Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 679 – 692

doi:10.3906/elk-1903-122

A modified relay-race algorithm for floorplanning in PCB and IC design

Mert VATANSEVER, Faik BAŞKAYA*
Department of Electrical and Electronic Engineering, Faculty of Engineering, Bogazici University, Istanbul, Turkey

Received: 21.03.2019 • **Accepted/Published Online:** 24.11.2019 • **Final Version:** 28.03.2020

Abstract: Floorplanning is a fundamental design step in the physical design of printed circuit boards (PCBs) and integrated circuits (ICs), as it handles the complexity of layout design. From a computational point of view, the floorplanning problem is an NP hard problem, and the size of the search space grows exponentially with increasing numbers of modules. Thus, the algorithm used is an essential factor for speed and quality of the floorplanning process. Although polynomial-time floorplanning algorithms can be implemented when solution space is limited to slicing floorplans, optimal solutions often exist only in the nonslicing floorplan search space. Various stochastic algorithms such as simulated annealing (SA), the genetic algorithm (GA), and the relay race algorithm (RRA) can be used with nonslicing floorplans. In this paper, a modified relay race algorithm (MRRA) is proposed. Based on the experimental results utilizing MCNC benchmarks, MRRA improved both solution quality and run time for area optimization when compared with SA, GA, and RRA.

Key words: Placement, floorplanning, CAD, VLSI, printed circuit board

1. Introduction

The number of components in a circuit and the interconnections between these components increase rapidly as technology improves over time [1]. Floorplanning optimizes the relative locations of the components in the layout to reduce the layout area and wire length of the interconnections, which affect the subsequent routing quality and overall physical design process significantly [2]. The representation method affects the floorplanning process, because it determines the scope of the search space and the complexity of transformation between the floorplanning representation and its corresponding floorplan. Researchers have proposed many representation methods such as Polish notation, bounded sliced grid (BSG), transitive closure graph (TSG), B*Tree, and sequence pairs [3].

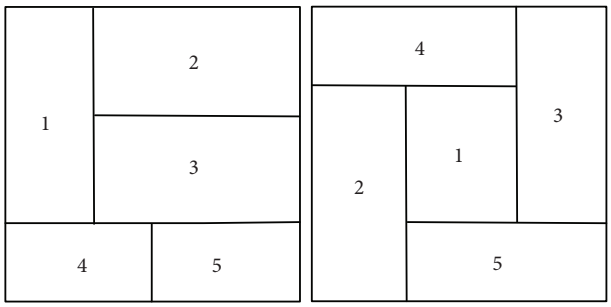
The most important factor that determines the time cost and solution quality of the floorplanning process is the algorithm used. Various floorplanning algorithms have been proposed by researchers, including simulated annealing (SA), genetic algorithms (GAs), and the relay race algorithm (RRA). SA was originally proposed as an optimization approach for placement and routing [4], but was later utilized by Wong and Liu to optimize the area of a floorplan [5]. Rebandengo and Reorda used the GA as an evolutionary algorithm [6]. Sheng et al. designed the RRA to overcome the shortcomings of SA and GAs [7]. In this paper, a modified relay-race algorithm (MRRA) has been proposed in order to improve the solution quality and time cost of RRA. Section 2 formulates the problem. Section 3 briefly summarizes the existing approaches for floorplanning. Section 4 proposes the MRRA approach and Section 5 presents experimental results. Section 6 concludes the paper.

*Correspondence: faik.baskaya@boun.edu.tr

2. Problem formulation

Floorplanning is the determination of relative module positions while considering objectives such as area and wire length minimization. The main inputs for floorplanning are a module set $M = \{m_1, m_2, m_3, \ldots, m_n\}$ where m_i are rectangular blocks with height h_i and width w_i, and a net set $N = \{n_1, n_2, n_3, \ldots, n_k\}$ where n_j are the interconnects between modules. Each net n_i, $1 \leq i \leq k$, has a length l_i, which can be computed between the centers of modules that are being connected, unless the pin locations of modules are provided as an additional input to the floorplanning.

There are two types of floorplans, which are called slicing and nonslicing floorplans [8]. Slicing floorplans can be represented by a binary tree, which is called the slicing tree, where the entire layout area is bisected repetitively in horizontal and vertical directions until each part includes only one module. In the slicing tree, the leaves represent the modules, vertices marked as H represent the horizontal bisections, and vertices marked as V represent the vertical bisections. On the other hand, nonslicing floorplans cannot be obtained by bisecting the layout area repetitively; therefore, slicing trees cannot be used to represent them. The constraint graph pair (CGP) method, which consists of a horizontal constraint graph (HCG) and vertical constraint graph (VCG), can be used to model these floorplans. HCG and VCG define the horizontal and vertical relations among the modules, respectively. Figures 1a and 1b depict instances of slicing and nonslicing floorplans, respectively. Slicing floorplans are easier to manipulate, and polynomial time algorithms are available for finding optimum floorplan solutions when restricted to slicing structures only. On the other hand, only nonslicing floorplans have a solution space that is P-admissible, which is guaranteed to contain an optimal solution [9]; therefore, an optimum floorplan solution for all problem instances is only possible with this floorplan type.



(a) Slicing Floorplan (b) Non-Slicing Floorplan

Figure 1. Types of floorplans.

