공학박사학위논문

An Adaptive Method of Mating Schemes in Genetic Algorithms

유전 알고리즘에서의 적응적 짝짓기 제도

2017년 2월

서울대학교 대학원
전기·컴퓨터공학부
정 찬 주

An Adaptive Method of Mating Schemes in Genetic Algorithms

by

Chanju Jung

School of Computer Science & Engineering
Seoul National University
2017

Abstract

Mating scheme is the way of selecting two parents to make offspring. It takes effect on the performance of genetic algorithms. In this thesis, we investigate mating schemes using the Hungarian method. The schemes include i) minimizing the sum of matching distances, ii) maximizing the sum, and iii) random matching for comparison. We apply the schemes to well-known combinatorial optimization problems, the traveling salesman problem and the graph bisection problem, and analyze how the quality of the best individual changes over generations. Based on the analysis, we suggest a simple hybrid mating scheme. The suggested hybrid scheme showed better performance than the non-hybrid schemes.

we also propose our main method, an adaptive mating method combining mating schemes. Our adaptive mating method selects one of the Hungarian schemes with voting. Every matched pair of individuals has the right to

vote for the mating scheme of the next generation. Its preference is closely related to the ratio of distance between parents over distance between parent and offspring. The proposed Hungarian adaptive method showed better performance than any single Hungarian mating scheme, the non-adaptive hybrid scheme, traditional roulette-wheel selection, and the other distance-based mating methods. The proposed Hungarian adaptive method selected a proper mating scheme with a periodic influx and local-optimization. We replaced the Hungarian schemes with finding local maximization or minimization. Our replaced adaptive mating scheme also outperformed single scheme finding local maximization or minimization.

Keywords : genetic algorithm, selection operator, mating schemes, Hungarian method, combinatorial optimization, treveling salesman problem, graph partition

Student Number : 2008-30885

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Genetic Algorithms (GAs) are inspired from natural evolution, and therefore, their main operators resemble the natural phenomena. Mating or marriage is important for each individual. There are various mating schemes, depending on the culture or species. Some studies suggest that people exhibit assortative mating, preferring partners with similar religion, habits, age, and other characteristics [HF11][Mil99]. However mating of two extremely similar individuals, such as in incest or inbreeding, is known to cause genetic disorders [Kin89].

Selection, crossover, and mutation are the key operators in GA. Most operations in GAs are closely related to the performance. These operations interact on each other. Selection or mating scheme is the way of selecting two parents to make offspring. A small change of a key operation may cause a dramatic change in result. Ochoa et al. [OMKRJ05] presented that an assortative mating is a good choice when the mutation rate is high while a disassortative mating is a good choice when that is low.

1.1 Motivation

Every space search algorithm needs to balance exploration and exploitation [HL99][ČLM13]. Exploration is the process of visiting wide re-

gions of a search space, whereas exploitation is the process of visiting the regions already known to have high fitness. The GA hyperparameters include the population size, crossover rate, and mutation rate. These hyperparameters and the genetic operators interact with each other. The choice of crossover and mutation operator changes the proper values of hyperparameters.

Most of the selection algorithms select the parents based on their fitness values. The higher their fitness value, the higher the chance of being selected. The selection pressure is defined as the degree to which the better individuals are favored. A higher selection pressure leads to a the faster convergence and may cause premature convergence. We can adjust the selection pressure in order to balance exploration and exploitation. Finding a proper selection pressure is difficult and ineffective. In this thesis, we propose mating schemes, which are the extreme cases of exploration and exploitation. In addition, we propose a combined method of our mating schemes. Our new method selects the best adaptive scheme in each generation.

1.2 Related Work

In GAs, mating scheme means the way of selecting two solutions to crossover. In mating or selection stage, the methods of mating are classified into three groups. The first one gives preference to similar solutions [Boo85][FF93]. It can be realized by a *restriction*. Restricted mating has been suggested in [Gol89].

This method assumes that dissimilar parents make worse offspring. It

has the characteristic of using good schemata already discovered. That is, it focuses on exploitation.

The second one is dissimilar mating. Goh et al. [GLR03] adopted the concept of gender. They allowed only pairs of different genders. Some other concepts were used in gender-based GAs. Garcia-Marinez and Lozano [GML05] selected female solutions widely first, and then they selected male solutions dissimilar with their corresponding female ones. This method focuses on exploration, and it tries to evade a premature converge and a fast diversity consumption of similar mating. It is realized mostly by a restriction. Ramezani and Lotfi [RL11] restricted a mating between family solutions such as parent and offspring. They banned ancestor-child or sibling crossover. It is an expansion of incest prevention [Mic96]. They solved function optimization problems and obtained good results. Fernandes et al. [FTMR01] compared positive assortative mating with negative one. They applied their GA to vector quantization problems, and got better results with negative assortative mating than a simple GA or positive assortative mating.

The last group tries to find a better mating scheme by combining two or more mating schemes. Ishibuchi et al. [INTN08][IS04] adjusted diversity and convergence with the control of selecting parents and the number of candidates. The first parent determined considering with how major(or extreme) solution would be selected. It is controlled by parameter δ. The second parent was selected to be a similar(or dissimilar) solution to the first one. It is controlled by parameter ω. They solved multi-objective knapsack problems. They adjusted parameters δ and ω by the method changing their strategy at a half of process. They showed the results were nearer to Pareto-

optimal solutions than a single strategy. They showed that the results were nearer to Pareto-optimal solutions than those by a single strategy.

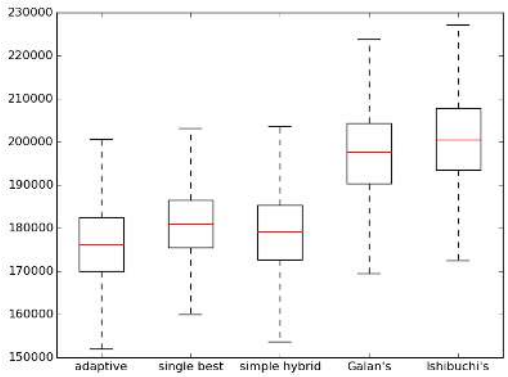
Galán et al. [GMP13] proposed a mating scheme that each individual has its mating preference value to balance exploitation and exploration. The mating preference can be from one to the number of population minus one. A low value of mating preference makes a match between solutions close to each other while a high mating preference makes a match between solutions far from each other. The preference is inherited or mutated like a normal gene. They tested their scheme in various environments of function optimization. They showed that their scheme outperformed a random mating or a scheme with a fixed preference value.

1.3 Contribution

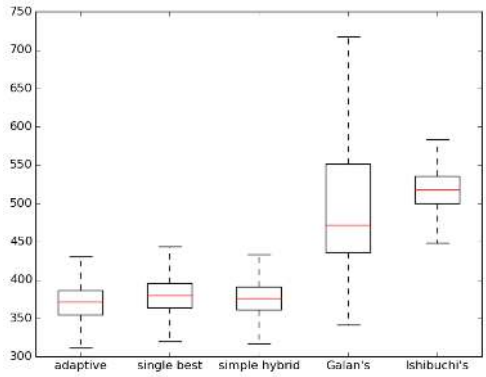
This thesis has two main contributions.

- We propose and observe Hungarian mating schemes. The schemes are the upper and lower bound of the sum of distances, and represent exploration and exploitation. We observe and compare the behaviors of our schemes.

- We propose an adaptive hybrid scheme of Hungarian mating schemes. This scheme dynamically selects a proper scheme in each generation. Our new scheme statistically outperforms single schemes and a simple hybrid scheme.



(a) pr152



(b) U1000.05

Figure 1: Main comparison results

Figure 1 shows the comparison result of two different problems. The label 'adaptive' denotes the distribution of the solution qualities of our adaptive method. It is described in Section 4.2.2. The label 'single best' part is the collection of best single Hungarian schemes. The 'simple hybrid' part is described in Section 4.1. Galán's and Ishibuchi's methods are distance-based mating schemes. The description and comparison results are reported in Section 4.2.8. Our adaptive algorithm shows significantly better results

than all other methods in almost all instances.

1.4 Organization

This thesis is organized in six chapters. In the next chapter, we review the Hungarian method, the properties of geometric crossover, the testing problems, and the distance metric. In Chapter 3, we explain our Hungarian mating schemes, and the primary approach of this thesis. In Chapter 4, we propose a simple hybrid method and an adaptive mating scheme for combinatorial optimization problems and present experimental results. We tested our adaptive mating scheme in various environments including a hybrid GA. We discuss the results in Chapter 5. The revised version of our adaptive mating scheme is tested in Chapter 6. Finally, we make conclusions in Chapter 7.

Chapter 2

Preliminary

2.1 Hungarian Method

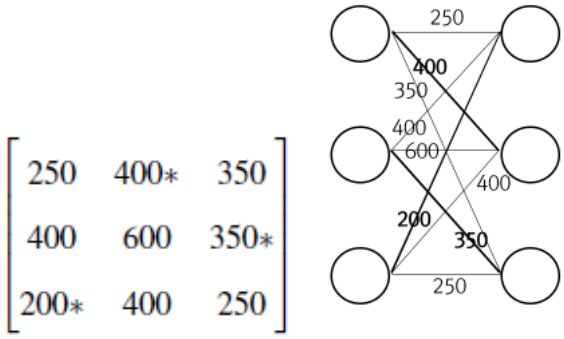
Consider a weighted complete bipartite graph with bipartition (X, Y), where $X = x_1, x_2, ..., x_{n/2}$, $Y = y_1, y_2, ..., y_{n/2}$, and each edge $(x_i, y_j) \in X \times Y$ has its weight w_{ij} . The optimal assignment problem is the problem of finding a maximum(or minimum) weight perfect matching in this weighted graph as follows:

$$\max_{\sigma \in \sum_{n/2}} \left(\sum_{i=1}^{n/2} w_{i\sigma(i)} \right) \text{ or } \min_{\sigma \in \sum_{n/2}} \left(\sum_{i=1}^{n/2} w_{i\sigma(i)} \right),$$

where σ is a permutation of size $n/2$.

The assignment problem can be represented as an N x N matrix. The matrix contains edge weights. In each row and column, we should select only one weight. Figure 2(a) shows the cost matrix of Figure 2(b). The marked position of the cost matrix is the optimal (minimum) assignment of the instance. Corresponding costs are represented as thick lines in Figure 2(b).

$n!$ ways of assigning n Xs and n Ys exist. $n!$ is intractable as n grows. With the use of Hungarian method, we can compute the optimal assignment in $O(n^3)$. The Hungarian method uses the following property: If a number



(a) Cost matrix

(b) Bipartite graph and optimal assignment

Figure 2: An instance of the optimal assignment problem

is added to or subtracted from all the entries of any one row or column of the matrix, then the optimal assignment for the resulting cost matrix does not change. The Hungarian method consists of five steps. In step one, the smallest entry in each row is subtracted from all the entries of its row. In step two, the smallest entry in each column is subtracted from all the entries of its column. Step three covers all the zeros of the matrix with the minimum number of horizontal or vertical lines. Step four is the checking step. If the sum of horizontal lines and vertical lines is less than n, then step five is performed. Otherwise, the zeros indicate the optimal assignment and the method terminates. In step five, the method subtracts the smallest entry in the uncovered entries from each uncovered entry. Then the method returns to step three.

Figure 3 shows a 4x4 matrix as an instance of the Hungarian method. Figure 3(a) is the given cost matrix, and Figure 3(b) is the result of step one.

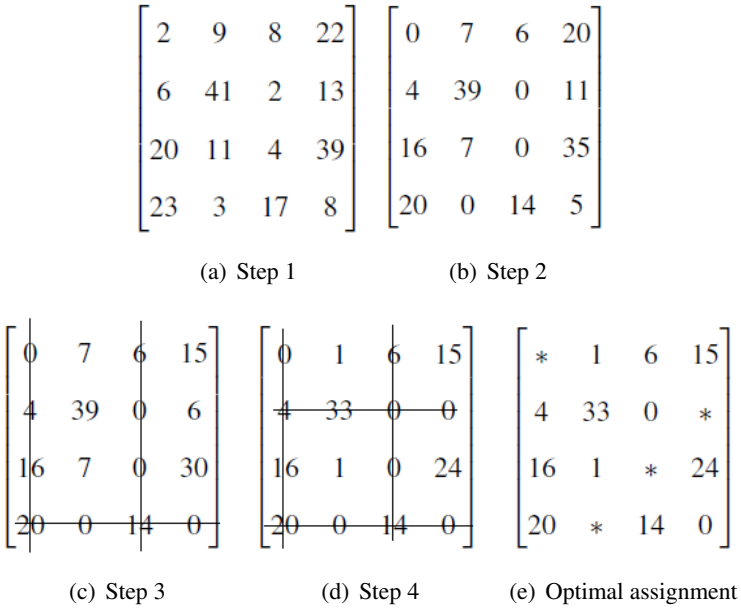


Figure 3: Example of the Hungarian method

In each row, two, two, three and four are subtracted. Figure 3(c) is the result of step two. In the first, second, and third columns, zero is subtracted. In the fourth column, five is subtracted. Figure 3(c) shows two vertical lines and one horizontal line to cover all the zeros. The number of lines is less than four, and then the method proceeds to the next step. In Step 5, the smallest cost in the uncovered number is six. Thus, the method subtracts six from the following entries: 7, 15, 39, 6, 7, and 30. The result is illustrated in Figure 3(d). The method then returns to step three. We need four lines to cover all the zeros. Figure 3(e) is the optimal assignment of the given cost matrix. Optimal matching is marked with a star(*) in Figure 3(e).

The Hungarian method [Kuh55] gives an optimum assignment, and it can be implemented in $O(n^3)$ time [PS82]. Avis [Avi83] has provided an ef-

ficient approximation. It can be implemented in $O(n^2)$ time. The Hungarian method has been used in various studies [HGH98][MKYM07][YKM08][YKMM12].

2.2 Geometric Operators

Geometric crossover and mutation are representation-independent search operators. Geometric crossover is defined as the offspring in the line segment between the parents for a certain distance. Moraglio [Mor07] proposed geometric crossover and mutation. He generalized pre-existing search operators used in GA and other evolutionary algorithms. He showed that all mask-based crossovers such as one-point, n-point crossovers, and uniform crossover [Sys89] for binary strings to any representation are geometric crossovers [AR06]. He showed that partially matched crossover [GL85], cycle crossover, and merge crossover are also geometric crossovers.

2.2.1 Formal Definitions

The formal definition of geometric crossover is as follows [Mor07]:

Definition: The image set Im[OP] of a genetic operator OP is the set of all possible offspring produced by OP with non-zero probability.

$IM[OP(p_1, p_2, ..., p_g)] = \{c \in S | f_{op}(c|p_1, p_2, ..., p_g) > 0\}$

Definition: A binary operator is a geometric crossover under the metric d if all offspring are in the segment between its parents.

Definition: A unary operator M is a topological e-mutation operator if $IM[M(p)] \subseteq B(p;e)$. $B(x;y) = \{y \in S | d(x,y) \leq r\}$, where r is a positive real number. r is the radius of the ball.

