ISCTE IUL
Instituto Universitário de Lisboa

Department of Information Science and Technology

Two-dimensional placement compaction using an evolutionary approach: a study

Rafael Luís Ferreira Valente

Dissertation submitted as partial fulfilment of the requirements for the degree of

Master in Telecommunications and Computer Engineering

Supervisor:

Ana Maria de Almeida, Assistant Professor,

ISCTE-IUL

September 2018

Acknowledgment

To my family, who stood by my side throughout this study and supported me in every situation.

To my beloved girlfriend, who had the patience and the kindness to understand that maybe we could not go out to dinner because I had set my priorities.

To my supervisor, my sincere appreciation for having the time and availability to answer to my annoying questions and always answer with a smile.

To my friends, that did not have a clue of what I was doing in my master thesis but still listened to me.

To my dogs, that heard me sigh in some occasions and came to me to comfort.

To all that I listed and others that I may have not mentioned, my sincere Thank You.

Abstract

The placement problem of two-dimensional objects over planar surfaces optimizing given utility functions is a combinatorial optimization problem. Our main drive is that of surveying genetic algorithms and hybrid metaheuristics in terms of final positioning area compaction of the solution. Furthermore, a new hybrid evolutionary approach, combining a genetic algorithm merged with a non-linear compaction method is introduced and compared with referenced literature heuristics using both randomly generated instances and benchmark problems. A wide variety of experiments is made, and the respective results and discussions are presented. Finally, conclusions are drawn, and future research is defined.

Keywords: two-dimensional rectangular placement; compaction; evolutionary strategy; genetic algorithm.

Index

Table Index

Figure Index

Abbreviations and Acronyms List

2-D – Two-Dimensional

2BP – Two-Dimensional Bin Packing

2D-SLOPP – Two-Dimensional Single Large Object Placement Problem

2K – Two-Dimensional Knapsack

2SP – Two-Dimensional Strip Packing

BF – Best-Fit

BFDH – Best-Fit Decreasing Height

BL – Bottom-Left

BLF – Bottom-Left Fill

C&P – Cutting and Packing

EA – Evolutionary Algorithm (or Approach)

FFDH – First-Fit Decreasing Height

FPGA - Field Programmable Gate Array

GA – Genetic Algorithm

GA-BLF – Genetic Algorithm with the Bottom-Left Fill heuristic

GCD - Guillotine Cutting Problem with Demand

GCV - Guillotine Cutting Problem with Value

GRASP – Greedy Randomized Adaptive Search Procedure

GSRC - Gigascale Silicon Research Center

HECA – Hybrid Evolutionary Compaction Approach

IFF – Improved First-Fit

LGFi – Improved Lowest Gap Fill

LGFi_{OF} – Improved Lowest Gap Fill non-oriented

LMAO - Left-Most Active Only

MCNC - Microelectronic Center of North Carolina

MERAM - Minimization of Enclosing Rectangle under Magnetic Attraction

OPC – One-Point Crossover

OPP - Orthogonal Packing Problem

RRC – Random Respectful Crossover

SA - Simulated Annealing

SMP - Search and Memorize for Packing

SPGAL – Strip Packing Genetic Algorithm with Layer approach

SPP – Strip Packing Problem

TSBP - Two Step Branching Procedure

VLSI - Very Large-Scale Integration

Chapter 1 – Introduction

1.1. Problem

Two-dimensional (2-D) placement problems are a specific class of Cutting and Packing (C&P) problems and are acknowledged as being NP-hard in general. Informally, a 2-D placement problem consists in allocating 2-D items within a placement area without overlapping. Further placement restrictions might have to be imposed. When the item's shapes are rectangular, orthogonal and guillotine cut constraints are commonly used for the most various reasons, namely for cutting, packing or routing. Additionally, the placement is intended to optimize given cost or utility functions. For the sake of simplification, hereinafter it is assumed that both items and the final placement are rectangular shaped.

The complexity associated with 2-D placement problems can vary from relatively simple problems, like disposing written articles in a magazine page, to much more complex challenges like placing components in a chip. In the latter, the complexity is derived from the sheer number of components to be allocated, since the typical dimension of a component has an order of magnitude of nanometres. 2-D placement problems can be differed as being offline or online. In the offline category, the input rectangles to be placed are known before building the packing. In the other hand, in online problems, rectangles are coming one at a time and a placement decision for the current rectangle must be done before the next rectangle is processed. Once a rectangle is packed, it is generally assumed that it will never move again.

Apart from the overlapping and orthogonal constraints, this study addresses two more restrictions, item orientation and guillotine cutting. The first limitation refers to the possibility of rotating by 90 degrees the items to be placed. Consequently, two concepts need to be clarified. When the width of a specific rectangle is superior to its height, the rectangle is said to be horizontally placed. Analogously, when the height is larger than the width, the rectangle is addressed as vertically placed. If both dimensions hold equal values, then the rectangle is a square and its orientation is ignored. The rotation of items implies that the number of potential solutions grows in a combinatorial way. Regarding the guillotine cutting constraint, items need to be allocated in such a way that allows an edge-to-edge guillotine cut or slice cut in the containing rectangular object, thus transforming it into two (sub)rectangles with smaller dimensions and without damaging allocated items. Slicing placement area division and individual items orientation are both inherently hard problems.

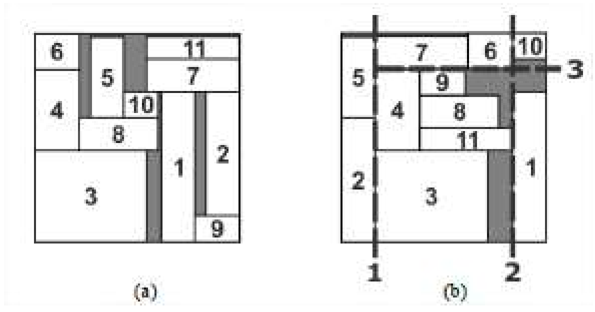


Figure 1 – Non-guillotine cuttable pattern (a) and guillotine cuttable pattern (b).

[*Source:* (Gomes & Mourão, 2012)]

In the present study, a hybrid approach for the generation of an optimal (minimal) placement with orientation and guillotine cutting constraints is proposed. This method consists in the integration of a Genetic Algorithm (GA) with a non-linear compaction algorithm (Almeida, Martins, & Rodrigues, 1998). Formally, the algorithm proposed is called Hybrid Evolutionary Compaction Approach and it will be further on addressed as HECA.

1.2. Motivation and relevance

Typically, C&P problems are NP-hard considering that they are unsolvable in a life time range using traditional algorithms. Therefore, the usual approaches are heuristics with a given percentage of deviation to optimality. Obviously, in some cases, optimal solutions can be known at hand and heuristics have proven to reach them in a relatively short computational time. Generally, these heuristics include a Bottom-Left (BL) algorithm (Baker, Coffman, Jr., & Rivest, 1980) to create an initial solution and then, an evolutionary approach (or genetic search) is applied to find the optimal placement solution.

A GA is a search method based in the Darwinian theory of natural selection and it was first introduced in (Holland, 1975). There are a great number of variations, but all of them respect three stages: reproduction, crossover and mutation, and selection. These are commonly called genetic operators. Initially, a population is randomly created and as the algorithm evolves, by means of crossover and mutation, only the fittest individuals progress to new generations.

1.3. Research questions and objectives

In the presented work, HECA will be tested for the 2-D placement problem with two associated constraints, possibility of component rotation and placement location limitation (to find a guillotine cutting pattern). This approach merges a GA and the Non-Dominance theory for multicriteria problems, studying its performance while comparing with usual approaches from related literature and benchmark instances.

Therefore, there is a main research question and other more specific research questions that, in the end of the research, must be answered.

Main research question:

- In what measure does the proposed hybrid approach compares with the current state of the art methods?

More specific research questions:

- Implementing different genetic operators leads to less wastage area or semi-perimeter?

1.4. Methodological Approach

The research methodology to be followed is based on the Agile methodology (Jonathan Rasmusson, 2014). The great focus of this software methodology is to deliver pieces of software incrementally (sprints), from the beginning of the project until its end, and not deliver the whole software by the end of the project, as it happens in classic software methodologies like Waterfall (Andrew Powell-Morse, 2016). The Agile methodology implementation in the present project can be visualized in the next section, as a work plan divided into sprints. However, having the work plan divided in iterations, there is always the need to adjust literature review and the algorithm development.

1.5. Dissertation structure and organization

The present study is organized in five chapters intended to reflect the different stages of this dissertation throughout its conclusion.

The first chapter introduces the research theme and objectives as well as a brief description of the study structure.

The second chapter states the theoretical framing translated by the referred relevant literature.

The third chapter is dedicated to the methodology followed in the process of gathering and treating data as well as data analysing methods.

The fourth chapter presents the experimental results and respective discussions that respect the appropriate methodology.

In the fifth and final chapter, study conclusions are introduced as well as recommendations, contributions for the scientific community, limitations and, finally, directions for future research are defined.

Chapter 2 –Literature review

The current chapter summarizes the relevant literature and illustrates some techniques for solving 2-D placement problems. Due to the NP-hard classification, instances for these types of problems are mainly solved using heuristic algorithms.

C&P problems are a broader class of 2-D placement problems. These types of problems can be found in industry (for the cutting and packing of materials like wood, glass and metal), transportation, logistics and warehousing. 2-D placement problems can be subdivided into other more specific classes. The 2-D bin-packing (2BP) problem, 2-D strip-packing (2SP) problem and 2-D knapsack (2K) problem are examples of 2-D placement problems that will be described further on. The 2BP problem can be formally described as: having a set of small rectangles and an infinite number of bins (larger rectangles), pack all the small rectangles, without overlapping and its edges must be parallel to the bin's edges, using the minimum number of bins. The description for 2SP problems is not much different. A given set of small rectangles must be packed in a single strip with limited width and infinite height. The main goal is to find a pattern for the rectangles placement that minimizes the strip's height. As for the 2K problem, a list of rectangles, with defined width, height and a value, is packed in a larger rectangular surface. The main drive is to maximize the total value of placed items.

2.1. 2BP problems

The 2BP problem with guillotine cutting constraint is addressed (Bansal, Lodi, & Sviridenko, 2005) where rectangles must be allocated in such a way that allows a recursively sequence of edge-to-edge cuts, parallel to the edges of the bin. The guillotine cutting constraint arises from the design of cutting machines and the programming complexity associated with it. With that in mind, guillotine cutting process is considered where the raw material value is relatively low when compared to the cutting process cost itself. A theorem is presented that admits an Asymptotic Polynomial Time Approximation Scheme (APTAS), i.e. an approximation algorithm, to solve a specific 2BP problem in polynomial time, since the 2BP problem is usually NP-hard. Moreover, a tree representation is used where a k-stage packing corresponds to a tree of depth k. The method contains a rounding technique that groups rectangles according to its dimensions to maintain a constant number of rectangle types without changing the resulting pattern. This allows the reduction of the tree's depth because a lot of edges on the tree are similar and can be merged. This routine starts by ordering the rectangle set by non-increasing height. Then, rectangles are divided into groups: big, horizontal, vertical and small (B, H, V and S, respectively). In the H group, rectangle's height is rounded to the highest value in that group. Analogously, in the V group, the same is done regarding the width dimension. Experimental results compare the best-known approximation algorithm at the time with the proposed method in the paper. The best-known approximation algorithm used for comparison guaranteed a deviation of 1.691 from the generated solution to the optimum one. The algorithm developed in the referred paper guarantees a deviation of 4/3 between the generated pattern and the optimum solution, in the worst-case scenario, thus proving the superiority of the developed algorithm.

A 2BP problem is addressed in (Pintea, Pascan, & Măcelaru, 2012) where several heuristics are compared, namely First-Fit Decreasing Height (FFDH), Best-Fit Decreasing Height (BFDH) and Bottom-Left Fill (BLF). A GA merged with the BLF heuristic is introduced. The comparison made in the referred paper confronts a greedy algorithm with the hybrid GA with the BLF heuristic (GA-BLF) developed. The results gathered show that GA-BLF has the best results and, therefore, is the best approach amongst the heuristics used (BLF with greedy and greedy standalone). The differences become more visible when the datasets used for testing increase in size.

2BP problem approaches can vary depending on certain constraints. In (Blum & Schmid, 2013), the authors propose an evolutionary algorithm (EA) merged with the Improved Lowest Gap Fill (LGFi) heuristic to solve a 2BP problem with a free guillotine cutting restriction. LGFi is a randomized two-stage one-pass heuristic for encountering solutions. To confront the EA-LGFi utility, a comparison, gathering two other heuristics (Set Covering Heuristic - SCH and Hybrid greedy randomized adaptive search procedure - GRASP) is made. The EA-LGFi method presented good results and it is able to solve to optimality four past unsolved problem instances. Furthermore, it is shown that the method obtained the best (minimum) value for the numbered of bins used in a specific instance.

2.2. 2SP problems

The wardrobe application problem (Gomes & Mourão, 2012) is one example of a C&P problem found in industry where a portuguese company is confronted with the challenge of managing efficiently the cuts of raw material in a wardrobe factory. The authors developed two new level-oriented heuristics: Bottom-Left First-Fit Decreasing Height (BLFFDH) and Guillotinable Bottom-Left First-Fit Decreasing Height (BLFFDHg) based in the FFDH and Bottom-Up Left-justified methods. The problem presented is classified as Two-Dimensional Rectangular Single Bin Size Bin Packing Problem (2DRSBSBPP) according to (Wäscher, Haußner, & Schumann, 2007)'s typology. The differences between both heuristics is that, in the last case, the resulting pattern ensures the guillotine cutting constraint. Several methods were used for comparison with the proposed ones. Apart from the new algorithms developed techniques like FFDH, Modified First-Fit Decreasing Height (MFFDH), Adaptative First-Fit Decreasing Height (AFFDH), Improved Lowest Gap Fill non-oriented ($LGFi_{OF}$), Bottom-Left Descend Width (BL-DW) and Bottom-Left Descend Height (BL-DH) were employed. All the heuristics present similar performances regarding wardrobe instances. Nevertheless, the $LGFi_{OF}$ method presents the best solution in half of the instances (12 of 24 instances) that were tested. The methods introduced in this paper outperform the two heuristics that they are based on, FFDH and BL, for every instance.

Usually, 2SP problem solving techniques need to address two constraints, item orientation and guillotine cutting, that help partition 2SP problems into four 2SP-subproblems: RF (with item orientation and without free guillotine cutting), RG (with item orientation and with free guillotine cutting), OF (without item orientation and without considering guillotine cutting), OG (without item orientation and with free guillotine cutting).

A 2SP problem is addressed as a container loading problem in (Bortfeldt, 2006). In the referred literature, a container must be loaded with predefined objects in such a way that the container layout is minimized. The authors introduce a strip packing GA with layer approach (SPGAL) that does not use encoded solutions. This method can also be interpreted as a Container Loading Problem with GA (CLP-GA) approach. The algorithm uses the BFDH heuristic to determine the initial solution. It is then used a search method diversification technique for items to be placed. After each object's placement, a routine is executed with the purpose of repositioning the already placed items. Consequently, the use of this process can lead to solutions that do not meet the guillotine cutting constraint. Therefore, this routine can only be used in 2SP instances that do not have this limitation. The testing results clarify SPGAL's potential for approaching strip-packing problems. Regardless of each 2SP problem subtype (RF, RG, OF, OG), the SPGAL outperformed every other method (BL, BLF, BFDH, parallel GA) in terms of occupied space in the container, i.e., the percentage of empty space is smaller when comparing to other methods. Nevertheless, it's true viability concerns 2SP very complex instances.

The 2SP problem with no wastage area is addressed (Lesh, Marks, McMahon, & Mitzenmacher, 2004). It is introduced a simple pruning approach that helps a branch-and-bound exhaustive search algorithm being truly efficient for instances with less than 30 rectangles, as experiments support. However, the perfect packing routine can be used in instances with more than 30 rectangles while using a divide-and-conquer strategy in order to relax the original problem into several sub-problems. The Smallest-Gap and BL heuristics are compared to decide which heuristic places rectangles with less dead space. Regarding the testing phase, in all instances tested, in only one was not found a perfect packing. Relatively to running times, facing the worst-case scenario where 29 rectangles need to be packed, the algorithm found a perfect packing in under 10 minutes. In average, the new approach found a feasible solution after scanning only 1% of the search space. Regarding the pruning strategy, it was shows that when used it can speed up the creation of a solution in an order of magnitude less, at least. Overall, the Smallest-Gap heuristic outperforms the BL one when it comes to placing rectangles with less wastage area.

