been increasing considerably in recent years due to multiple factors: the development of computers, the Internet, mobile communications, *big data*, cloud computing, artificial intelligence, 5G communication, optical communications, Internet of Things (IoT), automotive electronics, satellite communication and other applications of emerging technologies [1].



(a) *Wafers* of various sizes with the semiconductor chips. Image source: `https://commons.wikimedia.org/wiki/File:Wafer_2_Zoll_bis_8_Zoll_2.jpg`



(b) Quality information of semiconductor chips in a *wafer*. The positions in red indicate the defective chips. Image extracted from [5].

Figure 1: *Front-end* process in manufacturing integrated circuits.

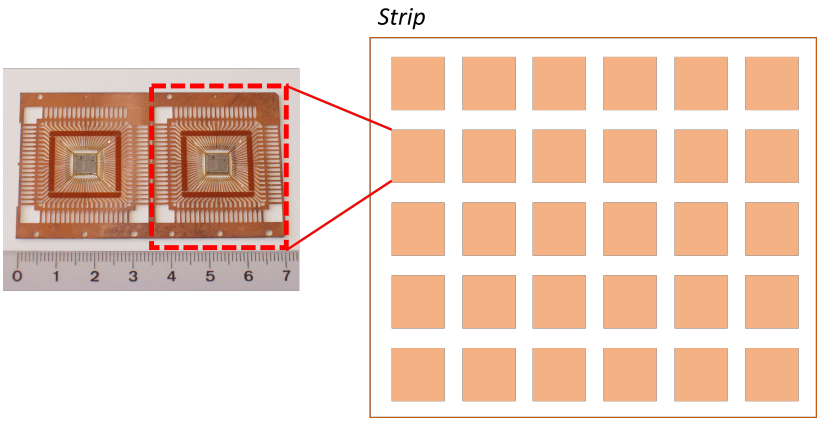


Figure 2: *Strip* with the slots where the semiconductor chips are deposited. Image adapted from: `https://commons.wikimedia.org/wiki/File:TQFP_Leadframe.jpg`

The IC industry is a branch of the semiconductor industry considered strategic and fundamental for a country. It is estimated that between 2021 and 2030, the global market will be between 400 and 537.5 billion dollars, with an annual growth rate of 3% [2]. Moreover, with the rapid development of global information and networks, the IC industry is an essential indicator of a country's economic and industrial strength and a key player in defense and national security [3].

The IC manufacturing process consists of four main stages: (i) the manufacture of the *wafer* with the semiconductors, (ii) the analysis of the *wafer*, (iii) packaging and assembly of the ICs, (iv) and the functional test of the IC [4]. The first two stages are known as the *front-end* process, and the last two as the *back-end* process.

In the *front-end* process, a *wafer* of semiconductor material (usually silicon) is processed and cut to produce semiconductor chips. Examples of *wafers* are shown in Figure 1(a). This process is complicated by the flow of reentrant products, high uncertainties in operations, and rapid changes in products and technologies [6]. Once this process is complete, each chip is examined for its electrical performance, classifying it accordingly as *good* or *bad* [7]. Figure 1(b) is shown as an example. This chip classification information is stored in a file and sent to the assembly process.

In the *back-end* process, only the good chips from the *wafer* are picked up by a robotic arm and attached to slots in a support structure (lead frames or substrate) on a *strip* [8]. Figure 2 shows an example of a *strip*. After all the chips have been collected and located, they must be retested in a final performance test. Since the robotic arm can move only one chip at a time, the total distance traveled by the robotic arm is a crucial point in the *back-end* process. To optimize this process, each chip's collection and assembly sequence must be considered; that is, in what order should the robotic arm take each chip and where should it be located to achieve the least possible displacement of the robotic arm. The described process is known as *semiconductor chip placement process* and can be formulated as a typical *pick-and-place* problem, which is NP-hard [9].

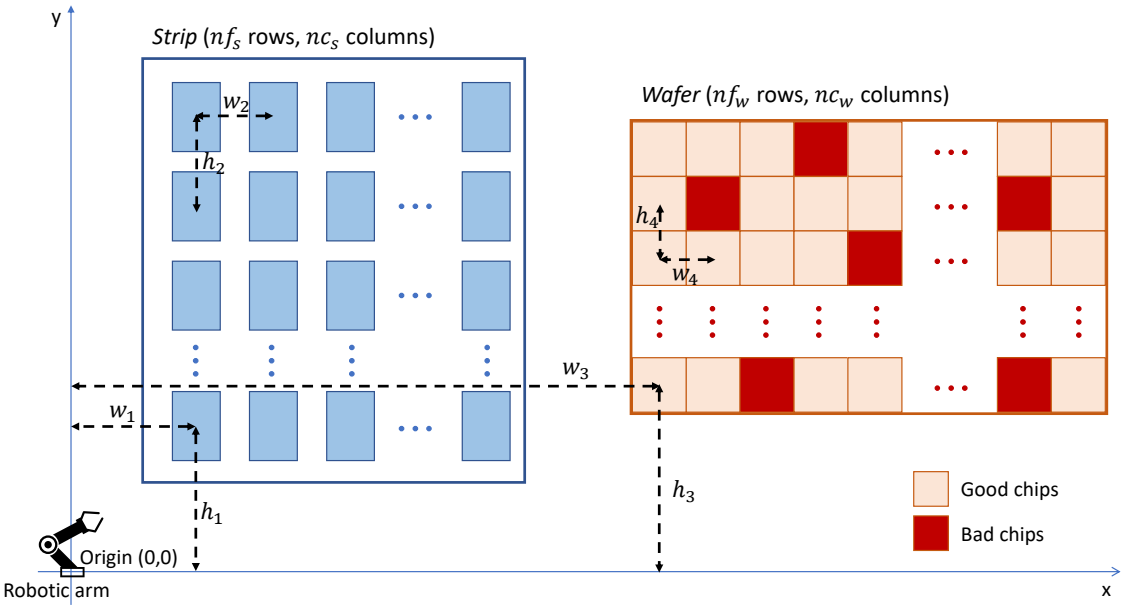


Figure 3: Definition of the problem and its parameters. Image adapted from [10].

This paper analyzes the mathematical formulation of the problem and the (approximate) solution proposals presented in two previous papers [8, 10]. These methods include eight heuristic rules that determine the movement of the robotic arm, a genetic algorithm that uses encoding based on binary matrices as chromosomes [8], and an interactive Q-learning algorithm with two agents: one for collection and the other for placement [10]. In our analysis, we identified the limitations of the mathematical formulation in [8] and proposed a new one that obtains an optimal solution for a given instance. Also, a random algorithm, a local search algorithm, a greedy algorithm, and a genetic algorithm are proposed for approximate solutions. The latter improved the results of the state-of-the-art methods for a set of test cases presented in the two works considered, and its outcomes are very close to the optimal values.

2 Materials and methods

2.1 Formulation of the problem and calculation of the total distance

At the beginning of the *back-end* manufacturing process of semiconductor chips, they are found in the *wafer*, and their quality information is available. In the problem considered in this work, we seek to determine the sequence of picking and placing each *good* chip of the *wafer* in the support structure or *strip*, which minimizes the total distance that the robot arm must move since its departure from the starting position to pick up the first good chip from a *wafer* to locating the last good chip on the *strip* and return to its starting position.

The total distance traveled by the robotic arm depends mainly on two factors: (i) the selection order of the chips in the *wafer* and (ii) the *strip* slot (lead frame) assigned to each chip. Therefore, the considered problem consists of finding the shortest path for the route of the robotic arm in the described process.

For the formulation of this problem, the following considerations are made [8, 10]:

1. Once the analysis of the *wafer* chips is finished, their classification is stored in a file, where a good chip is represented by 1 and a bad chip by 0.

2. The starting and ending point of the robotic arm is considered as the point $(0, 0)$; that is, the robotic arm starts from this point to pick up the first good chip from the *wafer* and returns to it after placing the last good chip on the *strip*.

3. It is assumed that the *strip* is to the left of the *wafer* and that both have a rectangular shape (see Figure 3). The *strip* has nf_s rows and nc_s columns, while the *wafer* has nf_w rows and nc_w columns.

4. The number of chips (N) to transport can be greater than the number of slots (M) available in a *strip* (i.e., $N >= M$). Once all the slots in a *strip* are filled, it is replaced by an empty one. Therefore, $c = \lceil N/M \rceil$ *strips* are needed to locate all the chips.

5. The robotic arm moves only horizontally or vertically.
6. The coordinates (x, y) of each chip in the *wafer* and of each slot in the *strip* can be determined through the parameters w_{1-4} and h_{1-4}, as can be seen in Figure 3.
7. w_1 represents the horizontal distance between the starting point of the robotic arm and the center of the first position column of the *strip*.
8. w_2 represents the horizontal distance between the centers of two consecutive columns of the *strip*.
9. w_3 represents the horizontal distance between the starting point of the robotic arm and the center of the first position column of the *wafer*.
10. w_4 represents the horizontal distance between the centers of two consecutive columns of the *wafer*.
11. h_1 represents the vertical distance between the starting point of the robotic arm and the center of the last row of positions of the *strip*.
12. h_2 represents the vertical distance between the centers of two consecutive rows of the *strip*.
13. h_3 represents the vertical distance between the starting point of the robotic arm and the center of the last row of *wafer* positions.
14. h_4 represents the vertical distance between the centers of two consecutive rows of the *wafer*.

