PAPER *Special Section on VLSI Design and CAD Algorithms*

A Novel High-Performance Heuristic Algorithm with Application to Physical Design Optimization

Yiqiang SHENG[†a)], ***Nonmember and*** **Atsushi TAKAHASHI**[†b)], ***Senior Member***

SUMMARY In this paper, a novel high-performance heuristic algorithm, named relay-race algorithm (RRA), which was proposed to approach a global optimal solution by exploring similar local optimal solutions more efficiently within shorter runtime for NP-hard problem is investigated. RRA includes three basic parts: rough search, focusing search and relay. The rough search is designed to get over small hills on the solution space and to approach a local optimal solution as fast as possible. The focusing search is designed to reach the local optimal solution as close as possible. The relay is to escape from the local optimal solution in only one step and to maintain search continuity simultaneously. As one of typical applications, multi-objective placement problem in physical design optimization is solved by the proposed RRA. In experiments, it is confirmed that the computational performance is considerably improved. RRA achieves overall Pareto improvement of two conflicting objectives: power consumption and maximal delay. RRA has its potential applications to improve the existing search methods for more hard problems.

key words: *NP-hard problem, optimization, conflicting objectives, physical design, placement*

1. Introduction

Search methods such as [1]–[11] are widely used to get near-optimal solution for NP-hard problem. In order to get a better solution efficiently, various algorithms such as iterative improvement method with random generation, simulated annealing [6]–[9], genetic algorithm [10] and variable neighborhood search [11] have been proposed so far. However, the efficiency of existing search algorithms is limited.

In order to improve search efficiency in search methods, we introduced relay-race algorithm (RRA) to explore similar local optimal solutions efficiently in [5]. In this paper, a detailed implementation of RRA is introduced, and the validity and efficiency of the implementation are investigated by trial experiments.

RRA consists of three stages: rough search, focusing search and relay. Rough search, focusing search, and relay are executed in turn and are repeated in the predetermined number.

Rough search is designed to get over small hills on the solution space and to approach a local optimal solution as fast as possible. Focusing search is designed to reach the local optimal solution as close as possible. During rough search and focusing search, a solution is iteratively modified whenever it is improved, and no degradation is allowed. A local optimal solution is obtained efficiently by a series of rough search and focusing search. Relay is one-time modification which is designed to escape from a local optimal solution. Even if a generated solution is worse than the current solution, it is always accepted. Relay is controlled by a global parameter to keep the search continuity. A generated solution is expected to be far enough from the current local optimal solution to escape from it and near enough to reach another local optimal solution.

To confirm the efficiency of RRA empirically, the placement optimization problem with multiple objectives is solved by RRA as a typical application. Nowadays, multi-objective problems have become common in practice. The optimization with conflicting objectives is facing big challenges to get good solution with short runtime. Pareto improvement of one objective with no degradation of another objective is one of the biggest challenges. It is important and urgent to explore more effective heuristic algorithms for multi-objective optimization. By using RRA, the Pareto improvement of placement with conflicting objectives is obtained. For area minimization, the runtime is reduced considerably for all tested benchmarks with better solution. For interconnect optimization, we gets low-power design with no degradation of delay in the best cases and in the worst cases.

2. Preliminary

The adjacency between solutions is defined by moving methods that modify a solution to another solution. A local optimal solution is a solution whose cost is not worse than any adjacent solutions. Whether a solution is a local optimal solution or not depends on the adjacency defined by moving methods. That is, a local optimal solution defined in terms of a set of moving methods is not necessarily a local optimal solution defined in terms of another set of moving methods.

Let Λ be the set of solutions. Let $C(S)$ be the cost of solution S in Λ. A moving method modifies a solution to another solution. The solution obtained from solution S by moving method M is denoted by $M(S)$. The composition of moving methods $\varphi = M_{|\varphi|}(\ldots M_2(M_1)\ldots)$ is said to be a series of modifications from one solution S_i to another solution S_j. Let $S_j = \varphi(S_i) = M_{|\varphi|}(\ldots M_2(M_1(S_i))\ldots)$ denote the solution obtained from a solution S_i to S_j by φ. The

Manuscript received March 14, 2014.
Manuscript revised June 28, 2014.

[†]The authors are with the Department of Communications and Computer Engineering, Tokyo Institute of Technology, Tokyo, 152-8550 Japan.
a) E-mail: sheng@eda.ce.titech.ac.jp
b) E-mail: atsushi@eda.ce.titech.ac.jp

DOI: 10.1587/transfun.E97.A.2418

length of φ which is the number of moving methods in φ is denoted by $|\varphi|$.

Let B be the set of moving methods which is used to define a solution space. A moving method in B is said to be *basic* hereinafter. In this paper, a local optimal solution defined in terms of B is simply referred by local optimal solution in the following unless otherwise specified. A sequence of moving methods is said to be *basic* if it consists of basic moving methods. In this paper, the distance between solutions in solution space is defined by using basic sequences. The distance dist(S_i, S_j) from S_i to S_j is defined as the minimum length of basic sequences from S_i to S_j. That is, dist(S_i, S_j) = min{ $|\varphi||\varphi$ in $P(S_i, S_j)$} where $P(S_i, S_j)$ is the set of basic sequences from solution S_i to solution S_j.

In most of search algorithms proposed so far, it is natural to consider that B consists of all moving method used in each search algorithm. There is no restriction on a moving method in B, but it is often assumed that a moving method M in B modifies a solution S slightly, and the difference between $C(S)$ and $C(M(S))$ is small. Moving methods used in SA are usually expected to satisfy this property.