A new best-fit approach is introduced (Burke, Kendall, & Whitwell, 2004) to solve two-dimensional rectangular stock cutting problem instances. Since the main objective is to place rectangles in a larger rectangle surface to minimize the height of the resulting placement pattern, the problem is categorized as a 2SP problem. Usually, literature approaches sort a list of items by an attribute (e.g., decreasing area) before allocating them. In this paper, the developed method dynamically searches the best pattern given an input list without being limited by the first placed item. Three niche placement policies are tested: leftmost (or rightmost, if mirrored), tallest neighbour or shortest neighbour. Experiments were made confronting the three niche placement policies where the leftmost approach presented the best solutions. The new heuristic outperformed BL and BLF heuristics as well as other literature metaheuristics (BLF + GA and BLF+ Simulated

Annealing) both in quality of the solutions created as well as in the execution time, that was faster than the other comparison algorithms.

2.3. 2K problems

A 2K problem is the 2-D Single Large Object Placement Problem (2D-SLOPP). One such example is addressed in (Pál, 2006). The problem concerns the creation of a cutting pattern of a set of small rectangular items on a larger rectangular plate, with the main objective of maximizing the sum of the profit (or, alternatively, minimizing the waste) of items to be cut. Constraints like guillotine cutting and item rotation are addressed. Three heuristics are developed and applied to SLOPP instances. One of the heuristics is an improvement of the First-Fit heuristic (IFF). The difference is when a row cannot hold more rectangles in its width, the rectangles to be placed are iterated to find those which can be allocated in the same column of already placed items. There are constraints associated to this process. Items to be placed cannot have superior width than the placed item and its height, added to the height of the already placed item, cannot be larger than the row's height. The other two heuristics developed (WDH1 and WDH2) are a variation of the WDH method (Wang, 1983). In the first one, items are sorted before being placed. In the latter, items are simply allocated without being sorted. The three heuristics (WDH1, WDH2 and IFF) were compared. In every instance, each of the methods showed reasonable results achieving a wastage rate between 0% and 5%. WDH1 (sorted rectangles) have the best results (less wastage area) when compared to WDH2 and IFF methods.

Addressing another heuristic procedure for solving 2D-SLOPP instances, a population heuristic based on a non-linear formulation is proposed in (Beasley, 2000). The author introduces a formulation in which a binary variable is used to know if an item is cut from the master surface or not. Two other integer variables are used to locate the centre of the item as its x and y coordinates. This formulation leads to the encoding of a solution used to generate the individuals of the population. Then, crossover and mutation operators are applied to evolve the population. The fitness function has a negative term that penalizes infeasible solutions. Computational results show that the developed heuristic is able to solve standard problems effectively and is even able to solve four instances to optimality in a matter of seconds.

Another 2D-SLOPP is referred in (Hadjiconstantinou & Iori, 2007). There are introduced greedy algorithms and a hybrid GA. The authors consider no item rotation, no guillotine cutting pattern and the value assigned to each small item to be packed is proportional to its area. There were implemented five heuristics (HC_{KP}, HC_{HV}, HC_{GAP}, HC_{ORD} and HC_{ORD2}) from literature based on a greedy procedure. Four of them (HC_{KP}, HC_{GAP}, HC_{ORD} and HC_{ORD2}) can lead to solutions that respect the guillotine cutting constraint and, therefore, cannot find the optimal solution for all instances. The approach to solve 2D-SLOPP instances has two phases: at first, simple upper bounds are defined, and an initial solution is created by the greedy algorithm. Then, in the second phase, the genetic search (that comprehends parent selection, elitism, immigration and a few crossover operators) is performed hybridized with an on-line heuristic. Regarding the developed evolutionary approach (GA_{2SLOPP}), the data representation is given by a permutation of the items. The resulting sequence is addressed as a Genotype and the leading placement solution produced is called Phenotype. The initial population is randomly generated, but it is implemented a mechanism that guarantees that no duplicated individuals are created, i.e., having the same height, width and, consequently, the same value. Individuals are sorted by non-increasing value. Only the best individual (highest value/fitness) from the current generation passed to the next generation. Genetic recombination is performed by choosing two parents from the current generation. The selection is done by based on a probability that is directly proportional to its fitness. Typically, two individuals are created from the parent genetic recombination. Then, the new offspring fitness is computed and only the best individual (highest value/fitness) progresses to the next generation. Finally, the new generation is completed through the random creation of the remaining individuals. This process id called Immigration and its main objectives are to diversify the search and prevent local minima problem. The evolutionary algorithm terminates if a computed solution is equal to the upper bound defined, or when the maximum computational time is attained or if the maximum number of generation is achieved. In the testing phase, using the GA_{2SLOPP} algorithm, all instances were solved to optimality but one. Computational times do not exceed the one-minute mark, except in two instances that reached four minutes of execution. The tests were intensely performed with different size and complexity and show that the new heuristic outperforms referred literature methods. In a matter of fact, the hybrid algorithm performs better with the increase of problem size (number of item types grows).

2.4. Other 2-D packing problems

Mrad et al. propose a branch-and-price algorithm (Mrad, Meftahi, & Haouari, 2013) based in Gilmore and Gomory model (classic travelling salesman problem) by considering solutions that are obtained by slitting stock sheets down its widths and, afterwards, across its heights. This is known as the two-stage guillotine cut constraint and it belongs in the 2SP problem category. This approach is used for obtaining integer optimal solutions for Gilmore and Gomory model. The branch-and-price method is compared with an arc-flow model-based solution in instances with up to 809 items. Tests were executed considering three cutting strategies: W - where patterns with only width cuts are considered in first cutting stage; H - where patterns with only height cuts are acknowledged in initial cutting stage; W&H - where both cutting directions are considered. The branch-and-price algorithm presented optimal solutions for W, H and W&H cutting strategies for some instances. The W&H cutting strategy outperformed W and H in six instances. The arc-flow model computing times are significantly larger than B&P method.

The revision of literature indicates that, for solving 2-D placement problems, when evolutionary methods are employed important results are achieved when using an informed initialization, i.e., with the help of another heuristic procedure that generates an initial solution that serves as an input for the GA to be executed and already incorporates important information.

Stockmeyer addresses a methodology to define Very Large Scale Integration (VLSI) layouts and proposes an approximation algorithm (Stockmeyer, 1983), to solve an optimization problem that considers cells that are to be allocated in a chip to be of rectangular shape with given dimensions in order to generate a slicing type chip layout. If the above conditions are met and an initial placement, represented by a hierarchical binary tree with locality/proximity relations, is given, the compaction of placement area can be solved in polynomial time.

Modern VLSI floorplanning considers the packing of blocks within a fixed die (outline) and then, additional constraints (interconnectivity and block positioning) are considered. Typical approaches to the Layout Design problem involves usage of Metaheuristics. This design problem involves conflicting problems: proximity definition, routing connectivity and compaction. Unfortunately, the general efficiency is highly constrained by the efficiency of the placement algorithms used.

A new placement heuristic is proposed (Ahmad, 2015) for the two-dimensional, orthogonal, and rectangular layout design involving non-identical hard modules. The objective function considers the distance between the centres of modules A and B as well as a multiplier X that can be manipulated by the user knowing that modules A and B are to be closer ($X > 0$) or apart ($X < 0$). Moreover, a "weight" is assigned to each module. Three fitness function were used: the maximization of the Contiguous Remainder, the minimization of the total inter-modules distance, and the subjective Quality Rating. The author underlines that the proposed method Minimization of Enclosing Rectangle under Magnetic Attraction (MERAM) has a quadratic ($O(n^2)$) computational cost. The proposed algorithm considers a small integer for the number of modules available for a given placement. In each placement decision, a reduction of the enclosing rectangle area of the pattern as well as the sum of inter-modules distance is sought. After the placement stage, a GA is used for optimization of the resulting packing pattern. Selection is done through a biased Roulette scheme based on the raw fitness of layouts. Mutation rate is 80% and it interchanges modules of different patterns, deletes or inserts modules in a given pattern. Crossover rate is 20% and is the one-point technique. A Replacement Strategy is used where a new offspring layout replaces the worst layout of the current population. However, since such strategy is used, and in order to promote diversity and avoid genetic drift, the mutation rate must be high. The BL, Improved BL and BLF algorithms were compared to the MERAM method in a set of benchmark problem instances. All but one problem, have randomly generated instances. Results show that the MERAM algorithm outperforms the other methods in all instances regarding space utilization, therefore presenting less dead space layouts. Descending order of length of the longer side proved to be the best sort method when creating better solutions. In terms of computational time, the MERAM algorithm was the slowest one given its increased complexity facing the other approaches.

In (T.-C. Chen & Chang, 2005), a three-stage fast algorithm based in Simulated Annealing (SA) is introduced which uses a B-tree* (Chang, Chang, Wu, & Wu, 2000) floorplan representation. Two types of modern floorplanning are studied: fixed-outline floorplanning and bus-driven floorplanning. In the first type, the Fast-SA dynamically changes the weights of its cost function to optimize wirelength for the outline constraint. The bus-driven floorplanning is especially difficult as it meets interconnect and block positioning constraints simultaneously. Therefore, the feasibility conditions of the B-tree* representation are explored, and the Fast-SA is used to find solutions. A B-tree* is an ordered binary tree that is used for modelling the block placement in linear time. Usually, for B-tree* generation, top-down approaches are used, i.e., the root represents the bottom-left block (first one) of the placement. Regarding child nodes, the one that is to the left of its parent node have its x-coordinate set as xchild = xparent + widthparent. The child nodes located to the right of its parent, share the same x coordinate. Regarding the three stages of the Fast-SA method introduced are: (1) the high-temperature random search stage, (2) the pseudo-greedy local search stage, and (3) the hill-climbing search stage. In the Simulated Annealing algorithm, when temperature T $\rightarrow$ infinity, inferior solutions are acceptable. When T $\rightarrow$ 0, the SA works as a greedy procedure and only very few inferior solutions are accepted. Regarding experimental tests, for fixed-outline floorplanning, results indicate that in average it is achieved a 100% success rate by the algorithm developed when compared to literature methods that only achieve rates ranging 10-90%. Regarding the bus-driven floorplanning, results show that the new algorithm reduces by 20% or 55% of dead space for the floorplanning with hard or soft macro blocks, respectively, when compare with literature procedures. Also, running times of the new approaches are much smaller than classical SA methods.

Two-dimensional placement of modules in VLSI circuits is addressed (Emmert & Bhatia, 2001). A new approach based in Tabu Search is introduced with the main purpose of speeding up the process of obtaining a high-quality solution. The method makes use of a two-step strategy where initially, circuit routability is improved and finally, the circuit performance is enhanced. The Tabu Search metaheuristic uses a list of moves and restrictions are added to certain moves that converge to local optima. The main objective function of the presented work is to minimize the total wire length. Therefore, the multi-terminal net is converted to a set of edges where each edge consists of a driving terminal and one driven terminal. Then, the Manhattan length of each edge is used to estimate the total wire length. To augment the circuit performance, the placement strategy minimizes the critical circuit edges. In the end, tests are performed against comparison methods based in Simulated Annealing and commercial tools and results show that the proposed algorithm outperforms the other approaches in achieving good solutions in much less time. Actually, the proposed approach reaches high quality solutions 20 times quicker than the SA methods, that suffer from a slow-start problem. The total wire length of circuits generated by the compared methods is similar to the total wire length produced by the introduced method.

The floorplanning of VLSI circuits is of utter importance since the physical layout of a chip determines its performance and reliability. A hybrid simulated annealing (HSA) algorithm is presented (J. Chen, Zhu, & Ali, 2011) for non-slicing VLSI floorplanning. The HSA uses a greedy algorithm to create an initial B*-tree, a new operation to explore the search space, and an innovative search strategy to balance global exploration and local exploitation. The cost function used is measured by both area and wirelength used in the chip. Regarding the layout representation by the B*-tree, the root represents the module on the bottom left-most corner of the floorplan. The right child of a given node is adjacent to its parent to its right side. The left child is located above and adjacent relative to its parent node. Modules to be placed are sorted decreasingly by an evaluation function that compares the height and width of each module with the same dimensions of the floorplanning region multiplied by a couple of weights. Two experiments were carried out in which area optimization and simultaneous area and wirelength optimization were used as objective functions. Since only the new approach and another comparison method were implemented on the same programming language on the same platform, the other comparison methods running times were not listed. Benchmark tests from MCNC were used to gather results and draw conclusions. The HSA algorithm showed the best improvement in the area only objective function tests. Regarding simultaneous area and wirelength optimization objective functions, tests made show that the new approach presents better average area results but slightly worse average wirelength usage, when comparing with other testing methods.

Dhiraj et al. introduce a bottom up approach for floor planning that makes use of a method of composite block formation (composite blocks are a group of placed modules) using macro modules (Dhiraj, Verma, & Kumar, 2013). After searching for a placement solution, the process involving a recursive method of bounding rectangle formation is executed. The main objective is to minimize the cost metric function that, since only bounding rectangles are considered, only the total area produced by placed modules is taken into account. As the developed algorithm uses a bottom-up approach, the representation of a solution is given by a binary tree. The testing phase was executed on standard MCNC benchmark circuits. Only hard modules were used for placement consideration. Results show that the new approach presents solutions with 85-99% area utilization in less time when compared to other methods. The major drawback of the introduced approach is that when building composite blocks recursively, the dead space inside becomes unreachable and keeps increasing in future searches.

The two-dimensional transistor placement for standard cell layout synthesis problem is addressed (Saika, Fukuui, Shinomiya, Akino, & Kuninobu, 1997). This problem is typified as a Large-Scale Integration (LSI) problem. A four-phase algorithm is introduced, with a cost function of optimizing the net extent (or wire length) of the placed transistors, that is able to optimize wiring directly and diffusion sharing indirectly. VLSI layout problems can generally be sub-divided into 3 groups: (1) transistor/module placement, (2) routing of internal nets (set of transistors) and (3) compaction. The introduced algorithm tackles the first sub-problem. The first phase of the algorithm is to generate transistor series by grouping them together. In the second phase, these groups are placed in a one-dimensional style. Here, a random layout is chosen and improved iteratively using a simulated annealing algorithm. The third task concerns about folding transistors that are relatively large, because layout optimization is considered. Finally, the last stage of the new approach is about placing all transistors in a two-dimensional style utilizing the placement generated in one-dimensional style. Results show that optimizing in two-dimensional style is better than just executing a one-dimensional style optimization as it reduces, in average, cell width in 33%. In the other hand, it also increases cell height in 14%, in average.

In online Field Programmable Gate Array (FPGA) two-dimensional placement, a fast and efficient algorithm for finding empty spaces is of utterly importance. FPGA is an integrated circuit which is programmable and, therefore, reconfigurable by the user.

Hand et al. introduce a new method that predicts empty spaces in form of overlapping maximal rectangles (Handa & Vemuri, 2004). In a worst-case scenario, time complexity is O(xy), where x is the number of columns, y is the number of rows and xy is the number of cells of the FPGA. Using the overlapping maximal empty rectangle strategy has proven to lead to less wastage area than only keeping non-overlapping empty rectangles. As FPGA is represented by a matrix, a new matrix representation is introduced to save time while scanning for empty spaces. It is referred a staircase technique for finding maximal empty rectangles. A staircase is called maximal when it encloses a maximal empty rectangle. A maximal staircase is created when it cannot be extended either to its right or to its bottom. Exhaustive tests were performed, and it is shown that the new approach only needs to scan less than 15% of the FPGA cells in order to find empty spaces. This also means that this approach needs 85% less time to find empty spaces. Moreover, since a new matrix representation of the FPGA is used, it is shown that the algorithm needs half of the memory when comparing to literature methods.

In (Almeida et al., 1998), a compaction heuristic based in (Stockmeyer, 1983) is presented within the context of Floorplan Design. Given the relative positioning of the components, the objective is to minimize the placement area to construct a circuit board. Typically, these relations tend to minimize the cost measurement of a placement that could be weighted by its area or semi-perimeter, for instance. In this paper, the main concern is to present an algorithm that addresses an optimal cell orientation and defines a satisfactory placement with guillotine cutting constraints. The algorithm developed provides the best solution in terms of minimizing the occupied area of placement in the circuit board, regardless of the components number, when comparing to the benchmarking Stockmeyer's algorithm.

A compaction algorithm is introduced in (Chekanin & Chekanin, 2017) where the method improves already built solutions following a six rule process to select already placed items, remove and reallocate them in a free space inside a container. The algorithm was created for solving container loading problems, i.e., a less general problem of orthogonal packing problems. The main goal is to find the densest placement of all objects inside the container respecting the following constraints: (i) item overlapping is not accepted; (ii) all object's edges are parallel to the container's edges; (iii) items are fully contained in the larger object. Also, item rotation is not allowed. Tests show that, in average, the compaction algorithm developed improves solutions, i.e., reduces the wastage space, by 2%.