From these considerations, the coordinates $(x_{s_{ij}}, y_{s_{ij}})$ can be calculated for each slot with row $i \in \{0, 1, \ldots, nfs - 1\}$ and column $j \in \{0, 1, \ldots, nc_s - 1\}$ of the *strip*. Similarly, the coordinates $(x_{w_{ij}}, y_{w_{ij}})$ can be calculated for each chip with row $i \in \{0, 1, \ldots, nf_w - 1\}$ and column $j \in \{0, 1, \ldots, nc_w - 1\}$ of the *wafer*. These coordinates are obtained as follows:

$$\begin{aligned} x_{s_{ij}} &= w_1 + w_2 * j \\ y_{s_{ij}} &= h_1 + h_2 * (nf_s - i - 1) \\ x_{w_{ij}} &= w_3 + w_4 * j \\ y_{w_{ij}} &= h_3 + h_4 * (nf_w - i - 1) \end{aligned}$$

Next, we define as *step* of the robotic arm the path it takes to pick up a certain chip from the *wafer* and place it in a certain slot on the *strip*. Thus, if we have N good chips, the number of steps is N. Therefore, a solution to the problem posed can be represented by two sequences:

1. The chip collection sequence of the *wafer*. Let $W(k)$ be the point p_{k_w} in the (2D) plane with the coordinates $(x_{p_{k_w}}, y_{p_{k_w}})$ of the chip that is collected in the step k ($k \in \{1, 2, ..., N\}$) of the path of the robotic arm.
2. The visiting sequence of the slots of the *strip*. Let $S(k)$ be the point p_{k_s} in the (2D) plane with the coordinates $(x_{p_{k_s}}, y_{p_{k_s}})$ of the slot where it will be deposited the chip that is picked up at step k ($k \in \{1, 2, ..., N\}$) of the robotic arm path. It is important to note here that, considering that the number of chips N can be greater than the number of slots M in a *strip*, certain positions may be repeated in this sequence (corresponding to different *strips*).

Due to the assumption that the robotic arm can only move horizontally or vertically, the *Manhattan distance* between two points (in the plane) p_1 and p_2 are considered as:

$$d_{p_1,p_2} = |x_{p_1} - x_{p_2}| + |y_{p_1} - y_{p_2}|$$

With these definitions, and taking into account that the robotic arm starts from the origin and returns to it after the process, then the total distance (TD) traveled is:

$$TD = d_{Orig,W(1)} + \sum_{k=1}^{N} d_{W(k),S(k)} + \sum_{k=1}^{N-1} d_{S(k),W(k+1)} + d_{S(N),Orig}$$

Figure 4 shows an example of four chips and four available slots, indicating the coordinates of each one. If the *wafer* chips are picked up in order from left to right and from top to bottom, and placed on the *strip* following the same order for the slots, then we have:

Strip (2 rows, 2 columns)

(26, 202)	(38, 202)
(26, 182)	(38, 182)

Wafer (2 rows, 2 columns)

(66, 52)	(70, 52)
(66, 48)	(70, 48)

Figure 4: Example case, with the coordinates (x, y) of good *wafer* chips (2 rows and 2 columns) and *strip* slots (2 rows and 2 columns).

$$\begin{aligned} S &= \{(26,202),(38,202),(26,182),(38,182)\} \\ W &= \{(66,52),(70,52),(66,48),(70,48)\} \end{aligned}$$

Therefore:

$$\begin{aligned} d_{Orig,W(1)} &= |0-66|+|0-52| = 118 \\ \sum_{k=1}^{N} d_{W(k),S(k)} &= (|66-26|+|52-202|)+(|70-38|+|52-202|) \\ &\quad +(|66-26|+|48-182|)+(|70-38|+|48-182|) = 712 \\ \sum_{k=1}^{N-1} d_{S(k),W(k+1)} &= (|26-70|+|202-52|)+(|38-66|+|202-48|) \\ &\quad +(|26-70|+|182-48|) = 554 \\ d_{S(N),Orig} &= |38-0|+|182-0| = 220 \\ TD &= 118+712+554+220 = 1604 \end{aligned}$$

2.2 State of the art methods considered

In this work, ten state-of-the-art methods are considered to establish the movement of the robotic arm. The first eight methods are rules defined in Table 1, where the *Pickup* column indicates the W chip collection sequence of the *wafer* and the *Location* column indicates the slot visit sequence S of the *strip* to locate the chips.

Another method considered in the literature is a genetic algorithm that is formulated from a *binary integer programming* (BIP) [8] model.

The state-of-the-art method with the best results for this problem is based on the [10] reinforcement learning approach. The authors developed an interactive Q-learning with two agents, one for picking and one for placing, which are trained to determine the path of the robotic arm interactively.

2.3 Test cases and parameters

We employ a test set of four input cases for each method considered in this paper. This set was proposed and used in previous works [8, 10], and is shown in Figure 6. For the cases *WaferA* and *WaferC*, there is a bivariate normal distribution with 80% and 90% good chips, respectively. In contrast, for the cases *WaferB* and *WaferD*, there is a uniform distribution with 80% and 90% good chips, respectively. In all cases, the number of rows and columns of the *wafer* equals 9, while for the *strip*, there are ten rows and four columns.

The parameters used for the experiments are the following [10, 8]: $nf_s = 10$, $nc_s = 4$, $nf_w = 9$, $nc_w = 9$, $w_1 = 16$, $w_2 = 12$, $w_3 = 66$, $w_4 = 4$, $h_1 = 22$, $h_2 = 20$, $h_3 = 18$, $h_4 = 4$.

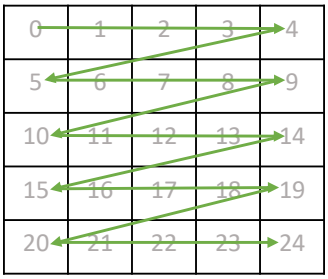
The results of the state-of-the-art methods, considering these test cases and parameters, are shown in Figure 7. Since the results for the eight selected rules are very close to each other, only the total distances for the $R1$ and LD rules are shown. It can be seen that the genetic algorithm outperforms the rules, although not by a considerable margin. In contrast, the reinforcement learning method performs significantly better than the others in all test cases.

2.4 Analysis of a BIP model and a genetic algorithm from a previous work

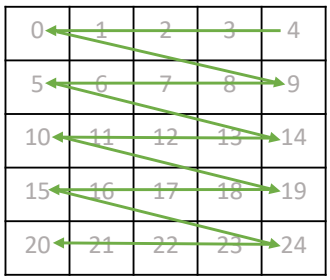
A previous work proposes a binary integer programming (BIP) model for the problem addressed [8]. It is presented here in some detail for the following reasons:

Table 1: Rules (heuristics) for determining the *wafer* chip collection sequence and their placement in the *strip* slots.

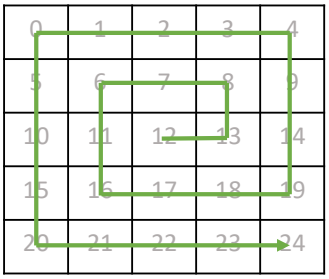
Rule	Pickup	Location
R1	Each good chip is collected from the *wafer* in a left to right, top to bottom order (Figure 5(a)).	Each collected chip is placed on the *strip* in order from left to right and top to bottom (Figure 5(a)).
R2	Each good chip is collected from the *wafer* in a left to right, top to bottom order (Figure 5(a)).	Each collected chip is placed on the *strip* following an order from right to left and from top to bottom (Figure 5(b)).
R3	Each good chip is collected from the *wafer* in a right-to-left, top-to-bottom order (Figure 5(b)).	Each collected chip is placed on the *strip* following an order from right to left and from top to bottom (Figure 5(b)).
R4	Each good chip is picked from the *wafer* in a right-to-left, top-to-bottom order, as shown in Figure 5(b).	Each collected chip is placed on the *strip* in order from left to right and top to bottom (Figure 5(a)).
RU	Each good chip is picked up from the *wafer* in a circular path, starting in the center and moving to the right and top (Figure 5(c)).	Each collected chip is placed on the *strip* in order from left to right and top to bottom (Figure 5(a)).
RD	Each good chip is picked up from the *wafer* in a circular path, starting in the center and moving to the right and bottom (Figure 5(d)).	Each collected chip is placed on the *strip* in order from left to right and top to bottom (Figure 5(a)).
LU	Each good chip is picked up from the *wafer* in a circular path, starting in the center and moving to the left and top (Figure 5(e)).	Each collected chip is placed on the *strip* in order from left to right and top to bottom (Figure 5(a)).
LD	Each good chip is picked up from the *wafer* in a circular path, starting in the center and moving to the left and bottom (Figure 5(f)).	Each collected chip is placed on the *strip* in order from left to right and top to bottom (Figure 5(a)).