3. Existing Search Algorithms

In typical search algorithms, an adjacent solution is randomly generated by a moving method. If the adjacent solution satisfies some condition, then the current solution is replaced with it. Otherwise, it is rejected, and the current solution is kept. In order to get a better solution efficiently, various ideas have been introduced.

A search algorithm with iterative improvement rejects an adjacent solution if it is worse than the current solution. It finishes when it regards the current solution as a local optimal solution. The final solution obtained by such algorithm is often far worse than a global optimal solution. When local optimum solutions are repeatedly obtained by iterative improvements from randomly generated initial solutions, the probability of finding a better solution becomes larger when the number of initial solutions is larger until a global optimum solution is found. However, an obtained local optimal solution is still independent of others. The efficiency is not good enough since the search history is not utilized.

Simulated annealing (SA) [6]–[9] accepts a non-improved adjacent solution with some probability according to temperature scheduling to escape from a local optimal solution. In order to obtain a better solution by SA, a solution space that satisfies the following properties is usually desired to be constructed. 1) The solution space is connected. This guarantees the existence of a search path from any initial solution to an optimal solution by a finite number of iterations. 2) The diameter of the solution space is small. This gives a chance to reach an optimal solution by a small number of iterations in best cases. 3) The difference of costs between adjacent solutions is small. This improves the stability and convergence of a search algorithm.

If the number of adjacent solutions in a solution space used in SA is large, then the connectivity and small diameter would be realized, and the number of local optimal solutions tends to be small. However, in the case of large number of adjacent solutions, the difference of costs between adjacent solutions tends to be large and the ratio of better solutions among adjacent solutions tends to be small. The probability that a generated adjacent solution is accepted becomes quite low. A set of moving methods in SA is designed so that the number of adjacent solutions is not so large and so that the difference of costs between adjacent solutions is not large. In general, a moving method that may cause a drastic change of solution is not used. This feature limits the efficiency in a global search of SA. Also, the number of required iterations including rejection from a local optimal solution to another local optimal solution tends to be large. Even if a solution space is explored globally in earlier stage of SA, similar local optimal solutions are not efficiently explored.

In order to improve the search ability of SA by using diverse moving methods with big changes, adaptive simulated annealing (ASA) was introduced in [6]. In ASA, a guide that changes the selection probability of each moving method adaptively is introduced. The selection probability of a moving method is increased by the guide when the frequency and amplitude of improvement achieved by the moving method in recent trials is high. The moving methods with big changes including group rotation, group exchange and a special crossover between the current solution and the best solution found so far are used in [6]. These moving methods are mainly selected in earlier stage of ASA and search efficiency is improved. However, it is still not efficient because the similar local optimal solutions are not explored in later stage of ASA.

Genetic algorithm (GA) [10] is a directed random search algorithm which is based on the evolution of a population of many individuals. As a population-based algorithm, GA requires a group of initial solutions to form a population. The crossover operator of GA is used to create new individuals as children from two existing individuals as parents. Since it is based on crossover of two different solutions in global scope even if mutation improves local search ability, the local search ability of GA is limited.

Variable neighborhood search (VNS) [11] utilizes search history in series of local searches. In VNS, the next local search starts from a k-th neighborhood of the local optimal solution obtained by the previous local search. Since k is gradually increased from 1, it is not efficient to get over small hills on the solution space.

Although various ideas have been used to improve the search efficiency as we discussed here, the efficiency of existing search algorithms is still limited. It is expected that a better local optimal solution exists near an obtained local optimal solution. A local optimal solution which is similar to an obtained local optimal solution is not effectively explored by existing algorithms.

4. Relay Race Algorithm

4.1 Overview

Relay race algorithm (RRA) is designed to approach a global optimal solution by exploring similar local optimal solutions more efficiently. Each runner is assigned to find one local optimal solution. A team is made up of a predetermined number of runners.

RRA consists of three stages: rough search, focusing search and relay. The detailed flow of RRA is as follows.

RRA starts with rough search, which is designed to approach one of local optimal solutions as fast as possible by using moving methods with big changes. A solution generated by a moving method is accepted if it is better than the current solution, and rejected otherwise. It finishes when the number of rejected trials reaches a predetermined number (N_r).

After rough search, focusing search is used to reach a local optimal solution as close as possible by using moving methods with small changes. A solution generated by a moving method is also accepted if it is better than the current solution, and rejected otherwise. It finishes if no solution is accepted during consecutive trials of length predetermined number (N_f).

After focusing search, a moving method with big changes, named relay, is used to escape from the current local optimal solution if the number of relays so far is less than the predetermined number (N_t). A solution generated by relay is always accepted. The relay is designed to improve global search ability while keeping search continuity.

After relay, RRA returns to rough and focusing searches again. The output of RRA is the best of all searched solutions.

In RRA, three types of moving methods, named focusing, rough and relay, are defined. Let Φ^f, Φ^r and Φ^e be the set of focusing moving methods, rough moving methods and relay, respectively. Φ^f, Φ^r and Φ^e are used in focusing search, rough search and relay, respectively.

In RRA, Φ^f is supposed to consist of moving methods with small changes. While, Φ^r and Φ^e are supposed to consist of moving methods with big changes, respectively. Also, a moving method in Φ^f is regarded as a basic moving method. That is, $\text{dist}(S, M(S)) = 1$ for any M in Φ^f, while $\text{dist}(S, M(S))$ is large in general for any M in Φ^r and Φ^e where S is a solution in Λ.