The objective of floorplanning is to optimize a layout according to a predefined cost function [8]. The most common consideration in this function is the area covered by the rectangular bounding box enclosing all modules. This requires minimization of the dead space, which is called white space. White space is the empty space that is not covered by any module in the floorplan. Another important consideration in this function is the total wire length, which has several types of evaluations such as minimum chain, Steiner tree, and half perimeter wire length (HPWL) methods. Steiner tree estimation is the most accurate but also most computationally expensive method, while HPWL is the most efficient and can still be used to compare the relative wire lengths of different solutions with respect to each other in an optimization engine. HPWL is obtained by dividing the perimeter of the rectangular bounding box that surrounds all the pins of a net by two [10]. A commonly used cost function in floorplanning is the weighted sum of area and wire length as given

by Equation 1, where C_a, C_w, and α represent the area cost, the wire length cost, and the weight factor, respectively. The weight factor α is associated with each objective and is user-defined.

$$Cost = \alpha.C_a + (1 - \alpha).C_w \tag{1}$$

Floorplan representation must be chosen according to the floorplan type, and this choice determines the complexity of the transformation and the scope of the search space [1]. Researchers have proposed several representation schemes in the last couple decades. Polish notation, bounded slicing grid, transitive closure graph, B*Tree, and sequence pairs are the most commonly used representation schemes [3]. In Table 1, the comparison of different floorplan representation schemes is represented. This comparison contains information about the flexibility and the computational complexity of these floorplan representation schemes.

Table 1. Comparison of different floorplan representations.

Representation	Flexibility	Complexity
Polish notation	Slicing	$O(n)$
Bounded slicing grid	General	$O(n^2)$
Transitive closure graph	General	$O(n^2)$
B*Tree	Compacted	$O(n)$
Sequence pair	General	$O(n^2)$

As shown in Table 1, Polish notation and B*Tree have better computational complexity and bounded slicing grid, transitive closure graph, and sequence pair approaches have better flexibility. Polish notation is an efficient representation scheme for slicing floorplans, but it cannot handle other floorplan types. The expression of Polish notation is the postfix ordering of a binary tree, which can be reached from the postorder traversal on a binary tree. Bounded slicing grid is a flexible representation scheme, but it cannot handle nonslicing floorplans, either. In a bounded slicing grid, n blocks are placed in a special n by n grid. The transitive closure graph method runs faster using less memory, but it cannot deal with slicing floorplans [3]. B*Tree representation is based on ordered binary trees and can model compacted floorplan structures. It is also an efficient representation scheme with smaller encoding cost. However, it is less flexible than bounded slicing grid, transitive closure graph, and sequence pair. Sequence pair is the most flexible representation scheme and it can handle all types of floorplans, but it has high encoding cost. Sequence pair is utilized as the representation method in this paper because of its flexibility advantage.

Sequence Pair representation is suitable for both slicing and nonslicing floorplans. A sequence pair $(\Gamma+, \Gamma-)$ is a pair of sequences of the n modules in a floorplan, where modules can be placed into different orders in each pair [9]. Horizontal and vertical constraints between each pair of modules can be inferred from the sequence pair to be used in generating the constraint graph pair (HCG and VCG). For instance, $(\Gamma+, \Gamma-) = (bacde, cabde)$ can be the sequence pair representation for one of the solutions of a floorplan that includes the module set a,b,c,d,e. For this sequence pair, d is placed after a in both $\Gamma+$ and $\Gamma-$, i.e. $< \ldots a \ldots d \cdots >$; thus, d has to be located to the right of a. Also in the same sequence pair, a is placed after b in $\Gamma+$, i.e., $< \ldots b \ldots a \cdots >$, and a is placed before b in $\Gamma-$, i.e. $< \ldots a \ldots b \cdots >$; thus, a has to be located below b. Replacing the orders of (a, b) or (a, d) in the sequences described above reverses the relative positions of these module pairs with respect to each other.

3. Existing floorplanning approaches

Developments in optimization field led numerous researchers to utilize modern optimization methods. Simulated annealing is the first modern optimization algorithm that has been used to optimize the floorplanning area. Population-based metaheuristic algorithms that imitate the social behavior of species and biological evolution were then utilized. The GA, ant colony, particle swarm optimization, and differential evolution are in this category, and they have been collectively named as evolutionary algorithms [3].

3.1. Simulated annealing

Simulated annealing (SA) has been proposed based on statistical mechanics theory and the analogy between solid annealing and optimization problems [4]. The utilization of the SA algorithm to solve the floorplanning problem was first introduced by Otten in 1983 [11]. SA resembles the cooling procedure of molten metal through annealing. In the cooling process of molten metal, the atoms have the highest mobility at high temperatures. As the temperature drops, the movement ability of the atoms is also reduced. Then the atoms are gradually organized to form crystals with the minimum energy state possible.

In SA, each state of the solid structure corresponds to an applicable solution of the problem. The energy of the state is the value of the cost function to assess the solution. The state of the minimum energy represents the optimal solution with the best value of the cost function. SA is a stochastic algorithm with iterative improvements. Each repetitive step includes an alteration of the current solution to a new solution. This action is called movement to a neighborhood. The current temperature of the state determines the acceptance probability of new solutions. Temperature updates are scheduled from the highest temperature to the lowest temperature, where the acceptance probability at higher temperatures is higher than the acceptance probability at lower temperatures. If the temperature is decreased rapidly, it is known as simulated quenching instead of simulated annealing. The main difference between SA and simulated quenching is the parameter used for temperature scheduling. In SA, the temperature needs to be decreased at a slower rate in order to reach the absolute minimum energy state.

3.2. Genetic algorithm

The GA has been utilized as a floorplanning algorithm after SA by researchers. Rebandego and Reorda were the first researchers to use the GA to solve the floorplanning problem [6]. They used the GA with Polish notation in 1996. Afterwards, Nakaya et al. and Lin et al. also presented GAs using Polish notation for the floorplanning problem [12, 13]. Gwee and Lim proposed a GA with heuristic-based decoder for IC floorplaning in 1999 [14]. This approach was able to achieve an efficient solution to the multiobjective area and wire length optimization problem of floorplanning. In 2006 Drakidis et al. and in 2007 Chatterjee and Manikas presented GA-based floorplanning approaches using sequence pair representation [15].