The definition is based on the notion of metric segment. The definition is independent with the representation of an individual. It is associated with the search space and a distance metric d. For example, in the 2D Euclidean space, the area with the function B makes a circle. Otherwise, in the Manhattan space, the area with the function B makes a diamond shape. The shape of a line segment is also dependent on the metric d. For example, in the 2D Euclidean space, the shape of a line segment forms a line. Otherwise, in the Manhattan space, the shape of a line segment forms a rectangle.

In the later parts of this thesis, all the crossovers we used are the geometric crossovers, and the mutations we used are geometric mutations.

2.3 Exploration Versus Exploitation Trade-off

All search algorithms have two sides of property. Exploration is the process of searching wide new regions, whereas exploitation involves visiting the regions already known to have high fitness. If a search algorithm is focused too much on exploration, then the algorithm may use too much time in low-fitness regions. Otherwise, if a search algorithm is focused too much on exploitation, it may find the local best solution in a small region. The local best solution can be worse than the global best solution. A good search algorithm balances well between exploration and exploitation.

In GAs, almost all parts of the GA affect the balance between exploration and exploitation. The parts are listed below:

- The structure of GA: steady-state GA changes one individual in one generation. It focuses on exploitation, whereas generational GA changes

more individuals in one generation. If steady-state GA finds a better solution than other individuals, the solution has high chance to be selected and the offspring may have a high chance to survive. Thus, the solution and their offspring may occupy whole individuals.

- Encoding: the encoding scheme affects most operators in GA, and the scheme changes the landscape of fitness.

- Selection operator and its parameters: this is the operator we are mainly concerned about. The selection or mating schemes can adjust the balance between exploration and exploitation.

- Crossover operator: the crossover operator is highly correlated with the problem to be solved. The crossover operator can also adjust the balance between exploration and exploitation. For example, one-point crossover leans on exploitation than five-point crossover does [DJS92].

- Mutation operator: the mutation operator is also highly correlated with the problem and their encoding. It disturbs the solution. The mutation operator can also adjust the balance between exploration and exploitation. For example, no mutation operator is an extreme case of exploitation and highly disturbing mutation focuses on exploration.

- Replacement operator: the replacement operator directly affects the convergence speed. Replacement the worst solution is more highly focused on exploitation than replacement with the parent solutions or the most similar solution.

- Hyperparameter: they include the number of individuals and mutation rate. A small number of individuals and low mutation rate lean more toward exploitation.

If a GA is highly focused on exploitation, then most individuals resemble each other in the early stage of GA; this situation called premature convergence. Controlling convergence speed is important to obtain a better solution.

2.4 Test Problems and Distance Metric

Combinatorial optimization problem is a set of optimization problems where one tries to find the best solution in a discrete solution space. Famous NP-hard problems related to combinatorial optimization include the traveling salesman problem(TSP) [HPR13], knapsack problems [BCM03], job scheduling [GdMMR05], and the graph partitioning problem, and so on. The solution spaces of these NP-hard problems are usually very complex requiring us to resort to heuristics such as evolutionary algorithms. There have been many attempts to solve combinatorial optimization problems using GAs [TB91][JB91][Lev93]. Some have successfully replaced the best-known solutions [KM04][MF00][SM02].

We considered TSP and the graph bisection problem as test problems. For the TSP, we are given a complete undirected graph G that has a nonnegative integer cost associated with each edge. TSP requires to find a Hamiltonian cycle (a tour that passes through all the vertices) of G with minimum cost.

Let $G = (V, E)$ be an unweighted undirected graph, where V is the set of vertices and E is the set of edges. K-way partition is a partitioning of the vertex set V into K disjoint subsets $C_1, C_2, ..., C_K$. A K-way partition is said to be balanced if the difference of cardinalities between the largest and the smallest subsets is at most one. The cut size of a partition is defined to be the number of edges with endpoints in different subsets of the partition. The K-way partitioning problem is the problem of finding a K-way balanced partition with minimum cut size. In this study, we made experiments on the case that K is equal to two. We call this problem "graph bisection" in this paper.

We used the quotient swap distance [YKMM12] for the phenotype distance metric in TSP. The quotient swap distance of between X and Y is defined by the the nearest one among swap distances (between X and every shifted Y), where the swap distance between X and Y is the minimum number of swaps to make X be equal to Y.

We used the quotient Hamming distance [YKMM12] for the phenotype distance metric in graph bisection. In a similar way, the quotient Hamming distance is defined by the nearest one of Hamming distances (between X and Y, and between X and $\overline{Y}$). The Hamming distance between two strings is the number of different positions at which the corresponding symbols are different.

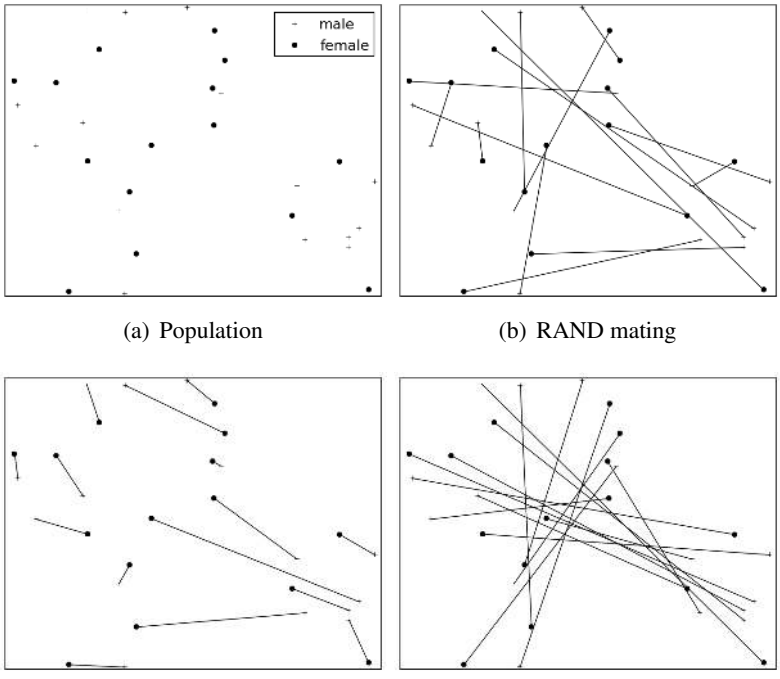
Chapter 3

Hungarian Mating Scheme

3.1 Proposed Scheme

Our GA has a population of size n. We use the concept of gender as in [GML05][GLR03][RL11]. Population is divided into $n/2$ male solutions and $n/2$ female ones. We made a mating scheme that all female solutions matches all different male ones in a generation. We calculated distances between each pair of male and female solutions, using the proper distance metric defined for each problem (see Section 2.4).

The tested mating schemes are illustrated in Figure 4. Figure 4(a) illustrates the distribution of a population. The individuals (x, y)s are distributed in a real square domain. The distance metric used in this illustration is the Euclidean distance. Thirty individuals are shown, where filled circles denote male individuals, and plus symbols denote female individuals. Figure 4(b) represents the scheme that matches randomly. It has only one rule that every individual should match with one of opposite gender (one-to-one matching). We used this scheme to compare with other schemes. We call this scheme "RAND". Figure 4(c) shows the result optimized using the following for-



(a) Population (b) RAND mating

(c) NEAR mating (d) FAR mating

Figure 4: Mating schemes

mula with the Hungarian method:

$$\min_{\sigma \in \sum_{n/2}} \sum_{i=1}^{n/2} d\left(m_i, f_{\sigma(i)}\right),$$

where m_is are male solutions, $f_{\sigma(i)}$s are their corresponding female ones, and d is a distance metric. Some individuals may have nearer points than the ones in the figure. But note that the minimization is considered globally. We name this scheme "NEAR". Figure 4(d) shows the result using the

Hungarian method to optimize:

$$\max_{\sigma \in \sum_{n/2}} \sum_{i=1}^{n/2} d\left(m_i, f_{\sigma(i)}\right).$$

It also considers the maximization globally. This scheme is called “FAR”.

The maximized(or minimized) sum of pairs matched by the Hungarian method is presented as a notable feature. NEAR method guarantees the minimum sum of distances in one-to-one matching. And it uses a deterministic method. So we can guess the characteristic of mating. NEAR method can be considered as an extreme strategy of decreasing diversity in one-to-one matching. Similarly, FAR method can be considered as an extreme method of preserving diversity in one-to-one matching. In other words, NEAR method is an extreme case of exploitation, and FAR method is an extreme case of exploration.

Table 1: Genetic parameter settings

	Traveling salesman problem
Encoding	Order-based encoding
Crossover	Partially matched crossover [GL85]
Mutation	Double bridge kick move [MSWO91] (50%)
Repair	-
Stop condition	1,000 generations

	Graph bisection problem
Encoding	Assignment of one gene for each vertex (zero or one)
Crossover	One-point crossover
Mutation	Random swap of some pairs of genes (50%)
Repair	Random repair until partition is balanced
Stop condition	500 generations

3.2 Tested GA

We implemented a generational GA. All male individuals are one-to-one matched with all female ones. A pair of individuals produces two offspring. An offspring nearer to father than the other offspring is set to be a male individual. The remaining offspring is considered as a female individual.

As population size, we used 50 male individuals and 50 female ones. As replacement, we used elitism [DJ75] in both genders. Among new 50 offspring and previous 50 individuals, we chose 50 best ones for the next generation.

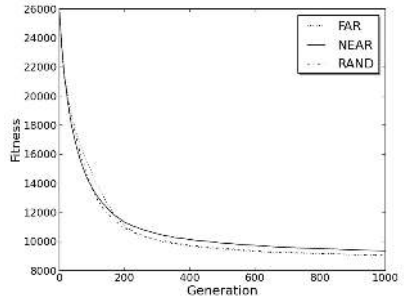
The other genetic parameters are described in Table 1. In order-based encoding of TSP, a permutation of city numbers means a travel path itself. In the graph bisection problem, each position of genes denotes each vertex. Vertices valued with zero form a partition, and vertices valued with one are grouped together in the other partition.

3.3 Observation

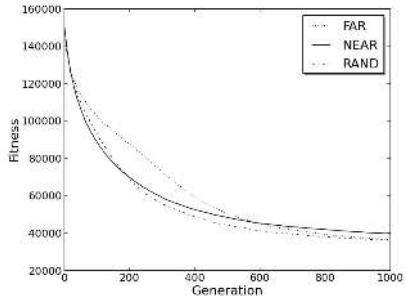
3.3.1 Traveling Salesman Problem

We selected four Euclidean instances from TSPLib [Rei91]. They are berlin52, kroA100, bier127, and pr152. The numbers in the instance name mean the number of cities of the instance.

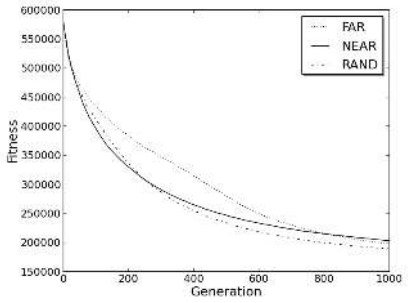
Figure 5 shows the fitness of the best individual according to generation. The plotted results are the average value over 1,000 runs. Table 2



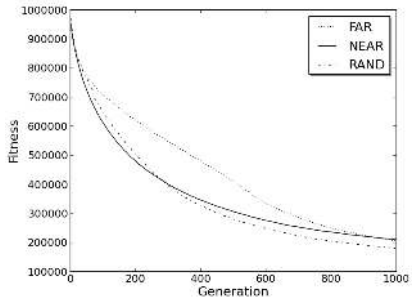
(a) berlin52



(b) kroA100



(c) bier127

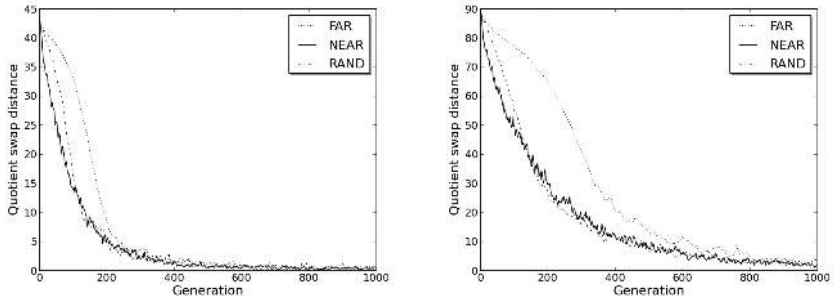


(d) pr152

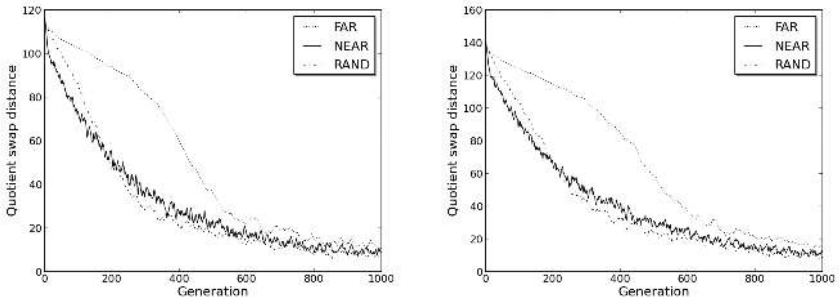
Figure 5: Fitness of mating schemes in TSP (the smaller, the better)

gives the average best fitness and the standard deviation per m generation. m means the number of cities of each instance. Figure 6 shows the diversity of a population. The diversity is measured by the average distance within a population. The diversity lines of RAND and NEAR are quite similar. But the line of FAR drops very slowly, especially in pr152. It means that there are many types of low-order schemata in a solution pool. At the end of running, the diversity of FAR drops near NEAR and RAND, and its quality exceeds NEAR.

In all instances, we observed a superiority of NEAR in early stage,



(a) berlin52 (b) kroA100



(c) bier127 (d) pr152

Figure 6: Average population distance of mating schemes in TSP

and RAND in later stage. RAND outpaced NEAR after generation $2m$ in all cases. As the size of the solution space grows, the time that crosses RAND and NEAR was delayed. When we consider the cases that FAR performed worse than RAND or NEAR, not only the period but also fitness difference increased. Our results show that NEAR scheme is a good strategy in sufficiently large instances of TSP. In the case of relatively small instances, FAR produced better results than RAND at the end. If a sufficient time is allowed to solve TSP, we can guess that FAR scheme has a potential to exceed RAND.