In Fig. 1, the behavior of RRA in solution space is illustrated. A solution is always improved in rough search and focusing search, while relay degrades a solution in general. The differences between rough search and focusing search are in moving methods and in terminal condition. The purpose of relay is to escape from a local optimal solution and to approach the next local optimal solution. In the following, the detail of each stage is explained.

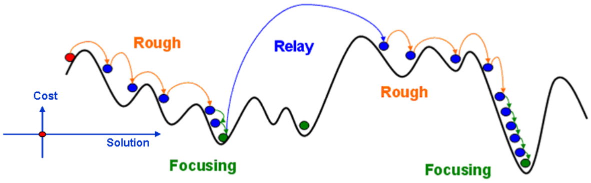

Fig. 1 Rough search, focusing search and relay.

4.2 Rough Search

For a given solution, rough search modifies the current solution iteratively by using rough moving methods Φ^r in the manner of greedy improvement. A candidate of the next solution is generated by a rough moving method from the current solution. If the cost of candidate is smaller than that of the current solution, the candidate becomes the current solution. Otherwise, the candidate is rejected and the current solution is kept. The terminal condition of rough search is that the number of rejected trials reaches the predetermined number N_r.

The parameter N_r is used to control the maximum scope of a single runner. In rough search, the goal is not a local optimal solution defined in terms of Φ^r, but a solution which is near to a local optimal solution defined in terms of $B = \Phi^f$. Rough search cannot get over any hills of the solution space defined in terms of Φ^r, but gets over small hills of the solution space defined in terms of B. A better solution would be obtained as trials are repeated. However it is expected that if the number of trials increases, then the probability of acceptance becomes quite low, and the efficiency of search degrades. Also, search continuity would be lost since the goal could be far from the given initial solution. Therefore, the number of rejected solution is limited by N_r to limit the search scope of a runner, but the number of accepted solution is not limited to approach a local optimal solution enough.

4.3 Focusing Search

For a given solution, focusing search modifies the current solution iteratively by using focusing moving methods Φ^f in the manner of greedy improvement. A candidate is generated by focusing moving method from the current solution. If the cost of candidate is smaller than that of the current solution, the candidate becomes the current solution. Otherwise, the candidate is rejected and the current solution is kept. Focusing search terminates if no candidate is accepted during consecutive trails of length predetermined number N_f.

The purpose of focusing search is to get a local optimal solution. However it is not easy to confirm that the current solution is a local optimal solution even if the number of adjacent solutions defined in terms of Φ^f is far smaller than that defined in terms of Φ^r. In order to keep the search efficiency, a current solution is regarded as a local optimal solution if no solution is accepted in consecutive N_f trials.

4.4 Relay

A solution generated by relay consists of two parts. One part is from a given solution and another part is randomly generated, though the latter is also allowed to be given by priori knowledge. The parameter R_e is used to defined the percentage of the randomly generated part of the solution. The part of randomly generated part is selected randomly.

The purpose of relay is to generate a solution which is far enough from a current solution to escape from a local optimum solution, and near enough to reach another local optimal solution.

5. Application

To confirm the computational performance of RRA, 2D rectangular placement problem in physical design is used.

5.1 Placement Problem

The placement is a typical NP-hard problem with conflicting objectives. 2D rectangular placement is to position modules or blocks into a fixed rectangular shape with interconnect optimization and area minimization. Placement problem is regarded as packing problem with interconnect optimization.

Let $M = \{m_1, m_2, \ldots, m_n\}$ denote the modules or blocks to be placed where n is the number of modules. Each m_i, where $1 \leq i \leq n$, has height h_i and width w_i. Let (x_i, y_i, r_i) of module m_i be the location and rotation on 2D orthogonal coordinate system where (x_i, y_i) represents the coordinates of the lower-left corner of module m_i, and r_i represents the rotation of m_i on xy-plane. $r_i = 1$ is the normal state of a module with vertical height and horizontal width, while $r_i = 0$ is rotated by 90 degree. See Fig. 2. The packing area is defined by the minimum bounding rectangle including all modules.

For the interconnect of placement, let $N = \{n_1, n_2, \ldots, n_l\}$ be the set of nets between modules where l is the number of nets. A net is a connection requirement between two or more modules and to be connected by wires. Let Len_i denote the estimated wire length of each net n_i, $1 \leq i \leq l$. Let P_i denote the estimated dynamic power, i.e. the interconnect power of net n_i.

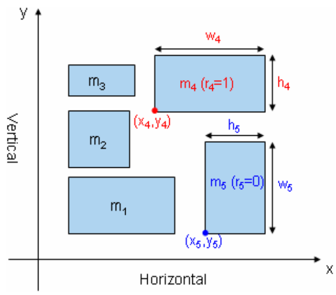

Fig. 2 Formulation of rectangular placement problem.

In short, the input is the set of modules $M = \{m_1, m_2, \ldots, m_n\}$ with height and width $\{(h_1, w_1), (h_2, w_2), \ldots, (h_n, w_n)\}$ and the net list $N = \{n_1, n_2, \ldots, n_l\}$. The constraint is no overlap between m_i and m_j, where $i \neq j$. The output is a set of location and rotation of modules $\{(x_1, y_1, r_1), (x_2, y_2, r_2), \ldots, (x_n, y_n, r_n)\}$ such that:

1. Minimize the power consumption $\sum_{1 \leq i \leq l} P_j$.
2. Minimize the maximal delay $\underset{1 \leq i \leq l}{Max}[Len_i]$.
3. Minimize the bounding area.