The utilization of the GA for floorplanning optimization starts with the randomly generated population of solutions. These solutions have random placement of modules along a defined rectangle of the circuit. Then the solutions are evaluated for their fitness values based on the predefined fitness function. The objective of this fitness function can be area or wire length optimization or optimization for both criteria. The modules correspond to genes in chromosomes. After the creation of the initial population, the algorithm follows the mentioned mechanisms repetitively until the specified number of generations is reached. In the crossover operation, two floorplan solutions are taken and they are used to generate a new floorplan arrangement as a new solution. These new solutions are called the offspring of the selected solutions that the crossover operation

is performed on. Afterwards, a mutation operation with a small probability is applied by flipping any module of the solution. Finally, the new population is evaluated and the solutions with the lowest fitness values are eliminated.

3.3. Relay race algorithm

Sheng et al. proposed the RRA for floorplanning problems to approach a global optimal solution by exploring similar local optimal solutions more efficiently within shorter computation times [7]. Sheng et al. stated that the RRA was designed to overcome the shortcomings of SA, which does not use the experience of past moves, and the GA, which selects the next generation according to a ranking function that has a high time cost despite it not being always necessary.

The RRA contains the three basic parts shown in Figure 2: focusing search, rough search, and relay. An algorithmic flowchart of the RRA and the details of searches are depicted in Figures 3a and 3b, respectively. The aim of the rough search is to pass over little hills in the search space and approach a local optimum as quickly as possible. The focusing search tries to reach as close to the local optimum as possible. The relay works for both running away from the local optimum with a single operation and maintaining the search continuity.

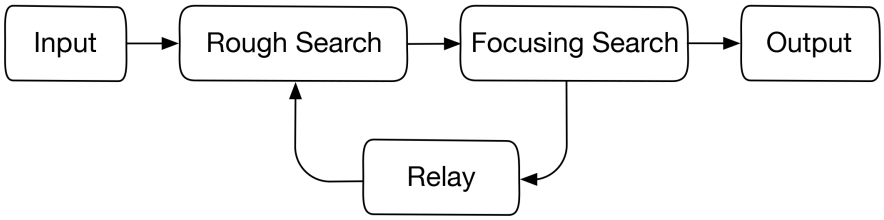


Figure 2. Basic steps of relay race algorithm.

Rough search begins with method selection. Three types of move methods are utilized in rough searches: group insertion, group exchange, and group rotation. In group insertion, the order of randomly selected modules in one sequence is changed. Group exchange is the exchange of randomly selected modules. Group rotation rotates randomly selected modules. The number of modules in the group is set to 10 to ensure that the rough moves affect more modules than the focusing moves.

Focusing search starts with the termination of the rough search. After the rough search is completed, the local optimal solution is transferred to focusing search. For the focusing search, three focusing move methods are utilized: insertion, exchange, and rotation. In an insertion move, the order of a single module is changed in one sequence. Rotation moves alter the orientation of a single module. Exchange move is the exchange of the order of two modules in both sequences $\Gamma+$ and $\Gamma-$.

Both rough search and focusing search follow similar procedures after the move method is selected. First, the current solution is modified to generate the next solution by utilizing the selected move method. Then the new solution is evaluated. If there is an improvement, the new solution becomes the current solution, followed by the updates of the best record and the guide. Otherwise, the new solution is rejected and the old solution is kept as the current solution. Rough search is terminated when the number of trials with no improvement reaches the predetermined number N_r. Focusing search is terminated when the number of consecutive trials with no improvement reaches the predetermined number N_f.

Relay is the last part of the outer loop. After completion of the rough search and focusing search, the relay operator generates a new solution from the current solution. This new solution consists of two parts. The

