Preprint of the paper published in:
IEEE Transactions on System, Man, and Cybernetics Part A, vol.32, no.4, pp.497-504 (2002)

Coping with Nonstationary Environments: A Genetic Algorithm using Neutral Variation

Teijiro Isokawa†, Nobuyuki Matsui†, Haruhiko Nishimura†, and Ferdinand Peper*

†Division of Computer Engineering, Himeji Institute of Technology,
2167 Shosha, Himeji, 671-2201, JAPAN
{isokawa,matsui}@comp.eng.himeji-tech.ac.jp

‡Studies of Information Science, Hyogo University of Education,
942-1 Shimokume, Yashiro-cho, 673-1494, JAPAN
haru@life.hyogo-u.ac.jp

*Nanotechnology Group, Communications Research Laboratory,
588-2 Iwaoka, Iwaoka-cho, Nishi-ku, Kobe, 651-2492, JAPAN.
peper@crl.go.jp

In nonstationary environments, it is difficult to apply traditional Genetic Algorithms(GAs) because they use strong selection pressure and lose the diversity of individuals rapidly. We propose a GA with neutral variation that can track environmental changes. The idea of this GA is inspired by Kimura's neutral theory. The scheme of this GA allows neutral characters, which do not directly affect the fitness with respect to environments, thus increasing the diversity of individuals. In order to demonstrate the properties of this GA, we apply it to a permutation problem called Ladder-Network, of which the imposed alignment on the output changes regularly. We show that the GA with neutral variation can adapt better to environmental changes than a traditional GA.

Keywords: Genetic Algorithm, Neutral Variation, Self-Adaptability, Nonstationary Environment, Ladder-Network.

1 Introduction

In the application of Genetic Algorithms(GAs)(Holland [1975], Goldberg [1989], Mitchell [1996], Fogel [2000], Forrest [1993]) to optimization problems, adoption of strong selection pressure accelerates the evolution of a population and enhances the optimization performance. Strong selection pressure, however, can be detrimental in practical applications, since it limits the diversity of the population that is evolving. This problem especially manifests itself in nonstationary environments, because to allow a GA to track solutions efficiently, population diversity covering a significant part of the solution-space is indispensable. That is, it is difficult for traditional GAs to track nonstationary environments if the diversity of individuals is lost rapidly due to a strong selection pressure.

It is important to maintain diversity in GAs, from the viewpoints both of tracking nonstationary environments and of finding optimal solutions of multi-modal functions. Therefore, many studies focus on this problem. They are divided into three major groups.

One way to cope with nonstationary environments is to introduce new genetic operators into the framework of GA. Cobb proposed the so-called Triggered Hypermutation operator that increases the mutation rate temporarily upon a detected decrease of the time-averaged best performance(Cobb [1990], Cobb and Grefenstette [1993]). A higher mutation rate can change the composition of a population, thereby enabling GA with Triggered Hypermutation to track environmental change. A more direct way to change the composition is Random Immigration, proposed in (Grefenstette [1992]), which replaces some individuals of a population with ones of which genotypes are perfectly at random in every generation.

Both Triggered Hypermutation and Random Immigration aim to increase the diversity of the population. Instead of increasing diversity, it is also possible to preserve it: Goldberg and Richardson proposed the shared

fitness function to evaluate the affinity of individuals(Goldberg and Richardson [1987]). According to this scheme, an increase in the number of individuals with a similar genotype(or phenotype) causes a decrease to their fitness, so population diversity is maintained. Another example of this technique is Thermodynamical GA(TDGA), proposed by Mori, which also uses the evaluation of diversity in the fitness function(Mori et al. [1995]).

Another way to cope with nonstationary environments is to expand the memory of a chromosome or of a population. Chromosome memory is typically expanded by the use of a diploid (pair of chromosomes) or a multiploid (group of chromosomes), as proposed in (Goldberg and Smith [1987], Dasgupta and McGregor [1992], Hadad and Eick [1997]). The diploid or multiploid representation is translated to a haploid (single chromosome) representation by using a dominance operator. This representation can be handled in the framework of traditional GA and at the same time carry redundant information. Population-based memory is typically expanded by storing individuals that have good performance to be re-introduced in later generations. In GA with a Case-based Memory, the worst individuals are replaced by the best individuals from previous generations when environmental changes are detected (Eggermont et al. [2001]). The Immune Algorithm(IA), inspired by the natural immune system, is another population-based memory approach (Fukuda et al. [1999]). To adapt a population, IA uses a memory for storing and retrieving past solutions.

One more way to cope with nonstationary environments is increasing the geographical spread of a population, thereby stimulating population diversity in a spatial way. In nature examples of this are subpopulations separated by mountain ranges or water, which eventually might lead to different subspecies. In the context of GA, this method is employed in (Collins and Jefferson [1991], Sarma and Jong [1999]). In these studies, the genetic information(or a population) is spatially distributed and their local neighborhood is defined. Mating between individuals(crossover) is restricted within their neighborhood, which is different from the traditional GAs where any individual has the potential for mating with any other. This setting prevents individuals to be homogeneous, so the diversity of a population is maintained.

Our strategy for constructing a GA that can track nonstationary environments is to use neutrality to the heredity of an individual without introducing new operators (Matsui et al. [1997], Isokawa et al. [1999]). This idea is inspired by Kimura's neutral theory of molecular evolution (Kimura [1983], Avise [1994]). In neutral theory, some of the heredity in an individual is unaffected by (or, in other words, neutral to) selection pressure, unlike orthogenesis, and allowed to be determined by chance, i.e., by so-called random genetic drift. This results in redundancy of the heredity of individuals, thus diversifying a population. Redundancy of genetic information, having hardly been regarded as important from the viewpoint of efficiency and performance in engineering, has invited only few examinations about the effect of the existence of neutral mutations. We believe, however, that it is an important framework to study evolutionary systems in nonstationary environments, and to obtain effective GAs for problems in such environments.

In this paper, we further proceed the model of a GA with neutral variation (Matsui et al. [1997], Isokawa et al. [1999]) by enlarging the scale of simulations and exploring the ability of this GA more quantitatively. We apply our GA to a permutation problem, a useful example being the so-called Ladder-Network, which is a network that permutes symbols in a string in accordance to its topology. This network allows the existence of neutral factors in the representation of its permutation function, and thus offers a good framework to test our proposed strategy by simulations.

2 Models and Methods

The permutation problem is defined as the problem of finding a description of a function that performs a certain permutation. Various descriptions can be candidates for this problem. Among them, the Ladder-Network serves as a suitable function description for our purposes: it performs permutations, and it incorporates neutral factors.