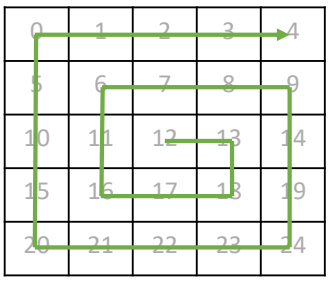
(a) Left-right from top-bottom



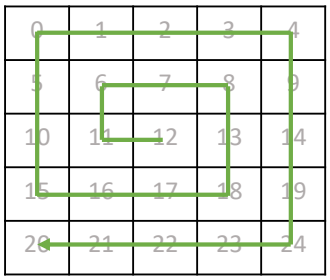
(b) Top-bottom right-left



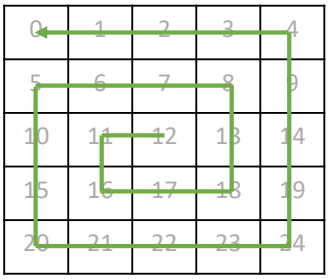
(c) Right-top starting at center



(d) Right-bottom starting at center



(e) Left-top starting at center



(f) Left-bottom starting at center

Figure 5: Rules for robotic arm movements. Image adapted from [8].

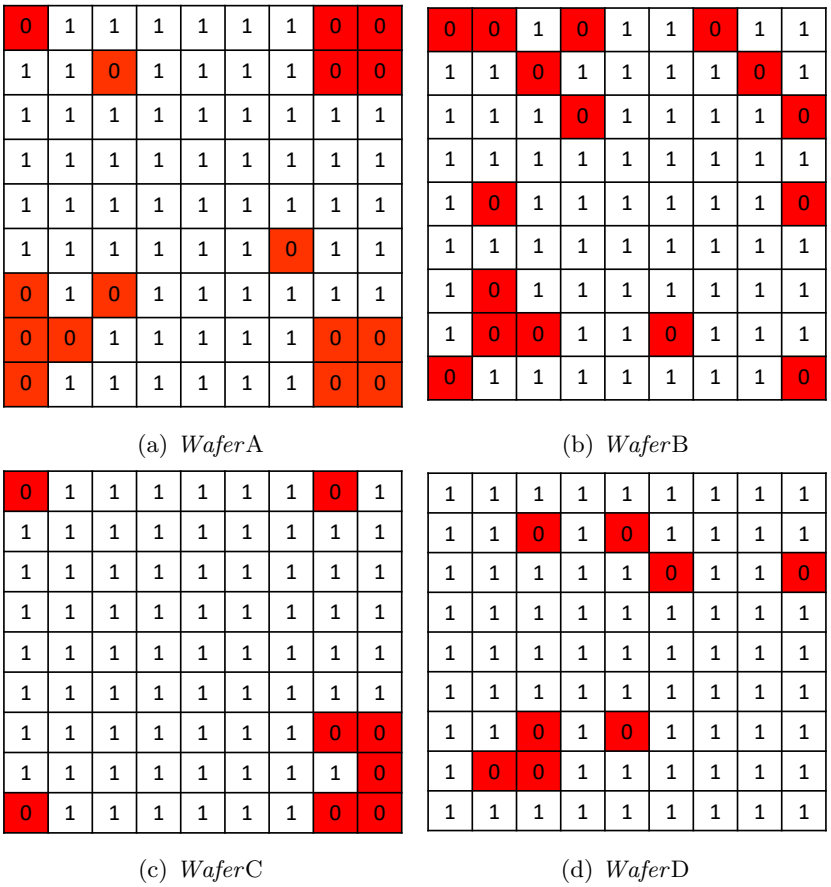


Figure 6: Test cases used in experiments. Image adapted from [8, 10].

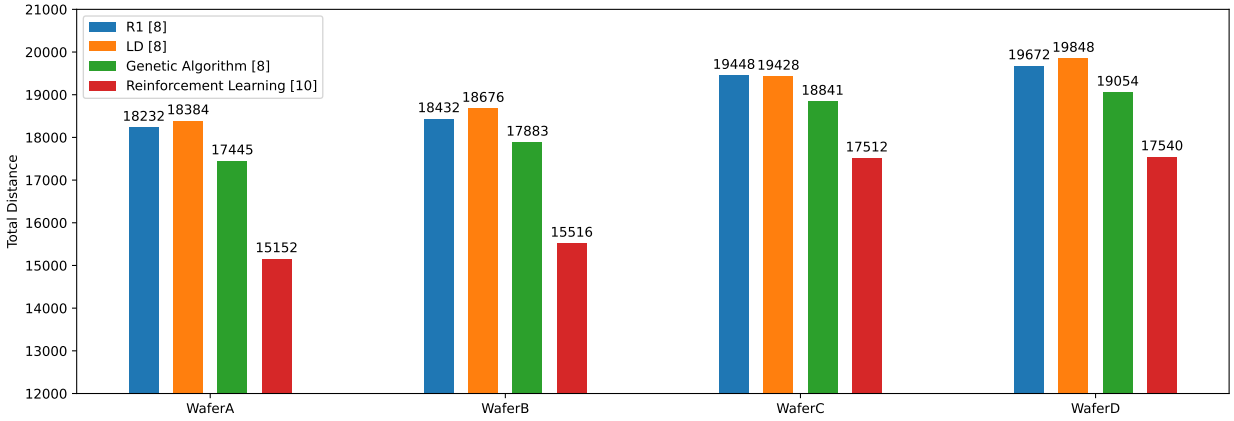


Figure 7: Total distance of the state-of-the-art methods for the four test cases. As the eight rules considered have similar results, only the results of two of them are shown.

- To point out its limitations, which will be addressed in the Subsection 2.8 by a new ILP model proposed in the present work, which will be used to obtain the optimal solutions for the test cases under consideration.
- To show how the representation of solutions used in their genetic algorithm and their criterion for discarding the generated individuals were determined. This will clarify the importance of the model proposed in our work (in Subsection 2.5).

For this BIP formulation, the authors assume that the number of good chips in the *wafer* (N) is equal to the number of slots in the *strip*. Furthermore, the index i identifies the chips ($i \in \{1, 2, \ldots, N\}$) and the index j identifies the slots of the *strip* ($j \in \{1, 2, \ldots, N\}$). Thus, the decision variables in this model are:

1. x_{ij}: binary variable that is equal to 1 if, as part of the path, the robotic arm goes from chip i of the *wafer* to slot j of the *strip*; and 0 otherwise.
2. y_{ji}: binary variable that is equal to 1 if, as part of the path, the robotic arm goes from slot j of the *strip* to pick up the next chip i of the *waffer*; and 0 otherwise.

The following distances are also defined:

1. d_{ij}^{ws}: distance from the chip location i in the *wafer* to the slot location j in the *strip*.
2. d_{ji}^{sw}: distance from the slot location j in the *strip* to the chip location i in the *wafer*.
3. d_{si}: distance from the origin s to the location of the chip i in the *wafer*.
4. d_{js}: distance from the location of the slot j in the *strip* to the origin s.

Based on these definitions, the model proposed in [8] is:

$$min \sum_{i=1}^{N} d_{si}x_{si} + \sum_{j=1}^{N} d_{js}y_{js} + \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^{ws}x_{ij} + \sum_{j=1}^{N}\sum_{i=1}^{N} d_{ji}^{sw}y_{ji}$$

Subject to the following constraints:

$$\sum_{i=1}^{N} x_{ij} = 1 \quad \forall j \in \{1, \ldots, N\} \tag{1}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \quad \forall i \in \{1, \ldots, N\} \tag{2}$$

$$x_{si} + \sum_{j=1}^{N} y_{ji} = 1 \quad \forall i \in \{1, \ldots, N\} \tag{3}$$

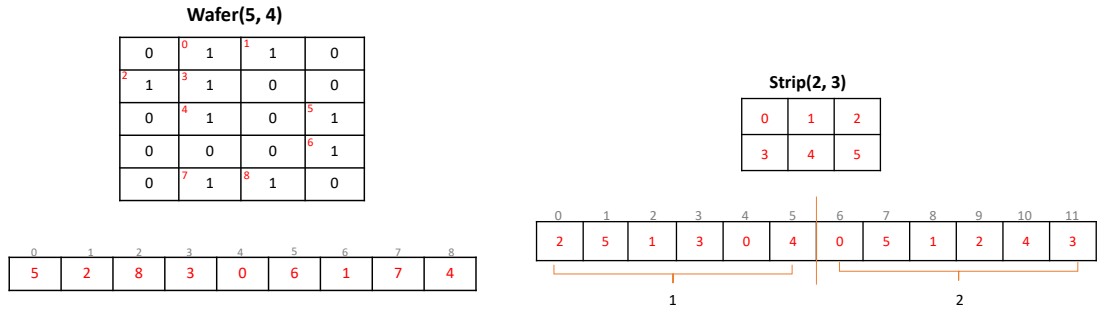
$$y_{js} + \sum_{i=1}^{N} y_{ji} = 1 \quad \forall j \in \{1, \ldots, N\} \tag{4}$$

$$\sum_{i=1}^{N} x_{si} + \sum_{j=1}^{N} y_{js} = 2 \quad \forall i, j \in \{1, \ldots, N\} \tag{5}$$

The constraint set (1) indicates that slot j is reached from a single chip. The constraint set (2) provides that the chip i be deposited in a single slot. The constraint set (3) establishes that the chip i can only be reached from the origin or a single slot of the *strip*. The constraint set (4) indicates that from slot j, it returns to the origin or goes to the position of a single chip in the *wafer*. Finally, the constraint set (5) establishes that the robotic arm must go from the origin to the location of the first chip to be picked up and return to the origin after placing the last chip in the corresponding slot.