5.2 Problem Representation

To search near-optimal solutions of 2D placement efficiently, many researches explored representations for 2D placement, such as sequence pair (SP) [12], BSG [13], O-tree [14], B*-tree [15], CBL [16], FAST-SP [17], Q-sequence [18], Selected SP [19] etc. In this paper, a typical representation, called sequence pair (SP) [12], is used to represent 2D placement.

In SP, a pair of sequences (Γ^+, Γ^-) of modules represents the vertical and horizontal relations between modules. Let $\Gamma^i = (\Gamma^i[0], \Gamma^i[1], \ldots, \Gamma^i[n-1])$ be one of two sequences where i is + or −. Let $F^i(m)$ be the order of module m in sequence Γ^i. 2D topology is regarded as a set of the relations of relative location between modules, i.e. "Above, Below, Left and Right (ABLR)" relations.

Let $(m_i \text{ A } m_j)$, $(m_i \text{ B } m_j)$, $(m_i \text{ L } m_j)$, and $(m_i \text{ R } m_j)$ denote the relation that "m_i is above m_j," "m_i is below m_j," "m_i is left of m_j," and "m_i is right of m_j," respectively. SP defines $(m_i \text{ L } m_j)$ and $(m_j \text{ R } m_i)$ when

$$F^+(m_i) < F^+(m_j) \text{ and } F^-(m_i) < F^-(m_j).$$

It defines $(m_i \text{ A } m_j)$ and $(m_j \text{ B } m_i)$ when

$$F^+(m_i) < F^+(m_j) \text{ and } F^-(m_i) > F^-(m_j).$$

For a given packing with n modules, the size of solution space by SP is $(n!)^2$ if the rotation of the modules is fixed. If the rotation of the modules is not fixed, then the size of solution space is $(n!)^2 2^n$. Normally SP is decoded in time complexity $O(n^2)$. The decoding time is improved to $O(n \log n)$, further to $O(n \log \log n)$ using FAST-SP [17], and even to $O(n)$ using Selected SP [19].

5.3 Moving Methods

To solve placement problem by RRA, three focusing moving methods, called rotation, exchange and insertion, are used. Firstly, the rotation changes the orientation of a module. When a rotation is applied to module m_i, the orientation r_i is changed to $1 - r_i$. Secondly, the exchange moving method exchanges the order of two modules in all sequences, e.g. in sequence pair (Γ^+, Γ^-), $F^+(m_i)$, $F^-(m_i)$, $F^+(m_j)$, and $F^-(m_j)$ are changed to $F^+(m_j)$, $F^-(m_j)$, $F^+(m_i)$, and $F^-(m_i)$, respectively. Thirdly, the insertion changes the order of a module in one sequence Γ^i.

When a move is applied to module m in Γ^i, $F^i(m)$ is changed to another value, say j, and the orders of modules whose order is between $F^i(m)$ and j are shifted accordingly.

To improve the search efficiency of RRA, three rough moving methods, named as group rotation, group exchange and group insertion, are used. The group rotation is repeated rotations of randomly selected modules with a given number. The group exchange is repeated exchanges of randomly selected pairs of modules with a given number. The group insertion is repeated insertions of randomly selected pairs of modules in a selected sequence with a given number. The given number is set to be ten in this paper to satisfy that the roughing moves are much larger than the focusing moves.

For the relay operator used in the placement problem, a part of layout is randomly generated to escape from the local optimum and the remaining part is inherited from the current layout to keep the search continuity. The percentage of the random part depends on the parameter of relay R_e. The random part of representation is selected from Γ^+, Γ^- and rotation. Although R_e could be any value between 0 and 1, $R_e = 1/q$ is used to implement trial experiments this time, where q is a positive integer. The selection of the random part for Γ^+, Γ^- and rotation is set to be same. For example, if $q = 4$ (i.e. $R_e = 0.25$) for Γ^+, one of four following parts Γ^+ $[0, n/4 - 1]$, Γ^+ $[n/4, n/2 - 1]$, Γ^+ $[n/2, 3n/4 - 1]$ and Γ^+ $[3n/4, n - 1]$ is randomly selected, where n is the number of modules. If Γ^+ $[n/4, n/2 - 1]$ is selected, then Γ^- $[n/4, n/2 - 1]$ and the rotation of $[n/4, n/2 - 1]$ are also selected.

6. Experiment

6.1 Parameter Setting

Table 1 Trial experiments for N_f.

N_f	Number of Trials	Cost	
		Start	End
1	5	2.73	2.71
5	221	2.77	2.65
10	1885	2.67	2.46
50	7561	2.73	2.35
100	15534	2.58	2.08
500	29539	2.76	1.55
1000	58216	2.69	1.42
5000	106131	2.85	1.45
10000	156782	2.57	1.48

Table 2 Trial experiments for N_r (N_f = 1000).