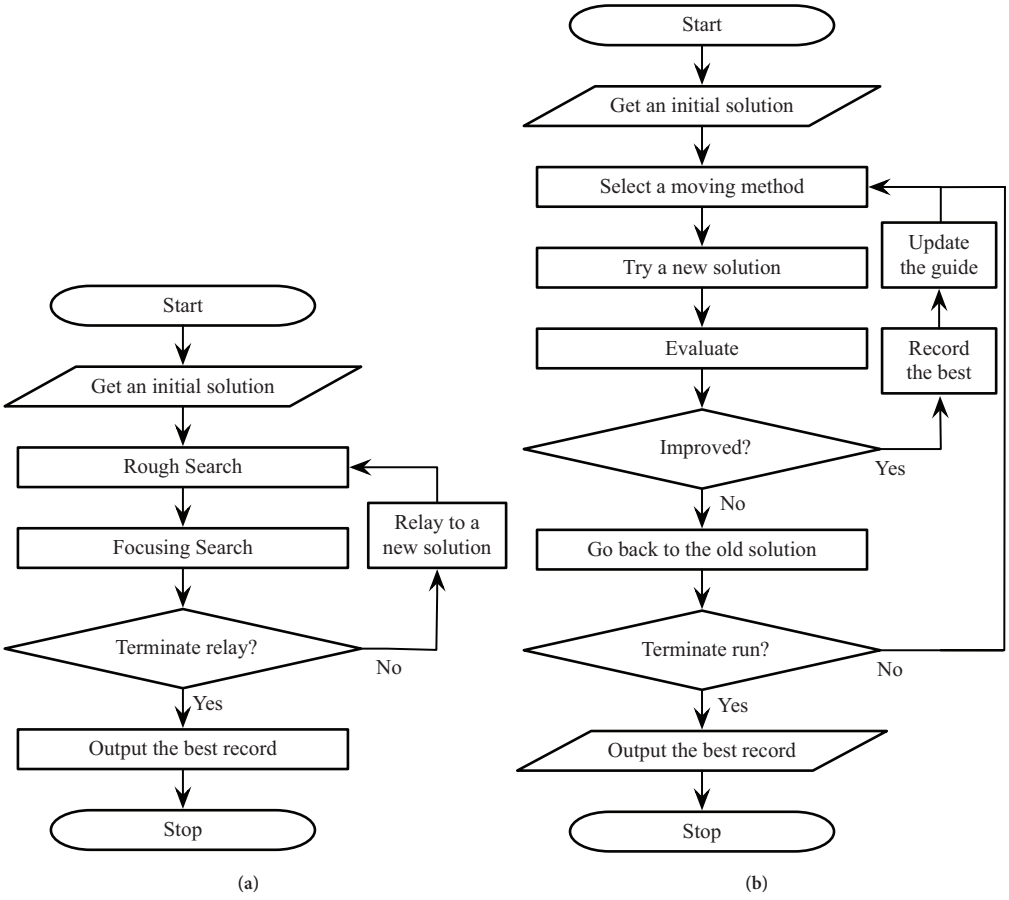


Figure 3. Flowchart of relay race algorithm (a) and rough/focus search routines (b).

first part of the new solution is inherited from the current solution. The second part of the new solution is randomly generated. The ratio of the solution's randomly generated part is defined as the parameter R_e. To find the number of modules that will be affected by the relay operation in a circuit with N_m modules, the $R_e \cdot N_m$ product is rounded to the nearest integer. The modules that will be in the randomly generated part are selected randomly.

Figure 4 illustrates the behavior of the RRA in the solution space. Only the solutions with improvement are accepted in both rough and focusing runs. The differences between the rough run and focusing run are in moving methods and in terminal condition. As Figure 4 indicates, the rough run gets over small hills and the focusing search gets a local optimal solution. On the other hand, the relay escapes from the local optimum solution and reaches near another local optimum solution. This process is repeated as many times as the number of runners on the team, N_t, in order to find the global optimum solution.

4. Modified relay race algorithm

Although the RRA was proposed to overcome the shortcomings of the SA and GA algorithms, there are shortcomings of the RRA, too. The MRRA is proposed here to improve the RRA by working on the following algorithmic choices of the RRA:

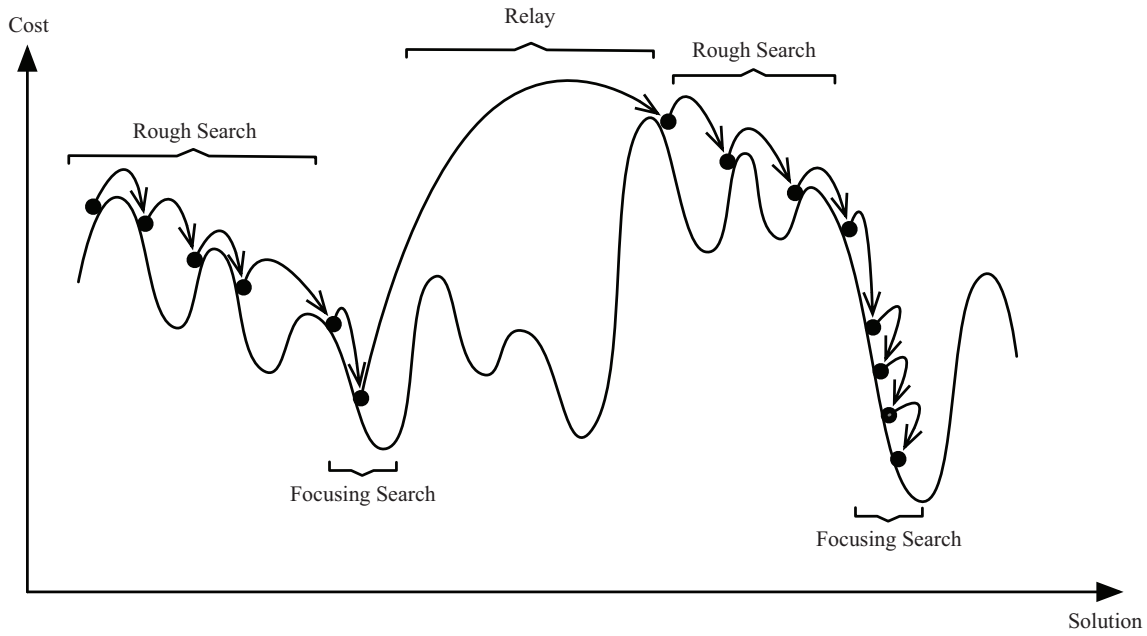


Figure 4. Behavior of relay race algorithm in the solution space.

. Searches of different solutions within the solution space are performed on a single path.

. The group size used in rough moves is a fixed number of 10, regardless of the size of the input circuit.

. Parameters N_r and N_f that determine the efficiencies of rough and focusing searches have been decided after a set of trial experiments and their values are fixed at 100 and 1000, respectively.

In the MRRA, the search is performed on a dual path to increase the chance of discovering better local optimum solutions. Moreover, the maximum number of iterations without improvement during rough search N_r and maximum number of consecutive iterations without improvement during focusing search N_f are not fixed for all problems; instead, their values are determined by multiplying the circuit size, N_m, by different coefficients. After extensive trial experiments to determine the best values of N_r, N_f, and N_t, the best empirical parameter set was determined as $3N_m$, $3N_r$, and 20, respectively.

The MRRA starts by getting an initial solution that can be either user-defined or randomly produced. Then rough search and focusing search are applied to the initial solution. Afterwards, the current solution enters the dual path search.

Figure 5a illustrates the flowchart of the MRRA, which includes two inner loops. While the first inner loop corresponds to the dual path search of the MRRA, the second inner loop corresponds to the single path search of the original RRA. The single path search phase continues until the total number of runners in the team for the relay N_t is reached. In the single path search, the value of the parameter R_e is chosen to be 0.1. As a result of the dual path search, the probability of exploring better local optimum solutions in distant regions is increased. Figure 5b depicts a more detailed description of the "Search Path" step in Figure 5a.