Based on the above, it is essential to point out that this model does not formulate a constraint set for the formation of cycles, so it is incomplete, obtaining, in any case, a lower bound for the distance of the optimal route. Furthermore, the four test cases considered do not meet the initial assumption since the number of chips in the *wafer* is greater than the number of slots in a *strip*.

In any case, in [8], this BIP model has not been used to obtain solutions; however, it has served as the basis for the implementation of a genetic algorithm that determines the values of the decision variables x_{ij}



(a) Representation of the *wafer* chip selection sequence (W_{seq}).

(b) Representation of the slot allocation sequence (S_{seq}) for each of the $c = \lceil N/M \rceil$ *strips* needed.

Figure 8: Representation of a solution through vectors relative to the *wafer* (W_{seq}) and the *strip* (S_{seq}). In this example, the robotic arm starts from the origin and goes to chip position 5 (in red) of the *wafer* (since $W_{seq}[0] = 5$). Then, it goes to slot position 2 (in red) of the *strip* (since $S_{seq}[0] = 2$) to place that chip there. After that, it goes to chip 2 of the *wafer* ($W_{seq}[1] = 2$) and places it in slot 5 of the *strip* ($S_{seq}[1] = 5$), and so on. After placing chip 6 ($W_{seq}[5] = 6$) in slot 4 of the *strip* ($S_{seq}[5] = 4$), it becomes full, so it is exchanged for another empty strip, and the process continues. Finally, having placed chip 4 of the *wafer* ($W_{seq}[8] = 4$) in slot 1 of the *strip* ($S_{seq}[8] = 1$), the robotic arm returns to the origin and the process ends. Therefore, in this example, the last three values of S_{seq} indicate the empty positions of the second *strip*.

(matrix X) and y_{ji} (matrix Y). Therefore, the matrices X and Y represent an individual in the population, and the constraints of the BIP model were taken into account to filter the new individuals generated through recombinations and mutations. It is essential to mention that the number of iterations (generations) is equal to 100, and the population size is equal to 200 [8]. Therefore, the number of evaluations of the objective function in the mentioned work is 20000.

The results for this method, shown in Figure 7, are not too far from the heuristic rules. However, before considering different combinations of parameters of the genetic algorithm to improve results, we were interested, in our work, in evaluating the usefulness of the representation used in [8] since a large part of the individuals could represent non-feasible solutions.

2.5 Proposed representation of a solution

In Subsection 2.1), the sequences $W(k)$ and $S(k)$ ($k \in \{1, 2, ..., N\}$) were defined, which indicate the chip collection sequence and the slot allocation sequence, respectively. Thus, $W(k)$ and $S(k)$ determine the path of the robotic arm. To present the algorithms proposed in this work, it is convenient to previously establish specific definitions regarding the representation used in them, which results from slight modifications to $W(k)$ and $S(k)$.

Suppose a *wafer* of dimensions (nf_w, nc_w) containing N good chips, which are numbered from 0 to $N-1$ following an order from left to right and from top to bottom. In the upper part of Figure 8(a) an example is presented for a *wafer* of $nf_w = 5$ rows, $nc_w = 4$ columns and $N = 9$ good chips. The selection sequence of the *wafer* chips can be represented by a vector W_{seq} of N elements, where the indices correspond to the order of chip selection and the elements in each position correspond to the selected *wafer* chips. In the example, the element $W_{seq}[2] = 8$ indicates that chip number 8 will be the third chip to be picked up by the robotic arm during its journey. Note that this representation is a permutation of the set of chip identifiers, which is illustrated at the bottom of Figure 8(a).

On the other hand, suppose a *strip* of dimensions (nf_s, nc_s) with $M = nf_s * nc_s$ slots, which are numbered from 0 to $M-1$ in order from left to right and top to bottom. In the upper part of Figure 8(b), an example is presented for a *strip* of $nf_s = 2$ rows and $nc_s = 3$ columns. It is important to remember that the number of good chips N can be greater than the number of slots M in a *strip*, so an amount $c = \lceil N/M \rceil$ of *strips* will be needed to locate all the chips. The allocation sequence of each slot in the path of the robotic arm can be represented by a vector S_{seq} of $c * M$ elements, where the indices represent the slot selection order and the elements at each position correspond to the selected *strip* slots. In the example we have that $c = \lceil 9/6 \rceil = 2$, so S_{seq} will have $c * M = 2 * 6 = 12$ elements. The first portion of S_{seq} (0 to $M-1$) will correspond to the slots of the first *strip*, while the remaining portion (M to $2M-1$) will correspond to the slots of the second *strip*. Thus, the element $S_{seq}[7] = 5$ indicates that the eighth chip picked up by the robotic arm will be placed in slot 5 of the second *strip*. Note that this representation consists of c permutations of the set of slot

identifiers (one permutation for each *strip*), as indicated at the bottom of Figure 8(b) for the given example. Furthermore, since $c * M \geq N$, only the first N elements of S_{seq} will be considered for the calculation of the total distance TD (the rest will correspond to empty slots in the last *strip*).

According to these definitions and characteristics of W_{seq} and S_{seq}, it can be affirmed that they will always represent feasible solutions for the problem considered, which implies an essential advantage concerning the proposal in [8].

2.6 Algorithms proposed in this work - Phase 1

Considering the analysis in Subsection 2.4, in the first phase, three methods are proposed to obtain approximate solutions for this problem.

The first method (Algorithm 1) consists of generating, through random permutations, the vectors W_{seq} and S_{seq}. Considering that in [8] 20000[1] evaluations of the objective function (total distance) were performed during the execution of its genetic algorithm for each case test, we have set this same quantity for the random generation of solutions for a fair comparison. The Fisher–Yates [11] algorithm is used to generate each permutation.

Algorithm 1: Random Method

Input : N, M, n_{iters}, location of slots and chips
Output: Best randomly generated solution (vectors W_{seq}^{best} and S_{seq}^{best})

$c = \lceil N/M \rceil$; `#Number of strips required`
$TD_{min} = \infty$;
for $i \leftarrow 1$ **to** n_{iters} **do**
  Generate a random permutation (of $\{0, \ldots, N-1\}$) for W_{seq};
  $S_{seq} = \emptyset$;
  for $j \leftarrow 1$ **to** c **do**
    Generate a random permutation (of $\{0, \ldots, M-1\}$) and add to the end of S_{seq};
  end
  Leave in S_{seq} only its first N elements;
  Calculate TD from W_{seq} and S_{seq};
  if $TD_{min} > TD$ **then**
    $W_{seq}^{best} \leftarrow W_{seq}$;
    $S_{seq}^{best} \leftarrow S_{seq}$;
    $TD_{min} = TD$;
  end
end
return W_{seq}^{best} *and* S_{seq}^{best}

The second proposed method consists of a local search (Algorithm 2), which starts with a randomly generated solution and has two stages in each process iteration. In the first stage, a local search is performed for each of the c blocks in S_{seq} (each block corresponds to a *strip*): each pair of elements at positions i and j is considered, and these elements are interchanged if this reduces the total distance. Each pair of elements in positions i and j of W_{seq} is considered in the second stage. In addition to swapping the values at those positions if it reduces the total distance, it also sets a flag indicating that a swap has been made. If the flag is active at the end of this second stage, both stages are repeated in the following iteration.

Finally, the third method is a greedy algorithm (Algorithm 3), which is based on a technique from the literature for the multi-robot coordination problem in *pick-and-place* tasks [12]. For the problem considered in this work, in each step of the proposed greedy algorithm, the closest chip to the position of the robotic arm is collected (from the *wafer*). Once there, it is deposited in the nearest available slot to the position of the robotic arm. Based on this description, it can be seen that this is a deterministic method.

2.7 Algorithms proposed in this work - Phase 2

In the second phase, different combinations of parameters are considered for a genetic algorithm based on the representation by permutation proposed in the present work (Algorithm 4). Unlike [8], this would ensure that feasible solutions are generated during the process since multiple recombination and mutation methods exist in the literature for this type of representation.

[1] This amount was arbitrarily determined in the reference work [8].

Algorithm 2: LocalSearchMethod

Input : N, M, location of slots and chips
Output: Solution resulting from the local search (vectors W_{seq} and S_{seq})

$c = \lceil N/M \rceil$; `#Number of strips required`
Generate W_{seq} and S_{seq} randomly; $TD_{min} = TD$ from W_{seq} and S_{seq} ;
repeat
  for $l \leftarrow 1$ **to** c **do**
    repeat
      $change_S = False$;
      foreach *positions* i, j *in block* l *of* S_{seq} **do**
        Swap the elements in i and j, generating $S^{'}_{seq}$;
        Calculate TD from W_{seq} and $S^{'}_{seq}$;
        if $TD_{min} > TD$ **then**
          $S_{seq} \leftarrow S^{'}_{seq}$;
          $TD_{min} = TD$;
          $change_S = True$;
        end
      end
    until $change_S == False$;
  end
  $change_W = False$;
  foreach *positions* i, j *of* W_{seq} **do**
    Swap the elements in i and j, generating $W^{'}_{seq}$;
    Calculate TD from $W^{'}_{seq}$ and S_{seq};
    if $TD_{min} > TD$ **then**
      $W_{seq} \leftarrow W^{'}_{seq}$;
      $TD_{min} = TD$;
      $change_W = True$;
    end
  end
until $change_W == False$;
return W_{seq} *and* S_{seq}