Table 3 shows the significance of results of Table 2 statistically. We used Welch's t-test[Wel47]. The t-value of $A - B$ in Table 3 is computed as follows:

$$t = \frac{\overline{X_A} - \overline{X_B}}{\sqrt{S_A^2/n_A + S_B^2/n_B}},$$

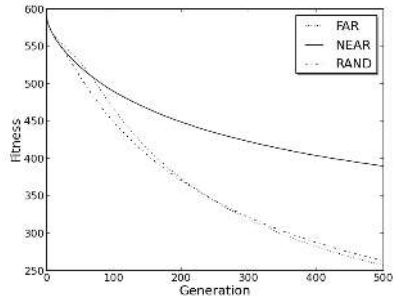
where X_A is the average of A, S_A is the standard deviation of A, and n_A is the number of runs of A. The lower p-value means the more significant result. We computed p-value with the absolute value of difference between two mating schemes. In most cases, p-values are very close to zero. It means FAR and NEAR schemes showed significant different results.

3.3.2 Graph Bisection Problem

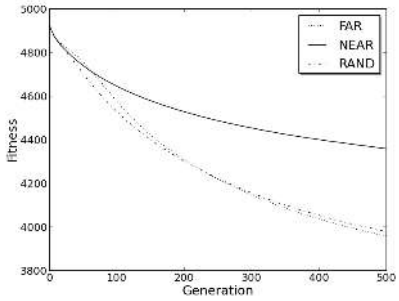
We tested on four popular instances with 1,000 vertices [KM04]. They have different ratios of edge size. The number of the right part of point in each instance name means the average degree of each vertex.

Figure 7 shows the fitness of the best individual according to generation. The plotted results are the average value over 1,000 runs as in TSP. The difference between graph bisection instances is smaller than that of TSP. As a reason, we supposed that the solution space sizes of graph bisection instances are the same. On the other hand, the solution spaces of TSP instances increase faster as m increases.

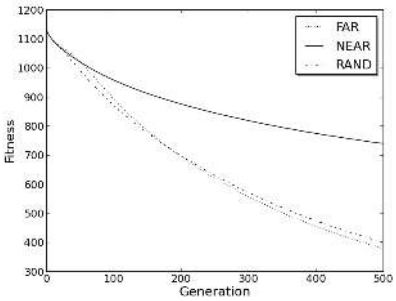
One of the common features in graph bisection is a goodness of FAR. In all graph bisection instances, FAR performed better than RAND and NEAR after the 300-th generation. NEAR performed worse than RAND and FAR in all instances. Figure 8 gives an explanation on the bad quality of NEAR. It



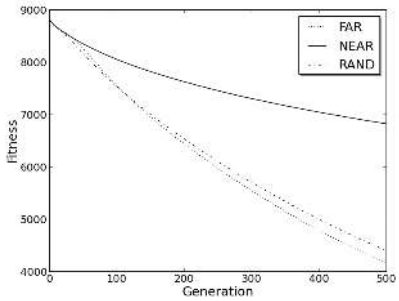
(a) G1000.2.5



(b) G1000.20



(c) U1000.05

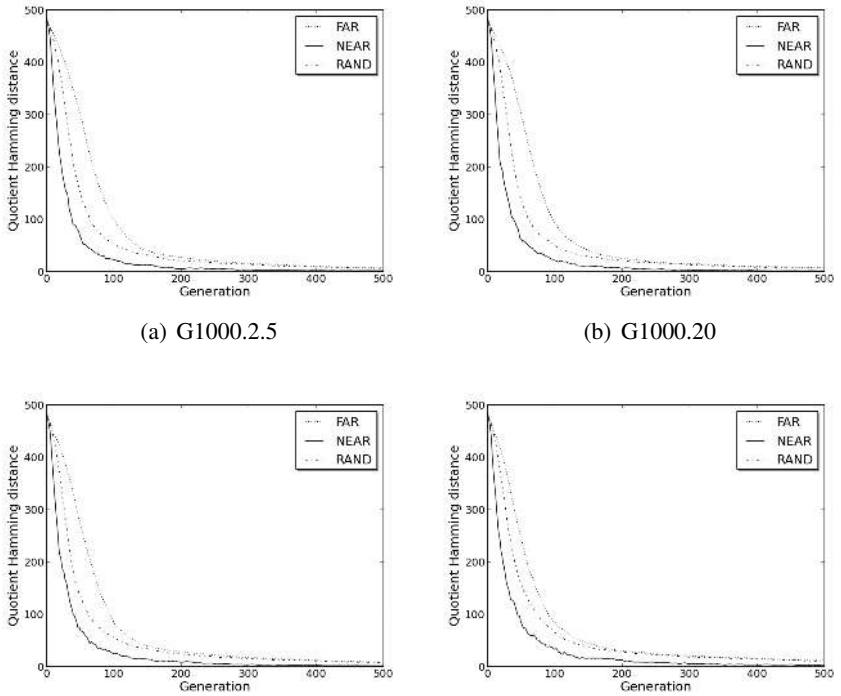


(d) U1000.40

Figure 7: Fitness of mating schemes in graph bisection (the smaller, the better)

describes the diversity of each instance. The diversity is measured by the average distance within a population. Every scheme suffers losing its diversity fast. But NEAR falls rapidly and almost zero near the 300-th generation.

Table 5 shows t-values and p-values of results in Table 4 with the same way as used in TSP. Different mating schemes made significant differences. The p-values are almost zero. The values of graph bisection problem shows larger difference than those of TSP.



(a) G1000.2.5 (b) G1000.20

(c) U1000.05 (d) U1000.40

Figure 8: Average population distance of mating schemes in graph bisection

Table 2: Results of TSP

Instance	Method	Generation m	Generation $2m$	Generation $3m$	Generation $4m$	Final($> 5m$)
		Ave(Std)	Ave(Std)	Ave(Std)	Ave(Std)	Ave(Std)
berlin52	FAR	1.77e4(7.64e2)	1.45e4(9.40e2)	1.23e4(9.97e2)	1.10e4(6.85e2)	**9.07e3(3.08e2)**
	NEAR	**1.66e4(5.92e2)**	1.36e4(5.31e2)	1.21e4(4.60e2)	1.13e4(4.37e2)	9.39e3(3.21e2)
	RAND	1.67e4(5.28e2)	**1.31e4(5.81e2)**	**1.15e4(3.85e2)**	**1.07e4(2.90e2)**	9.12e3(1.33e2)
kroA100	FAR	1.03e5(3.67e3)	8.76e4(5.82e3)	7.24e4(8.96e3)	5.88e4(7.72e3)	3.67e4(1.66e3)
	NEAR	**8.84e4(2.61e3)**	6.99e4(2.38e3)	5.96e4(2.18e3)	5.31e4(2.05e3)	3.97e4(1.41e3)
	RAND	9.36e4(3.29e3)	**6.88e4(3.45e3)**	**5.55e4(2.46e3)**	**4.85e4(1.82e3)**	**3.65e4(9.03e2)**
bier127	FAR	4.18e5(1.28e4)	3.65e5(1.72e4)	3.19e5(2.36e4)	2.74e5(2.42e4)	1.98e5(7.93e3)
	NEAR	**3.75e5(8.90e3)**	3.07e5(8.69e3)	2.70e5(7.91e3)	2.46e5(7.54e3)	2.03e5(6.35e3)
	RAND	3.89e5(3.97e3)	**3.06e5(8.29e3)**	**2.58e5(5.53e3)**	**2.31e5(4.03e3)**	**1.90e5(2.25e3)**
pr152	FAR	6.48e5(2.44e4)	5.42e5(3.33e4)	4.44e5(5.07e4)	3.43e5(5.07e4)	2.09e5(1.98e4)
	NEAR	**5.34e5(1.46e4)**	3.97e5(1.43e4)	3.22e5(1.36e4)	2.74e5(1.24e4)	2.11e5(9.62e3)
	RAND	5.72e5(1.76e4)	**3.97e5(2.01e4)**	**2.99e5(1.42e4)**	**2.45e5(1.16e4)**	**1.81e5(8.36e3)**

CPU: Intel(R) Xeon(R) E5530 2.40GHz. Average over 1,000 runs. m means the number of cities of each instance.
Ave: average (The smaller, the better.) / Std: standard deviation.

Table 3: Statistical test experiments of TSP

Instance	Method	Generation m	Generation $2m$	Generation $3m$	Generation $4m$	Final($> 5m$)
		t-value(p-value)	t-value(p-value)	t-value(p-value)	t-value(p-value)	t-value(p-value)
berlin52	FAR-NEAR	36.73(5.05e-188)	26.21(4.74e-116)	5.34(5.46e-8)	-13.45(2.38e-38)	-22.91(5.02e-94)
	FAR-RAND	35.36(1.20e-178)	39.36(1.00e-205)	23.77(1.11e-29)	10.30(5.05e-24)	-4.85(6.93e-07)
	NEAR-RAND	-3.36(3.96e-4)	19.28(5.43e-71)	32.55(2.10e-159)	35.47(2.3e-179)	24.63(2.10e-105)
kroA100	FAR-NEAR	118.25(0)	89.07(0)	45.39(1.84e-245)	22.56(9.70e-92)	-42.51(1.04e-226)
	FAR-RAND	61.12(0)	87.85(0)	59.22(0)	40.97(1.60e-216)	3.77(9.54e-5)
	NEAR-RAND	-45.64(4.6e-247)	8.13(6.14e-16)	38.97(3.8e-203)	52.84(5.2e-292)	59.54(0)
bier127	FAR-NEAR	87.97(0)	93.45(0)	61.92(0)	34.24(5.27e-171)	-16.46(2.23e-54)
	FAR-RAND	67.28(0)	97.34(0)	79.16(0)	54.96(9.06e-305)	29.58(5.7e-139)
	NEAR-RAND	-48.24(1.20e-263)	4.825e5(8.08e-7)	39.17(1.8e-204)	56.05(0)	61.03(0)
pr152	FAR-NEAR	126.11(0)	126.08(0)	73.48(0)	41.55(2.38e-220)	-2.54(5.50-e3)
	FAR-RAND	96.85(0)	117.97(0)	86.66(0)	59.38(0)	40.33(2.80-e212)
	NEAR-RAND	-82.12(0)	0.78(0.21e0)	35.96(9.6e-183)	54.23(2.1e-300)	72.52(0)

t-value: the bigger, the larger difference. / p-value: the smaller, the more significant.

Table 4: Results of graph bisection

Instance	Method	Generation 100	Generation 200	Generation 300	Generation 400	Final(=500)
		Ave(Std)	Ave(Std)	Ave(Std)	Ave(Std)	Ave(Std)
G1000.2.5	FAR	4.66e2(2.28e1)	3.73e2(1.49e1)	**3.19e2(1.17e1)**	**2.83e2(1.02e1)**	**2.57e2(9.27e0)**
	NEAR	4.90e2(8.13e0)	4.49e2(9.06e0)	4.23e2(9.38e0)	4.04e2(9.51e0)	3.90e2(1.02e1)
	RAND	**4.48e2(1.10e1)**	**3.71e2(9.72e0)**	3.22e2(9.20e0)	2.88e2(8.74e0)	2.64e2(8.73e0)
G1000.20	FAR	4.58e3(6.34e1)	4.31e3(4.31e1)	**4.15e3(3.53e1)**	**4.04e3(3.16e1)**	**3.96e3(3.00e1)**
	NEAR	4.65e3(2.43e1)	4.53e3(2.51e1)	4.45e3(2.70e1)	4.40e3(2.83e1)	4.36e3(2.91e1)
	RAND	**4.58e3(3.11e1)**	**4.30e3(2.78e1)**	4.16e3(2.72e1)	4.05e3(2.64e1)	3.98e3(2.68e1)
U1000.05	FAR	8.94e2(4.09e1)	**6.96e2(3.11e1)**	**5.58e2(2.77e1)**	**4.56e2(2.55e1)**	**3.80e2(2.41e1)**
	NEAR	9.59e2(1.63e1)	8.77e2(1.88e1)	8.20e2(2.05e1)	7.76e2(2.18e1)	7.41e2(2.37e1)
	RAND	**8.71e2(2.14e1)**	6.98e2(2.20e1)	5.72e2(2.22e1)	4.75e2(2.26e1)	4.02e2(2.25e1)
U1000.40	FAR	7.55e3(1.66e2)	**6.44e3(1.77e2)**	**5.55e3(2.06e2)**	**4.79e3(2.39e2)**	**4.16e3(2.64e2)**
	NEAR	8.06e3(5.86e1)	7.63e3(1.16e2)	7.31e3(1.34e2)	7.05e3(1.50e2)	6.83e3(1.63e2)
	RAND	**7.53e3(1.32e2)**	6.54e3(1.66e2)	5.71e3(2.00e2)	5.00e3(2.30e2)	4.40e3(2.55e2)

CPU: Intel(R) Xeon(R) E5530 2.40GHz. Average over 1,000 runs.
Ave: average (the smaller, the better.) / Std: standard deviation.

Table 5: Statistical test experiments of graph bisection

Instance	Method	Generation 100	Generation 200	Generation 300	Generation 400	Final(=500)
		t-value(p-value)	t-value(p-value)	t-value(p-value)	t-value(p-value)	t-value(p-value)
G1000.2.5	FAR-NEAR	-30.66(2.22e-146)	-137.34(0)	-220.07(0)	-276.16(0)	-304.39(0)
	FAR-RAND	22.20(1.98e-89)	4.55(2.95e-6)	-6.43(1.7e-10)	-12.70(1.12e-34)	-16.52(9.81e-55)
	NEAR-RAND	95.37(0)	186.22(0)	243.75(0)	284.27(0)	296.92(0)
G1000.20	FAR-NEAR	-32.57(1.68e-159)	-139.70(0)	-218.39(0)	-271.64(0)	-303.79(0)
	FAR-RAND	-3.01(1.32e-3)	4.08(2.37e-5)	-6.19(4.27e-10)	-12.63(2.26e-34)	-16.09(2.78e-52)
	NEAR-RAND	50.63(1.70e-278)	191.56(0)	246.32(0)	283.92(0)	304.11(0)
U1000.05	FAR-NEAR	-46.55(7.19e-253)	-157.4(0)	-239.6(0)	-301.7(0)	-337.2(0)
	FAR-RAND	15.41(1.39-e48)	-1.85(3.24-e3)	-12.25(1.50e-32)	-18.12(5.38e-64)	-20.60(3.41e-79)
	NEAR-RAND	102.51(0)	195.46(0)	258.93(0)	302.67(0)	328.59(0)
U1000.40	FAR-NEAR	-90.71(0)	-177.2(0)	-226.9(0)	-253.14(0)	-271.88(0)
	FAR-RAND	2.58(4.94-e3)	-12.19(2.64e-32)	-18.00(2.74-e63)	-20.22(8.27-e77)	-20.68(1.03-e79)
	NEAR-RAND	114.62(0)	170.91(0)	210.28(0)	235.98(0)	253.09(0)

t-value: the bigger, the larger difference. / p-value: the smaller, the more significant.