During dual path search, both paths implement the same operations. The only difference between these two paths is the R_e value, which is the ratio of the randomly generated part of the solution in the relay operation. The implementation of the searching process with two different paths increases the likelihood of

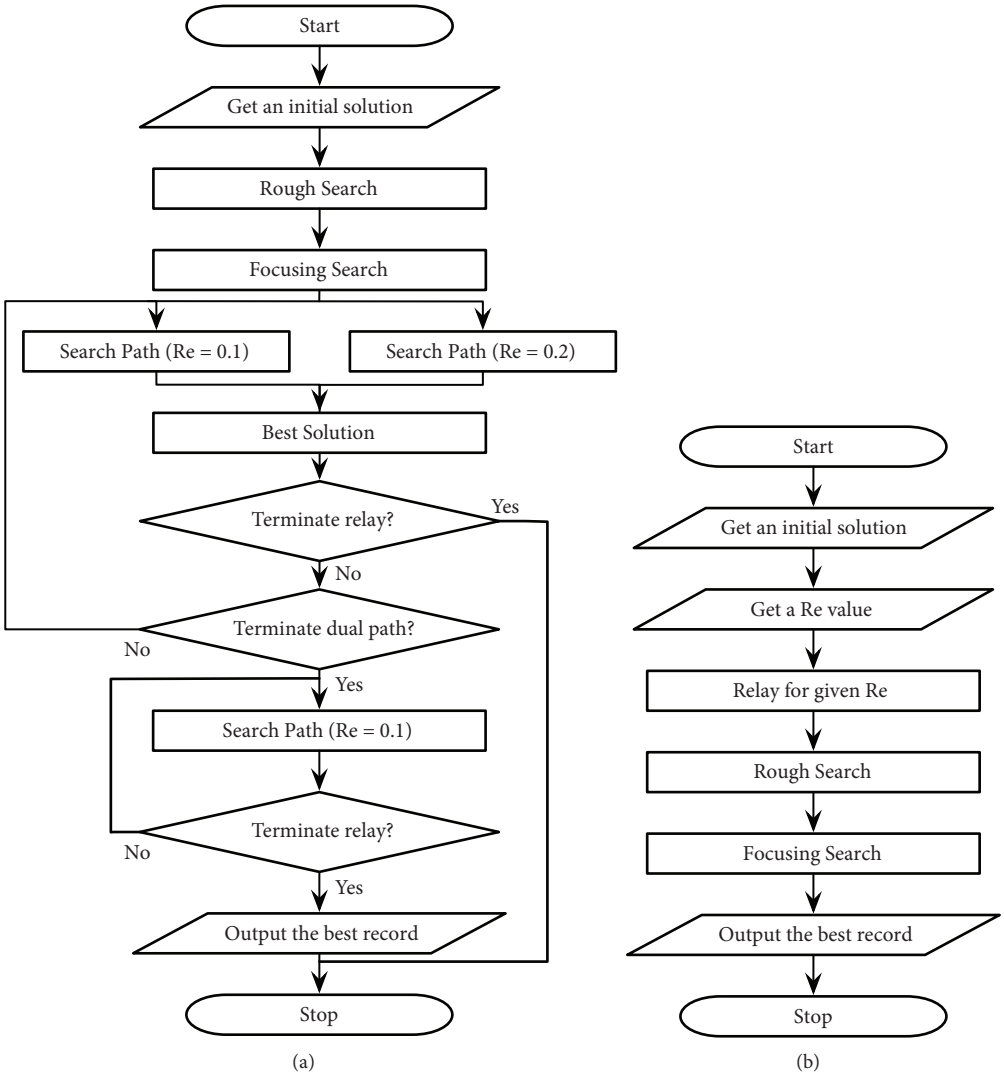


Figure 5. Flowchart of modified relay race algorithm (a) and search path for given R_e value (b).

achieving a better local optimal solution as the next solution. The value of the parameter R_e used in the first path process is chosen to be 0.1, as in the original algorithm. The value of the parameter R_e used in the second path process is chosen to be 0.2, which is larger than the R_e value used in the first path; therefore, the second path makes it possible to search in farther regions of the solution space. Since R_e corresponds to the mutation rate as mentioned in the previous section, the current solution is mutated at a rate of 0.1 and 0.2 in the first and second paths, respectively. Thus, there is an increased chance for exploring better local optimum solutions.

After both first and second paths complete their search operations, the best solutions of these paths are compared and only the better solution is kept as the next solution. However, if dual path search is applied until the algorithm is terminated, the computation time will increase too much. For this reason, there is also a termination condition for dual path search. When two consecutive solutions of the first path ($R_e = 0.1$) are better than the solutions of the second path ($R_e = 0.2$), the dual path search is terminated and the algorithm continues with the single path search afterwards.

4.1. Move methods

The MRRA utilizes three rough and three focusing move methods. Rough move methods are group rotation, group exchange, and group insertion. Focusing move methods are rotation, exchange, and insertion. These move methods are exactly the same as the move methods used in the RRA. Methods were kept the same as in the original algorithm in order to compare the differences in the results of the RRA and MRRA that are caused by modifying the approach.