Algorithm 3: Greedy Method

Input : N, M, slot and chip location
Output: Resulting solution of the greedy algorithm (vectors W_{seq} and S_{seq})

$c = \lceil N/M \rceil$; `#Number of strips required`
$p_{br} = p_{orig}$; `#Initial position of robotic arm`
for $i \leftarrow 0$ **to** $N - 1$ **do**
  if i mod $M == 0$ **then**
    Place an empty *strip*; `#All slots become available`
  end
  Let x be the *chip*, still in the *wafer*, closest to p_{br};
  $W_{seq}[i] = x$;
  $p_{br} = p_x$;
  Let y be the closest empty slot to p_{br} ;
  $S_{seq}[i] = y$;
  $p_{br} = p_y$;
end
return W_{seq} *and* S_{seq}

Algorithm 4: Genetic algorithm

Input : N, M, n_{iters}, pop_{size} (even), $elit$, slot and chip location
Output: Best solution generated during the process (vectors W_{seq}^{best} and S_{seq}^{best})

```
Generate a population of pop_size individuals randomly (similar to Algorithm 1);
Evaluate with the objective function (TD) each one of the generated individuals;
Determine W_seq^best and S_seq^best in the initial population;
for i ← 1 to n_iters do
    for i ← 1 to pop_size/2 do
        Apply binary tournament to select parents;
        Apply a recombination method to generate two children;
        Apply a mutation method to both children;
    end
    Evaluate with the objective function (TD) each one of the generated children;
    Considering the percentage of elitism elit, replace the current population with the generated
     children;
    Update the solutions W_seq^best and S_seq^best, if applicable;
end
return W_seq^best and S_seq^best
```

Four variants were considered for the application of the recombination and mutation methods for the proposed genetic algorithm:

- AG_1: Recombination and mutation apply only to W_{seq}.
- AG_2: Recombination and mutation apply only to S_{seq}.
- AG_3: Recombination and mutation apply to both W_{seq} and S_{seq}.
- AG_4: Recombination and mutation apply only to W_{seq}, or only S_{seq}, or both. This is randomly selected in each generation of individuals, with the same probability for all three options.

The crossover operators considered were: Partially Mapped Crossover (PMX), Order Crossover (OX), and Cycle Crossover (CX) [13]. The mutation operators considered were: Swap, Insert, Scramble, and Inversion [13]. In addition, mutation rates of 10%, 20% and 30% were considered; and elitism percentages of 0, 5% and 10%.

It is essential to clarify that since S_{seq} consists of c blocks of permutations of the set of slot identifiers (one permutation for each *strip*), the recombination and mutation operators apply separately to each of the c blocks of S_{seq}.

In preliminary tests (see the Appendix for more details) the four variants of the genetic algorithm were considered for the four test cases, with the combination of the indicated recombination and mutation operators, as well as the different mutation rates and elitism percentages. For each combination, 30 executions were carried out, comparing the average values of the total distance of the generated solutions.

The preliminary results determined that AG_3 obtains, on average, the best solutions for the cases considered, with the parameters of Table 2. It is essential to point out that, for a fair comparison, the number of generations and the population size are the same as in [8]. These parameters were used to obtain approximate solutions for the test cases considered, and the average of results (total distances) in 30 executions are reported in Subsection 3.2.

2.8 Obtaining exact solutions using ILP

When considering all the algorithms that seek to minimize the total distance traveled by the robotic arm, the following question arises: How close are the obtained results to the optimal distance for the considered test case? It is necessary to get this optimal distance, which is why an integer linear programming (ILP) model is proposed in the present work. Furthermore, this model generalizes and considers additional constraints concerning the BIP model from a previous work [8], exposed in the Subsection 2.4. Unlike the BIP in [8], with this new ILP model, we will obtain the exact solutions for the four cases used for the experiments. Therefore, we will compare the optimal distances for those test cases with those generated by the approximate methods proposed in our work.

The problem considered in this paper is of the *pick-and-place* type, so it can be modeled as a traveling salesman problem (TSP), where the starting position of the robotic arm, each good chip in the *Wafer* and

Table 2: Description of the parameters of the genetic algorithm (AG_3).

Representation	Permutations (W_{seq} and S_{seq})
Initialization	Random
Recombination	PMX
Mutation	Swap
Mutation Rate	30%
Parent Selection	Binary Tournament
Selection of survivors	Generational
Elitism Percentage	10%
Population Size	200
Number of generations	100
Number of children	200

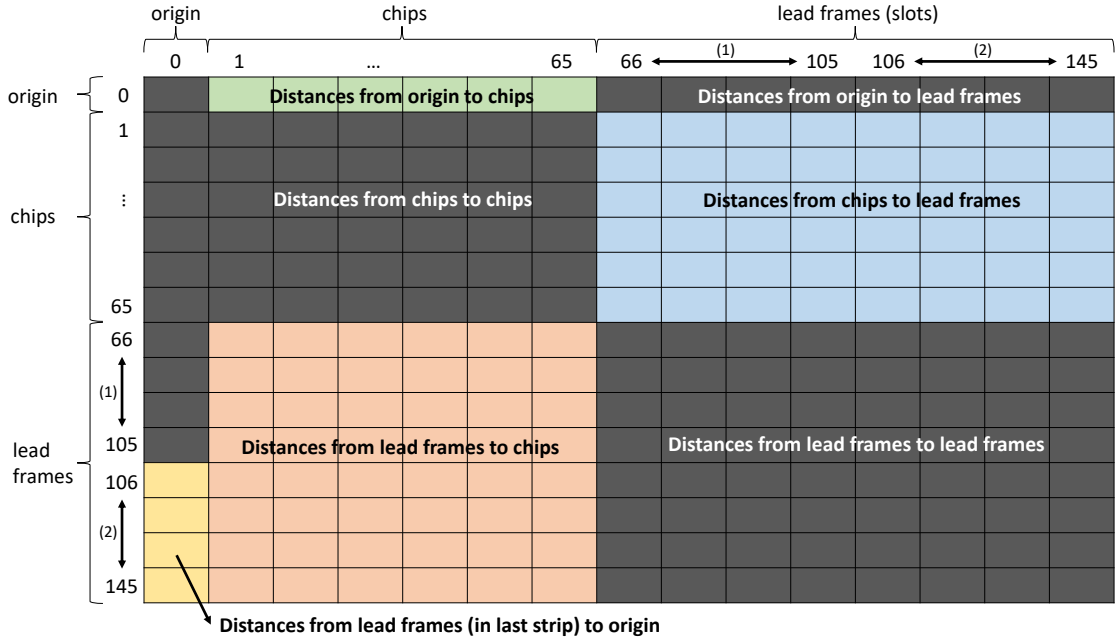


Figure 9: Example of distance matrix with $N = 65$ chips, $M = 40$ slots and $c = 2$ *strips* (needed to allocate all chips). The shaded regions indicate not allowed movements (due to model constraints). This example matrix represents the test cases *WaferA* and *WaferB*, although the same principle could be applied for a general case (given by the values of N and M).

each slot in the c *Strips* represent the cities. In this way, adding constraints for the movements not allowed, the ILP model can be formulated as a modification of the TSP, considering a matrix of distances D that represents the distances associated with each possible movement.

To generate this matrix of distances D, the following points are considered in order:

- The origin (row 0).
- The positions of the N good chips in the *Wafer* (rows 1 to N).
- The positions of the M slots of each of the c *Strips* needed to locate all the chips (rows $N + 1$ to $N + c * M$).

Thus, d_{ij} in D represents the distance from position i to position j. Figure 9 shows an example of a distance matrix defined in this way, considering $N = 65$, $M = 40$, and $c = 2$.

Regarding the constraints, it should be noted that:

- For the origin:
  - From there, the tour (of the robotic arm) starts.
  - It exits only once, and from there, the robot should go to a good *wafer* chip.

  - Must be visited once, from a slot of the last *strip* employed.

- For the *wafer* chips:
  - Each chip is reached and left only once.
  - From a chip, the robot arm can only go to one slot. Or, it can't go to the origin or another chip.
- For the slots of the *strips*:
  - For each slot of *strip* (except the last one), it must be arrived and exited once.
  - The slots of the last *strip* can be visited or not. If a slot in the last *strip* is visited, it must also be exited.
  - From a slot, the robot cannot visit another slot.

The number of vertices of the graph $n = 1 + N + c * M$ is considered for the proposed mathematical model. The decision variable x_{ij} indicates whether the robotic arm moves from the element represented by row i ($i = 0, 1, \ldots, n-1$) to the element represented by column j ($j = 0, 1, \ldots, n-1$), being equal to 1 if said movement is made and 0 otherwise. It is also introduced (as input data) $b_i \in \{0, \ldots, c-1\}$, which indicates the *strip* to which the vertex i belongs (if the vertex corresponds to the origin or a chip, $b_i = -1$).