Two exact algorithms are introduced (Clautiaux, Carlier, & Moukrim, 2007) to solve two-dimensional orthogonal packing problems (2OPP). This type of problems is transversal to 2BP, 2SP and 2K problems as items are constrained to be placed with its edges parallel to the containing surface edges. The first exact algorithm proposed (Left-Most Active Only - LMAO) is an improvement on a classical branch-and-bound algorithm, while the second one (Two Step Branching Procedure - TSBP) is a based on a two-stage enumerative algorithm. Also, space reduction procedures and lower bounds that can be used in LMAO are described. The two-step method is composed by two branch-and-bound algorithms. The first one (LMAO) computes the x-coordinates of the items. Then, for each feasible solution created, the second branch-and-bound algorithm is launched to determinate the y-coordinate of items included in the solution created by the first algorithm A reduction procedure is also introduced where the set of items to be placed are grouped by size. Firstly, "tall" items are allocated in the bin by decreasing height and the, "shallow" pieces are put above the "tall" objects using a bottom-left rule. Also, a technique is used to remove redundant solutions. A pseudo-symmetry procedure checks for solutions that can be symmetrical with respect to the y-axis. The solution that does not respect the leftmost-downward rule, it is not enumerated. Actually, both solutions can be discarded if not following the rule. The presented work focusses on reducing computational times. The branching scheme (TSBP) algorithm dramatically reduces execution times when compared to LMAO or MV - Martello and Vigo algorithm (Martello & Vigo, 1998). Moreover, TSBP can solve instances that the above referred methods are not capable of solving. TSBP is also compared to Fekete and Schepers method and it is able to solve instances which the latter cannot solve.

Grandcolas et al. present a new exact algorithm (Search and Memorize for Packing - SMP) for the 2-D orthogonal packing problem (Grandcolas & Pinto, 2015). Firstly, a relaxation of the problem is addressed and after the items placement, the horizontal axis of the bin is analysed, and items positions are defined to conclude if the sum of the object's heights does not exceed the height of the bin. If a valid placement is found, another procedure is executed to check if it can be extended to the original packing problem. seeking the items and positioning through the bin's vertical axis. Therefore, placements can be separated in Px and Py, position of items in the x and y axis, respectively. Then, a canonical placement can be constructed from Px if a reordering of the items can occur, allocating the items as much as possible to the left without violating the height constraint. Analogously, canonical placements can also be generated from Py. These canonical placements help realize a problem's feasibility, because they are derived from placements Px and Py. Restricted placements are defined in order to reduce the search space and unsuccessful placements are memorized to prevent dead ends. Moreover, the paper refers that ordering the items by some criteria before placing them helps reducing the search space. Orthogonal packing problems (OPP) and strip packing problems (SPP) were solved using the exact approach (SMP) and literature methods. Regarding OPP instances, items are ordered by decreasing height. The SMP method outperformed other approaches in every feasible instance. On unfeasible instances, the SMP algorithm is outperformed by Clautiaux methods (Clautiaux, Jouglet, Carlier, & Moukrim, 2008). The SMP method performs better with highly constrained problems, i.e., if the total area of the placed items is much smaller than the area of the bin, then the SMP method does not perform so well because the items share many intersections and the definition of their vertical positions is very hard. Relatively to SPP instances, ordering the items by increasing height provides better solutions than ordering by decreasing height.

An exact algorithm for the orthogonal packing problem is proposed (Joncour, Pêcher, & Valicov, 2012). The algorithm uses interval graphs for the characterization of solutions and MPQ-trees (Korte & Möhring, 1989) for recognition of the interval graphs. Interval graphs are used for describing each dimension of a given placement to define which rectangle is on top of one another. The algorithm presented is based in the one introduced by Fekete and Schepers. One of the improvements is to discard unnecessary couples of interval graphs that are generated by symmetrical solutions. The algorithm developed was tested in orthogonal packing problems and orthogonal knapsack problems. The exact method was able to solve all but one instance in the limit time of 1800 seconds. Analysing literature methods, only one algorithm had similar results as the one developed in the paper. Running times are also smaller for the exact algorithm when comparing with literature procedures. The algorithm presented is less efficient when dealing with instances of small size (small number of items to pack). Consequently, this approach should be only considered with larger instances.

Still in the OPP, a hybrid genetic algorithm based in random keys with a placement procedure is introduced (Gonçalves, 2007). The problem addressed in the paper falls in the category of 2K problem and its assignment kind is output maximization. Regarding the GA, three phases are executed: (1) decoding of the rectangle packing sequence that is coded in each chromosome, (2) placement strategy that takes advantage of the decoding done in the first phase to perform a packing of all rectangles, and (3) fitness function which uses a new procedure - Modified %Trim Loss - to evaluate the quality of generated solutions and gives feedback to the GA. A list of empty rectangular spaces (ERS) is kept updated to decide the placement that best-fits each rectangle. As stated in phase (3), the fitness function used is a Modified % Trim Loss where packings containing larger ERS's have more potential to have its trim loss reduced and, consequently, produce solutions closer to the optimal. Evidencing the crossover genetic operator, it is called the Parameterized Uniform Crossover where two chromosomes are randomly chosen, one from the best (fittest) group (TOP) and other from the rest of the population. Then, a crossover probability (cprob) is defined and for each random key (gene) of the chromosome, a random number ϵ [0,1] is generated. If it is inferior than the cprob, the gene chosen belongs to the fittest chromosome and it passes to the new offspring. Otherwise, the gene selected belongs to the least fit chromosome. The testing phase opposed the proposed hybrid meta-heuristic with three literature algorithms and results show that the new approach presents excellent quality of the solutions created in every instance in terms of trim loss percentage. Running times were not compared since the algorithms were developed in distinct programming languages and were tested in different machines. Future work is defined the extension of the new approach to the strip cutting problem and the weighted version of the orthogonal packing problem.

Aryanezhad et al. propose a new heuristic for the two-dimensional guillotine cutting stock problem (Aryanezhad, F. Hashemi, Makui, & Javanshir, 2012). The developed method is based in the First-Fit Decrease and Ranked Positional Weight heuristics. In this paper, the matter of selecting the appropriate sheet size is discussed as it is utterly essential for reducing the stock wastage. The developed method was compared against the BLF, Best-Fit + BLF and Interactive Approach (IA). Results show that the running times of the new approach are smaller than the other methods and, in some cases, are much smaller. Moreover, it is shown that increasing the problem size did not cause a major increase in computational times.

Two cutting problems and their variants that consider item rotation are investigated (Cintra & Wakabayashi, 2004). Moreover, a dynamic programming-based algorithm for the Two-Dimensional Guillotine Cutting Problem with Value (GCV), which is similar to the 2K problem, is introduced. It is also shown that if the items size is not so small when compared to the bin's size, the new algorithm reaches polynomial running times. The Two-Dimensional Guillotine Cutting Problem with Demand (GCD), that belongs to the 2BP problem category, is also addressed and a Column Generation method-based algorithm is presented to solve the previously stated problem. For the GCV problem, it is used the Discretization by Explicit Enumeration (DEE) algorithm that computes discretization points of the width or height of small rectangles. Alternatively, the Discretization using Dynamic Programming (DDP) method can also be used to find discretization points. Finally, the algorithms developed in this paper were tested in GCV and GCD instances, respectively. In almost every instance, optimal or quasi-optimal solutions were found. Moreover, the algorithms achieved polynomial time when the items size is not so small when compared with the bin's size.

The Two-Dimensional Cutting Stock Problem is addressed (Alvarez-Valdes, Parajon, & Tamarit, 2002) with two constraints: fixed item orientation and guillotine cutting patterns. The column generation approach introduced by Gilmore and Gomory (Gilmore & Gomory, 1961) is studied and a related subproblem is solved with the help of several algorithms. The main drive is to combine the column generation approach with a given algorithm that solves the column generation subproblem and executes a rounding procedure to obtain integer solutions for the stated problem in all types of demands in acceptable running times. Three algorithms for solving the column generation subproblem are developed: a constructive algorithm, a GRASP approach and a more time-consuming Tabu Search heuristic. The constructive algorithm keeps a list of rectangles to be cut and a list of pieces already cut. For each step, the smallest rectangle is chosen to be cut from the bottom-left corner of a piece. The GRASP procedure uses the previous algorithm for its constructive phase. In its improvement phase, waste rectangles are merged, if possible, and new rectangles are cut from it using, again, the constructive algorithm. The Tabu Search algorithm runs in three steps: (1) a rectangle from the solution (piece or waste) is chosen randomly; (2) all its neighbours are considered, one at a time; (3) for each pair of neighbours, a specific procedure using the GRASP algorithm is executed. Three strategies for the rounding procedures are compared: a simple rounding-up of the solution, a branch-and-bound method and, finally, a more complex method that rounds down the solution and then uses the GRASP algorithm to solve the residual problem. Tests were made to compare the constructive approach, the constructive approach + GRASP, the constructive approach + GRASP + Tabu Search and a literature method. The rounding procedure chosen was the most complex one. All the approaches presented good results having a wastage percentage of under 5%. The literature method outperformed the others but requires a significant amount of time. The constructive algorithm provided solutions with more wastage but also, the smallest running times.

A new heuristic dynamic decomposition algorithm for the two-dimensional rectangular packing problem is introduced (S. Wang, 2017). The work presented was motivated by the survey performed in (Lodi, Martello, & Monaci, 2002). The dynamic decomposition technique divides the container in two parts using a binary tree structure.

After each rectangle placement, the virtual container collection is updated, and two new virtual sub-containers are added which result from the vertical and horizontal split of the original container having the placed rectangle as reference. For each rectangle to be placed, the container collection is iterated to check which sub-container best-fits the current rectangle. The highest placing priority if assigned to larger rectangles. Finally, results show that the proposed algorithm has smaller running times than the comparison methods as it is able to improve, in average, by 10% of space usage regarding literature methods solutions. Additionally, it is shown that the new algorithm can be extended to the global layout of multi-vessel and three-dimensional layout optimization.

He et al. introduces a two-stage deterministic heuristic for solving two-dimensional rectangular packing problems (He, Huang, & Jin, 2012). The algorithm follows a best-fit strategy and it comprehends a constructive phase, for finding quickly an initial solution, and an improvement phase, for reducing the space utilization with the help of a tree search procedure. Moreover, the Best-Fit algorithm (BF) was also adapted to solve the constrained two-dimensional cutting problem. Two important concepts are introduced and explained that help to better understand the improvement phase: Action Space and Fit Degree. The definition of Action Space is given by having a maximum rectangular empty space where items can be placed. These Action Spaces can be overlapping amongst each other. The Fit Degree translates the compactness between an item being placed and the current Action Space. Furthermore, in the improvement phase, a tree search procedure is used, and, for each iteration, the algorithm selects the top X Action Spaces with larger Fit Degrees and applies them in the constructive phase until all items are placed or no more items can be held inside the selected Action Spaces. The proposed BF algorithm was tested against other literature methods. Two groups of instances were tested where, in the first group, all instances were solved to optimality by the BF algorithm and another literature method, but the BF algorithm presented much less computational time than the previous literature method that had to perform 100 iterations before finding an optimal solution. The second group of instances proved to be more challenging given that "only" 7 of the 13 instances were solved to optimality by the proposed approach. Regarding the two-dimensional cutting problem instances, the BF algorithm was compared to another literature method (SPA). Though the SPA outperformed the BFA in terms of more optimal solutions found, the latter was executed in only 1% of the SPA running time.

A new algorithm for the two-dimensional placement problem is proposed (Healy, 1999). It is claimed that it can reach a time complexity of O(n log n) given it relaxes the two-dimensionality into one-dimensionality using rectangle profiles as line-segments. Moreover, the algorithm searches for an empty space to place a given rectangle by considering that there is, in the worst-case scenario, O(n) bottom-left feasible locations. To keep track of these locations, a Composition of subsequent line-segments is generated by insertion, deletion and/or updating of profiles (line-segments) of the rectangles to be placed. Each Composition has a height-balanced binary search tree ordered by increasing abcissa. Furthermore, it is acknowledged that a feasible placement is one where there are no overlapping pieces. Unfortunately, no tests or results are presented to confirm the introduced algorithm viability.

From the literature referred in this section, (Almeida et al., 1998) is the only approach that presents the best sequence and orientation of guillotine cuts for the optimization problem. Additionally, none of the revisited literature mentions the use of a GA for the placement compaction problem.

Chapter 3 – Methodology

This chapter defines methods and techniques used to develop HECA for solving 2-D floorplanning problems for slicing arrangements. As stated in the introduction, for the sake of simplification, items (or basic rectangles) are to be placed in a rectangular area and grouped on what is called enclosing or enveloping rectangles, whose side dimensions are height and width. Enclosing rectangles implicitly hold the sequence of guillotine cuts needed to place the enclosed basic rectangles. Note that a guillotine cut has an associated direction that can either be vertical (V) or horizontal (H).

After generating a possible placement (an enclosing rectangle with all enveloping rectangles at any point of the evolution), the new algorithm progresses to the Compaction phase, where all the associated possible non-dominated solutions for this placement are sought by means of rotating, swapping or taking the mirror image of enveloping rectangles.

3.1. Heuristic Method for Placement Compaction - HECA

A genetic algorithm will be merged with the compaction method proposed by Almeida et al. (Almeida et al., 1998) to produce a heuristic that follows a classic genetic algorithm structure. A genetic algorithm is a search method based in the theory of natural selection. There are a great number of variations, but all of them respect common attributes: reproduction, crossover and mutation, and selection. Initially, a population is created and, as the algorithm evolves, by means of crossover and mutation, only the fittest individuals progress to new generations (Pál, 2006).

3.1.1. Placement Encoding

The representation chosen for the encoding of a placement solution is a complete binary tree. There are a few rules that imply the balance and construction of this type of tree. The tree must be full until, at least, the last but one level. Additionally, if the last level is not full, all leaf nodes (nodes without descendants) must be aligned to the left.

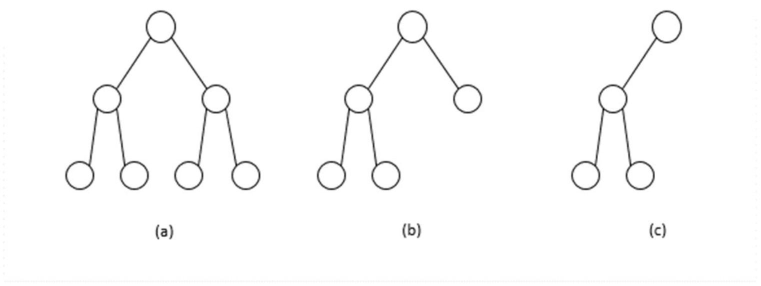


Figure 2 - (a) and (b) Complete binary trees; (c) Incomplete binary tree.

The leaves of the binary tree represent the basic rectangles to be placed. Internal nodes, nodes with descendant(s), represent enclosing rectangles. The binary tree root represents the final rectangular placement area and, ergo, all information regarding basic rectangles placement position, enclosing rectangles, and the sequence of positioning guillotine cuts required to effectuate this placement.

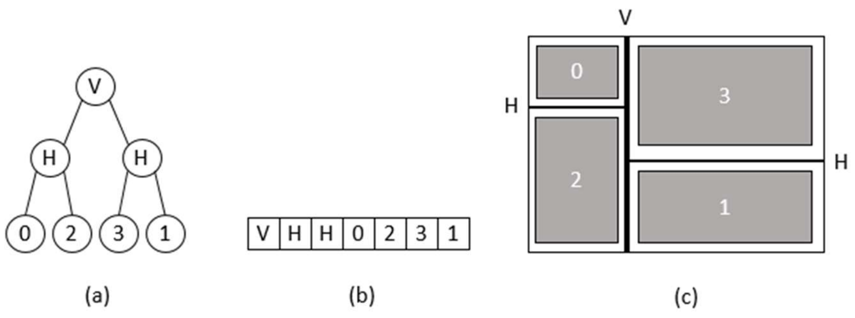


Figure 3 - Placement representation: (a) complete binary tree; (b) vector encoding; (c) associated layout.

The complete tree is encoded by a unidimensional vector (Figure 3b)). Typically, in evolutionary approaches, a solution is viewed as a chromosome. Therefore, chromosomes are vectors whose indexes (genes) either contain a basic rectangle's identification or the direction for a guillotine cut (v or h) standing for enclosing rectangles that simulate a potential placement.

The complete binary tree has advantages and disadvantages like all data representation models. Firstly, searching is possible in O(logn), instead of O(n) the latter being the time complexity associated with a linked list, for instance. Moreover, it is easier to sort a complete binary tree than to sort a linked list. Furthermore, insertion and deletion are also faster in a complete binary tree representation. Finally, a complete binary tree is easily represented as an array as seen in Figure 3b.