The corresponding placement of a sample initial SP $(\Gamma+,\Gamma-) = (32415, 12534)$ is shown in Figure 6a. The insertion move places a randomly selected module in one sequence into a random position. Figure 6b shows the placement of modules after the insertion move is applied to the module m_5. The exchange method changes the order of a randomly selected pair of modules in both the positive sequence $\Gamma+$ and the negative sequence $\Gamma-$. The placement of modules after the exchange move is applied to the modules m_3 and m_5 is displayed in Figure 6c. The rotation move changes the orientation of a randomly selected module. Figure 6d illustrates the placement of modules after a rotation move is applied to module m_4. A group insertion move inserts one randomly selected set of modules into a randomly selected set of positions in one sequence. A group exchange move exchanges randomly selected pairs of modules in both sequences. A group rotation move rotates a randomly selected set of modules. Unlike in the RRA, the number of modules in the group varies based on the total number of modules in the circuit, N_m. After experimenting with $0.3N_m$, $0.4N_m$, and $0.5N_m$, $0.4N_m$ has been decided as the group size in the MRRA rough moves.

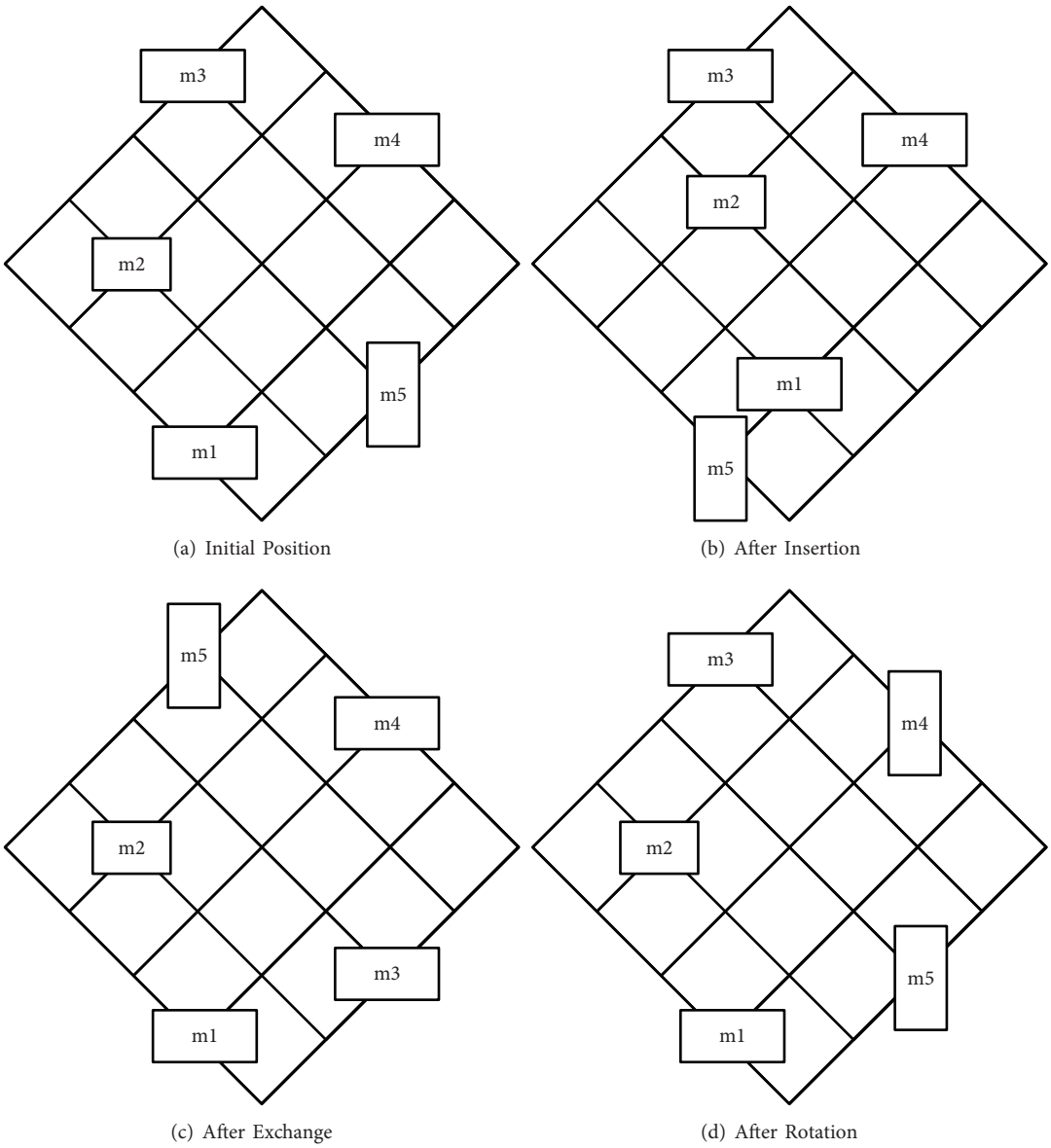
The move method is selected using the probability of move methods [7]. The probability of any method $p_{k+1}(i)$ is evaluated according to the old probability of the method $p_k(i)$ and the short-term improvement speed $s_k(i) = a_k(i).f_k(i)/\sum_{j=1}^{3} a_k(j).f_k(j)$, where i represents different move methods, $a_k(i)$ is the relative amplitude of improvement, and $f_k(i)$ is the frequency of the improvement. In detail, $a_k(i)$ is the relative $-\Delta C$ on average and $f_k(i)$ is the ratio of improved trials in the last t trials. For rough move methods and focusing move methods, t is chosen as 30 and 100, respectively. If the evaluation of the solution satisfies the condition $-\Delta C > 0$, $a_k(i)$ is calculated and updated. Otherwise, the probability of methods is not updated. The new probability $p_{k+1}(i)$ is given by $p'_{k+1}(i)/\sum_{j=1}^{3} p'_{k+1}(j)$ for each move method to keep the total probability 100%, where $p'_{k+1}(i)$ equals $(p_k(i) + s_k(i))/2$.