Based on these considerations, the objective function is defined as:

$$min \sum_{j=0}^{n-1} \sum_{i=0}^{n-1} d_{ij} x_{ij}$$

Subject to the following constraints:

$$\sum_{j=0}^{n-1} x_{0j} = 1 \tag{1}$$

$$\sum_{i=0}^{n-1} x_{i0} = 1 \tag{2}$$

$$\sum_{j=1}^{N} x_{0j} = 1 \tag{3}$$

$$\sum_{i=n-c*M}^{n-1} x_{i0} = 1 \tag{4}$$

$$\sum_{j=0}^{n-1} x_{ij} = 1 \quad \forall i \in \{1, \ldots, N\} \tag{5}$$

$$\sum_{i=0}^{n-1} x_{ij} = 1 \quad \forall j \in \{1, \ldots, N\} \tag{6}$$

$$x_{ij} = 0 \quad \forall i, j \in \{1, \ldots, N\} \tag{7}$$

$$x_{i0} = 0 \quad \forall i \in \{1, \ldots, N\} \tag{8}$$

$$\sum_{j=0}^{n-1} x_{ij} = 1 \quad \forall i \in \{N+1, \ldots, n-M-1\} \tag{9}$$

$$\sum_{i=0}^{n-1} x_{ij} = 1 \quad \forall j \in \{N+1, \ldots, n-M-1\} \tag{10}$$

$$\sum_{i=0}^{n-1} x_{ij} \leq 1 \quad \forall j \in \{n-M, \ldots, n-1\} \tag{11}$$

$$\sum_{j=0}^{n-1} x_{ij} = \sum_{j=0}^{n-1} x_{ji} \quad \forall i \in \{n-M, \ldots, n-1\} \tag{12}$$

$$x_{ij} = 0 \quad \forall i, j \in \{N+1, \ldots, n-1\} \tag{13}$$

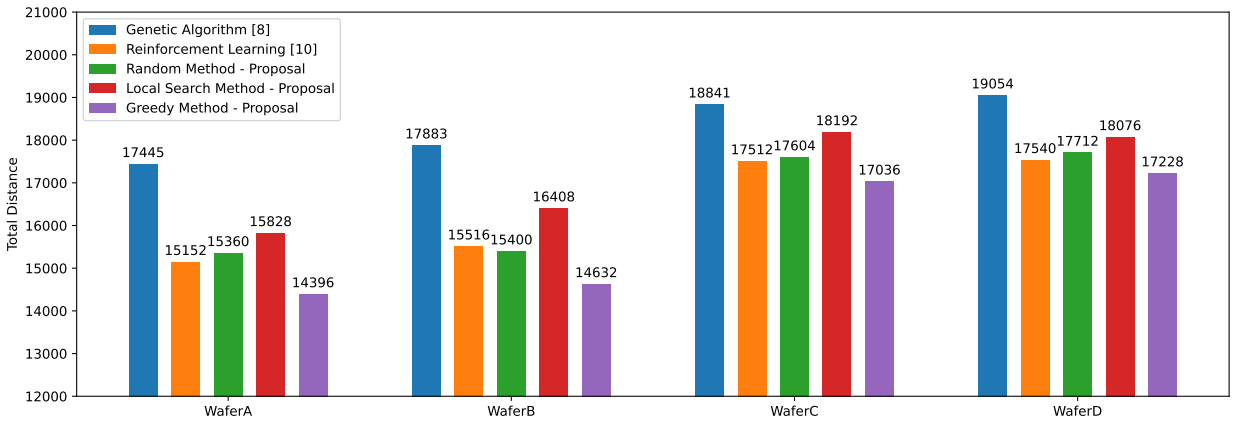


Figure 10: Comparison of the total distance obtained by the methods proposed in Phase 1 of this work and those of the state of the art. for the four test cases. Due to its low relative performance (see Figure 7), the results of the heuristic rules are not shown.

$$u_i - u_j + N * 2 * x_{ij} \leq N * 2 - 2 \quad \forall i, j \in \{0, \ldots, n-1\} \mid i \neq j \wedge i \neq 0 \wedge j \neq 0 \tag{14}$$

$$u_i \leq N * 2 - 2 \quad \forall i \in \{0, \ldots, n-1\} \mid i \neq 0 \tag{15}$$

$$b_i < b_j \rightarrow u_i + 2 \leq u_j \quad \forall i, j \in \{N+1, \ldots, n-1\} \tag{16}$$

The constraints (1)-(4) are relative to the origin. The constraint set (1) indicates that the robotic arm departs from the origin exactly once, and the constraint set (2) that the origin is reached exactly once. The constraint set (3) establishes that a chip is visited from the origin, while the (4) that one returns to the origin from some slot of the last *strip*.

The constraints (5)-(8) are relative to the *wafer*. The constraint set (5) indicates that each chip is exited exactly once, and the constraint set (6) that each chip is reached exactly once. The constraint set (7) states that a chip cannot be visited from another chip, while the constraint set (8) indicates that from a chip, it cannot go to the origin. Thus, these last two constraint sets ensure that from a chip, it goes to some slot of some *strip*.

The constraints (9)-(13) are relative to the *strips*. The constraint set (9) indicates that each slot of each *strip* (except the last) is exited exactly once, and the constraint set (10) that each of these is reached exactly once. The constraint sets (11) and (12) are similar and refer to the slots of the last *strip*, which can be visited up to once (and if they are visited, they must be exited). The constraint set (13) establishes that from one slot, another slot cannot be visited, so it should go to a chip or the origin (for the latter case, for slots of the last *strip*).

The constraints (14) and (15) represent the constraints that prevent the formation of subcycles, which were adapted from the Miller-Tucker-Zemlin [14] model for the elimination of subcycles (subtours). Finally, the constraint set 16) establishes that each *strip* must be filled before moving on to the next. It is essential to highlight that the constraint sets (13)-(16) were not considered in the BIP model of a previous work [8].

The proposed ILP model was implemented in CPLEX 20.12 to obtain the exact solutions of the four test cases considered in our work.

3 Results and discussion

3.1 Results - Phase 1 Algorithms

Figure 10 shows the results of each of the three methods proposed in the first phase of this work for the four test cases considered, comparing them with the results of the state-of-the-art methods.

Observing the results obtained by the genetic algorithm [8] and the random generation method, the latter achieves better outcomes for the same number of solutions considered during the genetic algorithm process (20000). Moreover, the improvement is substantial for each of the test cases: 12% for the *WaferA*, 13.9% for the *WaferB*, 6.6% for the *WaferC* and 7% for the *WaferD*. This result is remarkable since a genetic algorithm is expected to obtain better results than a method of random generation of solutions.

At this point, it is convenient to remember that in [8], binary matrices (X and Y) represent a solution that must comply with certain constraints for its feasibility. In addition, their work mentions that after generating

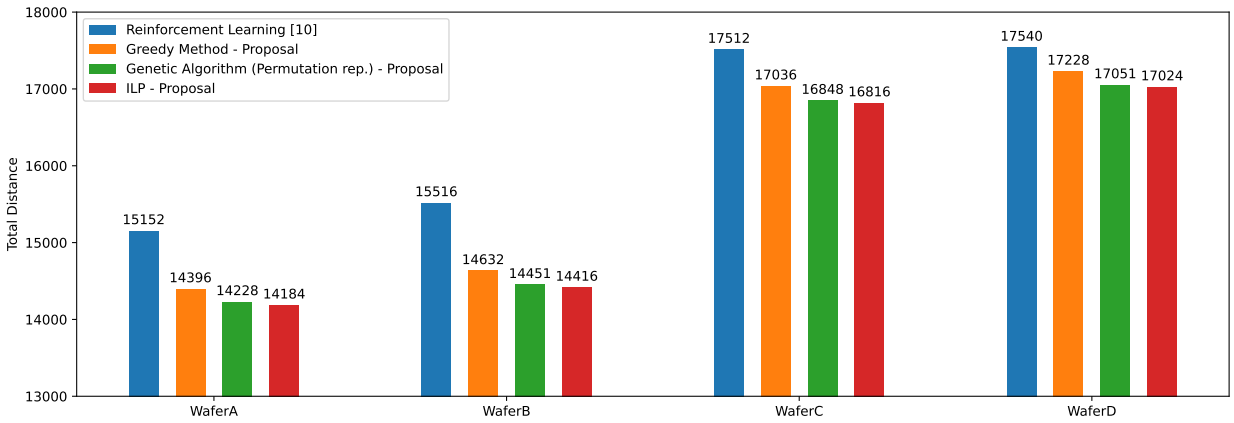


Figure 11: Comparison of the total distance obtained by the genetic algorithm proposed in Phase 2, the greedy method proposed in Phase 1, and the reinforcement learning method [10]. The optimal results obtained through the proposed ILP model are also shown.

new individuals (solutions) through recombination, it must be verified that they satisfy the constraints of the BIP model, ignoring those that do not comply with them. Therefore, the results of the tests in our work indicate that the percentage of non-feasible solutions generated (and hence discarded) during the execution of the genetic algorithm is considerably high. This inefficiency in developing feasible solutions limits the quality of the results obtained using this method.

It is essential to mention that [8] indicates that the genetic algorithm obtained better results than a random method that also generates 20000 solutions. For the same reasons exposed regarding the representation used, a significant percentage of these randomly generated solutions would be discarded for not representing feasible solutions. In our method, the permutation representation and the generation of random permutations ensure that all generated solutions are feasible.

On the other hand, Figure 10 shows that the results of the greedy method proposed in this work are better than those obtained by the reinforcement learning method [10] in the four test cases considered. The upgrade is 5% for the *WaferA*, 5.7% for the *WaferB*, 2.7% for the *WaferC* and 1.8% for the *WaferD*. Therefore, we can affirm that this work improves the results of the state-of-the-art methods for the test cases considered in previous works on this problem [8, 10].