Chapter 4

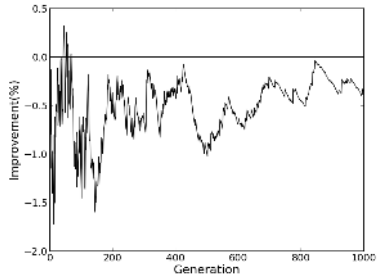
Hybrid and Adaptive Scheme

4.1 Simple Hybrid Scheme

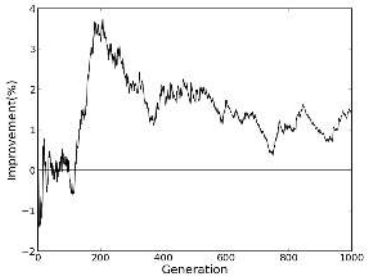
In our observation with RAND, NEAR, and FAR, we can check characteristics of each mating scheme. FAR is good with sufficient time resource. But it would not be a good choice with short time budget in TSP. NEAR decreases diversity relatively fast and shows premature convergence. But with a short period, it can be a good strategy. With this observation, we combined generation-best strategies in each problem. In TSP, we made a scheme changing from NEAR to RAND scheme. In the graph bisection problem, we made a scheme changing from RAND to FAR scheme. But, we decided the switching generation to be a half of the crossed generation[1] in our observation. Because we thought that the actual point of performance reversing would be before the crossing point. In graph bisection, our new mating scheme changes in the 100-th generation because they crosses near the 200-th generation. In TSP, the crossing point is depend on the number of cities m. We decided mating scheme to change at the m-th generation. Because Figure 5 shows crossing lines near the $2m$-th generation.

Figures 9 and 10 show comparison with the best scheme given in Sec-

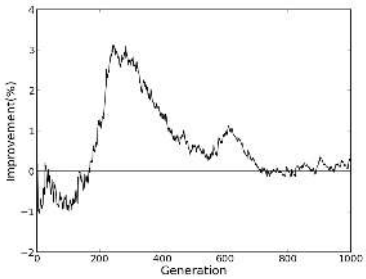
[1] It is in Figures 5 and 7



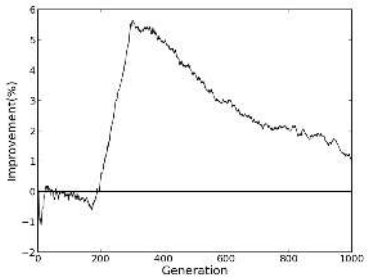
(a) berlin52



(b) kroA100



(c) bier127



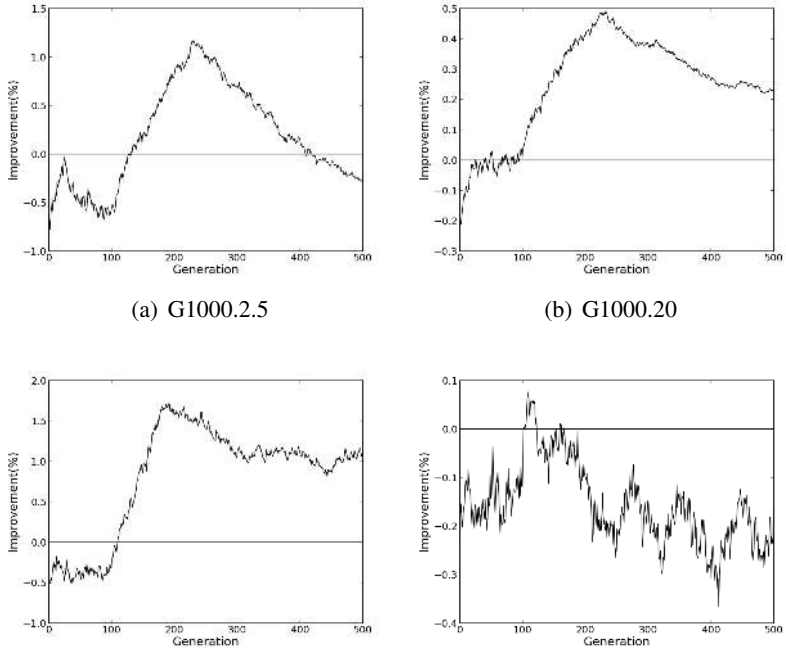
(d) pr152

Figure 9: Improvement over the best single scheme by generation in TSP

tion 3.3. Improvement is measured as follows:

$$\text{Improvement}\,(\%) = 100 \times \frac{f_{best} - f_{new}}{f_{best}},$$

where $f_{best} = \min(f_{\text{RAND}}, f_{\text{NEAR}}, f_{\text{FAR}})$ and f means fitness. In five among eight instances, our new scheme scored better at the end of running. After about 20% of generations, our new scheme performed better than the single best strategy in most generations. Improvement is notable at near our switching generations (m or 100). NEAR focuses on exploitation than FAR and RAND. FAR concentrates on exploration more than NEAR and RAND.



(a) G1000.2.5 (b) G1000.20

(c) U1000.05 (d) U1000.40

Figure 10: Improvement over the best single scheme by generation in graph bisection problem

Our hybrid mating schemes are balanced in the view of exploration and exploitation, and we guess that they create a synergy effect.

4.2 Adaptive Scheme

4.2.1 Significance of Adaptive Scheme

In Chapter 3, we reported that the best Hungarian mating scheme varies according to problems and their sizes. In the previous section, we predetermined the switching time before running their GAs. So it is hard to apply

the hybrid method to new problems or instances.

Galán et al. [GMP13] reported that a self-adaptive mating scheme can be better than traditional random mating, and their best-first and best-last mating. In the best-first mating, each solution pairs up with its nearest one in the order from the best solution to the worst one. In the contrast, in the best-last mating, each solution pairs up with their farthest one in the order from the best solution to the worst one. The best-first mating resembles NEAR method as the best-last mating resembles FAR method. NEAR and FAR are extreme cases of mating. The ideal mating scheme may exist in some middle point of NEAR and FAR as Galán et al. [GMP13] showed in function optimization.

Our new goal is to design a new adaptive method of the mating schemes. We want that: i) our new scheme works in various instances or problems, ii) it is adaptive, and iii) it outperforms any non-adaptive mating scheme. We will propose a new scheme satisfying these characteristics.

4.2.2 Proposed Method

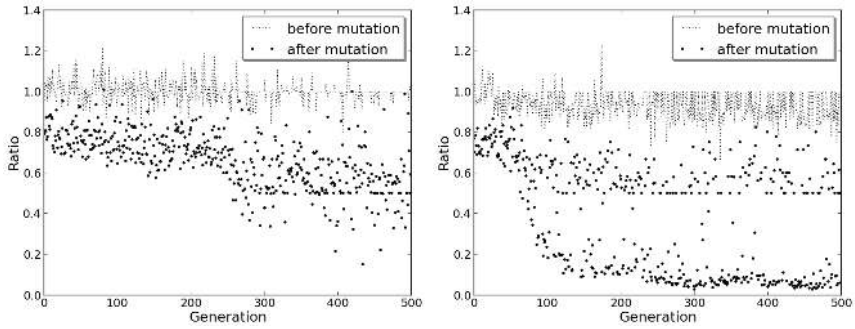
4.2.2.1 Voting Rules

We assume the same number of male and female solutions as Goh et al. did in [GLR03]. In each generation, our method selects FAR, RAND, or NEAR for the next generation. Our method does not simulate three schemes as they are. Instead, the appropriate scheme is adaptively adopted. For that, a Hungarian mating scheme for the next generation is selected with voting.

Algorithm 1 Voting rules

```
// input: two parents and two offspring
// output: FAR, NEAR, or RAND
// d(x,y): distance function between x and y
Function vote(p1, p2, o1, o2)
{
  if d(p1, p2) = 0, d(p1, o1) = 0, or d(o2, p2) = 0 then
    return FAR;
  end if
  ratio ← d(p1, p2) / (d(p1, o1) + d(o2, p2));
  if ratio < α then
    return FAR;
  end if
  if α ≤ ratio < β then
    return RAND;
  end if
  if ratio ≥ β then
    return NEAR;
  end if
}
```

Every pair of individuals has the right to vote. Our crossover operator generates two offspring, and their gender is randomly assigned. The voting is carried out after mutation. So our voting algorithm compares two parents and two offspring after mutation. Algorithm 1 describes the rules of voting. If one of the offspring is the same as one of the parents, this family votes to FAR. Otherwise, a ratio of distance between parents over the sum of the father-son distance and the mother-daughter distance is considered. If the ratio value is below α, this family votes to FAR. If the ratio is not less than α and below β, this family votes to RAND. The last case, in which the ratio is not less than β, this family votes to NEAR. After voting, the strategy which gets the most votes is set for the next generation.



(a) pr152 (b) G1000.2.5

Figure 11: Median of ratio values according to generation

4.2.2.2 Parameter Setting

We set α and β as 0.5 and 1, respectively. Figure 11 shows the median of the ratio values according to generation, for an instance of each test problem. The X-axis represents generation and the Y-axis represents our ratio values.

We call the median of the ratio values after crossover (before mutation) (thin line) BM. The median of our ratio values after mutation (thick points) is called AM. After crossover (before mutation), most of BM values are close to 1. BM does not change much while the diversity of population decreases. On the other hand, AM drops slowly as the diversity decreases.

A mutation operator moves an individual to nearby space. The distribution of moving distance by a mutation is independent of the distance between parents. The expected value of BM is one when we use a geometric crossover [MP04]. It will be proved in the next subsection. AM values over 1 appear frequently when the distance between parents is long enough.

It means that we have sufficient diversity to consume. So a family votes to NEAR. Besides, the lower bound of BM is 0.5 when we use a geometric crossover. It will also be proved in the next subsection. AM values below 0.5 appear by mutation effect. It is shown when the distance between parents is very close to 0. So a family votes to FAR. In other words, an influence of the mutation is estimated by the distance between parents. High influence of the mutation, or a low AM value means that the matched parents are too close to each other to produce new solutions while low influence of the mutation, or a high AM value means that the parents are far from each other so we can match nearer solutions.

4.2.3 Theoretical Support

A binary crossover operator is geometric if all offspring are in a convex segment between parents. So $d(p_1,p_2) = d(p_1,o) + d(o,p_2)$, where $d(p_1,p_2)$ is a distance between p_1 and p_2, p_is are parents, and o is an offspring. Let D be the distance between both parents. We assume that $D = d(p_1,p_2) \neq 0$, crossover is geometric [MP04], $p_1 \neq p_2$, $p_1 \neq o_1$, and $p_2 \neq o_2$. We remind that our ratio value is defined as

$$\frac{d(p_1,p_2)}{d(p_1,o_1)+d(o_2,p_2)},$$

where o_1 and o_2 are offspring obtained from a geometric crossover between p_1 and p_2.

Proposition 1. Under these assumptions, the expected value of our ra-

tio is 1. That is,

$$E\left[\frac{d(p_1,p_2)}{d(p_1,o_1)+d(o_2,p_2)}\right]=1.$$

Proof. It is enough to show that

$$E[d(p_1,o_1)+d(o_2,p_2)]=D.$$

$$\begin{aligned}
&E[d(p_1,o_1)+d(o_2,p_2)]\\
&=E[d(p_1,o_1)]+E[d(o_2,p_2)] \quad (\because E[\cdot] \text{ is linear})\\
&=E[d(p_1,o_1)]+E[d(p_1,p_2)-d(p_1,o_2)]\\
&(\because \text{Crossover is geometric})\\
&=E[d(p_1,o_1)]+E[D-d(p_1,o_2)]\\
&=E[d(p_1,o_1)]+D-E[d(p_1,o_2)] \quad (\because E[\cdot] \text{ is linear})\\
&=D \quad (\because E[d(p_1,o_1)]=E[d(p_1,o_2)])
\end{aligned}$$

□

Proposition 2. Under the same assumptions, the lower bound of our ratio value is 0.5. That is,

$$\frac{d(p_1,p_2)}{d(p_1,o_1)+d(o_2,p_2)}\geq\frac{1}{2}.$$

Proof. By the assumption of geometric crossover,

$$d(p_1,p_2)\geq d(p_1,o_1) \quad \text{and} \quad d(p_1,p_2)\geq d(o_2,p_2).$$

By summing the above inequalities, $2d(p_1,p_2)\geq d(p_1,o_1)+d(o_2,p_2)$. Hence,

we obtain

$$\frac{d(p_1, p_2)}{d(p_1, o_1) + d(o_2, p_2)} \geq \frac{1}{2}.$$

□

4.2.4 Experiments

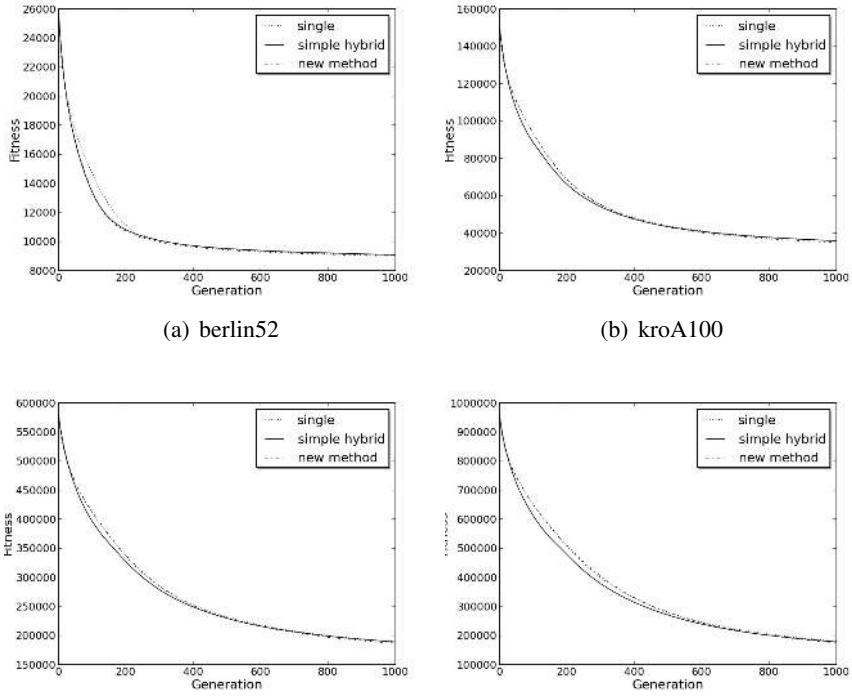
4.2.4.1 Tested GA

We use a generational GA. All male individuals are one-to-one matched with all female ones. A pair of individuals produces two offspring. One is male offspring, and the other is female one. The genders are assigned randomly. We used 50 male individuals and 50 female ones. As a replacement strategy, we used elitism [DJ75] in both genders. Among new 50 offspring and previous 50 individuals, we chose 50 best ones for the next generation. The other genetic parameters are same as in Section 3.2.

4.2.5 Traveling Salesman Problem

We selected four Euclidean instances from TSPLib [Rei91] as in chapter 3. They are berlin52, kroA100, bier127, and pr152. Each number in the instance name means the number of cities in the instance.

Figure 12 shows the fitness of the best individual according to generation. The plotted results are the average value over 1,000 runs. Table 6 gives the average best fitness and the standard deviation per 200 generations. ‘single best’ denotes the best single Hungarian mating scheme among three schemes (FAR, RAND, and NEAR). ‘simple hybrid’ is the strategy introduced in Section 4.1.