Comparing with B*-trees (Chang et al., 2000), where searching or inserting takes only O(1), which is faster than doing the same primitive tree operations when using the complete binary tree. However, the deletion takes O(n) in a B*-tree which is slower to perform than in a complete binary tree (O(logn)). It should be noticed that these operations are not to be performed by our approach, which uses a bottom up construction, without node insertions or deletions.

Another comparison can be made with the most classical representation via permutation of items (Hadjiconstantinou & Iori, 2007). When searching through a space of N items, a complete iteration can take up to O(n!). Comparing with complete binary trees, seeking an item, takes O(logn) in the worst-case scenario. Therefore, complete binary trees

3.1.2. Fitness Function

The fitness function must measure the quality of the chromosome adaptation, which for now will simply be the minimization of the space occupied by the positioning of the placement. This measure can then be defined in terms of the solution's overall area or the semi-perimeter. The area or the semi-perimeter of a solution is calculated using the tree root enclosing rectangle dimensions, that is, the vector's first index associated dimensions.

3.1.3. Initialization.

The initialization of the evolutionary population is achieved through one of two possibilities: controlled or random generation. In the former, basic rectangle's dimensions are pre-determined. In the latter, the dimensions of basic rectangles are randomly computed within a maximum rectangular dimension threshold.

Initially, a number N of basic rectangles are randomly generated. Thus, every chromosome, that is, each binary tree, defines a topological placement (layout) for all the N rectangles in a rectangular placement area via its slicing cuts and basic rectangle disposition (Figure 3). The initial guillotine cutting sequence, for each chromosome, is also randomly selected. Therefore, the initial population is composed by a parameterized number of chromosomes, whose cuts sequence and basic rectangles have a strong random component. All the created chromosomes are used in the first evolutionary cycle.

3.1.4. Genetic Operators.

The genetic algorithm developed follows the typical stages of an evolutionary approach: crossover that combines genes belonging to both progenitors thus creating a new individual, mutation, and selection of individuals for the next generation. In the forthcoming subsections, the genetic operators used are described.

3.1.4.1. Crossover

In the recombination phase, chromosomes are randomly chosen to act as parents and this process always occurs for each generation. In more detail, half of a new population of chromosomes is created in this phase, where two distinct parents are chosen to generate one offspring. Even though parent chromosomes are always distinct, there is a possibility that the newly created chromosome can be equal to one of the parents. In this case, the new offspring is accepted and incorporated in the new population. No special selection operator is used, i.e., the probability of recombination between parents is uniformly distributed. There were used two techniques for this first exploratory phase. The first one is the One-point crossover (OPC). Firstly, a vector index (cross-point) is randomly generated. Then, genes from one parent starting in the initial index to the cross-point are copied into the offspring and from there on, the genes of the other parent are copied, and a new individual is created. Note that each pair of parents generates only one child.

If the cross-point index falls between basic rectangles, there is a mechanism that keeps in a list, basic rectangles that have not been inserted in the offspring. In other words, if the gene to be copied is already in the chromosome it is discarded and a basic rectangle from the list is added instead, removing it from that list. This technique prevents that the same basic rectangle is copied twice to the new individual. Figure provides an illustration of the crossover process.

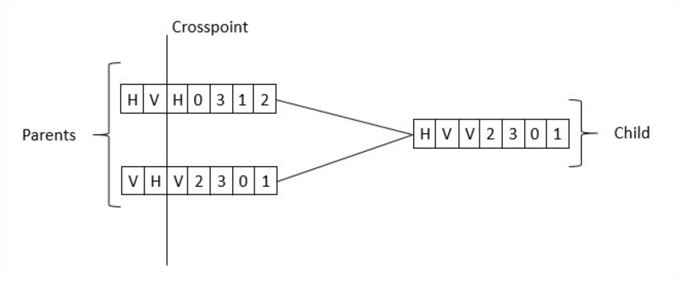


Figure 4 - One-point crossover example

The second crossover employed is the Random Respectful crossover (RRC) (Radcliffe, 1991). In this case, genes are added to the offspring only if the same gene is encountered in both parents in the same vector index, i.e., having two parents A[0..n] and B[0..n] and an iterator i = 0 → n, the gene presented in the i index is copied to the offspring only if A[i] and B[i] are the same. Otherwise, a random gene is introduced in the offspring using the technique described in the One-point crossover to prevent repeated genes. The pseudo-code for the Random Respectful crossover is the following:

```
Algorithm 1 – Random Respectful Crossover (RCC):
Input: chromosome A (vector) and chromosome B (vector); k, integer (vector length)
Output: chromosome C
Begin
    i ← 0;  list_of_basics_in ← $null$;
    While $i < k$:
            If(A[i] is the same as B[i])
            Then Begin
                    C[i] = A[i];
                    If C[i] is a basic rectangle
                    Then add to list_of_basics_in;
            Else C[i] = ChooseNewGene(list_of_basics_in);
            Increment i;
    EndWhile
End
```

3.1.4.2. Mutation

Mutation occurs with rare probability, namely 0.02. The values acknowledged to be the most successful for the mutation operator in a genetic algorithm are to be between 0.02 and 0.03. This genetic operator has the purpose of introducing higher genetic variability and prevent "genetic drift" – convergence to a sub-optimal solution. The mutation operator process occurs by generating a "mutation point": a random index. If an enclosing rectangle is chosen to be mutated and if it holds a vertical cut (V) it is swapped for a horizontal cut (H) and vice-versa. If the index points to a basic rectangle then another basic rectangle is randomly chosen and both trade places.

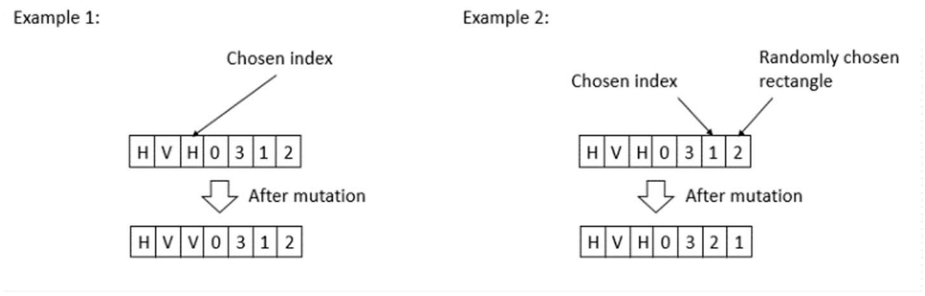


Figure 5 - Two mutation processes examples

3.1.4.3. Selection

At last, selection of individuals to be part of the next generation is achieved through the evaluation of the fitness of each individual. Half of the better fitted individuals from the current generation are transferred to the next generation. This process is known as Elitism. In the end of the selection phase, the new generation is completed since the other half of chromosomes is created in the crossover stage.

3.2. Compaction Method

Considering the randomness introduced and having a population with a given quantity of chromosomes and all the possible basic rectangle placements and rotations combinations, it is obvious that the search space grows in a combinatorial way. Therefore, to converge productively to a solution, a compaction method based in the theory of non-dominance (Almeida et al., 1998) is introduced.

A solution with dimensions (h_1, w_1) is said to be *non-dominated* if when comparing to another solution with dimensions (h_i, w_2) we have:

$$(h_1 \leq h_2 \land w_1 < w_2) \lor (h_1 < h_2 \land w_1 \leq w_2) \qquad (1)$$

An optimal solution for the area minimization (compaction) is a non-dominated one between all the others in the search space (Almeida et al., 1998). The strategy used to find the list of non-dominated solutions for a new enclosing rectangle and for all cutting directions consists in 3 steps:

- Construct the list of non-dominated solutions assuming a vertical cut will occur and combining all 90^0 rotations of the rectangles involved;
- Construct the list of non-dominated solutions assuming a horizontal cut is used and considering all 90^0 rotations combinations of the rectangles involved;
- Merge the two lists created discarding all dominated solutions.

In more detail, given a specific binary tree that represents a potential placement, at the bottom level containing the basic rectangles, each leave has an associated array that represents the two possible placements considering its possible different orientations: horizontal (highest dimension down) or vertical (highest dimension up). As the method progresses, in a bottom-up fashion, it reaches higher levels where enclosing rectangles are to be found. For these internal nodes, the above described procedure for the construction of non-dominated lists of solutions is performed, using the non-dominated solutions from all possible solutions associated with each of children nodes. When the algorithm reaches the tree root, the optimal global solution(s) are among the non-dominated potential placements, illustrating the slicing cut sequence and respective basic rectangles orientation. The solution to be chosen, i.e., the one that has the minimum area (or semi-perimeter) is the one which more efficiently compacts basic rectangles by simultaneously optimizing both the slicing cuts sequence and individual rectangle orientation.

The hybridization of the compaction method with the genetic algorithm occurs after the genetic operators are processed and before a new genetic iteration takes place. Basically, after a new population is formed, for each chromosome, the compaction method takes charge of rotating basic rectangles to create a new placement and check for non-dominated solutions. Figure 6 presents a visual description of the genetic algorithm hybridized with the compaction method.

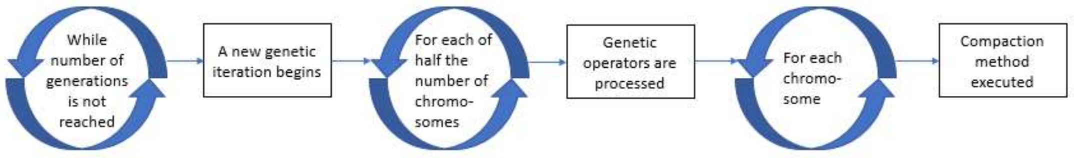

While number of generations is not reached
A new genetic iteration begins
For each of half the number of chromo-somes
Genetic operators are processed
For each chromo-some
Compaction method executed

Figure 6 - Genetic algorithm merged with the compaction method

Technically, each index of the chromosome has a dynamic list where all non-dominated solutions generated are saved. In the first index of the chromosome, the root of the binary tree can be found. In it, the dynamic list saved comprehends all non-dominated solutions generated throughout the tree respecting a bottom-up approach, starting from the lowest level of the binary tree where enclosing rectangles (that represent the guillotine cuts) are encountered and, therefore, the first parent nodes are found, i.e., parent nodes whose child nodes (basic rectangles) combined lead to enclosing rectangles. There is no need to start iterating from the level 0 of the tree since, in this level, only "leaf" nodes are found, i.e., there are no nodes below them. Figure 7 provides a visual explanation for the enclosing rectangles creation.

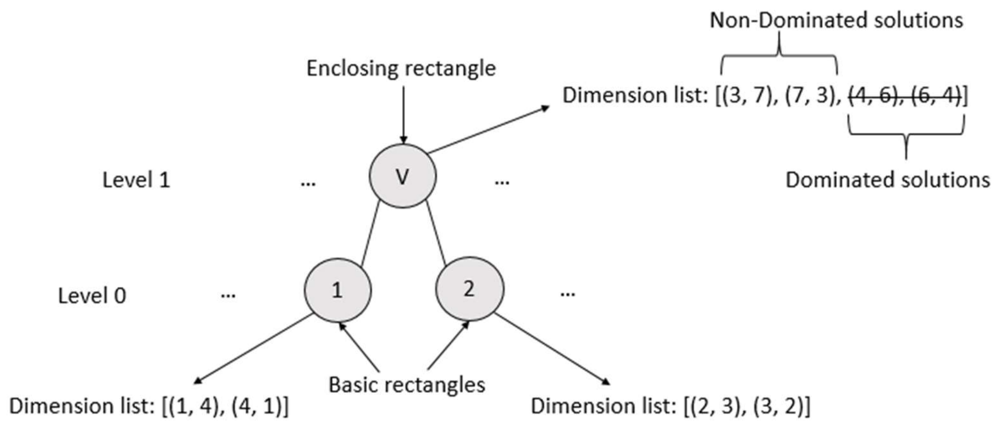

Non-Dominated solutions
Enclosing rectangle
Dimension list: [(3, 7), (7, 3), ~~(4, 6), (6, 4)~~]
Dominated solutions
Level 1
...
V
...
Level 0
...
1
2
...
Basic rectangles
Dimension list: [(1, 4), (4, 1)]
Dimension list: [(2, 3), (3, 2)]

Figure 7 - Enclosing rectangle creation

In each index of the dynamic list is an object that saves the non-dominated solution generated for that level and both child solutions that lead to the first solution. Moreover, the object also keeps the guillotine cut.

Finally, to know the fitness of the chromosome, the dynamic list is iterated to calculate either the area or the semi-perimeter of each non-dominated solution, given the chosen utility function, and the lowest value found through the dynamic list iteration is chosen as the best current placement. Then, this placement is saved in a variable in the chromosome as the current less wastage placement.

Concluding, the methodology followed to generate placements for the two-dimensional placement compaction problem using HECA comprehends a hybridization of a genetic algorithm with a non-linear compaction method. The data representation model chosen is the complete binary tree and the fitness function can be defined as either the area or the semi-perimeter. Regarding genetic operators, the crossover techniques are the OPC and the RRC, mutation occurs with a 0.02 probability and the selection of individuals to progress to the next population follows an elitist approach. Finally, after the genetic search process occurs, the compaction algorithm rotates the rectangles by 90º and discards non-dominated solutions to converge to a final more compact solution.

Chapter 4 – Results analysis and discussion

In this chapter, experimental results will be presented and discussed. Two major experiments were carried out. The first experiment confronted HECA with C&P classical approaches, which are described in subsection 4.1.1. The second experiment was intended to understand how HECA behaves against VLSI benchmark problems. In more detail, Gigascale Silicon Research Center (GSRC) hard benchmarks (Kahng & Markov, 1999) and Microelectronic Center of North Carolina (MCNC) benchmark instances (Microelectronics Center of North Carolina, 2012).

4.1. C&P experiment

Since only random instances are to be tested, the implementation of algorithms for comparison is indispensable to support the new approach viability and validity. Therefore, various algorithms were implemented, being two of them classical C&P (Baker et al., 1980; Johnson, Demers, Ullman, Garey, & Graham, 1974) algorithms. The third method implemented is a more recent algorithm (Sangwan, Verma, & Kumar, 2013).

4.1.1. Compaction methods description

Regarding classic approaches to C&P problems, the BLF heuristic (Baker et al., 1980) is one of the implemented comparison methods. Typical bin-packing problems are approached with this heuristic, either being used as stand-alone or hybridized with an evolutionary approach. Basically, BLF considers the best placement for a small rectangle in the lower leftmost space that can hold the small rectangle size. Figure 8 illustrates an item placement that considers the best-fit empty space for the current rectangle.

The implementation of this algorithm followed the pseudo-code found in (Pintea et al., 2012).

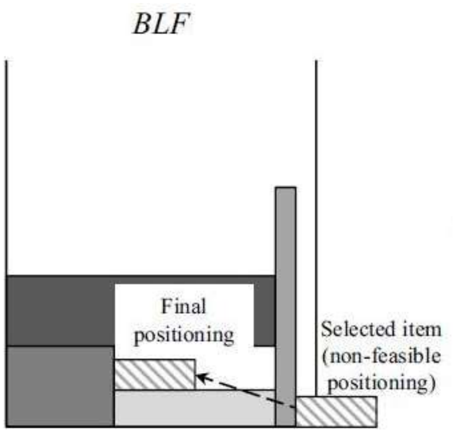


Figure 8 - Bottom-Left Fill algorithm approach to small rectangle placement [Source: (Oliveira, Neuenfeldt Júnior, Silva, & Carravilla, 2016)*]*

The FFDH meta-heuristic (Johnson et al., 1974) was also implemented. Initially, items are sorted by decreasing height before the placement phase occurs. The FFDH heuristic is typically used in strip-packing problems because of its approach by levels. The highest item is allocated in the first level, where it first fits. Consequently, this item also defines the level height. Once a rectangle cannot be allocated in the current level, because the remaining space is not sufficient to hold the rectangle size, a new level is created above the current one and the item is placed, also defining the new level's height. There are items that can be placed in previous levels, therefore decreasing the space wastage of the final solution. Figure 9 shows the method operation.

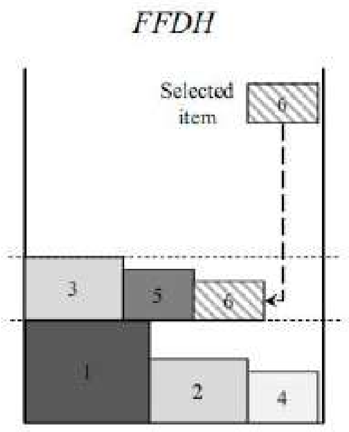


Figure 9 - First-Fit Decreasing Height approach to small rectangle placement [Source: (Oliveira et al., 2016)*]*

Modern VLSI floorplanning considers the packing of blocks within a fixed die (outline) and then, additional constraints (inter connectivity and block positioning) are taken into account (T.-C. Chen & Chang, 2005). A particularly difficult problem of floor planning in VLSI circuits is the bus-driven floor planning as it meets interconnect and block positioning constraints simultaneously. Having in mind approaches for floor planning in VLSI circuit problems, a more recent algorithm was implemented (Sangwan et al., 2013). The algorithm introduced is a bottom-up approach for placement and compaction of modules for floor planning in VLSI circuits that makes use of a method of composite block formation using macro modules. A composite block is a set of 2 or more modules and/or other composite blocks. This method was chosen for comparison as the full pseudo-code was available and there was no need to implement another data representation model.