It can be noted that, both in the improvement of the random method (concerning the genetic algorithm) and the greedy method (concerning the machine learning method), *WaferC* and *WaferD* have lower percentages of improvement than *WaferA* and *WaferB*. The difference between these case pairs is in the rate of defective chips; thus, the higher this percentage, the greater the opportunity for improvement.

Regarding the local search method, it is fundamental to point out that the results presented are the average of 30 runs. It can be seen that it improves the outcomes of the [8] genetic algorithm. However, the total distance values obtained are greater than those determined by the other methods shown in Figure 10. This is due to multiple local minima for the problem addressed, so the quality of the solution obtained by this method is highly dependent on the randomly generated initial solution.

3.2 Results - Phase 2 algorithms and proposed ILP

Figure 11 shows the results obtained by the genetic algorithm proposed and selected in Phase 2 and the optimal total distances for the test cases considered, comparing them with the results of the best state-of-the-art method and with the best algorithm proposed in Phase 1 of this work (greedy approach). It is necessary to highlight that the result of the genetic algorithm for each test case corresponds to an average of 30 executions.

As can be seen, the results obtained by the genetic algorithm (on average) are considerably better than those obtained by the reinforcement learning method [10] in the four test cases used. The upgrade is 6.1% for the *WaferA*, 6.9% for the *WaferB*, 3.8% for the *WaferC* and 2.8% for the *WaferD*. It can also be observed that the proposed genetic algorithm obtains better results than the greedy method proposed in Phase 1.

The results obtained by the genetic algorithm are very close to the optimal values determined by the ILP model proposed in this work. As seen in Table 3, the genetic algorithm obtains total distances that do not exceed the optimal value by more than 0.31% in the individual test cases. At the level of percentages, the difference with the reference method in the literature [10] is even more noticeable.

Table 3: Percentage increase of the optimal total distance (TD_{opt}) for each test case, obtained by the proposed ILP, concerning the reference state-of-the-art method and the best two algorithms proposed in this work.

Test case	TD_{opt} - ILP	Reinf. Learn. [10]	Greedy	Gen. Alg.
WaferA	14184	6.82%	1.49%	0.31%
WaferB	14416	7.63%	1.50%	0.24%
WaferC	16816	4.14%	1.31%	0.19%
WaferD	17024	3.03%	1.20%	0.16%

3.3 Execution environment and running times

Although the running time was not the central interest of this work, this subsection provides some relevant data in this scope. According to its authors, the reinforcement learning method [10] takes *less than a second*, but the execution environment is not mentioned. Neither the source code nor the executable for reproducing results is provided.

On the other hand, in [8], neither the execution times nor the environment used for testing is provided. Moreover, neither the source code nor the executable for reproducing results is given. However, the total number of evaluations of the objective function is 20000. Therefore, for a fair comparison, this total number was kept for the genetic algorithm proposed in Phase 2 of this work, as well as in the random method used in Phase 1.

The specifications of the environment used for our tests are: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 16GB RAM, and Windows 10 Home. The Phase 1 algorithms were implemented in Java and the genetic algorithm in Python 3. For each test case, the random method (Algorithm 1), the local search method (Algorithm 2), and the greedy method (Algorithm 3) were executed in less than one second; while the genetic algorithm (Algorithm 4) were executed in less than 5 seconds.

The proposed ILP model was implemented in CPLEX 20.1[2] for obtaining the exact solutions of the considered test cases. For *WaferA* and *WaferB*, the running time was one hour, while for *WaferC* and *WaferD*, it was 24 hours.

4 Conclusions

In this work, a combinatorial optimization problem was considered, where the aim is to reduce the total distance of a robotic arm for the collection of semiconductor chips in a *Wafer* and their placement in the slots of a *Strip*. This problem is crucial for the productivity of the *back-end* stage of the IC manufacturing process.

Because this problem is NP-hard, various heuristics and metaheuristics are proposed in the literature. In the first part, a genetic algorithm was analyzed. From this, a different representation of solutions (based on permutations) was proposed, which, together with a random method, produced better results for the test cases considered. This reveals that the results obtained by the genetic algorithm are limited by the original representation used and the selected recombination method.

In the second part, a greedy method for obtaining solutions was proposed, in which in each step, the closest *Wafer* chip is searched for and placed in the nearest available slot of the *Strip*. The experiments indicate that this proposed method obtains lower total distances than the best state-of-the-art method for this problem, considering four test cases used in previous works.

In the third part, a genetic algorithm was proposed that uses a representation based on permutations, which ensures that the solutions generated in the process are feasible. The results of this genetic algorithm are even better than those obtained by the greedy method. A new ILP model was also proposed to get optimal solutions, whose results show that the values obtained by the proposed genetic algorithm are very close to the optimum for the considered test cases, with a marked reduction in execution time.

The following future works are proposed:

- Consider other parameters for the problem and establish different criteria for generating larger and more diverse test cases. The behavior of the proposed algorithm in this new context should be analyzed, in which case a method could be proposed to determine a lower bound in polynomial time (for the time necessary to solve the proposed ILP model).

- Analyze the performance of other local search-based methods, such as tabu search or simulated annealing.

[2] `https://www.ibm.com/products/ilog-cplex-optimization-studio`

- Analyze the problem with multiple *wafers* and different mechanisms for collecting and placing chips.

All the implementations, test cases, and additional information about this work are publicly available at the following link: `https://doi.org/10.6084/m9.figshare.23995878`.

References

[1] Y. Hao, S. Xiang, G. Han, J. Zhang, X. Ma, Z. Zhu, X. Guo, Y. Zhang, Y. Han, Z. Song *et al.*, "Recent progress of integrated circuits and optoelectronic chips," *Science China Information Sciences*, vol. 64, no. 10, pp. 1–33, 2021.

[2] D. Kim and J. VerWey, "The potential impacts of the made in china 2025 roadmap on the integrated circuit industries in the us, eu and japan," *SSRN Electronic Journal*, 2019.

[3] X. Zhou, H. Chen, J. Chai, S. Wang, and B. Lev, "Performance evaluation and prediction of the integrated circuit industry in china: A hybrid method," *Socio-Economic Planning Sciences*, vol. 69, p. 100712, 2020.

[4] A. I. Sivakumar, "Optimization of a cycle time and utilization in semiconductor test manufacturing using simulation based, on-line, near-real-time scheduling system," in *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, 1999, pp. 727–735.

[5] C.-Y. Hsu, "Clustering ensemble for identifying defective wafer bin map in semiconductor manufacturing," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[6] H. Zhang, Z. Jiang, and C. Guo, "Simulation-based optimization of dispatching rules for semiconductor wafer fabrication system scheduling by the response surface methodology," *The International Journal of Advanced Manufacturing Technology*, vol. 41, no. 1, pp. 110–121, 2009.

[7] M. Fu, R. Askin, J. Fowler, M. Haghnevis, N. Keng, J. S. Pettinato, and M. Zhang, "Batch production scheduling for semiconductor back-end operations," *IEEE Transactions on semiconductor manufacturing*, vol. 24, no. 2, pp. 249–260, 2011.

[8] Y.-J. Park, G. Ahn, and S. Hur, "Optimization of pick-and-place in die attach process using a genetic algorithm," *Applied Soft Computing*, vol. 68, pp. 856–865, 2018.

[9] A. Srivastav, H. Schroeter, and C. Michel, "Approximation algorithms for pick-and-place robots," *Annals of Operations Research*, vol. 107, no. 1, pp. 321–338, 2001.

[10] G. Ahn, M. Park, Y.-J. Park, and S. Hur, "Interactive q-learning approach for pick-and-place optimization of the die attach process in the semiconductor industry," *Mathematical Problems in Engineering*, vol. 2019, 2019.

[11] R. A. Fisher and F. Yates, *Statistical tables for biological, agricultural and medical research.* Hafner Publishing Company, 1953.

[12] Y. Huang, R. Chiba, T. Arai, T. Ueyama, and J. Ota, "Robust multi-robot coordination in pick-and-place tasks based on part-dispatching rules," *Robotics and Autonomous Systems*, vol. 64, pp. 70–83, 2015.

[13] A. E. Eiben, J. E. Smith *et al.*, *Introduction to evolutionary computing.* Springer, 2003, vol. 53.

[14] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

Table 4: Summary of the preliminary results for parameter selection, considering the best average distances obtained for each of the test cases (30 executions) for each variant of the genetic algorithm.

Caso	Variante - GA	Elitismo (%)	Tasa de mutación (%)	Mutación	Cruzamiento	Distancia Total
WaferA	AG_1	5	20	Insert	CX	15155.47
	AG_2	0	20	Swap	CX	14304.93
	AG_3	0	20	Insert	CX	14217.47
	AG_4	10	30	Swap	CX	14224
WaferB	AG_1	0	30	Swap	PMX	14481.2
	AG_2	10	20	Swap	PMX	14523.73
	AG_3	10	30	Swap	CX	14444.13
	AG_4	10	30	Swap	PMX	14445.2
WaferC	AG_1	10	30	Scramble	PMX	17278.53
	AG_2	10	30	Swap	PMX	16922.53
	AG_3	0	30	Insert	CX	16837.73
	AG_4	10	30	Swap	PMX	16840.53
WaferD	AG_1	10	30	Scramble	PMX	17499.73
	AG_2	5	10	Swap	PMX	17119.2
	AG_3	5	20	Inversion	CX	17032.2
	AG_4	5	30	Swap	CX	17048.13

Appendix

It is mentioned in Subsection 2.7 that different combinations of parameters for a genetic algorithm based on permutation representation were tested for each of the four test cases. The parameters considered were as follows:

1. Crossover operators: Partially Mapped Crossover (PMX), Order Crossover (OX), and Cycle Crossover (CX) [13].
2. Mutation operators: Swap, Insert, Scramble, and Inversion [13].
3. Mutation rates: 10%, 20% and 30%.
4. Elitism percentages: 0, 5% and 10%.