(a) berlin52 (b) kroA100

(c) bier127 (d) pr152

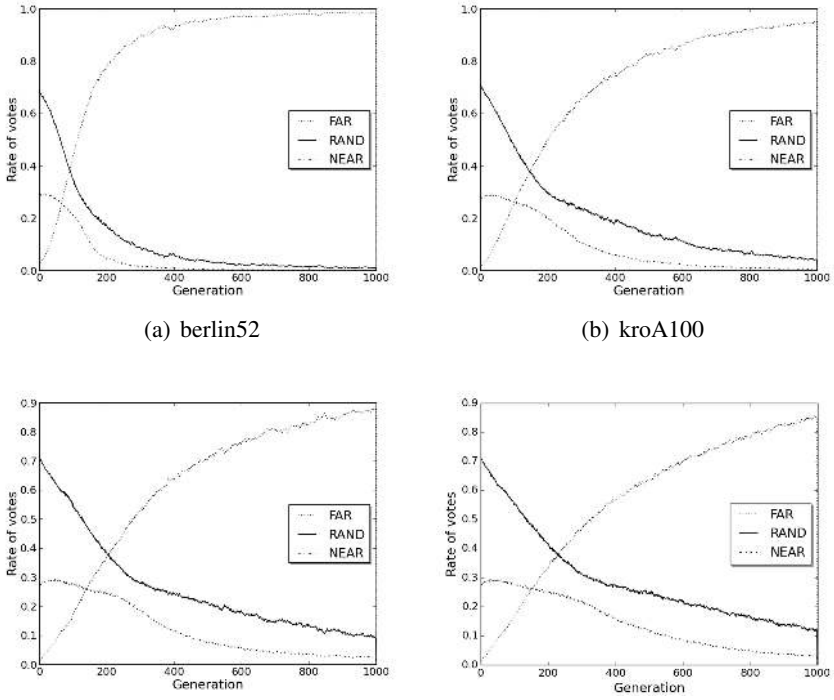
Figure 12: Fitness of mating schemes in TSP (the smaller, the better)

In early stages of each run, the simple hybrid method showed the best fitness. But at the end of each run, our method outperformed the others in all instances.

Table 7 shows the significance of results of Table 6 statistically. We used Welch's t-test [Wel47]. The t-value of $A - B$ in Table 7 is computed as follows:

$$t = \frac{\overline{X_A} - \overline{X_B}}{\sqrt{S_A^2/n_A + S_B^2/n_B}},$$

where $\overline{X_A}$ is the average of A, S_A is the standard deviation of A, and n_A is



(a) berlin52 (b) kroA100

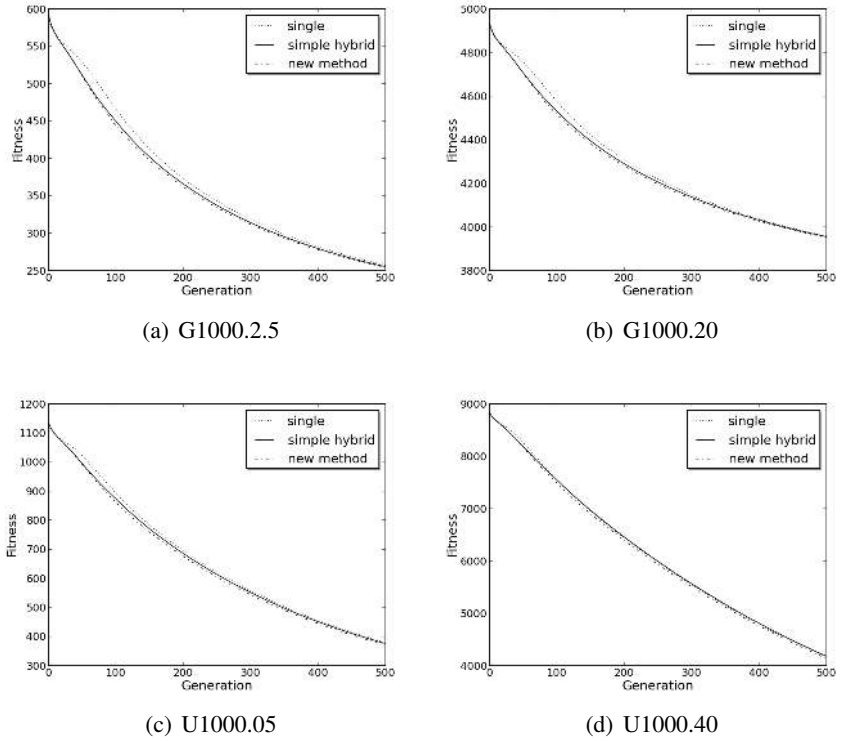
(c) bier127 (d) pr152

Figure 13: Voting rates of schemes in TSP

the number of A. The lower p-value means the more significant result. In most cases, p-values are very close to zero. A plus mark (+) means that it has passed t-test under significance level 0.01. Our method are significantly superior to the others.

Figure 13 shows the average voting rate of three schemes. In the early stage, RAND and NEAR gets higher chance to be elected. The graph shows the average over 1,000 runs. So NEAR is rarely selected in the early stage. As the diversity decreases, supporters of FAR increase. At the end of each run, almost all families vote to FAR. When we compare four instances in

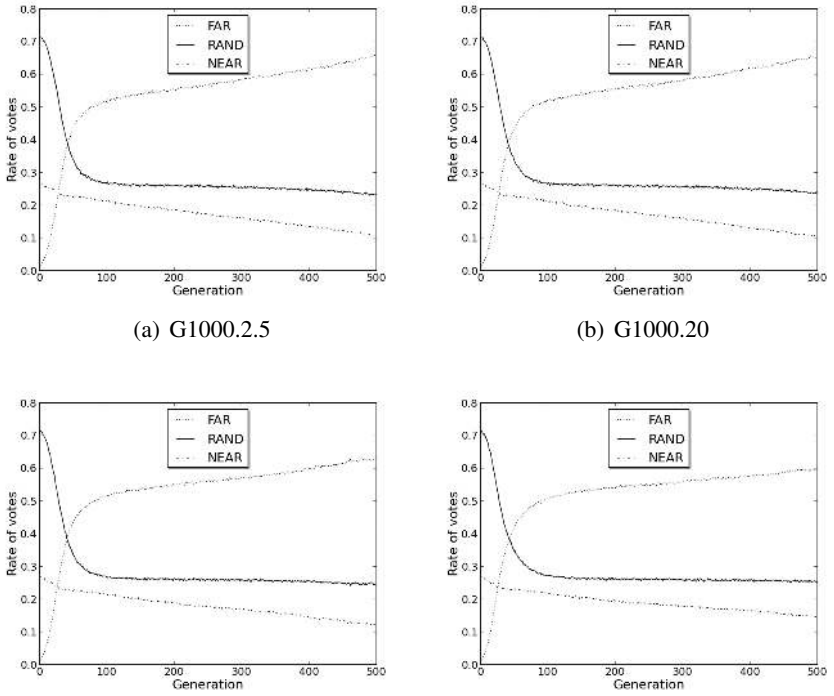
Figure 13, we could observe that our method is adaptive. Consuming a diversity in a small space was faster than that in a large space. So our algorithm changes the mating scheme from RAND (or rarely NEAR) to FAR. The speed of changing scheme in instance pr152 was slower than that in instance berlin52.



(a) G1000.2.5 (b) G1000.20

(c) U1000.05 (d) U1000.40

Figure 14: Fitness of mating schemes in graph bisection (the smaller, the better)



(a) G1000.2.5 (b) G1000.20

(c) U1000.05 (d) U1000.40

Figure 15: Voting rates of schemes in graph bisection

4.2.6 Graph Bisection Problem

We tested on four popular instances with 1,000 vertices [KM04]. They have different edge densities. The number of the right part of each name means the average vertex degree.

Figure 14 shows the fitness of the best individual over generation. The plotted results are the average over 1,000 runs as in TSP. Table 8 gives the average best fitness and the standard deviation per 100 generations. 'single best' and 'simple hybrid' are the same as in TSP. In almost all generations of all the instances, our method outperformed the others. Table 9 shows *t*-

test for the results in Table 8 with the same way as used in TSP. Our scheme significantly outperformed the others except one instance.

Figure 15 shows the average voting rate of three schemes. While TSP showed different speeds of changing schemes for each instance, the figures of four graph bisection instances are almost the same. The size of problem space may one of the most important factors to control population. Four instances of graph bisection have the same size of problem space. We reported that NEAR showed very poor results in this problem in Chapter 3. With this method, NEAR is almost abandoned because FAR increases very fast.

4.2.7 Comparison with Traditional Method

We compared our method with traditional roulette-wheel selection [Gol89]. The roulette-wheel selection provides higher chance to better individual. The probability to select an individual is defined as follows:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}.$$

In the maximization problem, f_i can be the fitness value of each individual. In the minimization problem, a proper f_i value should be calculated.

In our experiment, the f_i value is calculated as following function: $f_i = exp(-z_i/z_{max})$, where z_i is value of a cost function of an individual i and z_{max} is the maximum value of a cost function in the all individuals. We implemented a variant of traditional roulette-wheel selection. The variant select a male solution and a female solution using roulette-wheel selection. It is repeated until all solutions are one-to-one matched.

Table 10 shows the solution qualities and t-test results. Our method shows better performance in all instances. And our method was significantly better in six instances.

4.2.8 Comparison with Distance-based Methods

We compared our method with existing distance-based mating ones. We implemented variants of Ishibuchi et al.'s [IS03] and Galán et al.'s [GMP13] methods with two same-sized genders. Ishibuchi et al.'s method [IS03] selects one parent that is the farthest individual from the average among the results of repeated tournament selections of α times. Their method selects the other parent that is the nearest individual from the first parent among the results of repeated tournament selections of β times. We set α and β to be 9 as in [IS03]. The transformed variant selects the first parent from the female solutions, and selects the second parent from the male solutions. It is repeated until all solutions are one-to-one matched. Galán et al.'s method [GMP13] selects one parent that is the best. As the other parent, their method selects the $(\gamma-1)$-th nearest individual, where γ is the mating preference of the first parent. The mating preference is inherited in crossover, and it increases by 1 with probability 0.25 or decreases by 1 with probability 0.25, in mutation. The same as the variant of Ishibuchi et al.'s [IS03], we made this method select the first parent from the female solutions and the second parent from the male solutions. It is repeated until all solutions are one-to-one matched. All the conditions and settings excluding mating are the same as those in the experiments of the previous sections.

Table 11 compares the solution qualities of these two existing methods

and ours. For all instances of two test problems, our method significantly outperformed the others. Table 12 compares the computation time with respect to mating. Each value in Table 12 except mating proportion is measured in seconds. Our method took more time than Galán's method. But our method was faster than Ishibuchi's. Galán's method repeats finding the $(\gamma-1)$-th nearest individual whereas our method maximizes(or minimizes) the sum of distances. For graph bisection problem, computation times of instances of our method are similar to each other because the instances have the same number of nodes. In TSP, as the solution space grows, the proportion of mating time decreases, because the mating time of our method is mainly bounded by population size. As distance scale grows, mating time increases. It can be resolved by approximating the scale of distance values. Our mating method did not overburden the entire GA, and we also expect to reduce time burden through some improved implementation.

Table 6: Results of TSP

Instance	Method	Generation 200	Generation 400	Generation 600	Generation 800	Final(=1000)
		Ave(SD)	Ave(SD)	Ave(SD)	Ave(SD)	Ave(SD)
berlin52	single best	1.11e4(5.59e2)	9.71e3(4.23e2)	9.34e3(3.26e2)	9.71e3(2.85e2)	9.07e3(3.08e2)
	simple hybrid	1.08e4(4.34e2)	9.72e3(3.75e2)	9.39e3(3.48e2)	9.22e3(3.37e2)	9.10e3(3.18e2)
	new method	**1.07e4(4.68e2)**	**9.63e3(3.69e2)**	**9.30e4(3.45e2)**	**9.13e3(3.23e2)**	**9.02e3(3.09e2)**
kroA100	single best	6.87e4(4.10e3)	4.84e4(2.50e3)	4.13e4(1.71e3)	3.85e4(1.80e3)	3.65e4(9.03e2)
	simple hybrid	**6.62e4(3.20e3)**	**4.76e4(2.22e3)**	4.10e4(1.74e3)	3.78e4(1.58e3)	3.60e4(1.41e3)
	new method	6.78e4(4.09e3)	4.79e4(2.40e3)	**4.04e4(1.76e3)**	**3.71e4(1.56e3)**	**3.51e4(1.48e3)**
bier127	single best	3.36e5(9.09e3)	2.52e5(9.44e3)	2.18e5(7.03e3)	2.00e5(6.76e3)	1.90e5(2.25e3)
	simple hybrid	**3.27e5(1.09e4)**	**2.49e5(9.06e3)**	**2.16e5(7.22e3)**	1.99e5(6.70e3)	1.89e5(6.53e3)
	new method	3.38e5(1.38e4)	2.52e5(9.79e3)	2.57e5(7.83e3)	**1.97e5(6.80e3)**	**1.87e5(6.20e3)**
pr152	single best	5.07e5(1.66e4)	3.28e5(3.90e4)	2.47e5(1.36e4)	2.05e5(1.13e4)	1.81e5(8.36e3)
	simple hybrid	**4.78e5(1.75e4)**	**3.14e5(1.59e4)**	**2.40e5(1.32e4)**	2.01e5(1.12e4)	1.79e5(9.39e3)
	new method	5.09e5(2.65e4)	3.30e5(2.21e4)	2.45e5(1.60e4)	**2.01e5(1.23e4)**	**1.76e5(1.00e4)**

CPU: Intel Xeon E5530 2.40GHz. Average from 1,000 runs.
Ave: average (the smaller, the better) / SD: standard deviation.

Table 7: Statistical test of TSP

Instance	Compared method	t-test	t-value	p-value
berlin52	single best	+	3.01	1.33e-03
	simple hybrid	+	5.75	5.71e-09
kroA100	single best	+	24.38	1.0e-103
	simple hybrid	+	14.00	4.01e-42
bier127	single best	+	14.24	2.41e-42
	simple hybrid	+	8.65	9.72e-18
pr152	single best	+	10.79	4.46e-26
	simple hybrid	+	6.31	2.01e-10

t-value: the bigger, the larger difference.
p-value: the smaller, the more significant.
+: significantly better under level 0.01.