4.2. Cost function

The cost function of the MRRA has three components, which are area, wire length, and overlap costs. Area cost is the size of the smallest rectangular bounding box that contains all modules. Wire length cost is an approximation of the sum of length of all interconnects between the modules. Overlap is the amount of overlap between the paths of same and different interconnects. The area and wire length are the most common costs for the typical cost function in floorplanning. The overlap objective is inserted into the cost function to consider the interference between different signals. The cost function of the MRRA is given by Equation 2.

$$C_t = \alpha.C_a + \beta.C_w + \gamma.C_o \tag{2}$$

In Equation 2, α, β, γ are the coefficients of their corresponding objectives that satisfy $\alpha + \beta + \gamma = 1$. C_t, C_a, C_w, and C_o represent total cost function, cost function of area, cost function of wire length, and cost function of overlap, respectively. The cost function of area C_a estimates the area of the bounding rectangular shape, which is given by the minimum bounding rectangle, which includes all modules. The area is calculated



(a) Initial Position (b) After Insertion

(c) After Exchange (d) After Rotation

Figure 6. Moving methods used in RRA and MRRA.

by multiplying the total width W and the total height H. The cost function of wire length C_w estimates the total wire length that is used for the connection of all pins in the circuit. The half perimeter wire length method is utilized to obtain an approximation of the wire length for each net. The cost function of the overlap function C_o estimates the total overlap cost between nets. The calculation of the overlap cost is made according to the overlap coefficients.

5. Experiments and results

Parameter selection directly affects the performance and efficiency of the floorplanning algorithm. Therefore, the best empirical values of parameters N_f, N_r, and R_e were investigated by trial experiments of floorplanning

using the ami33 circuit from the Microelectronics Center of North Carolina (MCNC) benchmark suite. The MCNC benchmark is the most commonly used benchmark for comparing floorplanning algorithms; therefore, it is also used in this paper for comparison with other approaches. The MCNC benchmark suite consists of five circuits: apte, xerox, hp, ami33, and ami49. The details of the MCNC benchmark suite are shown in Table 2.

Table 2. MCNC benchmark circuit specifications.

Circuit	Modules	Nets	Terminals
apte	9	97	73
xerox	10	203	2
hp	11	83	45
ami33	33	123	42
ami49	49	408	22

In the RRA, the best empirical values of parameters N_f, N_r, and R_e were investigated with trial experiments and found as the fixed values of 1000, 100, and 0.1, respectively. When finding the best parameter values in the RRA, trial experiments were run for testing different values of N_f first. Next, different values of N_r were tried while N_f was kept constant at its best value of 100. Finally, R_e was also determined while keeping N_f and N_r fixed at the values determined in previous trial experiments. In the MRRA, combinations of different parameters were also used so that intermediate values of parameters were also implemented to obtain better results. The investigation for better parameter values also aims at speeding up the algorithm by limiting the time increase caused by the dual path search structure used in MRRA. The values of parameters N_f and N_r were not kept constant as in RRA, but they were automatically scaled with N_m, which is the number of modules in the circuit. The cost function coefficient α is set to 1 while β and γ are set to 0 so that the area optimization results can be compared with previous works.

The parameter values were adjusted according to the results of trial experiments. In particular, the final cost, run time, and product of these values that have been obtained as results of trial experiments were used as the most important values in determining the parameters. For trial experiments, 100 initial solutions were generated. Each trial experiment was conducted with this set of initial solutions for enforcing the same initial conditions in all tests.

In the first stage of trial experiments, the best combination of parameters N_f and N_t was investigated. For these experiments, the value of N_r has been set to three times the N_m value, which equals approximately the same value used in the RRA for the ami33 circuit. In the second stage of trial experiments, the best combination of N_r, N_f, and N_t was investigated. In these trial experiments, it was aimed to increase the value of the N_t parameter while decreasing the value of the N_r parameter and the value of the N_f parameter in order to allow more runners to try more solutions. In the third stage of trial experiments, N_r was chosen to be N_m and the value of N_t was increased while the value of N_f was decreased. It is seen that there is a trade-off between the final cost and the run time as a result of increasing N_t and decreasing N_f. After all trial experiments conducted in order to determine the best values of N_r, N_f, and N_t, the best empirical parameter set was determined as $(N_r, N_f, N_t) = (3.N_m, 3.N_r, 20)$. This parameter set was utilized for further experiments with the other circuits in the MCNC benchmark suite.

After the best empirical parameter set were determined, area optimization results of SA, GA, RRA, and MRRA were compared. For a fair comparison between the algorithms, all algorithms were implemented utilizing

the SP representation scheme while following the published details that can be accessed as closely as possible. These algorithms were applied to all circuits found in the MCNC benchmark suite in the Java environment on a 2.40 GHz PC with 8.00 GB memory. Since all these algorithms are stochastic, each algorithm was run 10 times for each circuit and each algorithm was started using the same initial solutions that were generated randomly at the beginning.

The values of the parameters used in the algorithms were chosen taking into account the values of the algorithms for which the run times were close. For the implementation of the RRA, N_r and N_f were selected as 100 and 1000, respectively, while $N_r = 3.N_m$ and $N_f = 3.N_r$ were chosen for the implementation of the MRRA. On the other hand, different values were used for N_t to provide close run times. The parameters of SA were set as follows: the initial temperature was determined by considering average difference cost of moving methods. The number of trials per each temperature and cooling rate were selected as 500 and 0.99, respectively. For the implementation of the GA, the population size and the number of generations were set to 100 and 1000, respectively. In addition, crossover rate and mutation rate were chosen as 0.8 and 0.3, respectively.

The average area cost and average run time comparisons are based on 10 trials for each algorithm, and they are shown in Tables 3 and 4, respectively. The RRA and MRRA have better results than SA and GA for all circuits in both categories.The MRRA also has the best results for all benchmarks among these four algorithms. The improvement of MRRA compared to RRA is between 0.03% and 1.75% for the average area costs. The MRRA also has considerable improvement between 9.4% and 24.9% for average run times.