In addition, four variants for the application of recombination and mutation methods were analyzed:

- AG_1: Recombination and mutation are applied only to W_{seq}.
- AG_2: Recombination and mutation apply only to S_{seq}.
- AG_3: Recombination and mutation apply to both W_{seq} and S_{seq}.
- AG_4: Recombination and mutation apply only to W_{seq}, or only S_{seq}, or both. This is randomly selected in each generation of individuals, with the same probability for all three options.

Selection of parameters and variant for the genetic algorithm

As the number of combinations is high (4 crossover methods * 3 mutation methods * 3 mutation rates * 3 elitism percentages * 4 variants of the genetic algorithm = 432 combinations), only the best nine results are shown for each variant and test case.

Tables 5, 6, 7, and 8 show the results for *WaferA*, *WaferB*, *WaferC*, and *WaferD*, respectively. It is important to mention that each distance value represents an average of 30 executions.

Table 4 summarizes the best distances obtained for each test case and each variant of the genetic algorithm, with the corresponding parameters. It can be seen that the AG_3 variant obtained the smallest distance (on average) for each test case, being significantly[3] better than AG_1 and AG_2. However, there is no significant difference concerning AG_4.

From the results of Table 4, we selected the parameters that appear most often in it, thus obtaining those indicated in Table 2. Although there is a tie between PMX and CX, PMX was finally selected as the crossover operator.

[3]ANOVA and Tukey HSD (as *post hoc*) statistical tests were performed to identify which variants are significantly different from each other.

Table 5: Parameters with the best average distances obtained for *WaferA* (30 executions), sorted by variant of the genetic algorithm.

Variante - GA	Elitismo (%)	Tasa de mutación (%)	Mutación	Cruzamiento	Distancia Total
AG_1	5	20	Insert	CX	15155.47
	5	30	Insert	PMX	15158.4
	10	30	Swap	PMX	15164.4
	10	10	Swap	PMX	15175.47
	10	20	Inversion	OX	15177.6
	5	10	Swap	PMX	15179.6
	0	10	Insert	CX	15202.27
	0	20	Insert	CX	15203.87
	0	30	Swap	PMX	15208.4
AG_2	0	20	Swap	CX	14304.93
	10	30	Swap	PMX	14305.07
	5	30	Swap	PMX	14310.27
	0	30	Swap	CX	14313.33
	5	10	Swap	PMX	14313.87
	10	20	Swap	PMX	14313.87
	5	20	Swap	PMX	14314
	10	10	Swap	PMX	14322.53
	0	10	Swap	CX	14327.47
AG_3	0	20	Insert	CX	14217.47
	10	30	Swap	CX	14223.2
	5	30	Swap	CX	14224.53
	0	30	Insert	CX	14224.93
	10	10	Swap	PMX	14228.13
	10	20	Swap	CX	14229.2
	5	20	Swap	CX	14229.47
	0	10	Swap	CX	14232.8
	5	10	Swap	CX	14237.2
AG_4	10	30	Swap	CX	14224
	5	30	Swap	CX	14224.67
	10	20	Swap	PMX	14229.73
	0	20	Swap	CX	14232.8
	10	10	Swap	PMX	14232.93
	5	20	Swap	PMX	14233.87
	5	10	Swap	PMX	14238.27
	0	10	Swap	CX	14241.87
	0	30	Scramble	PMX	14441.2

Table 6: Parameters with the best average distances obtained for *WaferB* (30 executions), sorted by variant of the genetic algorithm.

Variante - GA	Elitismo (%)	Tasa de mutación (%)	Mutación	Cruzamiento	Distancia Total
AG_1	0	30	Swap	PMX	14481.2
	5	30	Swap	PMX	15360.13
	10	30	Inversion	PMX	15371.33
	5	20	Swap	OX	15378.8
	5	10	Scramble	PMX	15381.6
	10	10	Swap	CX	15384.67
	10	20	Scramble	CX	15394.27
	0	10	Swap	CX	15413.33
	0	20	Scramble	CX	15429.07
AG_2	10	20	Swap	PMX	14523.73
	10	10	Swap	OX	14524.93
	5	30	Swap	PMX	14527.47
	5	20	Swap	PMX	14528.53
	0	30	Swap	CX	14529.2
	10	30	Swap	CX	14529.87
	5	10	Inversion	PMX	14532
	0	20	Insert	CX	14536
	0	10	Swap	CX	14546
AG_3	10	30	Swap	CX	14444.13
	5	30	Swap	CX	14446.53
	0	30	Insert	CX	14446.67
	0	20	Insert	CX	14448.13
	10	20	Swap	PMX	14449.73
	5	20	Swap	CX	14452.8
	10	10	Swap	PMX	14456.53
	5	10	Swap	PMX	14456.67
	0	10	Swap	CX	14459.47
AG_4	10	30	Swap	PMX	14445.2
	5	30	Swap	CX	14449.33
	0	20	Swap	CX	14451.6
	10	20	Swap	PMX	14452.53
	5	20	Swap	PMX	14452.67
	0	30	Swap	CX	14453.73
	10	10	Swap	PMX	14459.73
	5	10	Swap	PMX	14460.13
	0	10	Swap	CX	14467.2

Table 7: Parameters with the best average distances obtained for *WaferC* (30 executions), sorted by variant of the genetic algorithm.

Variante - GA	Elitismo (%)	Tasa de mutación (%)	Mutación	Cruzamiento	Distancia Total
AG_1	10	30	Scramble	PMX	17278.53
	5	30	Swap	PMX	17295.33
	10	20	Insert	PMX	17304.4
	5	20	Scramble	PMX	17304.93
	10	10	Inversion	PMX	17310.67
	5	10	Scramble	PMX	17318.4
	0	30	Swap	CX	17333.47
	0	10	Insert	CX	17339.07
	0	20	Swap	PMX	17341.2
AG_2	10	30	Swap	PMX	16922.53
	5	30	Swap	PMX	16923.87
	10	10	Swap	PMX	16925.07
	5	20	Swap	PMX	16925.33
	10	20	Inversion	CX	16926
	0	30	Swap	CX	16927.47
	5	10	Swap	PMX	16928.67
	0	20	Swap	CX	16930.53
	0	10	Swap	CX	16930.93
AG_3	0	30	Insert	CX	16837.73
	0	20	Insert	CX	16839.73
	5	30	Swap	CX	16840.8
	10	30	Swap	CX	16841.47
	0	10	Swap	CX	16846.8
	10	20	Swap	PMX	16846.8
	10	10	Swap	CX	16851.87
	5	20	Swap	CX	16852
	5	10	Swap	CX	16852
AG_4	10	30	Swap	PMX	16840.53
	5	30	Swap	PMX	16840.93
	10	20	Swap	CX	16844.93
	0	20	Swap	CX	16847.2
	0	30	Insert	CX	16848.67
	5	20	Swap	PMX	16849.47
	5	10	Swap	PMX	16855.2
	10	10	Swap	PMX	16857.47
	0	10	Swap	CX	16863.47

Table 8: Parameters with the best average distances obtained for *WaferD* (30 executions), sorted by variant of the genetic algorithm.

Variante - GA	Elitismo (%)	Tasa de mutación (%)	Mutación	Cruzamiento	Distancia Total
AG_1	10	30	Scramble	PMX	17499.73
	10	20	Scramble	PMX	17499.87
	5	10	Scramble	PMX	17502.8
	5	20	Insert	PMX	17504.93
	5	30	Swap	CX	17508.93
	10	10	Scramble	PMX	17512.53
	0	10	Inversion	CX	17520.93
	0	20	Inversion	CX	17522.67
	0	30	Insert	CX	17526
AG_2	5	10	Swap	PMX	17119.2
	10	30	Swap	CX	17120
	0	30	Scramble	CX	17120.27
	5	30	Swap	PMX	17120.53
	10	20	Swap	CX	17121.6
	10	10	Swap	PMX	17125.07
	5	20	Swap	CX	17127.07
	0	10	Inversion	CX	17133.87
	0	20	Inversion	CX	17134
AG_3	5	20	Inversion	CX	17032.2
	0	30	Insert	CX	17045.33
	5	30	Swap	CX	17047.6
	10	30	Swap	CX	17047.87
	0	20	Swap	CX	17049.33
	10	20	Swap	PMX	17051.73
	0	10	Swap	CX	17054.4
	5	10	Swap	CX	17056.93
	10	10	Swap	PMX	17057.73
AG_4	5	30	Swap	CX	17048.13
	10	30	Swap	CX	17048.53
	10	10	Swap	PMX	17049.73
	10	20	Swap	CX	17050.93
	0	30	Swap	CX	17052.27
	5	20	Swap	PMX	17053.6
	0	20	Swap	CX	17057.6
	5	10	Swap	PMX	17062.27
	0	10	Swap	CX	17066.27