Table 8: Results of graph bisection

Instance	Method	Generation 100	Generation 200	Generation 300	Generation 400	Final(=500)
		Ave(SD)	Ave(SD)	Ave(SD)	Ave(SD)	Ave(SD)
G1000.2.5	single best	4.48e2(1.10e1)	3.71e2(9.72e0)	3.19e2(1.17e1)	2.83e2(1.02e1)	2.57e2(9.27e0)
	simple hybrid	4.50e2(1.17e1)	3.66e2(9.36e0)	3.14e2(9.03e0)	2.80e2(8.82e0)	2.55e2(1.02e1)
	new method	**4.44e2(1.16e1)**	**3.62e2(9.55e0)**	**3.11e2(9.18e0)**	**2.78e2(8.77e0)**	**2.54e2(8.68e0)**
G1000.20	single best	4.58e3(3.11e1)	4.30e3(2.78e1)	4.15e3(2.70e1)	4.04e3(3.16e1)	3.96e3(3.00e1)
	simple hybrid	4.53e3(3.11e1)	4.29e3(2.83e1)	4.13e3(2.78e1)	4.03e3(2.72e1)	3.96e3(2.71e1)
	new method	**4.51e3(3.43e1)**	**4.28e3(2.90e1)**	**4.13e3(2.72e1)**	**4.02e3(2.64e1)**	**3.95e3(2.61e1)**
U1000.05	single best	8.71e2(2.14e1)	6.96e2(3.11e1)	5.58e2(2.77e1)	4.56e2(2.55e1)	3.80e2(2.41e1)
	simple hybrid	8.75e2(2.20e1)	6.86e2(2.25e1)	5.52e2(2.26e1)	4.51e2(2.26e1)	3.76e2(2.28e1)
	new method	**8.64e2(2.26e1)**	**6.76e2(2.24e1)**	**5.44e2(2.23e1)**	**4.45e2(2.24e1)**	**3.71e2(2.20e1)**
U1000.40	single best	7.53e3(1.32e2)	6.44e3(1.77e2)	5.55e3(2.06e2)	4.79e3(2.39e2)	4.16e3(2.64e2)
	simple hybrid	7.55e3(1.32e1)	6.46e3(1.77e2)	5.57e3(2.14e2)	4.81e3(2.48e2)	4.18e3(2.74e2)
	new method	**7.48e3(1.34e2)**	**6.40e3(1.27e2)**	**5.50e3(2.03e2)**	**4.76e3(2.35e2)**	**4.14e3(2.65e2)**

CPU: Intel Xeon E5530 2.40GHz. Average from 1,000 runs.
Ave: average (the smaller, the better) / SD: standard deviation.

Table 9: Statistical test of graph bisection

Instance	Compared method	t-test	t-value	p-value
G1000.2.5	single best	+	6.15	5.66e-10
	simple hybrid	+	2.82	2.44e-03
G1000.20	single best	+	5.48	2.69e-08
	simple hybrid	+	2.55	5.30e-03
U1000.05	single best	+	8.25	2.42e-16
	simple hybrid	+	5.18	1.32e-07
U1000.40	single best	$\sim$	1.54	6.23e-02
	simple hybrid	+	3.47	2.62e-04

t-value: the bigger, the larger difference.
p-value: the smaller, the more significant.
+: significantly better under level 0.01.
$\sim$: not significantly different under level 0.01.

Table 10: Comparison of results on two test problems

Problem instance	Our method		Roulette-wheel selection			
	Avg	Std	Avg	Std	t-test	p-value
berlin52	9.02e3	3.09e2	9.04e3	3.13e2	$\sim$	4.3e-01
kroA100	3.51e4	1.48e3	3.58e4	1.52e3	+	2.5e-05
bier127	1.87e5	6.20e3	1.87e5	7.07e3	$\sim$	3.9e-01
pr152	1.76e5	1.00e4	1.79e5	9.53e3	+	2.5e-03
G1000.2.5	2.54e2	8.68e0	2.64e2	8.84e1	+	2.2e-25
G1000.20	3.95e3	2.61e1	3.98e3	2.35e1	+	8.3e-30
U1000.05	3.71e2	2.20e1	4.04e2	2.17e1	+	3.7e-43
U1000.40	4.14e3	2.65e2	4.53e3	2.56e2	+	1.6e-43

CPU: Intel Xeon E5530 2.40GHz.
Avg: average (the smaller, the better) / Std: standard deviation.
p-value: the smaller, the more significant.
+: significantly better under level 1.00e-02.
$\sim$: not significantly different under level 0.01.

Table 11: Comparison of results on two test problems

Problem instance	Our method		Galán et al. [GMP13]				Ishibuchi et al. [IS03]			
	Avg	Std	Avg	Std	t-test	p-value	Avg	Std	t-test	p-value
berlin52	9.02e3	3.09e2	9.27e3	3.32e2	+	1.6e-59	9.24e3	3.38e2	+	4.4e-45
kroA100	3.51e4	1.48e3	3.81e4	1.60e3	+	3.5e-228	3.77e4	1.41e3	+	9.2e-213
bier127	1.87e5	6.20e3	1.96e5	6.78e3	+	1.2e-158	1.97e5	6.46e3	+	4.4e-188
pr152	1.76e5	1.00e4	1.96e5	1.12e4	+	1.6e-227	2.00e5	1.07e4	+	3.5e-290
G1000.2.5	2.54e2	8.68e0	3.01e2	3.13e1	+	5.2e-248	3.04e2	1.15e1	+	0*
G1000.20	3.95e3	2.61e1	4.09e3	9.47e1	+	1.0e-245	4.09e3	3.21e1	+	0*
U1000.05	3.71e2	2.20e1	4.96e2	8.40e1	+	9.7e-246	5.18e2	2.64e1	+	0*
U1000.40	4.14e3	2.65e2	4.99e3	5.94e2	+	5.9e-220	5.31e3	2.23e2	+	0*

CPU: Intel Xeon E5530 2.40GHz. Average from 1,000 runs.
Avg: average (the smaller, the better) / Std: standard deviation.
p-value: the smaller, the more significant.
+: significantly better under level 1.00e-02.
*: it means that this value is less than 1.0e-300.

Table 12: Results of computation time

Problem instance	Our method			Galán et al. [GMP13]			Ishibuchi et al. [IS03]		
	Mating	Total	Proportion of mating(%)	Mating	Total	Proportion of mating(%)	Mating	Total	Proportion of mating(%)
berlin52	7.68	81.49	9.4	0.63	70.31	0.9	68.31	140.07	48.5
kroA100	14.60	283.97	5.1	0.68	252.33	0.3	244.85	505.28	48.4
bier127	20.04	451.89	4.4	0.62	405.57	0.2	388.40	796.24	48.7
pr152	25.24	634.28	3.9	0.65	570.15	0.1	546.99	1119.84	48.8
G1000.2.5	8.26	54.81	15.0	0.60	45.94	1.3	9.06	55.74	16.2
G1000.20	7.76	61.91	12.5	0.56	52.74	1.1	9.10	63.39	14.3
U1000.05	8.98	56.45	15.9	0.57	47.02	1.2	8.86	55.22	16.0
U1000.40	8.15	66.63	12.2	0.50	57.95	1.0	9.02	68.41	13.1

Average CPU seconds from 1,000 runs on Intel Xeon E5530 2.40GHz.

Chapter 5

Tests in Various Environments

5.1 Hybrid GA

A hybrid GA, or a memetic genetic algorithm [HKS05] uses both genetic operators and a local optimization algorithm. In a hybrid GA environment, the GA parts relatively concentrate more on exploration. Otherwise, the local optimization parts are highly concentrated on exploitation [RN99]. Even if we use a simple local optimization, the hybrid GA may leans too much toward exploitation because a local optimization algorithm is powerful. Thus, a hybrid GA, especially local optimization with a steady-state GA, has a high chance to be premature converge. The GA operators in hybrid GA should focus exploration to obtain better solutions.

Local optimization algorithms consume almost(over 99.9%) of the running time in a hybrid GA. An effective implementation of local optimization needs to be determined.

5.1.1 Experiment Settings

We tested our adaptive mating scheme with hybrid GA. The FAR, RAND, and NEAR schemes are also tested for comparison. We used 2-opt algorithm as a local optimization algorithm. An individual after a mutation operator is the input of the 2-opt algorithm. The 2-opt algorithm swaps a

possible pair of genes and calculates the gain. The gain is defined in each problem specification. If the gain is more than zero, then the algorithm fixes swapped genes and finds another possible pair. The algorithm iterates these operations until no pair with a gain over zero remains to be swapped. The other operations and settings of the experiments are the same as in those in Chapter 4.

5.1.2 Results and Discussions

Table 13 compares the solution qualities of our adaptive method and the three Hungarian methods. Table 14 displays the t-test results. In a hybrid GA environment, local optimization is highly focused on exploitation. The NEAR method is an extreme case of exploitation. Thus, the solution quality of the NEAR method is worse than that of the other methods except berlin52. The FAR method has an advantageous position in the environment. The RAND method shows almost the same performance in TSP, but it shows statistically significant worse performance than the FAR method in the graph bisection problem. Our method shows slightly worse performance than the FAR method in some instances of TSP. However, the differences are not statistically significant at 0.01. Our method shows better performance in the graph bisection problem than in TSP. In the two instances of the graph bisection problem, our adaptive method shows a statistically significant better result than any other single Hungarian method.

Figure 16 shows the average voting rate of three schemes in TSP. The overall shapes of the voting rate are similar with those shown in Figure 13. As problem spaces grow, the speed of changing scheme is slower. However

the decreasing speed of voting to RAND and NEAR methods is faster than that in previous experiments because the local optimization consumes diversity rapidly. In this environment, the FAR method may be an optimal method in the Hungarian mating schemes. Our method selects the FAR method in the early stage of hybrid GA.

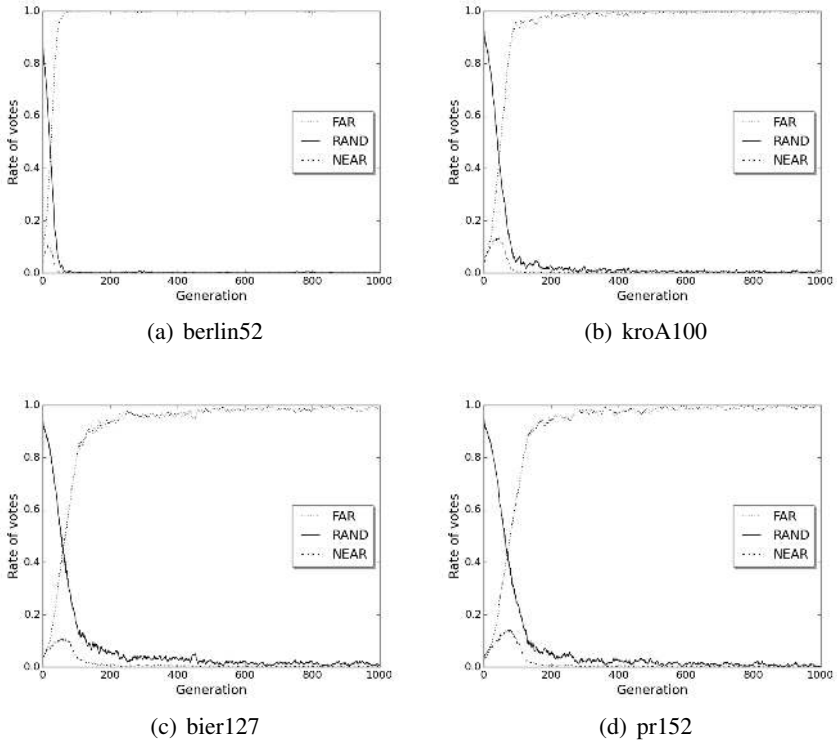
Figure 17 shows the average voting rate of the three schemes in graph bisection problem; the result is very similar to that shown in Figure 15. Compared with TSP, the local optimization method does not consume diversity rapidly possibly because of problem characteristics. The partial swap in TSP may directly cause to better solution qualities. However, in the graph bisection problem, the particular swap of two nodes affects the whole nodes and edges. Thus, it has difficulty in directly finding a better solution with the 2-opt local algorithm.

5.2 GA with New Individuals

The environments of the previous experiments do not produce new schema. A geometric crossover produces the offspring within the line segment. Exploration is left with the mutation operator. The FAR method may have an advantage in these environments. In this section, we tested our method in an environment with a periodic influx of new individuals.

5.2.1 Experiment Settings

We tested our adaptive mating scheme with a periodic influx of new individuals. For each 10 generations, 20% of the male and female solutions



(a) berlin52 (b) kroA100

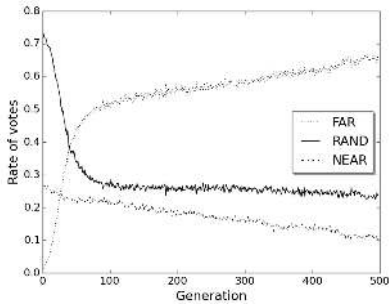
(c) bier127 (d) pr152

Figure 16: Voting rates of schemes in TSP with local optimization.

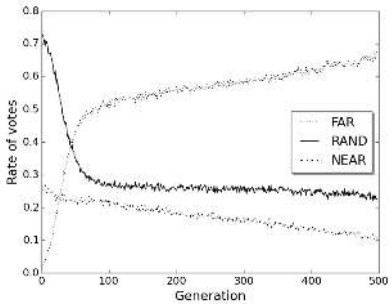
are dropped, and new random solutions replace them. The FAR, RAND, and NEAR schemes are also tested for comparison with our method in the same environment. The other operations and settings of experiments are the same as those in Chapter 4.

5.2.2 Results and Discussions

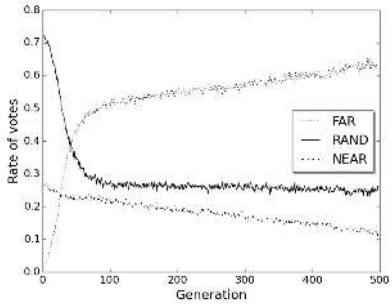
Table 15 compares the solution qualities of our adaptive method and the three Hungarian methods. In TSP, the FAR method shows the best performance among the single Hungarian methods in one instance and NEAR



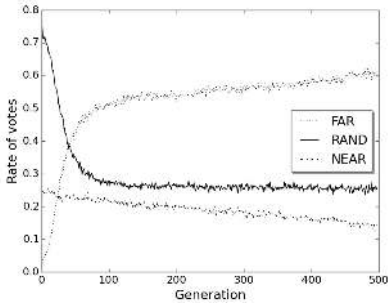
(a) G1000.2.5



(b) G1000.20



(c) U1000.05



(d) U1000.40

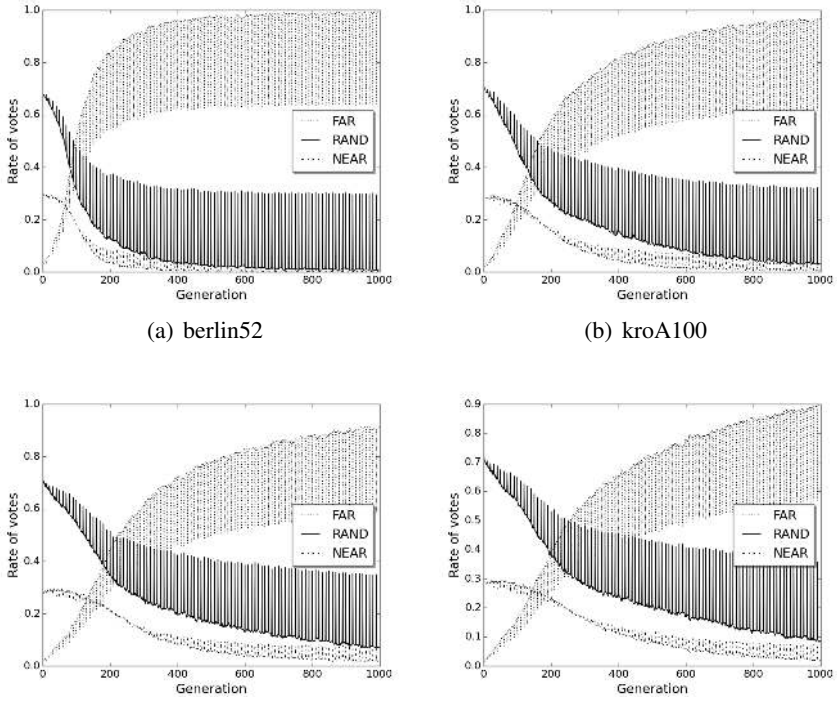
Figure 17: Voting rates of schemes in graph bisection with local optimization.

is the best in three instances. The performance of our adaptive method is statistically similar to that of the best single Hungarian method as shown in Table 16. In the graph bisection problem, our adaptive method shows statistically better performance than the three Hungarian methods.