N_r	Number of Trials			Cost		
	Rough	Focus	Total	Start	End of Rough	End of Focus
1	2	58377	58379	2.69	2.68	1.44
5	11	56821	56832	2.69	2.51	1.48
10	57	48240	48297	2.69	2.37	1.41
50	536	23754	24290	2.69	1.96	1.45
100	2563	14578	17141	2.69	1.81	1.38
500	5891	12488	18379	2.69	1.76	1.43
1000	12498	11425	23923	2.69	1.62	1.39
5000	25520	10251	35771	2.69	1.57	1.47
10000	46341	6664	53005	2.69	1.51	1.42

For the parameter setting of RRA, the best empirical values of parameters N_f, N_r, and R_e are investigated by trial experiments of placement using ami49 benchmark. Although the empirical values of parameters are not necessarily valid for different problems with different sizes or different objectives, the method below to get the values is still applicable.

In order to get the best value of N_f, the trial experiments of focusing search stage is executed without rough search and relay. All initial solutions are randomly generated. As shown in Table 1, the number of trails in focusing search stage increases as N_f increases. The cost of placement obtained by focusing search stage is decreasing continuously when N_f increases from 0 to 1000. However, the cost is almost same when N_f is larger than 1000. This shows that a local optimal solution is obtained when N_f is set larger than 1000. So we set $N_f = 1000$ for ami49 benchmark.

To get the best value of N_r, the trial experiments of rough search stage is executed without relay, but focusing search stage where $N_f = 1000$ follows. The same initial solution randomly generated is used. As shown in Table 2, the cost of solution at the end of rough search stage decreases as N_r increases, but the cost of a solution at the end of focusing search stage is almost same for all cases. Also, as N_r increases, the number of trials in focusing search stage decreases. This shows the usefulness of rough search stage in order to approach local optimal solution as fast as possible. However, the number of trials in rough search stage increases. The total number of trials is decreasing when N_r increases from 0 to 100. When N_r is larger than 100, the total number of trials starts to increase. So we set $N_r = 100$ for ami49 benchmark.

To get the best value of R_e, the trial experiments of RRA with $N_f = 1000$, $N_r = 100$, and $N_t = 10$ are executed. All initial solutions are randomly generated.

In Table 3, the results of $R_e = 1.00$ are shown. Each row corresponds to a runner in RRA. A solution generated by relay is random, and no search history is utilized. The number of trials and the cost of a solution at each stage vary among runners, but the deviation is not so large.

In Table 4, Table 5 and Table 6, the results of R_e = 0.20, 0.10 and 0.05 are shown, respectively. Each row corresponds to a runner in RRA. A solution generated by relay is generated by the solution obtained by the previous runner. In each relay, each sequence of SP of the previous runner is divided into 5, 10, and 20 parts, respectively, and one part is randomly selected for random part. As relay is repeated, the number of trials, the start cost, and the end cost of each runner decrease. This shows that the RRA explores near lo-

Table 3 Trial experiments when $R_e = 1.00$ ($N_r = 100$, $N_f = 1000$, $N_t = 10$).

	Number of trials			Cost		
	Rough	Focus	Total	Start of Rough	End of Rough	End of Focus
1	2131	14194	16325	2.58	1.85	1.67
2	2772	15083	17855	2.84	1.76	1.54
3	2413	17643	20056	2.55	1.89	1.50
4	2097	14081	16178	2.68	1.91	1.59
5	2368	17518	19886	2.80	1.96	1.42
6	2133	12087	14220	2.66	1.92	1.65
7	2819	16841	19660	2.88	1.62	1.42
8	2524	17918	20442	2.61	1.80	1.37
9	2604	12178	14782	2.87	1.92	1.68
10	2731	18056	20787	2.53	1.69	1.38

Table 4 Trial experiments when $R_e = 0.20$ ($N_r = 100$, $N_f = 1000$, $N_t = 10$).

	Number of trials			Cost		
	Rough	Focus	Total	Start of Rough	End of Rough	End of Focus
1	2347	14858	17205	2.73	1.80	1.63
2	1269	11771	13040	2.16	1.61	1.49
3	948	10811	11759	1.98	1.65	1.44
4	608	10025	10633	1.84	1.57	1.40
5	655	10119	10774	1.97	1.59	1.34
6	507	9220	9727	1.84	1.52	1.37
7	572	9280	9852	1.97	1.44	1.30
8	504	8803	9307	1.78	1.46	1.25
9	482	7910	8392	1.73	1.39	1.27
10	487	8351	8838	1.67	1.40	1.18

Table 5 Trial experiments when $R_e = 0.10$ ($N_r = 100$, $N_f = 1000$, $N_t = 10$).

	Number of trials			Cost		
	Rough	Focus	Total	Start of Rough	End of Rough	End of Focus
1	2381	14964	17345	2.94	1.79	1.62
2	1029	11243	12072	1.97	1.57	1.48
3	714	9721	10235	1.83	1.58	1.43
4	371	9378	9549	1.61	1.48	1.35
5	382	8939	9321	1.75	1.49	1.32
6	248	8763	8911	1.62	1.41	1.30
7	214	8074	8288	1.72	1.39	1.27
8	182	7349	7531	1.55	1.37	1.22
9	143	7229	7372	1.42	1.25	1.16
10	129	6803	6932	1.44	1.32	1.13

Table 6 Trial experiments when $R_e = 0.05$ ($N_r = 100$, $N_f = 1000$, $N_t = 10$).