The Ladder-Network consists of a set of vertical bars, each of which can be connected to its direct neighboring bar(s) by horizontal bars(rungs). Its input consists of strings containing a number of symbols equal to the number of vertical bars. Each symbol flows downwards along the vertical bar it is on, and upon encountering a rung, it moves along the rung toward the opposing vertical bar, after which it continues its journey downward, following the same procedure of exchanging bars along rungs, until it finally arrives at the output side at the bottom of the network (see Fig.1). Obviously, this procedure results in a permutation of the input string at the output side of the network. For example, the input string (ABCD) is changed into (BDCA) by the network in Fig.1. It is noted that the movement of a symbol encountering rungs which connect at the same height to both sides of one vertical bar as in Fig.2 is undefined, and we do not treat this kind of networks.

The Ladder-Network described above has a neutral factor with respect to its functionality because there

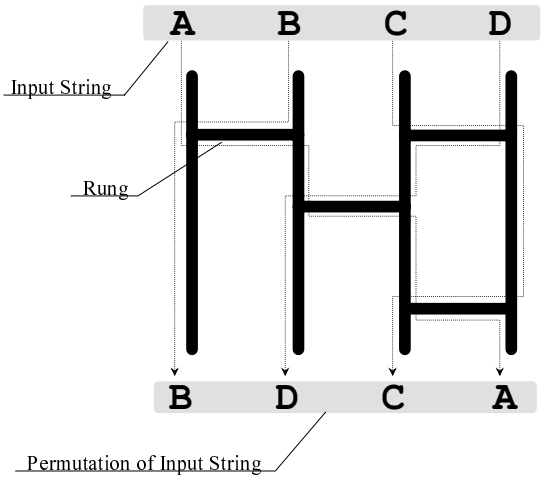


Figure 1: An example of Ladder-Network and its permutation between input and output strings.

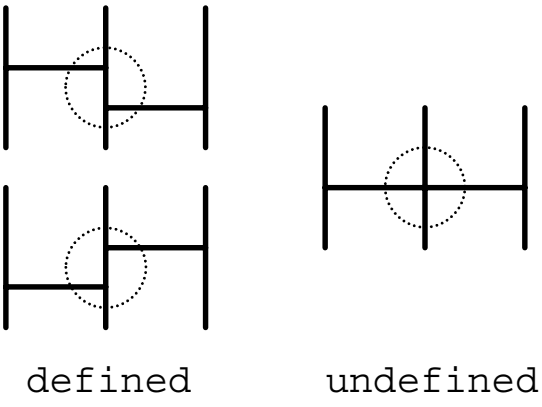


Figure 2: The connecting form of rungs to a vertical bar (surrounded by the dotted circle).

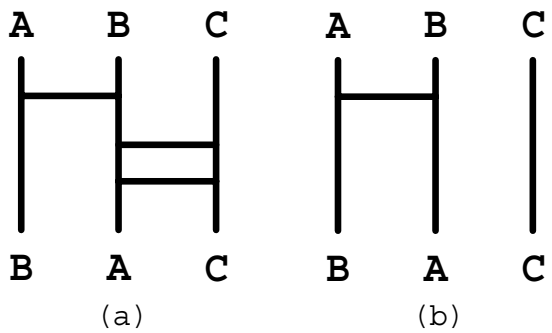


Figure 3: An example in which the permutation is invariant under the deletion of a double rung.

is a multitude of networks that achieve a certain permutation. Such a case occurs if, due to the placement of a multitude of rungs between two vertical bars, the output of the network is the same as when the rungs would be absent, for example in Fig.3. The structure of the network shown in Fig.3(a) is different from the one shown in Fig.3(b), but the output permutations of these two networks are the same. In other words, the Ladder-Network in Fig.3 has a neutral factor.

To design a GA for finding a Ladder-Network structure that performs a given permutation, the network is encoded by a genotype. Encoding the genotype of a Ladder-Network is done in the following way. First, the actual Ladder-Network (Fig.4(a)) is deformed in a topology-preserving way, such that it can be projected onto a checkerboard-like lattice, each shaded cell of which contains at most one rung (Fig.4(b)). Then, the value 1 or 0 is given to each shaded lattice cell depending on whether a rung is present or not in the cell (Fig.4(c)). Finally, the genotype of the Ladder-Network is obtained by rearranging the lattice into a one-dimensional sequence (Fig.4(d)). By this procedure, we acquire the genotype $\boldsymbol{g}(i)$ of the Ladder-Network, where i is the index of the Ladder-Network. We define the phenotype of a Ladder-Network as the permutation it imposes on its input. Clearly, a Ladder-Network's genotype unambiguously determines its phenotype, but not the other way around.

Considering that the problem to be solved is finding a Ladder-Network that performs a certain target permutation, we define the fitness of a Ladder-Network in terms of how near its associated permutation is to the target permutation. Let us consider a set $\{x_1, \cdots, x_n\}$ and the sequences $\boldsymbol{X}_\alpha = (x_{\alpha 1}, \cdots, x_{\alpha n})$ and $\boldsymbol{X}_\beta = (x_{\beta 1}, \cdots, x_{\beta n})$, with $x_{\alpha i}, x_{\beta i} \in \{x_1, \cdots, x_n\}$.
The distance between $\boldsymbol{X}_\alpha$ and $\boldsymbol{X}_\beta$ is defined by:

$$\text{distance}(\boldsymbol{X}_\alpha, \boldsymbol{X}_\beta) = \sum_i |\text{Pos}(\boldsymbol{X}_\alpha, x_i) - \text{Pos}(\boldsymbol{X}_\beta, x_i)| \tag{1}$$

where $\text{Pos}(\boldsymbol{X}, x_i)$ means the position of x_i in the array $\boldsymbol{X} = (x_1, \cdots, x_n)$.

For example, the distance between $\boldsymbol{X}_\alpha = (ABCD)$ and $\boldsymbol{X}_\beta = (BDCA)$ is given by

$$\begin{aligned} \text{distance}(\boldsymbol{X}_\alpha, \boldsymbol{X}_\beta) &= |1-4| + |2-1| + |3-3| + |4-2| \\ &= 3 + 1 + 0 + 2 \\ &= 6. \end{aligned}$$

Using the distance measure of Eq.(1), we define the fitness function of the i-th individual as

$$\text{fitness}(i) = \text{distance}(\boldsymbol{X}_i, \boldsymbol{X}_T) + c \cdot a(i)/N \tag{2}$$

where $\boldsymbol{X}_i$ and $\boldsymbol{X}_T$ are the output and target arrays, respectively, N is the total number of lattices, $a(i)$ is the number of rungs in the i-th individual, and c is a constant.