Figure 18 shows the average voting rate of the three schemes in TSP. Figure 19 shows the average voting rate of the three schemes in graph bisection problem. The overall changes of voting rates are similar to those in previous experiments. However, in both figures show that, the voting rate to the RAND method increases periodically because of the periodic influx of

new individuals. Almost all the families of randomly generated new individuals vote with the RAND method. Our method shows the reaction of an unexpected event.

Our adaptive method reacts well in the two new environments. The method not only shows good performances in solution qualities but also changes into a proper mating method in each generation. Within the hybrid GA, local optimization highly concentrates on exploitation. Thus, our adaptive method selects the FAR method in the early stage. With an influx of new individuals, the method has a high chance to select the RAND method periodically.



(a) berlin52 (b) kroA100

(c) bier127 (d) pr152

Figure 18: Voting rates of schemes in TSP with an influx

Table 13: Comparison of results on two test problems with local-opt

Problem instance	Adaptive method		FAR method		RAND method		NEAR method	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
berlin52	7.54e3	1.81e1	7.54e3	0	7.55e3	2.57e1	7.55e3	2.26e1
kroA100	2.16e4	2.55e2	2.15e4	1.98e2	2.16e4	2.03e2	2.16e4	2.45e2
bier127	1.21e5	1.35e3	1.21e5	1.23e3	1.21e3	1.30e3	1.22e4	1.43e3
pr152	7.62e5	1.24e3	7.58e5	1.07e3	7.60e5	1.32e3	7.66e5	1.18e3
G1000.2.5	2.54e2	7.95e0	2.58e2	9.62e0	2.65e2	1.03e1	3.90e2	1.12e1
G1000.20	3.95e3	2.86e1	3.96e3	2.90e1	3.98e3	2.60e1	4.36e2	3.07e1
U1000.05	3.67e2	2.34e1	3.77e2	2.20e1	4.02e2	2.10e1	7.43e2	2.15e1
U1000.40	3.97e3	2.60e2	3.97e3	2.63e2	4.22e3	2.44e2	6.67e2	1.55e2

CPU: Intel Xeon E5530 2.40GHz. Average from 100 runs.
Avg: average (the smaller, the better) / Std: standard deviation.

Table 14: Statistical test results on two test problems with local-opt

Problem instance	vs. FAR method		vs. RAND method		vs. NEAR method	
	p-value	t-test	p-value	t-test	p-value	t-test
berlin52	8.54e-02	$\sim$	1.93e-01	$\sim$	2.61e-01	$\sim$
kroA100	2.71e-02	$\sim$	4.26e-01	$\sim$	2.48e-02	$\sim$
bier127	2.21e-01	$\sim$	4.58e-01	$\sim$	2.12e-03	+
pr152	1.63e-02	$\sim$	1.04e-01	$\sim$	2.79e-03	+
G1000.2.5	4.91e-04	+	4.57e-12	+	8.18e-102	+
G1000.20	3.13e-02	$\sim$	1.59e-12	+	7.95e-101	+
U1000.05	1.36e-03	+	1.94e-19	+	1.98e-109	+
U1000.40	4.33e-01	$\sim$	2.10e-12	+	2.44e-97	+

CPU: Intel Xeon E5530 2.40GHz. Average from 100 runs.
Avg: average (the smaller, the better) / Std: standard deviation.
+: the adaptive method is significantly better under level 0.01.
$\sim$: not significantly different under level 0.01.

Table 15: Comparison of results on two test problems with a periodic influx

Problem	Adaptive method		FAR method		RAND method		NEAR method	
instance	Avg	Std	Avg	Std	Avg	Std	Avg	Std
berlin52	9.01e3	3.42e2	8.98e3	3.21e2	9.06e3	3.07e2	9.36e3	3.22e2
kroA100	3.60e4	1.36e3	3.67e4	1.63e3	3.60e4	1.48e3	3.95e4	1.36e3
bier127	1.89e5	6.59e3	1.99e5	8.56e3	1.89e5	6.32e3	2.02e5	6.52e3
pr152	1.81e5	1.07e4	2.18e5	2.38e4	1.81e5	9.68e3	2.09e5	9.34e3
G1000.2.5	2.55e2	7.18e0	2.58e2	8.75e0	2.63e2	9.36e0	3.91e2	1.15e1
G1000.20	3.95e3	2.75e1	3.97e3	2.91e1	3.98e3	3.00e1	4.36e2	2.68e1
U1000.05	3.67e2	2.46e1	3.79e2	2.55e1	4.02e2	2.14e1	7.43e2	2.29e1
U1000.40	4.23e3	2.71e2	4.19e3	2.71e2	4.43e3	2.59e2	6.83e2	1.54e2

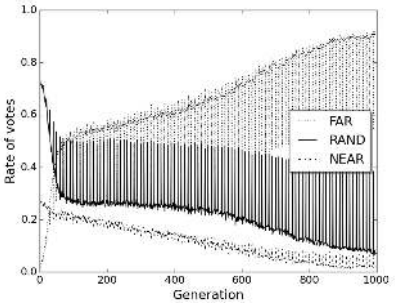
CPU: Intel Xeon E5530 2.40GHz. Average from 1,000 runs.

Avg: average (the smaller, the better) / Std: standard deviation.

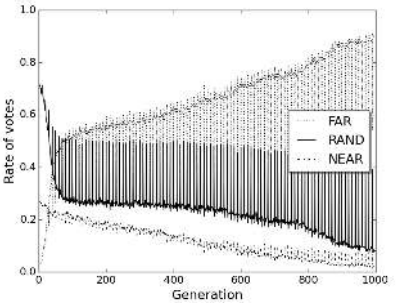
Table 16: Statistical test results on two test problems with local-opt

Problem instance	vs. FAR method		vs. RAND method		vs. NEAR method	
	p-value	t-test	p-value	t-test	p-value	t-test
berlin52	1.44e-02	$\sim$	1.12e-03	+	2.24e-104	+
kroA100	8.68e-27	+	2.93e-01	$\sim$	9.37e-208	+
bier127	2.50e-148	+	2.78e-01	$\sim$	3.59e-198	+
pr152	1.10e-241	+	2.17e-01	$\sim$	5.67e-77	+
G1000.2.5	8.11e-21	+	3.72e-91	+	0*	+
G1000.20	1.52e-22	+	4.89e-69	+	0*	+
U1000.05	4.93e-04	+	3.72e-122	+	0*	+
U1000.40	5.90e-04	+	6.82e-56	+	0*	+

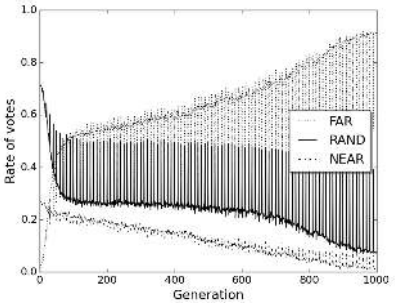
CPU: Intel Xeon E5530 2.40GHz. Average from 1,000 runs.
Avg: average (the smaller, the better) / Std: standard deviation.
+: the adaptive method is significantly better under level 0.01.
$\sim$: not significantly different under level 0.01.
*: it means that this value is less than 1.0e-300.



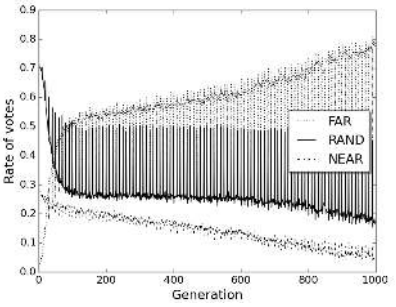
(a) G1000.2.5



(b) G1000.20



(c) U1000.05



(d) U1000.40

Figure 19: Voting rates of schemes in graph bisection with an influx

Chapter 6

A Revised Version of Adaptive Method

6.1 Hungarian Mating Scheme

Our adaptive scheme uses the Hungarian mating schemes to get a better solutions. Our adaptive scheme has advantages and disadvantages. The advantages are mentioned in previous chapters. The disadvantages are as follows: i) the Hungarian mating scheme consumes $O(n^3)$ time where n is population size ii) FAR and NEAR method is the extreme case of exploration and exploitation.

To overcome these disadvantages, we designed an approximated version of FAR and NEAR methods. The methods are named SEMI-FAR and SEMI-NEAR. The SEMI-FAR matches a female solution with the farthest male solution and the SEMI-NEAR matches a female solution with the nearest male solution. The unmatched male solutions are only considered to match. The new methods are faster than NEAR and FAR methods because the new methods does not consider the optimal sum.

6.2 Experiment Settings

Figure 2 shows the revised version of our adaptive mating scheme. The scheme uses SEMI-FAR method instead of FAR method. And the new

Algorithm 2 Voting rules

```
// input: two parents and two offspring
// output: SEMI-FAR, SEMI-NEAR, or RAND
// d(x,y): distance function between x and y
Function vote(p1, p2, o1, o2)
{
  if d(p1, p2) = 0, d(p1, o1) = 0, or d(o2, p2) = 0 then
    return SEMI-FAR;
  end if
  ratio ← d(p1, p2) / (d(p1, o1) + d(o2, p2));
  if ratio < α then
    return SEMI-FAR;
  end if
  if α ≤ ratio < β then
    return RAND;
  end if
  if ratio ≥ β then
    return SEMI-NEAR;
  end if
}
```

scheme uses SEMI-NEAR method instate of NEAR method. The remaining parts of the method is same with Algorithm 1. We compared this mating scheme with SEMI-FAR and SEMI-NEAR and our Hungarian adaptive mating scheme in Chapter 4.

6.3 Results and Discussions

Table 17 compares the solution qualities of our adaptive methods and the SEMI-FAR and SEMI-NEAR methods. Table 18 shows the results of t-test. In TSP, SEMI-FAR and SEMI-NEAR shows worse results in all of four instances. Our new adaptive method shows significantly better results than SEMI-NEAR and SEMI-FAR in all TSP instances. Our new adaptive

method shows better result in berlin52, but the difference is not statistically significant. In graph bisection problem, SEMI-FAR outperforms our old adaptive method. And our new adaptive method is statistically better than SEMI-FAR.

We reported that the best mating scheme is different with each problem. In TSP, the best scheme is the hybrid of NEAR and RAND. In graph bisection, the best scheme is RAND and FAR. SEMI-FAR shows good results in graph bisection problem. And our new adaptive scheme shows better results than SEMI-NEAR and SEMI-FAR in all instances of two problems.

Table 17: Comparison of results on two test problems

Problem instance	Hungarian adaptive method		SEMI-NEAR method		SEMI-FAR method		New adaptive method	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
berlin52	9.02e3	3.09e2	9.40e3	3.43e2	9.07e3	3.32e1	9.00e3	3.14e1
kroA100	3.51e4	1.48e3	3.93e4	1.68e2	3.65e4	1.70e2	3.54e4	1.68e2
bier127	1.87e5	6.20e3	1.98e5	7.17e3	1.97e3	7.67e3	1.90e4	5.88e3
pr152	1.76e5	1.00e4	2.06e5	9.45e3	2.05e5	1.52e3	1.85e5	1.25e3
G1000.2.5	2.54e2	8.68e0	3.80e2	1.04e1	2.50e2	7.33e0	2.47e2	8.02e0
G1000.20	3.95e3	2.61e1	4.33e3	3.51e1	3.91e3	2.73e1	3.92e2	2.87e1
U1000.05	3.71e2	2.20e1	7.23e2	2.66e1	3.53e2	2.54e1	3.48e2	2.37e1
U1000.40	4.14e3	2.65e2	6.68e3	1.71e2	3.94e3	2.66e2	3.93e2	2.73e2

CPU: Intel Xeon E5530 2.40GHz.
Avg: average (the smaller, the better) / Std: standard deviation.

Table 18: Statistical test results on two test problems

Problem instance	vs. SEMI-NEAR		vs. SEMI-FAR		vs. Hungarian adaptive method	
	p-value	t-test	p-value	t-test	p-value	t-test
berlin52	1.38e-119	+	3.10e-06	+	6.84e-02	$\sim$
kroA100	3.43e-284	+	2.71e-42	+	4.83e-06	-
bier127	1.84e-136	+	7.57e-98	+	9.81e-26	-
pr152	2.44e-228	+	4.49e-158	+	1.17e-75	-
G1000.2.5	0*	+	1.07e-19	+	2.00e-58	+
G1000.20	0*	+	1.37e-34	+	1.02e-73	+
U1000.05	0*	+	9.71e-07	+	3.81e-83	+
U1000.40	0*	+	1.92e-01	$\sim$	2.92e-79	+

CPU: Intel Xeon E5530 2.40GHz. Average from 100 runs.
Avg: average (the smaller, the better) / Std: standard deviation.
+: the new adaptive method is significantly better under level 0.01.
-: the new adaptive method is significantly worse under level 0.01.
$\sim$: not significantly different under level 0.01.
*: it means that this value is less than 1.0e-300.