After searching for a placement solution, the process involving a recursive method of bounding rectangle formation is executed. The main goal of the referred algorithm is to minimize the cost metric function that, since only bounding rectangles are considered, only the total area produced by placed modules is considered.

In the referred paper, the authors also mention how composite blocks can be constructed: 2BLOCK, 3BLOCK or 4BLOCK, meaning that a composite block can be built from 2, 3 or 4 modules/other composite blocks until a placement containing all modules is found. Item rotation constraint is also addressed, i.e., modules can be rotated by 90 degrees. Nevertheless, routing of modules was not implemented for comparison with our approach. The algorithm was implemented with other slight changes. In the referred paper, a variable (Cn) is used that acts as a stop condition for a while loop to place available modules. Since the computation of this variable does not guarantee that all modules are placed, this variable was calculated as the number of the remaining modules to place after each loop. Moreover, the computational time and the resulting pattern area or semi-perimeter were the parameters to compare with our approach.

The decision of implementing this algorithm, with a few changes, was made because of the compaction approach, since this topic is not extensively addressed in literature.

4.1.2. C&P test results

In the following, all the tests were performed in a computer with 8GB RAM (DDR3) and an Intel Core i5 3317U 1.70GHz CPU.

The results exhibited use random placement instances with 100, 250, 500 and 1000 basics. The basic rectangle's dimensions, width and height, were randomly generated (with a minimum and a maximum dimension threshold – [1, 50]). For each basic rectangle instance, the average value of 12 runs was taken for the computational time (in milliseconds) and fitness results of each algorithm to be compared. Regarding the new evolutionary approach presented, HECA, for each instance, 50 chromosomes are created randomly at the beginning. These chromosomes constitute the initial population that is evolved throughout 100 generations, with a mutation probability of 0.02 and the crossover operator chosen is the One-point technique. The fitness function evaluates area or semi-perimeter subject to what was intended for the final optimization minimization. For both table headers, methods are described as HECA, BLF, FFDH and 'BUA' for the bottom-up approach (Sangwan et al., 2013).

Table 1 - Results for the area as fitness function

	HECA		**BLF**		**FFDH**		**BUA**	
# basics	**fitness**	**elapsed time (ms)**	**fitness**	**elapsed time (ms)**	**fitness**	**elapsed time (ms)**	**fitness**	**elapsed time (ms)**
100	**34398**	7801	51920	1	84961	1	129050	1
250	**66010**	20918	68992	2	180727	1	302500	2
500	164135	44039	**93899**	2	372830	1	656000	2
1000	444423	89621	**125378**	4	718669	3	1253750	5

Table 1 presents the results for the area as the fitness function selected for the C&P experiment.

Table 2 - Results for the semi-perimeter as fitness function

	HECA		**BLF**		**FFDH**		**BUA**	
# basics	**fitness**	**elapsed time (ms)**	**fitness**	**elapsed time (ms)**	**fitness**	**elapsed time (ms)**	**fitness**	**elapsed time (ms)**
100	**69290**	6923	69552	1	195596	1	6272516	2
250	261928	17819	**155386**	2	394150	1	39894356	3
500	741955	42431	**246634**	2	775031	1	166696421	3
1000	1734004	88507	**313865**	5	1682367	3	629861909	5

In Table 2, it is possible to visualize experimental data, for the C&P tests, regarding the semi-perimeter as the fitness function.

Analysing the results presented in Table 1 and Table 2, it is perceptible that as the number of basic rectangles grows, the computational time of all algorithms is proportionally larger, what was to be expected. Moreover, HECA takes more time to find a solution, in average, in four orders of magnitude when comparing to the second slower algorithm, BUA. Nevertheless, the new approach can find a solution from 8 to 90 seconds and 7 to 89 seconds, approximately, in 100 and 1000 basic rectangles instances when the area or the semi-perimeter is chosen as the fitness function, respectively. To be more specific, in the new hybrid evolutionary approach, it is the genetic search that weights more in the computational time. Since the number of basic rectangles is larger, the chromosome (vector) that holds basic rectangles is, consequently, longer, and the genetic cycle takes proportionally more time to conclude. Nevertheless, these are acceptable computational times when a heuristic is applied to VLSI chip design.

Before comparing fitness values, it is important to state that the BLF heuristic did not place all the basic rectangles in every instance. This happens because the width of the placement generated by HECA is used as input for the width of the container for the BLF method to create a solution. As HECA considers item rotation and the BLF method does not, it is possible that a given number of basic rectangles cannot be placed in the container because the width of the solution produced by HECA by allowing rotations is smaller than the basic rectangles width presented for the BLF method. This is the reason why BLF presented higher fitness values than the new approach in 5 of the 8 instances tested. Disregarding BLF fitness values, the new approach outperformed all the other comparison algorithms, i.e., the solutions created have the lowest values of area or semi-perimeter. This translates in a better compaction of the placed items or, alternatively, less wastage of the available space.

4.2. VLSI experiment

In this section, HECA was confronted against VLSI benchmarks, namely GSRC hard problems (Kahng & Markov, 1999) and MCNC (Microelectronics Center of North Carolina, 2012) instances. Firstly, GSRC results will be presented and discussed and then, MCNC experimental tests are described and its deliberation is introduced.

4.2.1. MCNC benchmarks

The MCNC circuit benchmark instances (Microelectronics Center of North Carolina, 2012) were used to counterweigh optimal placement with our new approach and other comparison methods. Table 3 describes the number of modules and total sum of individual rectangle areas for each MCNC instance. The total area of the modules corresponds to adding up all the individual basic rectangle areas.

Other methods for comparison are indispensable to support the new approach viability and validity. Therefore, various algorithms were chosen among the related literature, being one of them a classical C&P - the FFDH meta-heuristic (Johnson et al., 1974). Having in mind approaches for floorplanning in VLSI circuit problems, some more recent algorithms were chosen like a bottom-up approach for placement and compaction of modules for floorplanning in VLSI circuits that makes use of a method of composite block formation using macro modules (Sangwan et al., 2013). Both approaches were directly implemented, which justifies the possibility of presenting computing times. To complete the comparison in terms of evolutionary algorithms, the new hybrid evolutionary heuristic is directly confronted with the results found in literature of the same nature. Therefore, experimental results presented in (G. Chen, Guo, & Chen, 2010; J. Chen et al., 2011; Chyi-Shiang, Kanesan, & Harikrishnan, 2013; He, Ji, & Li, 2015; Sangwan et al., 2013; Sivaranjani & Kawya, 2013) are used.

Table 3 - MCNC benchmark instances attributes

MCNC Circuit	Number of Modules	Total Area of Modules (mm^2)
apte	9	46.5616
xerox	10	19.3503
hp	11	8.8306
ami33	33	1.1564
ami49	49	35.4454

The tests were performed in a computer with 8GB RAM (DDR3) and an Intel Core i5 3317U 1.70GHz CPU.

For each MCNC instance, tests were performed 12 times and the average value for the computational time and area occupation is presented. Regarding the new evolutionary approach proposed, for each instance, 50 chromosomes are randomly created for the initial population, that is then evolved throughout 100 generations. The fitness function evaluated the occupied area for each solution chromosome.

The methods presented in Table 4 and Table 5 headers are: HECA, 'BUA' for the bottom-up approach (Sangwan et al., 2013), 'HSA' for the hybrid simulated annealing method (J. Chen et al., 2011), 'PSO' for the PSO-based intelligent decision algorithm (G.

Chen et al., 2010), ‘CABF’ for the cost reduction technique (Chyi-Shiang et al., 2013), ‘DRA’ for the dynamic reduction heuristic (He et al., 2015), and, finally, ‘HPSO’ for the hybrid PSO (Sivaranjani & Kawya, 2013). The values shown in bold are the lower bound values.

Table 4 - HECA and other implemented methods for comparison results

MCNC Circuit	HECA		FFDH		BUA	
	Area (mm^2)	Dead space (%)	Area (mm^2)	Dead space (%)	Area (mm^2)	Dead space (%)
apte	**46.5919**	**0.065**	49.6974	6.31	47.9141	2.82
xerox	**19.3577**	**0.038**	23.9421	19.18	20.7317	6.66
hp	**8.8367**	**0.069**	12.9698	31.91	14.8078	40.37
ami33	**1.1581**	**0.147**	1.7762	34.89	5.7819	79.99
ami49	**35.464**	**0.052**	58.4389	39.35	124.0994	99.07

Table 4 introduces a comparison between HECA and other implement methods using MCNC benchmarks.

Table 5 - HECA and related literature area (mm^2) results

MCNC Circuit	HECA	HSA	PSO	CABF	DRA	HPSO
apte	**46.59**	46.92	46.92	47.3	46.92	47.44
xerox	**19.36**	20.01	20.38	19.9	19.94	20.2
hp	**8.84**	9.01	-	9.01	8.95	-
ami33	**1.16**	1.20	1.29	1.20	1.17	1.29
ami49	**35.46**	36.48	38.93	36.51	35.96	40.6

Table 5 presents the results gathered from related literature and executing HECA in MCNC benchmarks.

Analysing the results presented in Table 4, it is perceptible that the new approach generated placements with area occupation very close to the total sum of the modules individual areas (or optimal placement). This fact is supported by the percentage of dead space being nearly 0.15% in the worst-case scenario.

Results shown in Table 4 and Table 5 confirm the quality of placements produced by HECA since it presents the best solutions, i.e., floorplanning layouts with less dead area. Either comparing with the implemented comparison algorithms results (Table 4) or related literature methods results (Table 5), the new approach outperforms all methods in area minimization. In more detail, when comparing with the second-best referenced literature method, the DRA, the new approach produced a solution with less dead space by 0.7% in the apte instance (smallest difference value of dead space) and 2.9% in the xerox instance (highest difference value of dead space).

It can also be seen that the new approach consistently produces more compact solutions, from the smaller to the larger size instances.

4.2.2. GSRC benchmarks

The GSRC hard benchmarks (Kahng & Markov, 1999) are used to compare the results between the new hybrid evolutionary approach and others found in the related literature (G. Chen et al., 2010; Chyi-Shiang et al., 2013; He et al., 2015; C.-Y. Wang, Luo, & Jan, 2009) regarding VLSI floorplanning layout. In Table 6, optimal placement area values are visualized. These data help to understand the quality of solutions produced by a given approach, by comparing, in this case, the optimal area against the area produced by a placement of a certain approach. The values presented have as much decimal places as possible for more precision while comparing our new approach and other methods.

Table 6 - GSRC benchmark optimal placement area values

GSRC circuit	Optimal placement area (mm^2)
n10	0.22168
n30	0.20859
n50	0.19858
n100	0.17930
n200	0.17540
n300	0.27317

The tests were performed in a computer with 8GB RAM (DDR3) and an Intel Core i5 3317U 1.70GHz CPU. For each GSRC instance, tests were performed 40 times and the average value for the area occupation is presented. Regarding the new hybrid evolutionary approach, for each instance, 50 chromosomes are created randomly for the initial population, that is then evolved throughout 100 generations. The fitness function evaluated the occupied area for each solution chromosome.

In Table 7, results from HECA and related literature algorithms are presented. The methods presented in Table 7 header are: HECA, 'PSO' for the PSO-based intelligent decision algorithm (G. Chen et al., 2010), 'FAE' for the Full-and-Elimination approach (C.-Y. Wang et al., 2009), 'CABF' for the cost reduction technique (Chyi-Shiang et al., 2013), and, finally, 'DRA' for the dynamic reduction heuristic (He et al., 2015). The values shown in bold are the lower bound values.

There is related literature where area values are given in square millimetres (mm2) (G. Chen et al., 2010; Chyi-Shiang et al., 2013), or area utilization (%) (He et al., 2015; C.-Y. Wang et al., 2009). To be coherent in the result appreciation and discussion, all values are presented in square millimetres. Therefore, equation (2) was applied to transform area utilization in square millimetres.

$$Area\ utilization\ (mm^2) = \frac{Optimal\ placemen\ \ (mm^2)}{Area\ utilization\ (\%)} * 100 \qquad (2)$$

Moreover, Wang et al. and Chyi-Shiang et al. tested variations of the same GSRC instance (same modules with different number of pins, terminal and nets) for their algorithm FAE, because their work also includes wirelength as an objective function. In these cases, only the best solutions in terms of area minimization were used in the comparison with the new approach.

Table 7 - HECA and related literature area (mm^2) results

GSRC circuit	HECA	PSO	FAE	CABF	DRA
n10	**0.222**	-	-	0.227	0.228
n30	**0.209**	0.234	-	0.216	0.213
n50	**0.199**	0.222	-	0.206	0.202
n100	0.182	-	0.197	0.187	**0.181**
n200	0.202	-	0.185	0.185	**0.176**
n300	0.304	-	0.289	0.296	**0.274**

It is also important to note that in the referred literature, in some cases, not all GSRC benchmark instances were tested. Therefore, when no result is defined, a " - " character appears.

Analysing results shown in Table 7, it can be concluded that the new approach outperforms other methods for the smaller size instances (n10, n30 and n50). Regarding larger size instances (n100, n200 and n300), the DRA heuristic outperforms not only the new approach but also other related literature work. The data representation used in the DRA method (B*-tree) is also an advantage when comparing to the binary tree representation of HECA, since B*-trees are VLSI problems oriented.

Having in mind that the new approach is not a VLSI floorplanning problem oriented but a more general two-dimensional placement and compaction approach, it is safe to say that for small size instances the new approach is a valid pick for tackling VLSI floorplanning problems. Regarding larger size instances, specific VLSI floorplanning algorithms are better suitable producing less area utilization solutions.

4.2.3. Crossover operators and evolutionary population size influence

Another set of experiments were done to study how two different crossover genetic operators (One-Point crossover and Random Respectful crossover) influence HECA's potential for creating solutions for the two-dimensional placement problem with guillotine cutting constraint. Moreover, the size of the evolutionary population was also studied, being set to range in {750, 1500, 2250, 3000, 4500} individuals. Since HECA faired worst over the three largest size GSRC hard benchmarks, these were used for this test, namely the instances n100, n200 and n300.

The experiments were performed in a computer with 8GB RAM (DDR3) and an Intel Core i5 3317U 1.70GHz CPU. For each GSRC instance, tests were performed 12 times and the average value for the area occupation and elapsed time was noted. Regarding the new hybrid evolutionary approach, for each GSRC instance, different number of chromosomes were tested: 50, 150, 300, 450 and 600 individuals. In each chromosome instance, individuals are created randomly for the initial population, that is then evolved throughout a variable number of generations: 750, 1500, 2250 and 3000. Finally, the fitness function evaluated the occupied area for each solution produced.

Results presentation and discussion will be performed, firstly, for the fitness as the evaluation parameter and, lately, the elapsed time experiment will be approached, and conclusions will be drawn.

The tables presented in Appendice B (Tables 8 to 13) describe the results regarding the variation of the fitness parameter for the OPC and RRC. Nevertheless, in order to better understand how the fitness value is influenced with the different crossover techniques, Figure 10, Figure 11 and Figure 12 were introduced to translate a more visual and graphical meaning to the data presented in the previous tables.

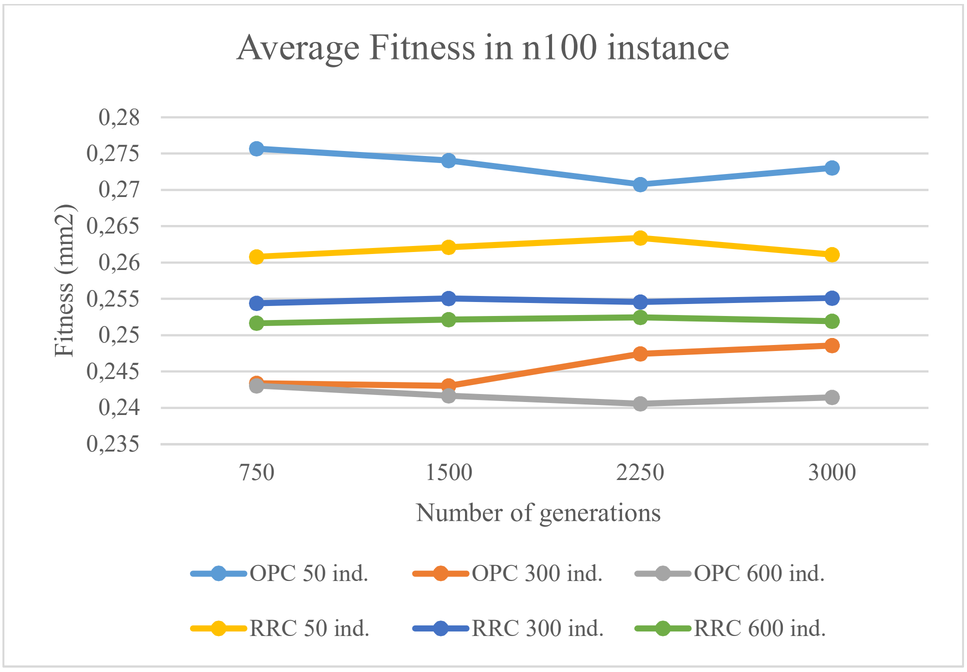


Figure 10 - Fitness comparison for n100 instance

Figure 10 shows the results for the fitness parameter variation with the two crossover genetic operators for the n100 instance presented in Table 8 and Table 11.