	Number of trials			Cost		
	Rough	Focus	Total	Start of Rough	End of Rough	End of Focus
1	2586	14228	16814	2.73	1.73	1.58
2	972	9986	10958	1.68	1.56	1.45
3	622	9205	9827	1.61	1.50	1.43
4	382	8443	8825	1.62	1.49	1.44
5	311	7593	7904	1.60	1.47	1.38
6	197	6526	6723	1.61	1.46	1.35
7	241	7249	7490	1.59	1.44	1.33
8	125	6296	6421	1.58	1.43	1.30
9	133	5972	6105	1.55	1.47	1.34
10	220	5275	5495	1.52	1.43	1.29

cal optimal solutions efficiently by utilizing search history. Several trial experiments were executed by changing R_e, and we select $R_e = 0.10$ for ami49 benchmark that achieves the best final cost by 10 runners.

6.2 Cost Function and Comparison

The placement problem includes three objectives, including area, power and delay. Since the power and the delay are conflicting, we have to get Pareto improvement of multi-objective optimization. Let us define the total cost function as following equation.

$$C_t = \alpha \cdot C_p + \beta \cdot C_d + \gamma \cdot C_a$$

Where α, β and γ are user-defined coefficients that satisfy $\alpha + \beta + \gamma = 1$. The total cost function (C_t) is made up of the cost of power (C_p), the cost of delay (C_d) and the cost of area (C_a).

For power estimation, the dynamic power of a net n_i is proportional to $C(n_i)$, $V_{dd}(n_i)^2$, $f(n_i)$ and $S(n_i)$, where $C(n_i)$ is the capacitance of the net n_i, $V_{dd}(n_i)$ is the voltage of power supply, $f(n_i)$ is the clock frequency, and $S(n_i)$ is switching probability of the net. The relative value, which is the power divided by the limit of lowest power, is used as the value of C_p because the relative values are scalable. The $C(n_i)$ is proportional to the length of net, and we use Len_i to represent its value. We assume that $V_{dd}(n_i)$ and $f(n_i)$ are same for each net and $S(n_i)$ is randomly defined between 0 and 1 in case of no specific information. The power is simplified as the function of Len_i and $S(n_i)$. The wire length estimation (Len_i) of each net (n_i) is gotten by the half perimeter wire length ($HPWL$) which is defined by the half length of the perimeter of the smallest bounding box among all modules that the given net connects.

For delay estimation, the maximal delay among all nets is used. The relative value, which is the maximal delay divided by the limit of the shortest wire length estimation (Len_i) of nets, is used as the value of C_d. The area estima-

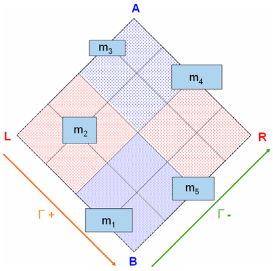

Fig. 3 Representation of rectangular placement problem.

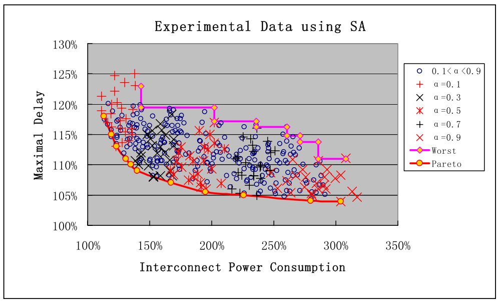

Fig. 4 The best and worst cases of experimental data using SA.

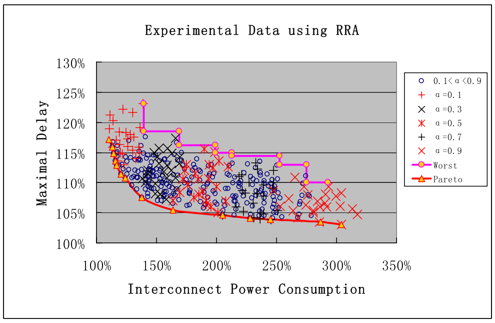

Fig. 5 The best and worst cases of experimental data using RRA.

tion is given by the minimum bounding rectangle including all modules. The relative value, which is the bounding area divided by the area of total modules, is used as the value of C_a.

For a fair comparison, all algorithms are implemented in Python environment on 2.16 GHz PC with 3.00 GB memory. With regard to ami49_X and MCNC benchmarks, a set of experiments was implemented. The ami49_X is generated by duplicating ami49 with X times. It has the scalable number of blocks (49*X) and nets (408*X). To compare with the published results fairly, MCNC benchmarks are used for area minimization. The parameters of RRA used in this experiments are $Re = 0.1$, $N_f = 1000$, $N_r = 100$. N_t is set to 1000/X for ami49_X, and is set to 10 for MCNC benchmarks. The parameters of SA are set as follows. The starting temperature $T_0 = 10000$, the ending temperature $T_e = 1$. The number of trials p in each temperature and the temperature coefficient a are set to $p = 1000/\mathrm{X}$ and $a = 0.98$ for ami49_X, and $p = 500$ and $a = 0.99$ for MCNC benchmarks. Results are normalized by using a lower bound of the cost function.

In the case of interconnect optimization, γ is set to 0 and $\alpha + \beta$ is set to 1. We are using ami49_4 benchmark for comparison this time. To get the Pareto frontiers of two conflicting objectives, α is randomly set from 0.1 to 0.9, and 240 solutions are picked up for comparison. The result of SA and the result of RRA are shown in Fig. 4 and Fig. 5, respectively. In this experiment, the time limit is set to 30 minutes, and the runtime of RRA is less than or equal to the runtime of SA. We get more than 30% improvement of interconnect power consumption with no degradation of performance. It obtains overall Pareto improvement. The "overall" means that Pareto improvement is achieved in all tested range of objectives, such as power consumptions from 100% to 350%, by multiple runs of RRA with various different choices of weighting coefficients.