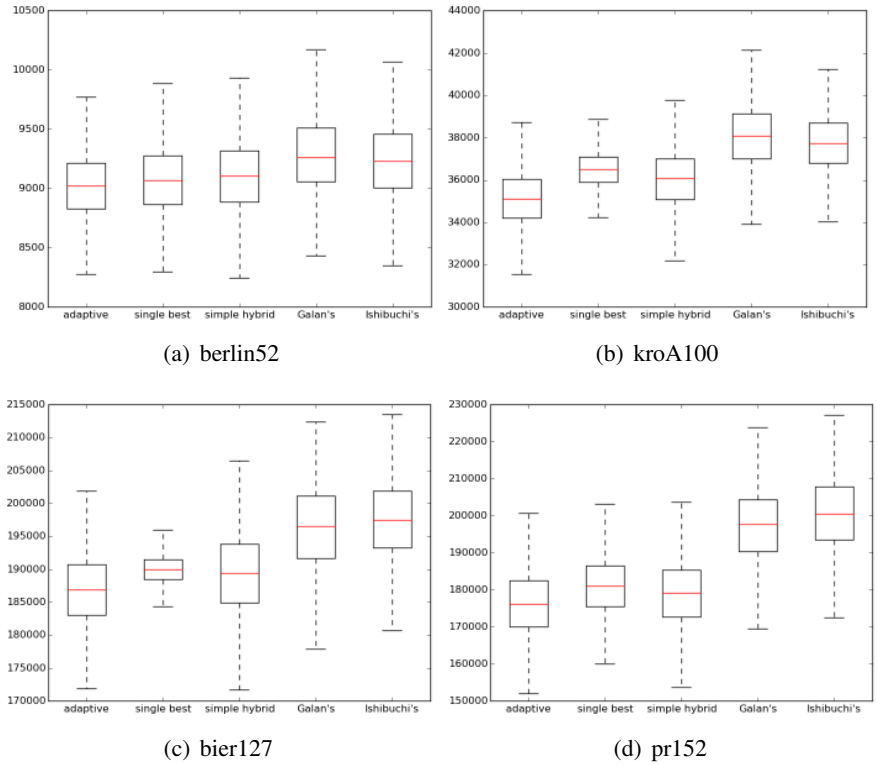
Chapter 7

Conclusion

7.1 Summary

Our study showed that mating scheme can be a very important part of genetic algorithm. We analyzed the effect of mating schemes in TSP and graph bisection problem. The problems themselves and the sizes of their solution spaces may take effect on performance. But we could observe the characteristics of mating schemes. Without an artificial influx of a new individual, NEAR mating causes losing diversity rapidly. But within large solution space or short time budget, NEAR mating can be a good choice.

The comparisons of solution qualities are displayed in Figure 20 and Figure 21. Our adaptive mating scheme shows better distribution than any other compared methods. Our adaptive mating scheme assesses the matched distance of individuals with their offspring. The NEAR scheme focuses on exploitation while FAR scheme focuses on exploitation. Our scheme tries to find a balanced point between exploration and exploitation in each generation. In various environments such with local-optimization or an influx of individuals, our adaptive scheme selects the proper schemes. Our adaptive scheme acts properly not only with the Hungarian schemes, but also with greedy methods.



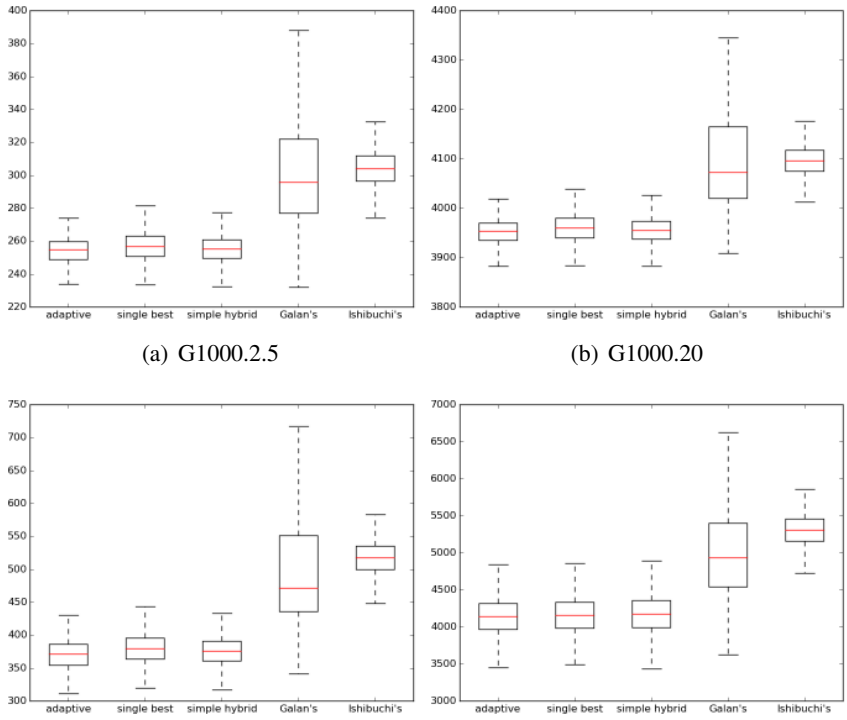
(a) berlin52 (b) kroA100

(c) bier127 (d) pr152

Figure 20: TSP comparison results

7.2 Future Work

We set the threshold parameters as 0.5 and 1.0 with some observation and theoretically justified them. But we expect that the method of dynamically adjusting these values may produce better results. Real-coded problems [DSKM09] [DBD03] [CC98] may have different characteristics from combinatorial optimization. More various problems such as function optimization [HJK95] can be tested with our scheme. There is room for further improvement and we will study the presented scheme with various opera-



(a) G1000.2.5 (b) G1000.20

(c) U1000.05 (d) U1000.40

Figure 21: Graph bisection comparison results

tions such as crossover, mutation rates, replacement, and local-optimization for future work.

Bibliography

[AR06] Moraglio Alberto and Poli Riccardo. Product geometric crossover. In *Parallel Problem Solving from Nature - PPSN IX: 9th International Conference*, pages 1018–1027, 2006.

[Avi83] David Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13:475–493, 1983.

[BCM03] Mousbah Barake, Pierre Chardaire, and Geoff P McKeown. The probe metaheuristic and its application to the multi-constraint knapsack problem. In *Metaheuristics: computer decision-making*, pages 19–36. Springer, 2003.

[Boo85] Lashon B. Booker. Improving the performance of genetic algorithms in classifier systems. In *Proceedings of the International Conference on Genetic Algorithms*, pages 80–92. Lawrence Erlbaum Associates, 1985.

[CC98] Fi-John Chang and Li Chen. Real-coded genetic algorithm for rule-based flood control reservoir management. *Water Resources Management*, 12(3):185–198, 1998.

[ČLM13] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)*, 45(3):35, 2013.

[DBD03] Ioannis G Damousis, Anastasios G Bakirtzis, and Petros S Dokopoulos. Network-constrained economic dispatch using real-coded genetic algorithm. *IEEE Transactions on Power Systems*, 18(1):198–205, 2003.

[DJ75] Kenneth Alan De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975.

[DJS92] Kenneth A De Jong and William M Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of mathematics and Artificial intelligence*, 5(1):1–26, 1992.

[DSKM09] Kusum Deep, Krishna Pratap Singh, Mitthan Lal Kansal, and C Mohan. A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Applied Mathematics and Computation*, 212(2):505–518, 2009.

[FF93] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. volume 1, pages 416–423. Citeseer, 1993.

[FTMR01] Carlos Fernandes, Rui Tavares, Cristian Munteanu, and Agostinho Rosa. Using assortative mating in genetic algorithms for vector quantization problems. In *Proceedings of the ACM symposium on Applied computing*, pages 361–365. ACM, 2001.

[GdMMR05] José Fernando Gonçalves, Jorge José de Magalhães Mendes, and Mauricio GC Resende. A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, 167(1):77–95, 2005.

[GL85] David E. Goldberg and Robert Lingle. Alleles, loci, and the traveling salesman problem. In *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pages 154–159, 1985.

[GLR03] Kai Song Goh, Andrew Lim, and Brian Rodrigues. Sexual selection for genetic algorithms. *Artificial Intelligence Review*, 19(2):123–152, 2003.

[GML05] Carlos Garcia-Marinez and Manuel Lozano. Hybrid real-coded genetic algorithms with female and male differentia-

tion. In *Proceedings of the Congress on Evolutionary Computation*, pages 896–903, 2005.

[GMP13] Severino F. Galán, Ole J. Mengshoel, and Rafael Pinter. A novel mating approach for genetic algorithms. *Evolutionary Computation*, 21(2):197–229, 2013.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[HF11] Susanne Huber and Martin Fieder. Educational homogamy lowers the odds of reproductive failure. *Public Library of Science One*, 6(7):e22330, 2011.

[HGH98] Peter Hahn, Thomas Grant, and Nat Hall. A branch-and-bound algorithm for the quadratic assignment problem based on the Hungarian method. *European Journal of Operational Research*, 108(3):629 – 640, 1998.

[HJK95] Christopher R Houck, Jeff Joines, and Michael G Kay. A genetic algorithm for function optimization: a matlab implementation. *NCSU-IE TR*, 95(09), 1995.

[HKS05] William Eugene Hart, Natalio Krasnogor, and James E Smith. Memetic evolutionary algorithms. In *Recent advances in memetic algorithms*, pages 3–27. Springer, 2005.

[HL99] Lin Hansheng and Kang Lishan. Balance between exploration and exploitation in genetic search. *Wuhan University Journal of Natural Sciences*, 4(1):28–32, 1999.

[HPR13] Karla L Hoffman, Manfred Padberg, and Giovanni Rinaldi. Traveling salesman problem. In *Encyclopedia of Operations Research and Management Science*, pages 1573–1578. Springer, 2013.

[INTN08] Hisao Ishibuchi, Kaname Narukawa, Noritaka Tsukamoto, and Yusuke Nojima. An empirical study on similarity-based mating for evolutionary multiobjective combinatorial optimization. *European Journal of Operational Research*, 188(1):57–75, 2008.

[IS03] Hisao Ishibuchi and Youhei Shibata. A similarity-based mating scheme for evolutionary multiobjective optimization. In *Lecture Notes in Computer Science*, pages 1065–1076, 2003.

[IS04] Hisao Ishibuchi and Youhei Shibata. Mating scheme for controlling the diversity-convergence balance for multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1259–1271, 2004.

[JB91] Donald R Jones and Mark A Beltramo. Solving partitioning problems with genetic algorithms. In *ICGA*, pages 442–449, 1991.

[Kin89] Helen M. Kingston. ABC of clinical genetics. techniques of DNA analysis. *BMJ*, 299(6690):34–37, 1989.

[KM04] Yong-Hyuk Kim and Byung-Ro Moon. Lock-gain based graph partitioning. *Journal of Heuristics*, 10(1):37–57, 2004.

[Kuh55] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[Lev93] David M Levine. A genetic algorithm for the set partitioning problem. In *Fifth International Conference on Genetic Algorithms*, pages 481–487, 1993.

[MF00] Peter Merz and Bernd Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transaction Evolutionary Computation*, 4(4):337–352, 2000.

[Mic96] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[Mil99] Jessica Miller. *Human Development: A Lifespan View, Instructor's Resource Manual*. Cengage Learning, 1999.

[MKYM07] Alberto Moraglio, Yong-Hyuk Kim, Yourim Yoon, and Byung-Ro Moon. Geometric crossovers for multiway graph partitioning. *Evolutionary Computation*, 15(4):445–474, 2007.

[Mor07] Alberto Moraglio. *Towards a geometric unification of evolutionary algorithms*. PhD thesis, University of Essex, Wivenhoe Park, Colchester, 2007.

[MP04] Alberto Moraglio and Riccardo Poli. Topological interpretation of crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 3102, pages 1377–1388, 2004.

[MSWO91] Olivier Martin and Edward W. Felten Steve W. Otto. Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5(3):299–251, 1991.

[OMKRJ05] Gabriela Ochoa, Christian Mädler-Kron, Ricardo Rodriguez, and Klaus Jaffe. Assortative mating in genetic algorithms for dynamic problems. In *Applications of Evolutionary Computing*, pages 617–622, 2005.

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., 1982.

[Rei91] Gerhard Reinelt. TSPLIB - a traveling salesman problem library. *INFORMS Journal on Computing*, 3(4):376–384, 1991.

[RL11] Fatemeh Ramezani and Shahriar Lotfi. IAMGA: intimate-based assortative mating genetic algorithm. In *Proceedings of the Swarm Evolutionary and Memetic Computing Conference*, pages 240–247, 2011.

[RN99] Miguel Rocha and José Neves. Preventing premature convergence to local optima in genetic algorithms via random offspring generation. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 127–136. Springer, 1999.

[SM02] Dong-Il Seo and Byung-Ro Moon. Voronoi quantizied crossover for traveling salesman problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 544–552, 2002.

[Sys89] Gilbert Syswereda. Uniform crossover in genetic algorithms. In *In Proceedings of the third international conference on Genetic algorithms*, pages 2–9, 1989.

[TB91] E-G Talbi and Pierre Bessiere. A parallel genetic algorithm for the graph partitioning problem. In *Proceedings of the 5th international conference on Supercomputing*, pages 312–320, 1991.

[Wel47] Bernard L Welch. The generalization of 'Student's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[YKM08] Yourim Yoon, Yong-Hyuk Kim, and Byung-Ro Moon. Feasibility-preserving crossover for maximum k-coverage problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 593–598, 2008.

[YKMM12] Yourim Yoon, Yong-Hyuk Kim, Alberto Moraglio, and Byung-Ro Moon. Quotient geometric crossovers and redun-

dant encodings. *Theoretical Computer Science*, 425:4–16, 2012.

국문 초록

짝짓기 제도는 자식 해를 만들기 위하여 두 부모를 선택하는 방법을 말한다. 이는 유전 알고리즘의 동작 전반에 영향을 끼친다. 본 논문에서는, 헝가리안 방법을 사용한 짝짓기 제도에 대해 연구하였다. 그 제도들은 대응되는 거리의 합을 최소화하는 방법, 최대화하는 방법, 그리고 비교를 위해 랜덤하게 대응시키는 방법들을 가리킨다. 본 논문에서는 이 제도들을 잘 알려진 문제인 순회 판매원 문제와 그래프 분할 문제에 적용하였다. 또한 세대별로 가장 좋은 해가 어떻게 변화하는지 분석하였다. 이러한 분석에 기초하여, 본 논문에서는 간단히 결합된 짝짓기 제도를 제안하였다. 제안된 제도는 결합되지 않은 제도에 비해 더 좋은 결과를 보였다.

본 논문에서는 또한, 본 논문의 핵심 방법인 짝짓기 제도를 결합하는 방법을 제안한다. 본 논문의 적응적인 짝짓기 방법은 세 헝가리안 제도 중 하나를 선택한다. 모든 짝지어진 쌍은 다음 세대를 위한 짝짓기 방법을 결정할 투표권을 갖게 된다. 각각의 선호도는 부모해간 거리와 부모해와 자식해의 거리의 비율을 통해 결정된다. 제안된 적응적 방법은 모든 단일 헝가리안 짝짓기 제도, 비적응적으로 결합된 방법, 전통적인 룰렛 휠 선택, 기존의 다른 거리 기준 방법들보다 좋은 결과를 보였다. 제안된 적응적 방법은 정기적인 해집단의 유입과 지역 최적화와 결합된 환경에서도 적절한 제도를 선택했다. 본 논문에서는 헝가리안 방법을 최대 혹은 최소의 지역 최적점을 찾는 방법으로 교체했다. 이 방식 역시 지역 최적점을 찾는 단일 방법들보다 좋은 결과를 보였다.