Table 3. Comparison of the final area costs of each algorithm.

MCNC	Average (mm^2)				Standard Deviation (mm^2)			
	SA	GA	RRA	MRRA	SA	GA	RRA	MRRA
apte	47.687	48.579	47.570	47.481	0.375	0.523	0.271	0.410
xerox	20.930	21.120	20.607	20.307	0.552	0.397	0.288	0.062
hp	9.971	9.822	9.528	9.361	0.256	0.164	0.192	0.135
ami33	1.438	1.327	1.275	1.259	0.055	0.018	0.019	0.019
ami49	46.256	41.907	39.714	39.702	1.885	0.823	0.447	0.541

Table 4. Comparison of the run times of each algorithm.

MCNC	Average (s)				Standard deviation (s)			
	SA	GA	RRA	MRRA	SA	GA	RRA	MRRA
apte	1.871	1.936	1.460	1.322	0.168	0.220	0.189	0.481
xerox	1.853	1.695	0.841	0.631	0.214	0.215	0.317	0.150
hp	2.779	1.662	1.102	0.928	0.210	0.160	0.311	0.376
ami33	43.036	19.854	13.103	10.294	0.369	0.547	0.817	0.642
ami49	85.255	36.799	27.404	22.587	0.209	0.603	2.003	1.384

6. Conclusion

In this paper, a heuristic approach named MRRA is proposed to solve the floorplanning problem. The MRRA was designed to improve the speed and quality of the RRA by overcoming the shortcomings of the RRA. In the

MRRA, a dual path search is designed to increase the probability of exploring a better local optimal solution as the next solution. In both search paths, rough and focusing runs are implemented and the only difference between these two paths is the ratio of the randomly generated part of the solution R_e in the relay operation. The dual path search has its own termination condition in order not to increase the run time. Moreover, parameters N_r and N_f are determined according to the detailed analysis, which also considers the number of modules in the circuit to improve the efficiency of the algorithm. The efficiency of the MRRA is proven by applying it to the floorplanning problem in physical design optimization. Based on the comparisons of the experimental results utilizing the MCNC benchmark suite, the MRRA is better than SA, GA, and RRA in terms of average cost and average run time of area optimization. The MRRA reduced the average run time by an average of 17.5% according to the RRA. With regard to comparison results, the proposed MRRA has the potential to improve more NP-hard problems.

As shown in the comparisons in Section 5, the improvement of the MRRA varies according to the MCNC benchmark. The difference in the number of modules of the circuits may be the cause of this situation. Although the parameter values used in the MRRA were determined as a result of a detailed analysis, they may need to be changed according to the region where they are located in search space. Searching by more than one path, as evidenced by the MRRA, increases the efficiency. However, the most suitable number of initial paths and their termination conditions can be determined to increase the improvement of efficiency. In addition, these multiple paths can be operated on different cores to further decrease the computation time.

References

[1] Singh RB, Baghel AS, Agarwal A. A review on VLSI floorplanning optimization using metaheuristic algorithms. In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques; Tamil Nadu, India; 2016. pp. 4198-4202.

[2] Banerjee S, Ratna A, Roy S. Satisfiability modulo theory based methodology for floorplanning in VLSI circuits. In: 2016 Sixth International Symposium on Embedded Computing and System Design; Patna, India; 2016. p. 91-95.

[3] Laskar NM, Sen R, Paul PK, Baishnab KL. A survey on VLSI floorplanning: its representation and modern approaches of optimization. In: 2015 International Conference on Innovations in Information, Embedded and Communication Systems; Coimbatore, India; 2015. p. 1-9.

[4] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983; 220 (4598): 671–680.

[5] Wong DF, Liu CL. A new algorithm for floorplan design. In: 23rd ACM/IEEE Design Automation Conference; Las Vegas, NV, USA; 1986. pp. 101–107.

[6] Rebaudengo M, Reorda MS. GALLO: A genetic algorithm for floorplan area optimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 1996; 15 (8): 943–951.

[7] Sheng Y, Takahashi A, Ueno S. Relay-race algorithm: a novel heuristic approach to VLSI/PCB placement. In: 2011 IEEE Computer Society Annual Symposium on VLSI; Chennai, India; 2011. pp. 96-101.

[8] Moni DJ. Certain optimization techniques for floorplanning in VLSI physical design. PhD, Anna University, Guindy, India, 2009.

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

[10] Ray BNB, Tripathy AR, Samal P, Das M, Mallik P. Half-perimeter wirelength model for VLSI analytical placement. In: 2014 International Conference on Information Technology; Bhubaneswar, India; 2014. pp. 287–292.

[11] Otten R. Efficient floorplan optimization. In: International Conference on Computer Design; Port Chester, NY, USA; 1983. pp. 499-502.

[12] Nakaya S, Koide T, Wakabayashi S. An adaptive genetic algorithm for VLSI floorplanning based on sequence-pair. In: 2000 IEEE International Symposium on Circuits and Systems, Emerging Technologies for the 21st Century; Geneva, Switzerland; 2000. pp. 65–68.

[13] Lin CT, Chen DS, Wang YW. An efficient genetic algorithm for slicing floorplan area optimization. In: 2002 IEEE International Symposium on Circuits and Systems; Phoenix-Scottsdale, AZ, USA; 2002. pp. I-II.

[14] Gwee BH, Lim MH. A GA with heuristic-based decoder for IC floorplanning. Integration 1999; 28 (2): 157-172.

[15] Drakidis A, Mack RJ, Massara RE. Packing-based VLSI module placement using genetic algorithm with sequence-pair representation. IEE Proceedings - Circuits, Devices and Systems 2006; 153 (6): 545–551.