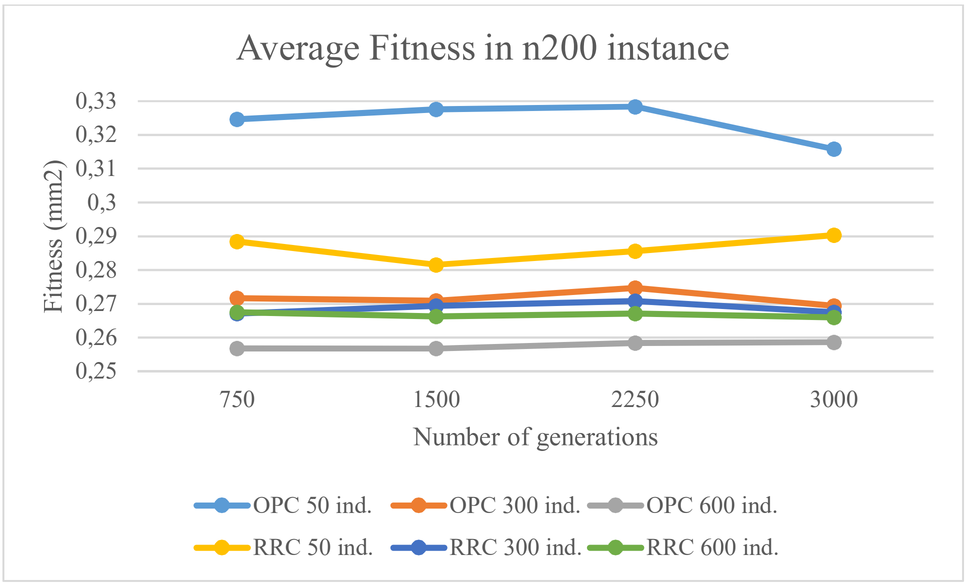


Figure 11 - Fitness comparison for n200 instance

Figure 11 shows the results for the fitness variation for the n200 instance with the two crossover techniques that are presented in Table 9 and 12.

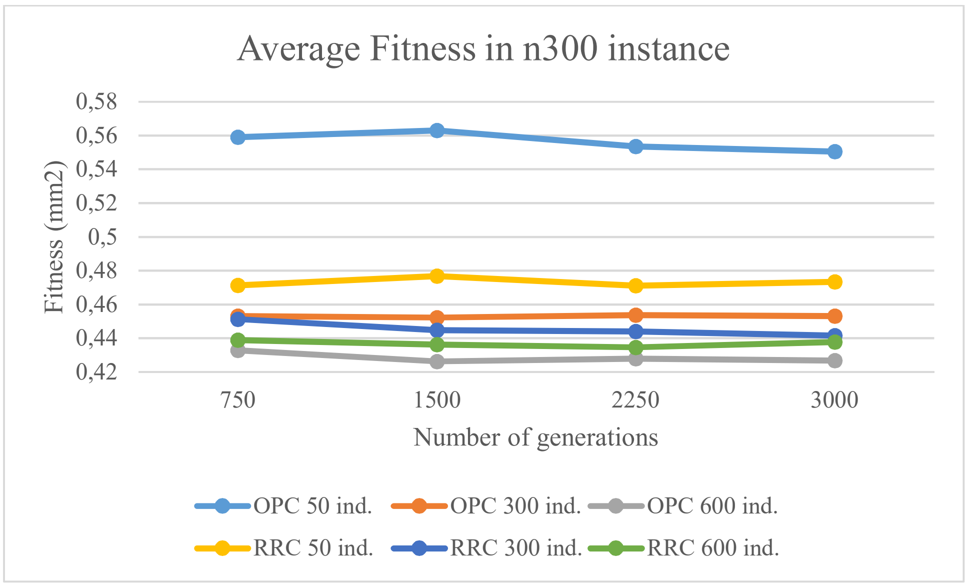


Figure 12 - Fitness comparison for n300 instance

In Figure 12, it is possible to understand how the fitness parameter evolves through the experiments whose results are projected in Table 10 and Table 13.

In Figure 10, Figure 11 and Figure 12 only instances with 50, 300 and 600 chromosomes were shown so it is visually clearer how the number of individuals and generations increase affects the fitness of generated placements.

Regarding Figure 10 and comparing the values for the same chromosome instance and different crossover technique, OPC 50 ind. RRC 50 ind., OPC 300 ind. with RRC 300 ind., and OPC 600 ind. with RRC 600 ind., it is possible to conclude that for the 50 and 300 individual instances, the RRC crossover helps to produce solutions with better (lower) fitness. However, in the instance with the largest number of individuals, 600 chromosomes, the OPC crossover overcomes the other technique.

Relatively to Figure 11 and Figure 12, which visually represents the results described in Table 9 and Table 12, and Table 10 and Table 13, respectively, the same behaviour can be seen as it happened with the n100 instance presented in Figure 10. It is also important to note that, in general, both crossover operators help HECA to produce more compact solutions, in average, when the number of individuals augments.

With the previous analysis, it can be concluded that for less or equal than 300 individual instances, the RRC technique is the best choice as a crossover genetic operator for the evaluated fitness parameter.

Regarding the variation of the elapsed time parameter for the OPC and RRC techniques, Table 14, Table 15, Table 16, Table 17, Table 18 and Table 19 (shown in Appendice B) evidences the results executed with the new hybrid approach.

Once again, a visual presentation is needed to infer conclusions and truthfully analyse the data introduced in the previous tables. Therefore, Figure 13, Figure 14 and Figure 15 translate the results shown in Table 14 and Table 17, Table 15 and Table 18 and, finally, Table 16 and Table 19, respectively.

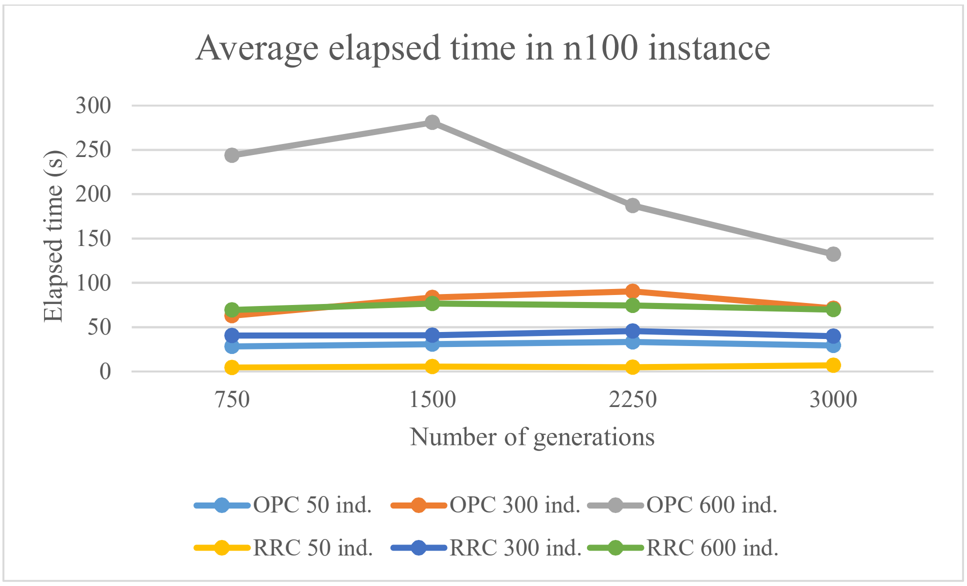


Figure 13 - Elapsed time comparison for n100 instance

Figure 13 presents the results in the n100 instance for 50, 300 and 600 individuals and the elapsed time needed for HECA to produce a final placement, either with the OPC or the RRC techniques.

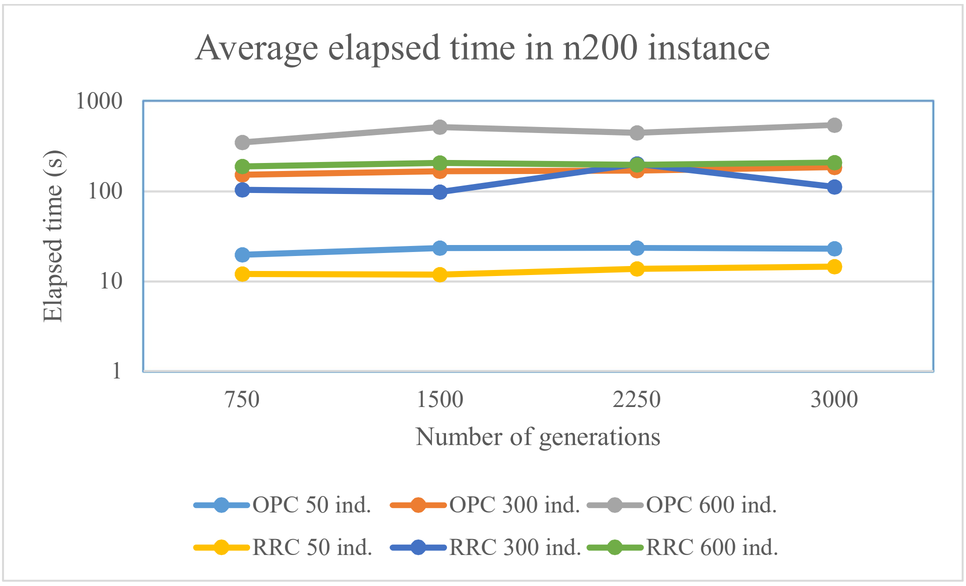


Figure 14 - Elapsed time comparison for n200 instance

In Figure 14, it is possible to visualize the results for the n200 instance experiments and how the elapsed time is affected as the number of chromosomes increases.

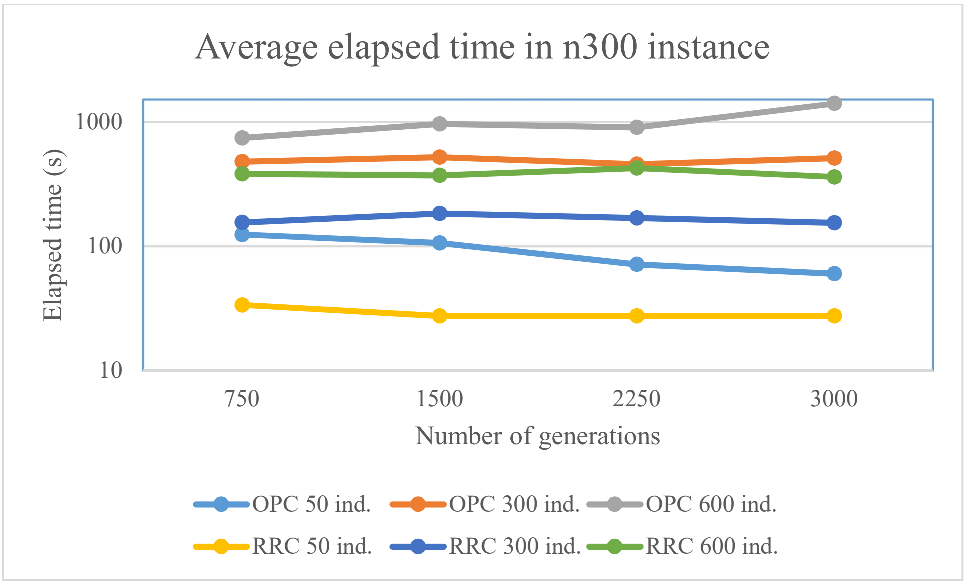


Figure 15 - Elapsed time comparison for n300 instance

Finally, Figure 15 represents the tests performed using the n300 instance for the elapsed time parameter while variating the number of individuals and generations.

Analysing Figure 13, it is clear that for each instance where the number of individuals is different (50, 300 and 600 chromosomes), the computational time needed to produce a solution is always smaller when using the RRC technique. Moreover, it can be seen that every line but one presented in Figure 13 is almost flat, which translates in approximately the same elapsed time, no matter what the number of generations is. The difference in the elapsed time is given by the number of chromosomes in each instance, i.e., the higher the number of chromosomes the longer the time spent to find a solution.

Figure 14 and Figure 15 exhibit the same behaviour between both crossover techniques as it is shown in Figure 13, therefore concluding that the RRC method outperforms the OPC technique for every instance but one regarding the elapsed time parameter. This exception is visualized in Figure 14 for the 300 individuals instance and with 2250 generations. It is also important to elicit that Figure 14 and Figure 15 elapsed time axis is logarithmic, which translates in an even larger difference between both crossover methods.

The objective of the study for the two crossover techniques is to know, effectively, which one is more suitable to HECA in order to produce a solution with the best commitment between fitness and elapsed time, i.e., less area wastage in less time. Therefore, it is safe to say, and given the data and visual representation shown, that the best crossover technique is the RRC.

More intensive experiments were performed with HECA using n100 and n200 instances and the RRC crossover technique, to understand how the variation of the number of chromosomes helps to produce a more compact placement. Moreover, two stopping conditions were implemented to limit the elapsed time for HECA to run, as these tests were much more time consuming than the previous ones. The stopping conditions are:

- If HECA does not generate a solution more compact (smaller fitness) than in the previous X generations then the method stops executing;
- If HECA finds the optimal solution, then the algorithm ends its execution.

Note that X is a variable that the user can parameterize. In this experiment, the number of generations used for the stopping condition is 2500.

The experiments were performed in a computer with 8GB RAM (DDR3) and an Intel Core i5 3317U 1.70GHz CPU.

In Table 20 and Table 21 (introduced in Appendice C), results are introduced. The discussion of this data is more efficient if a visual element is presented. Therefore, Figure 16, Figure 17, Figure 18 and Figure 19 are shown below.

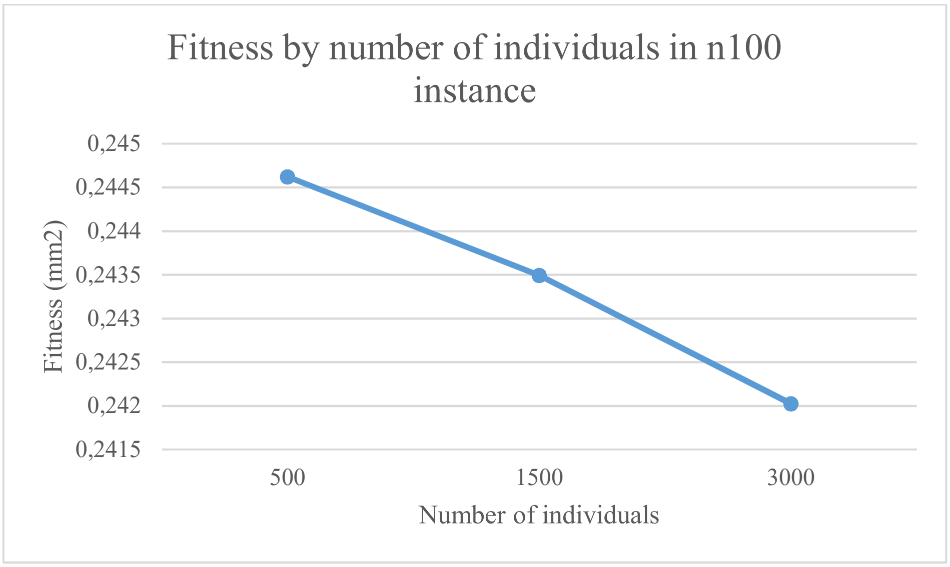


Figure 16 - Fitness by number of individuals in n100 instance

Figure 16 translates the fitness variation as the number of individuals increases for the n100 instance.