The smaller the fitness value of an individual, the closer the individual's permutation is to the target permutation, i.e., the fitter the individual is. It is noted that the number of rungs becomes neutral with respect to selection when it is not directly suppressed through Eq.(2), that is, when $c = 0$.

We finally compose a GA using the genotype and the phenotype of a Ladder-Network and the fitness function. A population which includes some individuals is prepared, and the rungs of each individual are randomly placed. Each individual's fitness with respect to an imposed target is calculated by using the fitness function. Then, the individuals for the next generation are selected in accordance with their fitness values using so-called Elitist Selection described below, and reproduced.

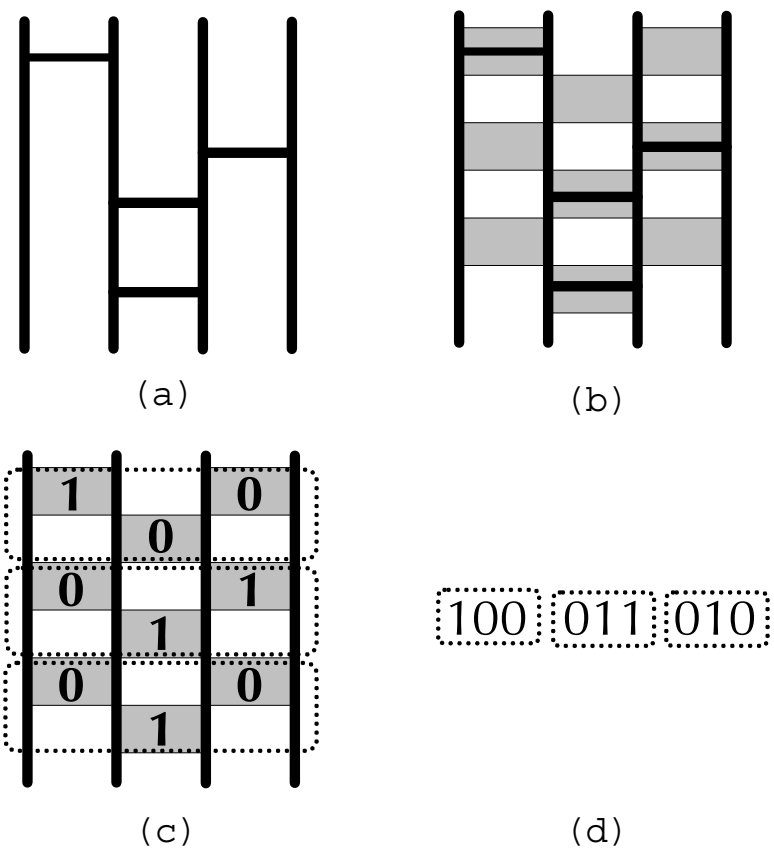


Figure 4: The encoding scheme of the genotype of the Ladder-Network.

In elitist selection, individuals in the top stream, as ranked by their fitness values, are always selected and preserved for the population of the next generation. Individuals not selected are replaced with duplicates of the selected ones, the number of individuals being constant between generations. Figure 5 shows an example of the selection and duplication operations in our GA. First, the individuals are ranked by their fitness values. The top 20% of the individuals, those which have lower fitness values, are selected and preserved. The remaining 80% of the individuals are discarded. Four copies of individuals are made from each of the preserved individuals, therefore, the number of individuals remains same as in the previous generation. After selection and duplication, the crossover and mutation operations are applied on the genotypes of the individuals. Crossover on the genotypes is accomplished by randomly selecting a certain number of pairs of individuals and exchanging partial genes between each pair of individuals with a certain probability (the crossover rate). Mutation on the genotypes is done by changing the values of the genes in all the individuals with a certain probability (the mutation rate). A new population is prepared after mutation is completed, and this population undergoes the same reproduction cycle again.

Summarizing, the GA that searches for a Ladder-Network structure performing a certain target permutation works by iteratively applying the loop of fitness evaluation, selection, duplication, crossover, and mutation (see Fig.6).

3 Results

3.1 Conditions of computer simulations

The scheme described above allows us to seek for a network of which the output is as near as possible to a certain target. Computer simulations are done to test the adaptability of a population of Ladder-Networks to a changing environment. Figure 7 shows an example of a task changing every M generations, where the process of finding one imposed input-output relation is called a task.

We carried out the simulations under the following conditions.

1. The size of the lattice of the Ladder-Network is 10 rows by 10 columns, giving a total of 100 bits as

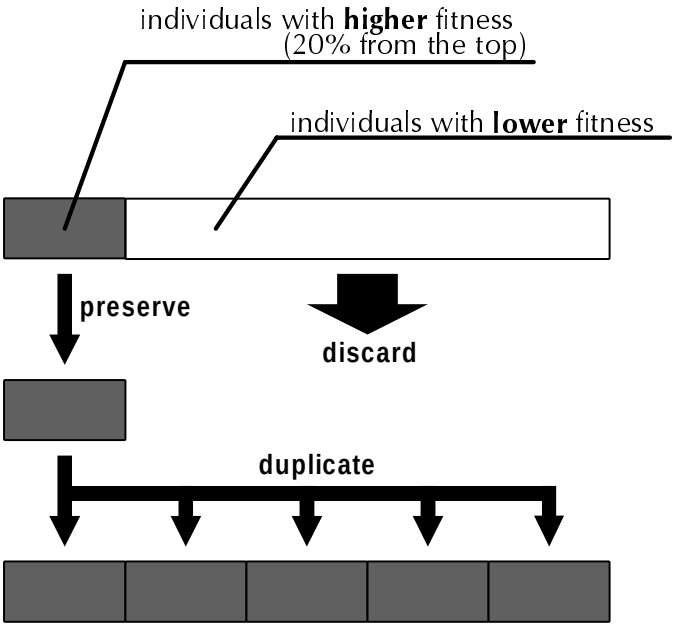


Figure 5: Selecting and Duplicating processes for a population.