In Fig. 6 and Fig. 7, comparisons of best-case and worst case of SA and RRA are shown, respectively. As shown in Fig. 6, RRA obtains at least 29% Pareto improvement with the constraint of less than 107.5% maximal delay. To check the worst cases more precisely, 120 solutions are further obtained by setting α to 0.1, 0.3, 0.5, 0.7, and 0.9. As a result, we confirm that RRA got near 24% worst-case mitigation on average for power consumption with no degradation of maximal delay as shown in Fig. 7. Similar results are gotten for all tested ami49_X, where X is from 1 to 12.

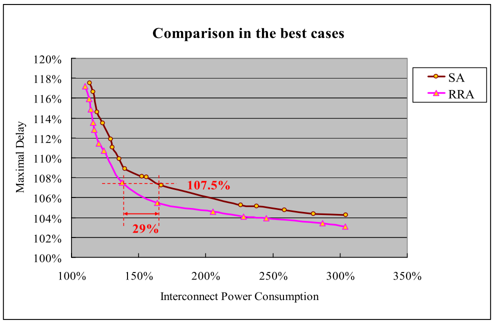

Fig. 6 Best-case improvement from SA to RRA.

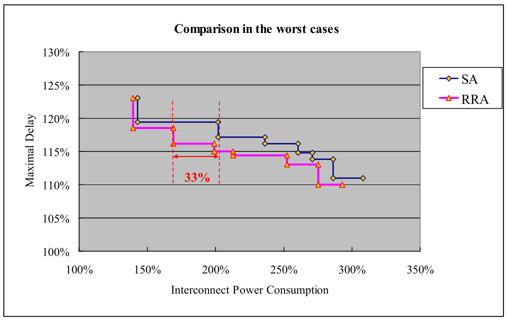

Fig. 7 Worst-case improvement from SA to RRA.

In case of area optimization, γ is set to 1 and $\alpha + \beta$ is set to 0. As shown in Table 7, the best, average and worst cases among 50 trials are gotten within 12 minutes. As shown in Table 8, the specific improvement of RRA is from 0.19% to 2.70%. RRA reduced at least 40% runtime with better so-

Table 7 Area minimization using SA and RRA (within 12 min).

MCNC	Best (mm^2)		Average (mm^2)		Worst (mm^2)	
	SA	RRA	SA	RRA	SA	RRA
apte	47.08	46.92	47.38	47.29	47.71	47.54
xerox	19.8	19.8	20.51	20.44	21.27	21.11
hp	9.03	8.95	9.18	9.14	9.38	9.25
ami33	1.19	1.18	1.24	1.22	1.30	1.27
ami49	37.23	36.52	37.96	37.17	38.97	38.01
ami49_2	74.13	72.99	75.98	74.46	77.65	75.78
ami49_4	149.2	145.8	152.2	148.4	156.7	150.7

Table 8 Average area improvement from SA to RRA.

MCNC	Solution (mm^2)		Runtime (s)		Improvement (%)	
	SA	RRA	SA	RRA	Solution	Runtime
apte	47.38	47.29	4.1	1.6	0.19%	61.0%
xerox	20.51	20.44	1.9	0.8	0.36%	57.9%
hp	9.18	9.14	2.7	1.1	0.45%	59.3%
ami33	1.24	1.22	22	13	1.73%	40.9%
ami49	37.96	37.17	45	26	2.23%	42.2%
ami49_2	75.98	74.46	194	91	2.15%	53.1%
ami49_4	152.2	148.4	720	371	2.70%	48.5%

Table 9 Comparison between RRA and the latest ASA [6].

MCNC	Solution (mm^2)		Runtime (s)		Improvement (%)	
	ASA	RRA	ASA	RRA	Solution	Runtime
apte	47.33	47.29	2.9	1.6	0.09%	44.8%
xerox	20.48	20.44	1.1	0.8	0.21%	27.3%
hp	9.17	9.14	1.5	1.1	0.34%	26.7%
ami33	1.23	1.22	15	13	0.86%	13.3%
ami49	37.68	37.17	29	26	1.44%	10.3%
ami49_2	75.17	74.46	121	91	1.00%	24.8%
ami49_4	150.3	148.4	519	371	1.34%	28.5%

lution by comparison to SA with same representation. As shown in Table 9, the comparison between RRA and the latest proposed ASA [6] shows the considerable improvement of both solution and runtime.

In this paper, the single-objective function problem is solved multiple times independently with fixed values of coefficients to obtain the Pareto frontiers of conflicting objectives. Based on the experimental data, the proposed RRA is more efficient to get Pareto optimal solutions comparing with SA under the same conditions. However, the local optimal solutions obtained by iterative improvements might be improved when coefficients are dynamically changed during search as mentioned in [7]. Changing the coefficient during search in RRA to improve the search efficiency is one of our future works.

7. Conclusion

In this paper, relay-race algorithm (RRA) is investigated. In RRA, rough search is designed to approach a local optimum solution as fast as possible and focusing search is designed to approach the local optimum solution as close as possible. Relay escapes from a local optimal solution efficiently while keeping search continuities. The efficiency of RRA is confirmed by applying it to placement problem in physical design optimization problems. Based on the experimental data using MCNC and ami49_X benchmarks, the big improvement of area, power and delay is achieved by using RRA. The overall Pareto improvement between power and maximal delay is obtained. In the worst cases, the big average improvement of power is obtained without any degradation of maximal delay. With regard to its impact, the proposed RRA has potential to improve more NP-hard problems.