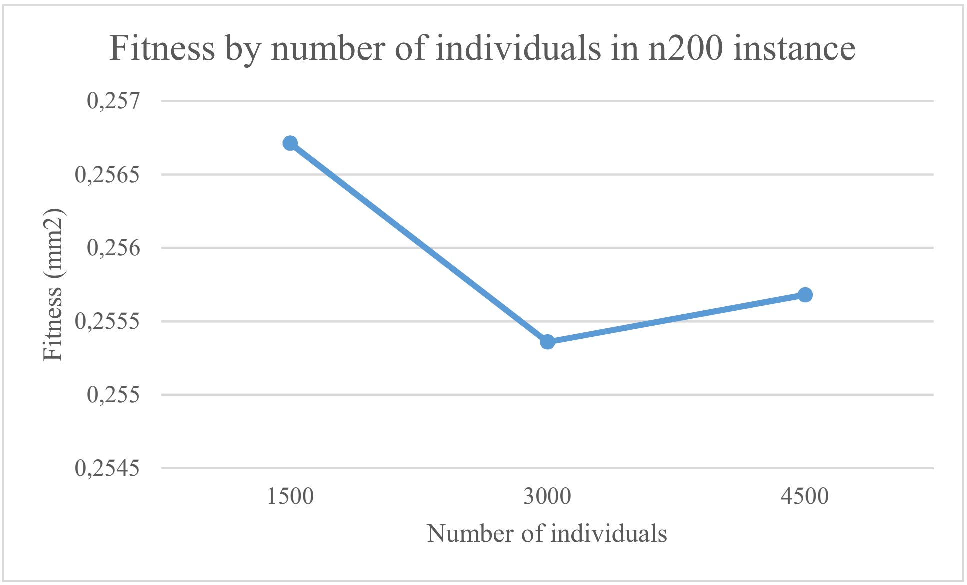


Figure 17 - Fitness by number of individuals in n200 instance

Figure 17 presents a visual element of how the fitness parameter evolves throughout the experiment with the n200 instance where the number of chromosomes grows from 1500 to 4500 individuals.

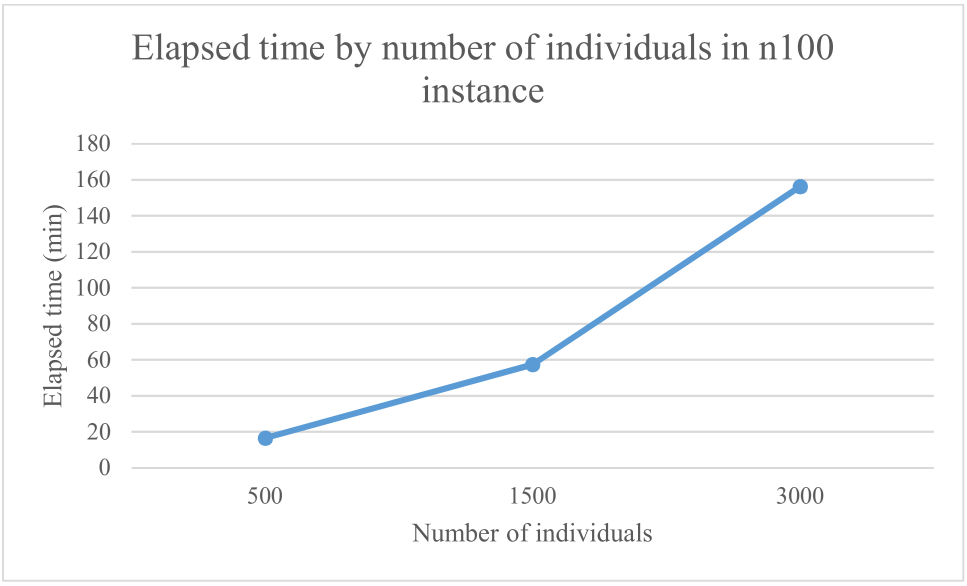


Figure 18 - Elapsed time by number of individuals in n100 instance

Figure 18 introduces a graphic with the elapsed time where a non-linear increase can be seen when the number of individuals augments for the n100 instance.

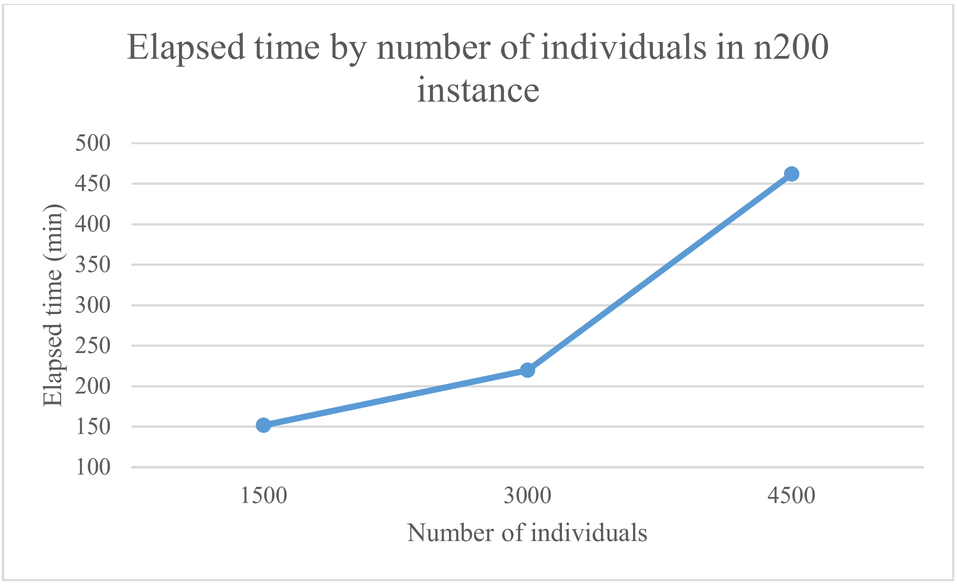


Figure 19 - Elapsed time by number of individuals in n200 instance

At last, Figure 19 provides an understanding of the elapsed time evolution directly related to the number of chromosomes increasing when the n200 instance is tested.

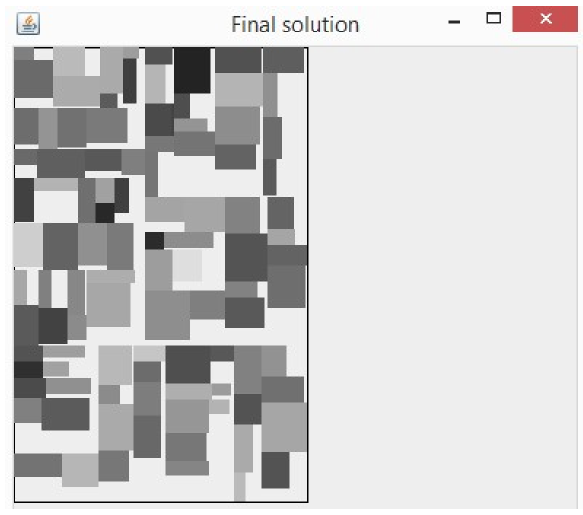
Analysing Figure 16 and Figure 17, the fitness (area) of solutions produced by HECA decreases as the number of individuals grows, in almost every experiment with n100 and n200 instances. The exception can be seen in Figure 17, where the fitness value increases when the number of individuals grows from 3000 to 4500 chromosomes, i.e., the solution created with 3000 chromosomes has less wastage area then the one generated with 4500 individuals. This can be due to the genetic drift phenomena, i.e., in the 4500 chromosomes instance, HECA entered a minimum local optima area and was not able to produce a placement with a lower value of fitness than the solution produced with 3000 individuals.

As for the elapsed time parameter variation, measured in minutes in these experiments, it is faster to find a solution for the n100 instance with the smallest number of individuals (500), as expected. In this case, it took 16 minutes and 33 seconds, in average, to find a final placement. Still in the n100 experiment, and looking at the instance with 3000 individuals, HECA can generate a solution in nearly 2 hours and 36 minutes. As for the n200 instance, in the least number of individuals (1500), HECA could find a solution in, approximately, 2 hours and 32 minutes. When the test with the highest number of individuals (4500) was executed, the new hybrid approach was able to generate a final placement in almost 7 hours and 42 minutes.

When correlating this elapsed time differences with the placement's area (or fitness) created, it can be concluded that increasing the number of individuals does not help to create solutions proportionally compact. In other words, the increase of the number of individuals will lead to much more computation time and it will not reverberate in much less wastage area solutions. Ideally, fitness and computational time would be inversely proportional until a certain fitness threshold – the optimal solution.

After HECA finds the final solutions for each benchmark tested, the placement created pops up in a window, so the user can inspect how basic rectangles were disposed and where empty spaces are located.

Figure 20 and Figure 21 show placements generated for n100 and n200, and n300 instances, respectively. The black line that surrounds basic rectangles represents the final enclosing rectangle.



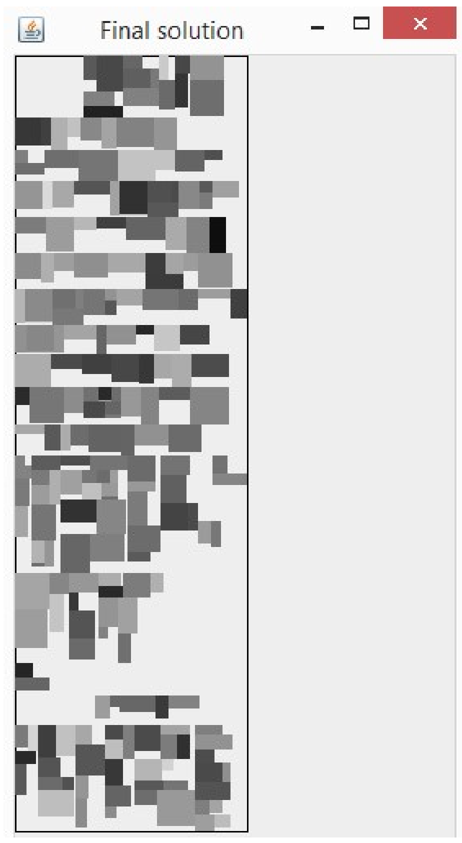


Figure 20 - Placement creation for the n100 (left) and n200 (right) instances

Figure 20 presents two generated placements using HECA for the n100 (left) and the n200 (right) instances.

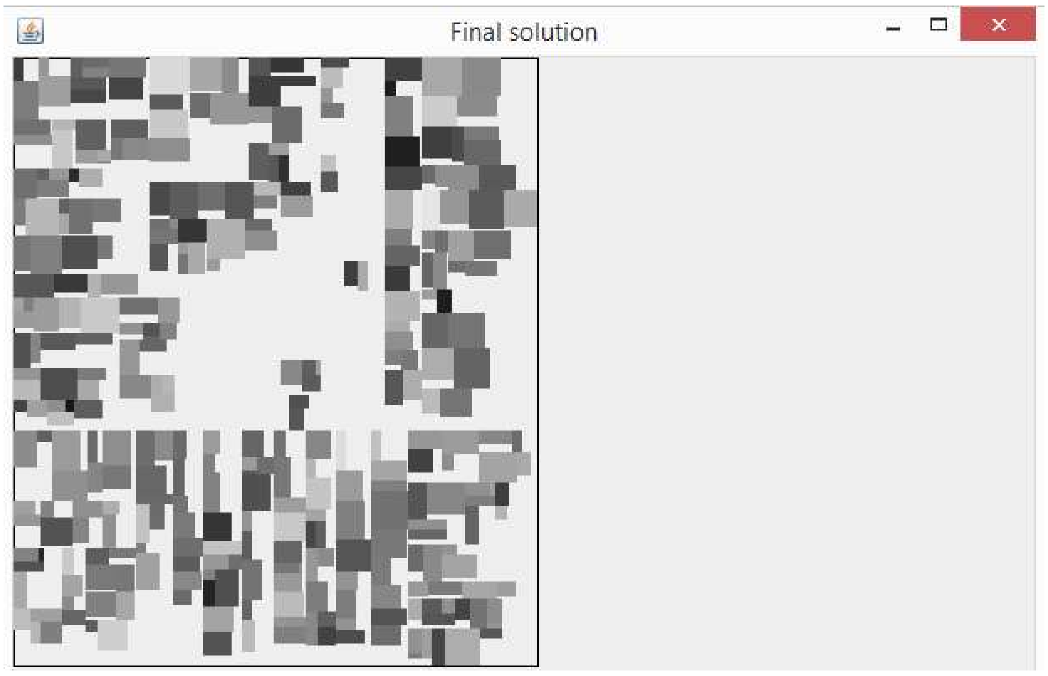


Figure 21 - Placement created for the n300 instance

Figure 21 presents a solution created by the new approach for the n300 instance.

When analysing the placements created, it can be concluded that, opposed to what happens for the smaller instances and benchmarks, HECA produces solutions with a lot of empty spaces for these larger instances. This is explained by the guillotine cutting constraint that ensures that a clean guillotine cut can be performed in the placement, edge-to-edge, without damaging any basic rectangle, limiting the space available to allocate rectangles.

Chapter 5 – Conclusions and recommendations

5.1. Main conclusions

A new hybrid evolutionary compaction approach was introduced to address the placement problem of two-dimensional objects over rectangular surfaces while using certain utility functions. Two constraining placement rules have been assumed: basic rectangles could be rotated, and they had to be disposed in such a way that would permit an edge-to-edge guillotine cut in the final enclosing rectangle.

This problem was categorized as NP-hard as it is unsolvable in a lifetime period with the help of classic algorithms for larger quantities of rectangular components. The newly proposed heuristic merges a genetic algorithm with a non-linear compaction method to produce solutions with a balanced commitment between the elapsed computational time and the compactness of the generated placement.

A study was presented in which the new algorithm was compared with referred literature and a given number of VLSI typical benchmark problems.

In Chapter 4, results were shown, and its discussion was introduced as it helps to understand how HECA presents feasible solutions and how much time it takes to generate them.

Recalling research questions stated in section 1.3, the main inquiry is to know in which measure does HECA compares with the referred literature methods? As it was described in the previous Chapter, HECA can present suitable solutions that rival with state-of-the-art algorithms. It is also important to remember that none of these algorithms respond to the guillotine cutting constraint. The absence of this limitation helps to find solutions more compact and consequently, with less empty spaces.

Another secondary research question is to understand how HECA behaves when having distinct genetic operators. The study that compared RRC and OPC techniques answers exactly to this question as, firstly, HECA was executed with the OPC technique for experiments introduced in sections 4.1 and 4.2.1. Lately, in section 4.2 it was shown that the RRC technique leads HECA to produce more compact solutions than the OPC method for larger size instances with 300 or less chromosomes to explore.

Finally, given the results and the respective discussion presented in Chapter 4, it can be concluded that HECA generates placements with more wastage area as the instance size increases (number of items to allocate grows), but in a satisfactory computational time. In section 5.4, recommendations are made for future work that may help produced solutions closer to the optimal.

5.2. Contributions to the scientific community

5.2.1. Scientific level implications

The general two-dimensional placement problem is a well-known challenge in the scientific community as researchers have addressed this problem for 50 years, approximately. As the scientific community knows, this type of problems are real as common people face them in many different areas of the industry (for the cutting and packing of materials like wood, glass and metal), transportation, logistics and warehousing. Moreover, 2-D placement problems are also found in simple problems, like disposing written articles in a magazine page, or much more complex challenges like placing components in a chip.

The HECA method follows a not so typical approach as it uses a non-linear compaction method. As it was determined after studying the referred literature, researchers do not often use compaction algorithms in its heuristics. Therefore, HECA can add value to the scientific community in such a way that can, maybe, lead to new methods that use new compaction methods as they have proven its value.

5.3. Study limitations

The new hybrid approach presented is an alternative two general two-dimensional placement problems. Nevertheless, the experimental phase strongly focused on VLSI floorplanning layout problems. The data representation model (complete binary tree) used in HECA can lead to solutions that are sub-optimal when comparing to other VLSI oriented approaches. Moreover, the guillotine cutting constraint helps to produce placements whose area or semi-perimeter is larger than desired. As seen in Chapter 4, HECA behaves worst in larger size instances and the data representation model chosen can be intimately related to it.

5.4. Proposals for future research

The study presented here can certainly be extended to a Philosophiæ Doctor dissertation as there is still much more work to be done. Therefore, future research can be focused on exploring different data representations like the B*-tree (Chang et al., 2000), for instance. Also, other different genetic operators can be tested, not only crossover techniques, but also distinct selection methods. Experimental tests can be rerun, and conclusions can be drawn. The genetic drift phenomena can also be addressed in more detail by varying the mutation probability or even devising new mutations.

Bibliography

Ahmad, A.R. (2015). A New Hierarchical Placement Algorithm for Two-Dimensional Rectangular Layout Design [International Journal of Operations and Quantitative Management, 21(2), pp. 79-98, June 2015]. *International Journal of Operations and Quantitative Management*, *21*, 79–98.