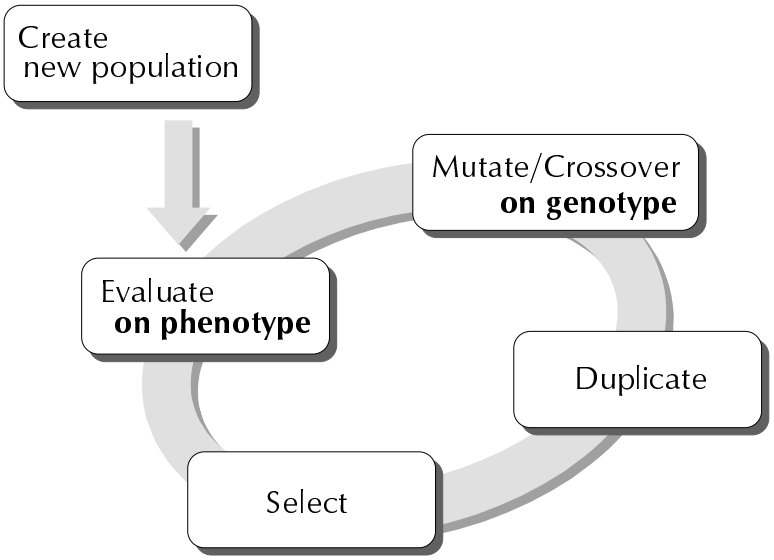


Figure 6: The flowchart of the Genetic Algorithm procedure.

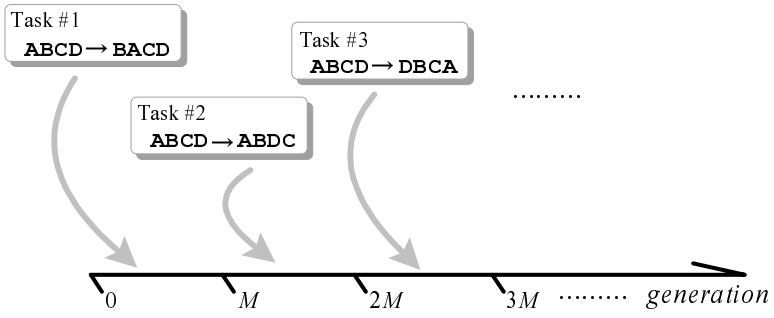


Figure 7: An environmental condition changing every M generations.

Task No.	Generation	Input string	Target string
1	0 – 999	ABCDEFGHIJK	CJADIEGKHFB
2	1000 – 1999	ABCDEFGHIJK	BADHEIGFCKJ
3	2000 – 2999	ABCDEFGHIJK	DGBKHCFJIAE
4	3000 – 3999	ABCDEFGHIJK	AGJKHCBFIED
5	4000 – 4999	ABCDEFGHIJK	GDEKJHCBFIA
6	5000 – 5999	ABCDEFGHIJK	EJBCGIDFAHK
⋮	⋮	⋮	⋮

Figure 8: An example of a set of tasks.

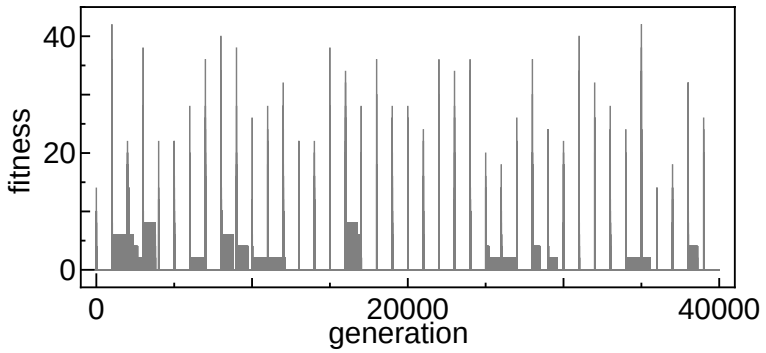
genes.

2. The target permutation changes every $M = 1000$ generations.
3. The number of individuals in the population is 100.
4. The number of tasks is 40. Examples of tasks are shown in Fig.8.
5. The number of pairs of individuals randomly selected for crossover is 50.
6. The crossover rate is 0.2.
7. The mutation rate is 0.005.
8. The number of individuals preserved in each generation is 20.

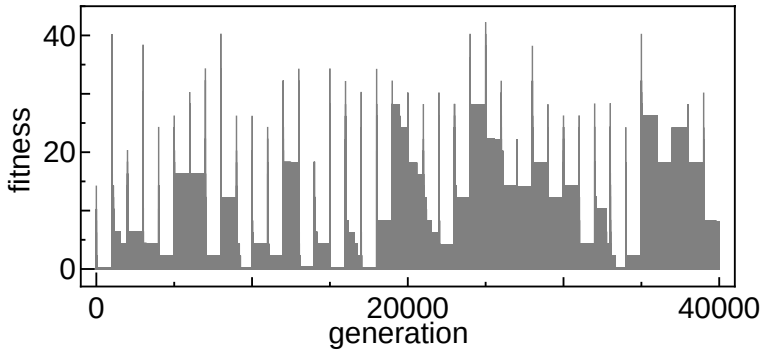
In the following sections, we explore the adaptability of the GA in the case it includes neutrality and in the case it does not. We include neutrality by setting $c = 0$ in Eq.(2) and leave it out by setting $c > 0$. Selection of individuals is often very sensitive to the value of the fitness. This implies that some care is necessary in setting the value of c in the case $c > 0$. In the case of elitist selection, which we employ here, however, the fitness order of individuals is more important than the fitness itself for each individual to be selected for the population of the next generation. The fitness order of individuals does not change when $0 < c < 2$ because the minimum difference of distance$(\boldsymbol{X}_i, \boldsymbol{X}_T)$ over all possible pairs $\boldsymbol{X}_i$ and $\boldsymbol{X}_T$ always exceeds the maximum over all c in the range $(0, 2)$ of the value $c \cdot a(i)/N$. We may thus choose any value of c in the range $(0, 2)$ without affecting the selection procedure in the GA. For convenience, we set $c = 1$. The background for this choice is that, since $0 < a(i)/N < 1$, the fractional part of the fitness value will be due to the part $c \cdot a(i)/N$ in Eq.(2), whereas the integer part will be due to the part distance$(\boldsymbol{X}_i, \boldsymbol{X}_T)$ in Eq.(2). This simplifies comparison with the fitness value in the case $c = 0$, since that is only made up from an integer part. This is of particular importance for meaningful comparison of simulations for the case $c = 0$ (the case including neutrality) with simulations for the case $c > 0$ (the case without neutrality).