Acknowledgment

The authors would like to thank Prof. Kunihiro Fujiyoshi at Tokyo University of Agriculture and Technology, and thank Dr. Tayu, Mr. Shinoda, Mr. Inoue and Mr. Zhao at Tokyo Institute of Technology for their discussion and support.

References

[1] P. Subbaraj, S. Sankar, and S. Anand, "Parallel genetic algorithm for VLSI standard Cell placement," International Conference on Advances in Computing, Control, and Telecommunication Technologies, pp.70–84, 2009.

[2] F. Mao, N. Xu, and Y. Ma, "Hybrid algorithm for floorplanning using B*-tree representation," Proc. Third International Symposium on Intelligent Information Technology Application, pp.228–231, 2009.

[3] M. Tang and X. Yao, "A memetic algorithm for VLSI floorplanning," IEEE Trans. Syst. Man. Cybern., vol.37, no.1, pp.62–69, 2007.

[4] D. Stefankovic, "Adaptive simulated annealing: A near-optimal connection between sampling and counting," 48th Annual IEEE Symposium on Foundations of Computer Science, pp.183–193, 2007.

[5] Y. Sheng, A. Takahashi, and S. Ueno, "Relay-race algorithm: A novel heuristic approach to VLSI/PCB placement," Proc. IEEE Computer Society Annual Symposium on VLSI, pp.96–101, 2011.

[6] Y. Sheng and A. Takahashi, "A new variation of adaptive simulated annealing for 2D/3D packing optimization," Information Processing Society of Japan (IPSJ) Transactions on System LSI Design Methodology, vol.6, pp.94–100, 2013.

[7] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," IEEE Trans. Evol. Comput., vol.12, no.3, pp.269–283, 2008.

[8] S.Y. Ho, Y.K. Lin, and W.C. Chu, "An orthogonal simulated annealing algorithm for large floorplanning problems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.12, no.8, pp.874–877, 2004.

[9] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," Science, vol.220, no.4598, pp.671–680, 1983.

[10] A. Drakidis, R.J. Mack, and R.E. Massara, "Packing-based VLSI module placement using genetic algorithm with sequence-pair representation," IEE Proc. Circuits, Devices and Systems, pp.545–551, 2006.

[11] P. Hansen and N. Mladenovic, "Variable neighborhood search: Principles and applications," European Journal of Operational Research, vol.130, no.3, pp.449–467, 2001.

[12] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.15, no.12, pp.1518–1524, 1996.

[13] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani, "Module placement on BSG-structure and IC layout applications," Proc. International Conference on CAD, pp.484–491, 1996.

[14] P.N. Guo, C.K. Cheng, and T. Yoshimura, "An O-tree representation of non-slicing floorplan and its applications," Proc. Design Automation Conference, pp.268–273, 1999.

[15] Y.C. Chang, Y.W. Chang, G.M. Wu, and S.W. Wu, "B*-trees: A new representation for non-slicing floorplans," Proc. Design Automation Conference, pp.458–463, 2000.

[16] X. Hong, G. Huang, Y. Cai, S. Dong, C.K. Cheng, and J. Gu, "Corner block list: An effective and efficient topological representation of non-slicing floorplan," Proc. International Conference on Computer Aided Design, pp.8–12, 2000.

[17] X. Tang and D.F. Wong, "FAST-SP: A fast algorithm for block placement based on sequence pair," Proc. IEEE Asia South Pacific Design Automation Conference, pp.521–526, 2001.

[18] C. Zhuang, K. Sakanushi, L. Jin, and Y. Kajitani, "An enhanced Q-sequence augmented with empty-room-insertion and parenthesis trees," Proc. Design, Automation and Test in Europe Conference and Exhibition, pp.61–68, 2002.

[19] C. Kodama and K. Fujiyoshi, "Selected sequence-pair: An efficient decodable packing representation in linear time using sequence-pair," Proc. IEEE Asia South Pacific Design Automation Conference, pp.331–337, 2003.

Yiqiang Sheng received his Master's degree from Nankai University, Tianjin, China, in 2003. He worked at Sanyo Electric Co., Ltd. in Osaka, Japan, from 2003 to 2008 and at Nokia Corporation in Tokyo, Japan, from 2008 to 2010. He did research at Osaka University, Osaka, Japan, from 2010 to 2011. He received his Doctoral degree from Tokyo Institute of Technology, Tokyo, Japan, in 2014. His current research interests are in design optimization and heuristics.

Atsushi Takahashi received his B.E., M.E., and D.E. degrees in electrical and electronic engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1989, 1991, and 1996, respectively. He had been with the Tokyo Institute of Technology as a research associate from 1991 to 1997, and as an associate professor from 1997 to 2009, and from 2012. He had been with the Osaka University as an associate professor from 2009 to 2012. He visited University of California, Los Angeles, U.S.A., as a visiting scholar from 2002 to 2003. He is currently with Department of Communications and Computer Engineering, Graduate School of Science and Engineering, Tokyo Institute of Technology, as an associate professor since 2013. His research interests are in VLSI layout design and combinatorial algorithms. He is a member of IEEE, ACM, and IPSJ.