Almeida, A. M. ., Matins, Q. V., & Rodrigues, R. D. (1998). Optimal Cutting Directions and Rectangle Orientation Algorithm. *European Journal Of Operational Research*, *109*(3), 660–671. https://doi.org/10.1016/S0377-2217(97)00085-4

Alvarez-Valdes, R., Parajon, A., & Tamarit, J. M. (2002). A computational study of LP-based heuristic algorithms for two-dimensional guillotine cutting stock problems. *OR Spectrum*, *24*(2), 179–192. https://doi.org/10.1007/s00291-002-0093-3

Andrew Powell-Morse. (2016). Waterfall Model: What Is It and When Should You Use It? Retrieved January 20, 2018, from https://airbrake.io/blog/sdlc/waterfall-model

Aryanezhad, M.-B., F. Hashemi, N., Makui, A., & Javanshir, H. (2012). A simple approach to the two-dimensional guillotine cutting stock problem. *Journal of Industrial Engineering International*, *8*.

Baker, B. S., Coffman, Jr., E. G., & Rivest, R. L. (1980). Orthogonal Packings in Two Dimensions. *SIAM Journal on Computing*, *9*(4), 846–855. https://doi.org/10.1137/0209064

Bansal, N., Lodi, A., & Sviridenko, M. (2005). A Tale of Two Dimensional Bin Packing. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (pp. 657–666). Washington, DC, USA: IEEE Computer Society. https://doi.org/10.1109/SFCS.2005.10

Beasley, J. E. (2000). A population heuristic for constrained two-dimensional non-guillotine cutting, (May).

Blum, C., & Schmid, V. (2013). Solving the 2D bin packing problem by means of a hybrid evolutionary algorithm. *Procedia Computer Science*, *18*, 899–908. https://doi.org/10.1016/j.procs.2013.05.255

Bortfeldt, A. (2006). A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, *172*(3), 814–837. https://doi.org/10.1016/j.ejor.2004.11.016

Burke, E. K., Kendall, G., & Whitwell, G. (2004). A New Placement Heuristic for the Orthogonal Stock-Cutting Problem. *Oper. Res.*, *52*(4), 655–671. https://doi.org/10.1287/opre.1040.0109

Chang, Y.-C., Chang, Y.-W., Wu, G.-M., & Wu, S.-W. (2000). B*-Trees: a new representation for non-slicing floorplans. *Proceedings - Design Automation Conference*.

Chekanin, V. A., & Chekanin, A. V. (2017). Compaction algorithm for orthogonal packing problems. *IOP Conference Series: Materials Science and Engineering*, *248*, 012024. https://doi.org/10.1088/1757-899X/248/1/012024

Chen, G., Guo, W., & Chen, Y. (2010). A PSO-based Intelligent Decision Algorithm for VLSI Floorplanning. *Soft Comput.*, *14*(12), 1329–1337. https://doi.org/10.1007/s00500-009-0501-6

Chen, J., Zhu, W., & Ali, M. (2011). A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, *41*, 544–553.

Chen, T.-C., & Chang, Y.-W. (2005). Modern Floorplanning Based on Fast Simulated Annealing. In *Proceedings of the 2005 International Symposium on Physical Design* (pp. 104–112). New York, NY, USA: ACM. https://doi.org/10.1145/1055137.1055161

Chyi-Shiang, H., Kanesan, J., & Harikrishnan, R. (2013). Cost reduction in bottom-up hierarchical-based VLSI floorplanning designs. *International Journal of Circuit Theory and Applications*, *43*(3), 286–306. https://doi.org/10.1002/cta.1939

Cintra, G. F., & Wakabayashi, Y. (2004). Dynamic Programming and Column Generation Based Approaches for Two-Dimensional Guillotine Cutting Problems. *Lecture Notes in Computer Science*.

Clautiaux, F., Carlier, J., & Moukrim, A. (2007). A new exact method for the two-dimensional orthogonal packing problem. *European Journal of Operational Research*, *183*(3), 1196–1211. https://doi.org/https://doi.org/10.1016/j.ejor.2005.12.048

Clautiaux, F., Jouglet, A., Carlier, J., & Moukrim, A. (2008). A new constraint programming approach for the orthogonal packing problem. *Computers & Operations Research*, *35*(3), 944–959. https://doi.org/https://doi.org/10.1016/j.cor.2006.05.012

Dhiraj, Verma, S., & Kumar, R. (2013). A bottom up approach for placement and compaction of standard modules in VLSI circuit. *Proceedings of the 2013 International Conference on Advanced Electronic Systems, ICAES 2013*, 133–137. https://doi.org/10.1109/ICAES.2013.6659377

Emmert, J. M., & Bhatia, D. K. (2001). Two-dimensional Placement Using Tabu Search. *VLSI Design*, *12*(1), 13–23. Retrieved from 10.1155/2001/92781

Gilmore, P. C., & Gomory, R. E. (1961). A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research*, *9*(6), 849–859. Retrieved from http://www.jstor.org/stable/167051

Gomes, J., & Mourão, M. C. (2012). Heuristics for a 2D Guillotine Cutting Problem : A Wardrobes Application Heuristics for a 2D Guillotine Cutting Problem :

Gonçalves, J. F. (2007). A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem. *European Journal of Operational Research*, *183*(3), 1212–1229. https://doi.org/https://doi.org/10.1016/j.ejor.2005.11.062

Grandcolas, S., & Pinto, C. (2015). A New Search Procedure for the Two-dimensional Orthogonal Packing Problem. *Journal of Mathematical Modelling and Algorithms in Operations Research*, *14*(3), 343–361. https://doi.org/10.1007/s10852-015-9278-z

Hadjiconstantinou, E., & Iori, M. (2007). A hybrid genetic algorithm for the two-dimensional single large object placement problem. *European Journal of Operational Research*, *183*(3), 1150–1166. https://doi.org/10.1016/j.ejor.2005.11.061

Handa, M., & Vemuri, R. (2004). An Efficient Algorithm for Finding Empty Rectangles for Online FPGA Placement. *Proceedings of the Design Automation Conference*, 960–965.

He, K., Huang, W., & Jin, Y. (2012). An efficient deterministic heuristic for two-dimensional rectangular packing. *Computers and Operations Research*, *39*(7), 1355–1363. https://doi.org/10.1016/j.cor.2011.08.005

He, K., Ji, P., & Li, C. (2015). Dynamic reduction heuristics for the rectangle packing area minimization problem. *European Journal of Operational Research*, *241*(3), 674–685. https://doi.org/https://doi.org/10.1016/j.ejor.2014.09.042

Healy, P. (1999). An Optimal Algorithm for Rectangle Placement, 1–10.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. *Ann Arbor MI University of Michigan Press*. https://doi.org/10.1137/1018105

Johnson, D. S., Demers, A., Ullman, J. D., Garey, M. R., & Graham, R. L. (1974). Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. *SIAM Journal on Computing*, *3*(4), 299–325. https://doi.org/10.1137/0203025

Jonathan Rasmusson. (2014). What is Agile? Retrieved January 20, 2018, from http://www.agilenutshell.com/what_is_agile

Joncour, C., Pêcher, A., & Valicov, P. (2012). MPQ-trees for the orthogonal packing problem. *Journal of Mathematical Modelling and Algorithms*, *11*(1), 3–22. https://doi.org/10.1007/s10852-011-9159-z

Kahng, A. B., & Markov, I. (1999). GSRC Benchmark Module and Netlist for Floorplanning and Placement. Retrieved June 22, 2018, from http://vlsicad.eecs.umich.edu/BK/GSRCbench/HARD/

Korte, N., & Möhring, R. (1989). An Incremental Linear-Time Algorithm for Recognizing Interval Graphs. *SIAM Journal on Computing*, *18*(1), 68–81. https://doi.org/10.1137/0218005

Lesh, N., Marks, J., McMahon, A., & Mitzenmacher, M. (2004). Exhaustive approaches to 2D rectangular perfect packings. *Information Processing Letters*, *90*(1), 7–14. https://doi.org/10.1016/j.ipl.2004.01.006

Lodi, A., Martello, S., & Monaci, M. (2002). Two-dimensional packing problems: A survey. *European Journal of Operational Research*, *141*(2), 241–252. https://doi.org/https://doi.org/10.1016/S0377-2217(02)00123-6

Martello, S., & Vigo, D. (1998). Exact Solution of the Two-Dimensional Finite Bin Packing Problem. *Management Science*, *44*.

Microelectronics Center of North Carolina. (2012). MCNC Benchmark Netlists for Floorplanning and Placement. Retrieved June 10, 2018, from https://s2.smu.edu/~manikas/Benchmarks/MCNC_Benchmark_Netlists.html

Mrad, M., Meftahi, I., & Haouari, M. (2013). A Branch-and-Price Algorithm for the Two-Stage Guillotine Cutting Stock Problem. *The Journal of the Operational Research Society*, *64*.

Oliveira, J. F., Neuenfeldt Júnior, A., Silva, E., & Carravilla, M. A. (2016). A Survey on Heuristics for the Two-Dimensional Rectangular Strip Packing Problem. *Pesquisa Operacional*, *36*(2), 197–226. https://doi.org/10.1590/0101-7438.2016.036.02.0197

Pál, L. (2006). A Genetic Algorithm for the Two-dimensional Single Large Object Placement Problem. *3rd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, Timisoara*.

Pintea, C. M., Pascan, C., & Măcelaru, M. H. (2012). Comparing several heuristics for a packing problem. *International Journal of Advanced Intelligence Paradigms*, *4*(3/4), 268. https://doi.org/10.1504/IJAIP.2012.052071

Radcliffe, N. J. (1991). Forma Analysis and Random Respectful Recombination. In *ICGA*.

Saika, S., Fukuui, M., Shinomiya, N., Akino, T., & Kuninobu, S. (1997). A two-dimensional transistor placement algorithm for cell synthesis and its application to standard cells. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E80–A*(10), 1883–1890. https://doi.org/10.1109/ASPDAC.1997.600335

Sangwan, D., Verma, S., & Kumar, R. (2013). A bottom up approach for placement and compaction of standard modules in VLSI circuit. *Proceedings of the 2013 International Conference on Advanced Electronic Systems, ICAES 2013*.

Sivaranjani, P., & Kawya, K. K. (2013). Article: Performance Analysis of VLSI Floor planning using Evolutionary Algorithm. *IJCA Proceedings on International Conference on Innovations In Intelligent Instrumentation, Optimization and Electrical Sciences*, *ICIIIOES*(9), 42–46.

Stockmeyer, L. (1983). Optimal orientations of cells in slicing floorplan designs. *Information and Control*, *57*(2–3), 91–101. https://doi.org/10.1016/S0019-9958(83)80038-2

Wang, C.-Y., Luo, C., & Jan, G. E. (2009). An efficient Full-and-Elimination approach for floorplan area minimization. *2009 International Conference on Microelectronics - ICM*, 244–247.

Wang, S. (2017). Solving Rectangle Packing Problem Based on Heuristic Dynamic Decomposition Algorithm Shi WANG 1,2,3 1, (Eeta), 187–196.

Wäscher, G., Haußner, H., & Schumann, H. (2007). An improved typology of cutting and packing problems. *European Journal of Operational Research*, *183*(3), 1109–1130. https://doi.org/10.1016/j.ejor.2005.12.047

Attachments and Appendices

Appendice A

The present master thesis was supported by the implementation of HECA in Java programming language. The source code was uploaded to Github and can be inspected through this link.

Appendice B

Table 8 - One-Point crossover fitness (mm^2) experiment for n100 instance

Technique		**One-Point crossover**				
Instance		**n100**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	0.275679	0.254276	0.243385	0.240435	0.243039
	1500	0.274062	0.247843	0.243023	0.243102	0.24167
	2250	0.270765	0.2506	0.247432	0.243059	0.240557
	3000	0.273053	0.25597	0.248577	0.2404	0.241414

Table 9 - One-Point crossover fitness (mm^2) experiment for n200 instance

Technique		**One-Point crossover**				
Instance		**n200**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	0.324654	0.292038	0.271715	0.265286	0.256796
	1500	0.327535	0.279454	0.27091	0.259893	0.256773
	2250	0.328371	0.281939	0.27472	0.258422	0.258441
	3000	0.315782	0.285891	0.269403	0.257281	0.258652

Table 10 - One-Point crossover fitness (mm^2) experiment for n300 instance

Technique		**One-Point crossover**				
Instance		**n300**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	0.559067	0.47838	0.453071	0.436428	0.432825
	1500	0.562954	0.467189	0.452257	0.433819	0.4263
	2250	0.553471	0.483142	0.453623	0.440167	0.427853
	3000	0.550443	0.472511	0.453196	0.433251	0.426845

Table 11 - Random Respectful crossover fitness (mm^2) experiment for n100 instance

Technique		Random Respectful crossover				
Instance		**n100**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	0.260801	0.256326	0.254391	0.252072	0.251641
	1500	0.262123	0.255482	0.255053	0.253743	0.252127
	2250	0.263386	0.25561	0.254562	0.253185	0.252462
	3000	0.261105	0.25765	0.255114	0.253187	0.251941

Table 12 - Random Respectful crossover fitness (mm^2) experiment for n200 instance

Technique		Random Respectful crossover				
Instance		**n200**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	0.288429	0.271651	0.267121	0.267392	0.267543
	1500	0.281578	0.274177	0.269455	0.265838	0.266271
	2250	0.285614	0.271477	0.270819	0.265419	0.267124
	3000	0.290341	0.274009	0.267506	0.266697	0.266031

Table 13 - Random Respectful crossover fitness (mm^2) experiment for n300 instance

Technique		Random Respectful crossover				
Instance		**n300**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	0.471236	0.4487	0.451308	0.441697	0.438912
	1500	0.476751	0.45017	0.444865	0.444551	0.436224
	2250	0.471137	0.450287	0.444085	0.444346	0.434646
	3000	0.473298	0.446759	0.441517	0.442826	0.437832

Table 14 - One-Point crossover elapsed time (s) experiment for n100 instance

Technique		One-Point crossover				
Instance		**n100**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	28.17975	74.68342	62.9365	99.28792	243.6471
	1500	30.85858	79.68992	83.50275	89.31708	280.841
	2250	33.361	28.86942	90.30442	107.2638	187.1361
	3000	29.19817	27.64908	71.05708	299.9696	132.2273

Table 15 - One-Point crossover elapsed time (s) experiment for n200 instance

Technique		**One-Point crossover**				
Instance		**n200**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	19.62308	79.4985	151.7869	667.0067	346.1704
	1500	23.31225	86.49025	166.0496	337.6389	510.3709
	2250	23.36658	81.756	168.4658	306.4758	441.9373
	3000	22.9675	75.88508	183.4288	308.334	540.3663

Table 16 - One-Point crossover elapsed time (s) experiment for n300 instance

Technique		**One-Point crossover**				
Instance		**n300**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	123.9931	226.9599	478.1856	604.0179	742.981
	1500	106.3093	268.3282	520.932	658.2953	962.7007
	2250	71.34425	221.4392	456.9921	609.6691	900.4454
	3000	60.033	230.3652	509.8691	607.3908	1405.347

Table 17 - RRC elapsed time (s) experiment for n100 instance

Technique		**Random Respectful crossover**				
Instance		**n100**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	4.591833	17.38133	40.60725	57.67533	69.29133
	1500	5.657667	20.62208	40.87858	50.16233	76.66575
	2250	4.754583	20.52625	45.57375	52.8465	74.5275
	3000	6.854333	21.71592	39.67325	49.37775	69.95667

Table 18 - RRC elapsed time (s) experiment for n200 instance

Technique		**Random Respectful crossover**				
Instance		**n200**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	11.99558	53.43125	103.0671	152.3075	187.739
	1500	11.79533	41.725	97.23258	163.8788	204.9038
	2250	13.69283	47.50867	199.5325	174.5134	195.708
	3000	14.44542	49.00675	111.3689	169.5244	207.465

Table 19 - RRC elapsed time (s) experiment for n300 instance

Technique		**Random Respectful crossover**				
Instance		**n300**				
# Individuals		**50**	**150**	**300**	**450**	**600**
# Generations	**750**	33.62958	87.979	154.7622	231.7078	382.7508
	1500	27.36383	87.84767	182.8613	213.8203	370.9229
	2250	27.40783	98.50067	168.4204	232.9308	424.7484
	3000	27.36625	92.21283	154.0199	253.575	360.6311

Appendice C

Table 20 – Fitness, elapsed time and # generations executed in n100 instance

# Individuals	Fitness (mm^2)	# Generations	Elapsed time (min)
500	0.24462	5171	16.55
1500	0.24349	4675	57.42
3000	0.24203	5025	156.32

Table 21 - Fitness, elapsed time and # generations executed in n200 instance

# Individuals	Fitness (mm^2)	# Generations	Elapsed time (min)
1500	0.25671	5449	151.78
3000	0.25536	4879	219.6
4500	0.25568	4065	461.83