3.2 Results of simulations

First, we show how well a population adapts to a nonstationary environment due to the inclusion of neutrality, and compare the adaptability to the case when there is no neutrality in the genotypes of individuals. The evolution of the fitness of the best individual in each generation is shown in Fig.9 for the cases including and excluding neutrality. Note that individuals with lower fitness values are more adaptive. In both cases we find that at every 1000 generations the value of the fitness becomes suddenly higher, i.e., the fitness becomes suddenly worse. This phenomenon is caused by the change of task every 1000 generations. The value of the fitness decreases in many tasks in the case there is neutrality, while it remains high in the case there is no neutrality. Comparing these two results of the fitness (shaded areas), we see that individuals in the case without neutrality ($c = 1$ in Eq.(2)) are less able to adapt their structures to the tasks that change over time. The first term of $fitness(i)$ being zero for a certain individual means that the individual causes the same permutation as the imposed task: this is called the achievement of the task. From Fig.9, we count the number of achieved tasks. The number of achievements in the case including neutrality is 31, and the number in the case without neutrality is 6. The selection pressure on the number of rungs decreases the activity of random genetic drift in each task and this makes the network lose its adaptability.



(a) with neutrality



(b) without neutrality

Figure 9: Evolution of fitness in the case including neutrality and in the case excluding neutrality, where the region under the curve is shaded for ease of comparison of these two cases.

In order to confirm the reliability of the above differences in adaptability, the simulations in Fig.9 are repeated for 100 kinds of initial populations under 40 different task sequences. Figure 10 shows the histograms for 4000 data points, each corresponding to the individual with the best fitness in each task($M = 1000$). Here, the fractional part of the fitness value that appears in the case there is no neutrality is omitted. From Fig.10(a) we find that the number of cases at which a certain best fitness is achieved in the case there is neutrality almost concentrates around zero, but in the case there is no neutrality, the number of cases is almost uniformly spread over the range [0,20]. This indicates that the case there is neutrality is superior in its adaptability as compared to the case there is no neutrality. A similar tendency of the difference in adaptability is confirmed by changing the crossover rate to 0.4 and the mutation rate to 0.01 as shown in Fig.10(b).

Next, we investigate the effect of selection pressure on the structural diversity of the Ladder-Network. The number of rungs in individuals is a measure for the structural difference, because individuals that have different numbers of rungs always have different structures. Figure 11 shows the evolution of the number of rungs of the best individual in each generation in the same simulation as in Fig.9. As to the number of rungs in the case there is neutrality, it keeps fluctuating even after the fitness falls to zero. This means that there are more than one individuals with the same function and the diversity of individuals is kept even after the task is achieved. On the contrary, in the case there is no neutrality, the number of rungs has the tendency to be stationary in each task. In other words, the individuals are very similar to each other with respect to the number of rungs, i.e., there is less diversity than in the case that includes neutrality.

In both cases, we see the number of rungs fluctuating within about 30 rungs on the whole evolution process. Especially, in the case including neutrality, the number of rungs changes within a certain range between 40 and 70, even though we do not control it, due to the setting $c = 0$ in Eq.(2). This large fluctuation seems to be caused by selection pressure, though the number of rungs is not affected directly. We prepare Fig.12 for checking this phenomenon. This shows the changes of the average number of rungs in all individuals in the absence of the selection operation. The absence of the selection operation corresponds to the situation that only crossovers and mutations take place in the GA in Fig.6. The number of rungs reaches almost 50 (half of the total number of lattice cells in an individual) on both the initial conditions of 20 and 80 rungs. This is due to stochastic diffusion in the random mutation processes. From Fig.12, it is deduced that when the selection operation is not employed at all during evolution, the number of rungs does not fluctuate much. In Fig.11 we see a similar tendency for the number of rungs to converge to 50 in the case including neutrality, though there is more fluctuation due to the selection operation.

In the above discussion, we have referred only to the number of rungs, but not to the actual structure of the networks. Actually there can be several different networks that have the same number of rungs. In order to quantify structural changes of the individuals, we define the difference between two networks, Diff, as follows:

$$\mathrm{Diff}(\boldsymbol{g}(A), \boldsymbol{g}(B)) = \frac{H(\boldsymbol{g}(A), \boldsymbol{g}(B))}{N} \tag{3}$$

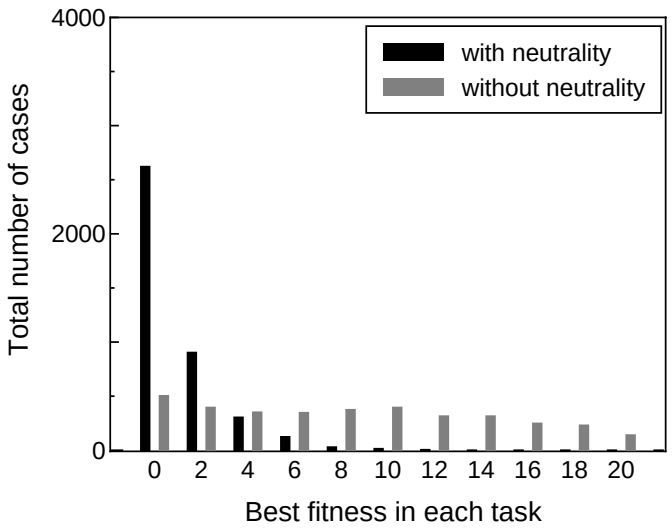
where $H(\boldsymbol{g}(A), \boldsymbol{g}(B))$ is the Hamming distance between genotypes of networks $\boldsymbol{g}(A)$ and $\boldsymbol{g}(B)$, and N is the total number of bits in the genes of individuals.

To examine diversity, we calculate the value of Diff between individuals in the same generation. We denote this Diff as Diff_{intra} in the following discussion. From the data of Fig.9, for all combinations of two individuals, Diff_{intra} is calculated and the maximum value and average value of Diff_{intra} are also calculated. Figure 13 shows the maximum Diff and average Diff in the generations from 24000-th to 24030-th as a typical example. The maximum value of Diff_{intra} in the case including neutrality is 1.5 times as large as the value in the case excluding neutrality from the 24003-th generation to the 24009-th generation. For each task, the average of the maximum values of Diff_{intra} for 30 generations from the generation that a new task is imposed is calculated. Figure 14 shows the averages of the maximum values of Diff_{intra} for all tasks. In this figure we see the tendency that the averages in the case including neutrality is higher than that in the case excluding neutrality. By summing up Diff_{intra} $\times$ the number of cases for each Diff_{intra} of Fig.14, we get the following quantities for the two cases.

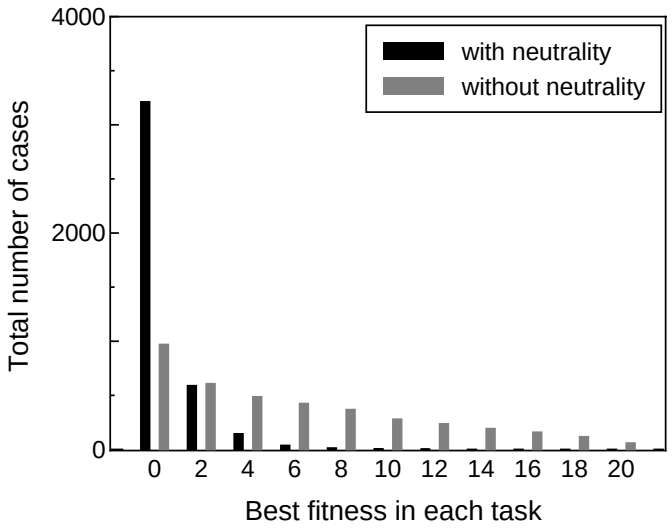
with neutrality	0.092
without neutrality	0.076

These quantities equal the average value of Diff_{intra} weighted over a total of 40000 generations. The average value in the case including neutrality is about 20% larger than in the case excluding neutrality. This difference between the two cases relates to the differences in performance of tracking environmental changes.

In order to discuss diversity further, we calculate Diff between different generations. This is accomplished by substituting the best individual in the t-th generation for A and the best individual in the $(t-1)$-th



(a)



(b)

Figure 10: Histograms of fitness values ((a): crossover rate is 0.2 and mutation rate is 0.005, (b): crossover rate is 0.4 and mutation rate is 0.01).

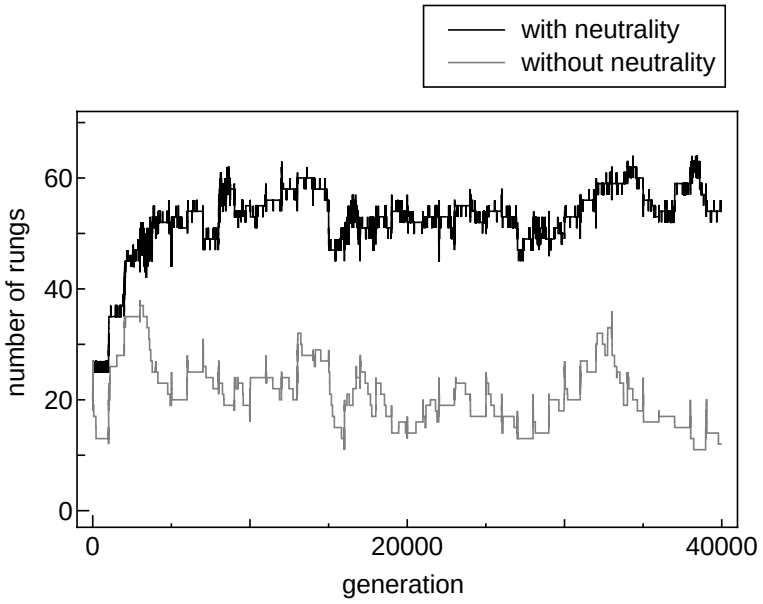


Figure 11: Evolution of the number of rungs in the case including neutrality and in the case excluding neutrality.

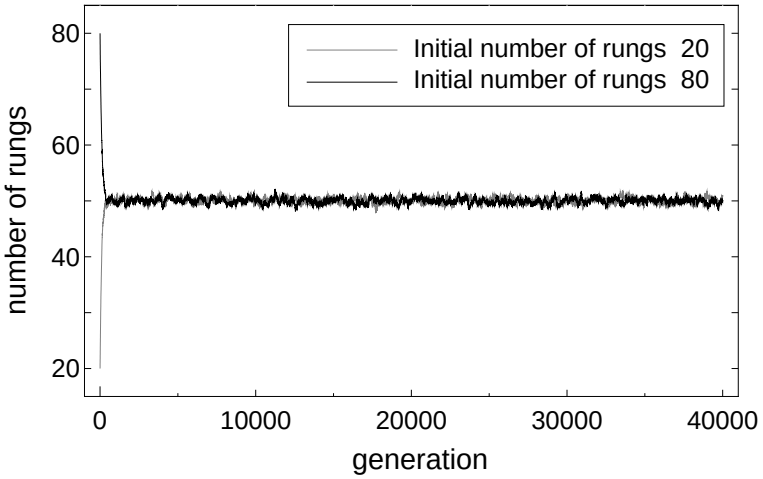


Figure 12: Changes of the number of rungs in the case there is no selection operation.

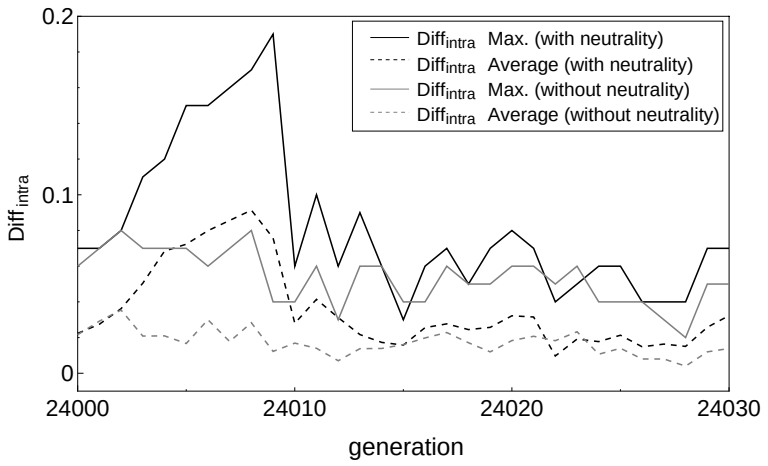


Figure 13: The maximum Diff_{intra} and average in a population from the 24000-th generation and 24030-th generation.

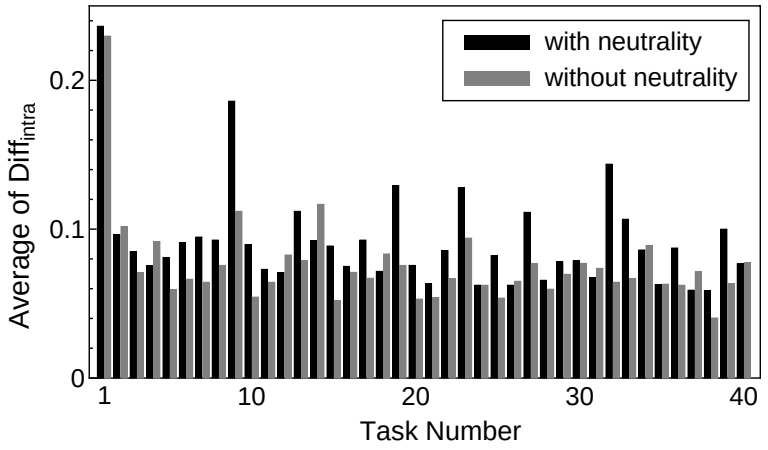
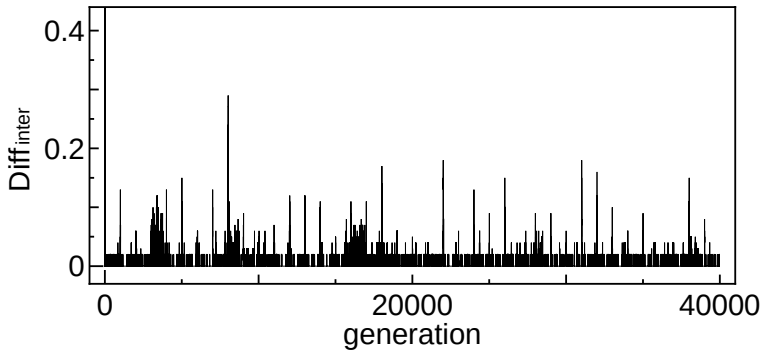
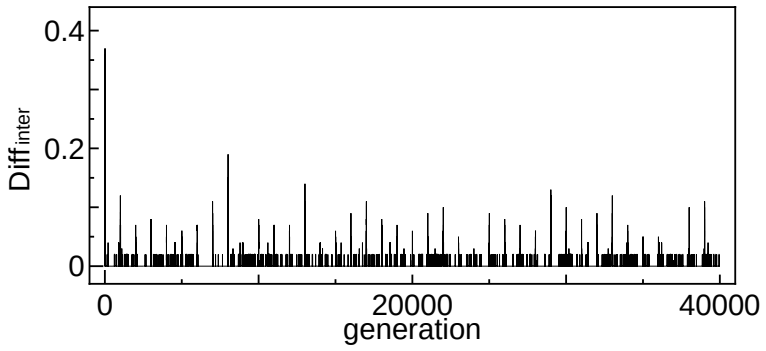


Figure 14: The average of the maximum Diff_{intra} for all tasks.



(a) with neutrality



(b) without neutrality

Figure 15: Changes of Diff_{inter} in the case there is neutrality and in the case there is no neutrality.

generation for B in Eq.(3): this Diff is denoted by Diff_{inter} in the following. The Diff_{inter} between the best individual in the t-th generation and that in the $(t-1)$-th generation is calculated from the data in Fig.9 for t in the interval $[1, 39999]$. Figure 15 shows the changes of Diff_{inter} in the case including neutrality and in the case without neutrality in a simulation with the same tasks as shown in Fig.9. The value of Diff_{inter} in the case there is neutrality is higher than in the case without neutrality. This means diversity in the case including neutrality is retained better than in the case excluding neutrality, and this result shows as well as in Fig.11.

Next, we explore the relation between adaptability and Diff_{inter}. Figure 16 shows the changes of the fitness and of Diff_{inter} in two cases (with and without neutrality) in the neighborhood of the 18000-th generation, which is the point of change from the 18-th task to the 19-th task, as a typical example. During 20 generations after the 18000-th generation, i.e., after the target string is changed, Diff_{inter} in the case including neutrality(solid line) changes more than in the case without neutrality(dotted line), indicating that a population changes faster after a change of task in the case including neutrality. At the same time, the fitness value decreases gradually to zero after a sudden increase at the change of task in the case including neutrality. After the fitness becomes zero, a non-zero value of Diff_{inter} sometimes appears in the case including neutrality, while in the case excluding neutrality Diff_{inter} remains zero. This phenomenon shows that in the case including neutrality there exist several individuals whose fitness value is zero, and the diversity in a population is maintained after the target permutation is found.

Finally, in Fig.17 we show the occurrence of Diff_{inter} throughout the same process (40000 generations) as in Fig.9. In the case including neutrality, Diff_{inter} is substantially represented in the range of the larger value of Diff(0.05 to 0.2), as compared to the case excluding neutrality.

By summing up Diff_{inter} weighted by its occurrence in Fig.17, we get the following quantities for the two

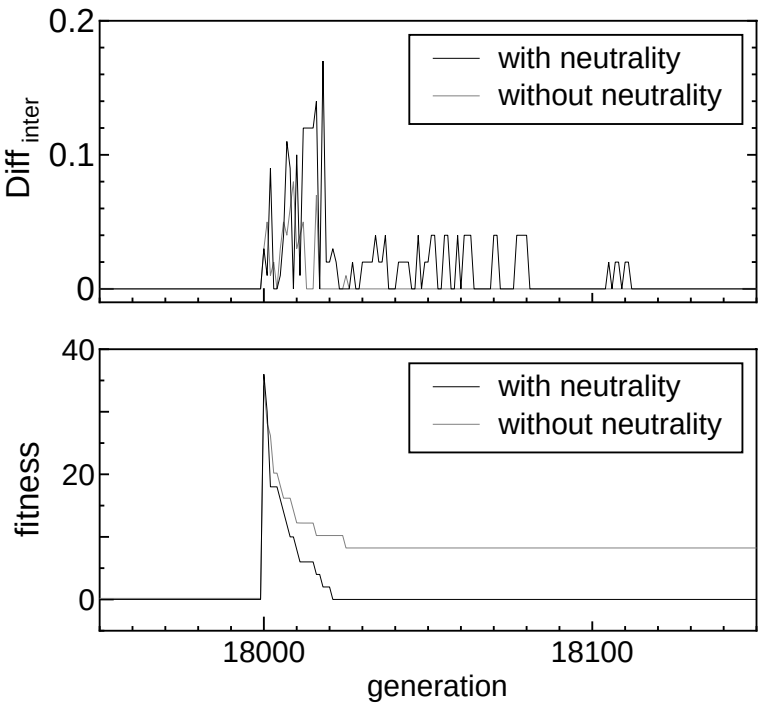


Figure 16: Changes of Diff_{inter} and the fitness around 18000-th generation.

cases:

with neutrality	171
without neutrality	55

These quantities reflect the degree of the structural diversity, and therefore, the degree of robustness against changes in tasks, i.e., adaptability.

From these results we conclude that introducing neutrality increases the structural diversity in a population and this leads to the formation of a population that can track environmental changes.

4 Discussion and Conclusion

In this paper we propose a GA with neutral variation that can track environmental changes, and demonstrate the properties of this GA by applying it to a permutation problem. In the permutation problem, we represent a permutation function as a Ladder-Network because it can include neutrality. We investigated to what extent neutral factors make a difference as to the adaptability of the network structure for a nonstationary task. By comparing the number of tasks fitting the evolution's target, we found that the adaptability in the case including neutrality is much higher than that in the case excluding it. Furthermore, we measured the structural diversity of both the case including neutrality and the case excluding neutrality, and conclude that population diversity is maintained much better in the case including neutrality than in the case excluding it. From these results, we conclude that adaptability can be enhanced by including a neutral factor in a Genetic Algorithm.

The framework of our model is inspired by Kimura's neutral theory, but we do not intend to design a strict biological model of neutral theory. Our purpose is rather to establish a framework for engineering based on neutral theory or related theories and to explore the resulting adaptive properties. The Ladder-Network lends itself well for the inclusion of neutrality in the permutation problem, and this is particularly due to the fitness function being nonuniquely determined, i.e., the fitness function gives the same value for a multitude of equivalent genotypes. The technique outlined in this paper of including neutrality in a problem is applicable to other nonstationary GA as well, provided a suitable nonuniquely determined fitness function can be defined.

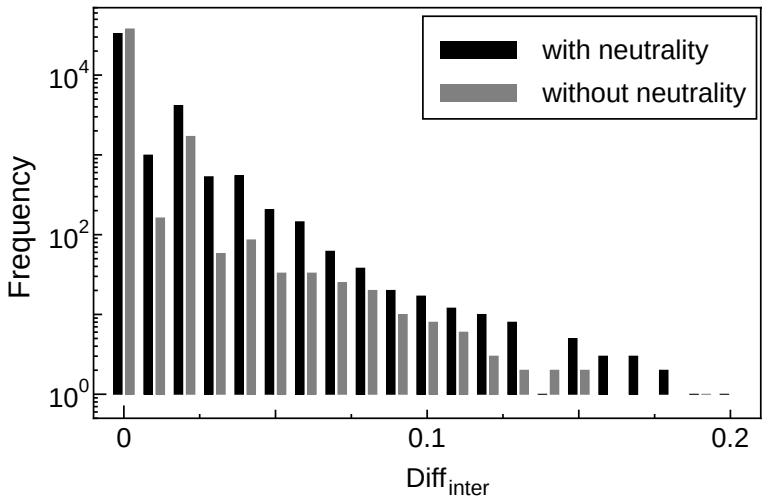


Figure 17: Histogram of Diff_{inter}, where a zero frequency is omitted because of the logarithmic scale.

References

J. C. Avise. Molecular Markers. In *Natural History and Evolution*, pages 16–43. Chapman & Halls, NY, 1994.

H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report 6760 (NLR Memorandum), Naval Research Lab., Washington, D.C., 1990.

H. G. Cobb and J. J. Grefenstette. Genetic algorithms for tracking changing environments. In S. Forrest, editor, *ICGA93: Proceedings of the 5th International Conference on Genetic Algorithms*, pages 523–530, San Mateo, CA, 1993. Morgan Kaufmann.

R. J. Collins and D. R. Jefferson. Selection in massively parallel genetic algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 249–256, San Mateo, CA, 1991. Morgan Kaufmann.

D. Dasgupta and D. R. McGregor. Nonstationary Function Optimization using the Structured Genetic Algorithm. In R. Männer and B. Manderick, editors, *PPSN 2: Proceedings of the 2nd Parallel Problem Solving from Nature*, pages 145–154, Amsterdam, 1992. Elsevier.

J. Eggermont, T. Lenaerts, S. Poyhonen, and A. Termier. Raising the Dead: Extending Evolutionary Algorithms with a Case-Based Memory. In J. Miller, M. Tomassini, P. L. Lanzi, C. Ryan, A. Tettamanzi, and W. B. Langdon, editors, *EuroGP'2001: Proceedings of the 4th European Conference on Genetic Programming*, pages 280–290, Berlin, 2001. Springer. Lecture Notes in Computer Science 2038.

D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ, 2nd edition, 2000.

S. Forrest. Genetic Algorithms: Principles of Natural Selection Applied to Computation. *Science*, 261: 872–878, 1993.

T. Fukuda, K. Mori, and M. Tsukiyama. Parallel Search for Multimodal Function Optimization with Diversity and Learning of Immune Algorithm. In D. Dasgupta, editor, *Artificial Immune Systems and Their Applications*, chapter 11, pages 210–219. Springer, Berlin, 1999.

D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*, pages 41–49, Hillsdale, NJ, 1987. Lawrence Erlbaum.

D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J. J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*, pages 59–68, Hillsdale, NJ, 1987. Lawrence Erlbaum.

J. J. Grefenstette. Genetic algorithms for changing environments. In R. Männer and B. Manderick, editors, *PPSN 2: Proceedings of the 2nd Parallel Problem Solving from Nature*, pages 137–144, Amsterdam, 1992. Elsevier.

B. S. Hadad and C. F. Eick. Supporting polyploidy in genetic algorithms using dominance vectors. In P. J. Angeline, R. G. Reynolds, J. R. McDonnell, and R. Eberhart, editors, *Proceedings of the 6th Annual Conference on Evolutionary Programming*, pages 223–234, Berlin, 1997. Springer. Lecture Notes in Computer Science 1213.

J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.

T. Isokawa, N. Matsui, and H. Nishimura. A Genetic Algorithm Inspired by the Neutral Theory and Its Application to the Formation of Ladder-Network. *Transactions of the Society of Instrument and Control Engineers*, 35(11):1462–1468, 1999. (in Japanese).

M. Kimura. *The Neutral Theory of Molecular Evolution.* Cambridge University Press, Cambridge, 1983.

N. Matsui, T. Isokawa, H. Nishimura, and R. Nagura. Self-Adaptability in Dynamic Formation of Ladder-Network by Genetic Algorithm. In M. H. Hanza, editor, *Proceeding of the 15th IASTED International Conference*, pages 205–208. IASTED/ACTA Press, 1997.

M. Mitchell. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, MA, 1996.

N. Mori, J. Yoshida, H. Tamaki, H. Kita, and Y. Nishikawa. A Thermodynamical Selection Rule for the Genetic Algorithm. In D. B. Fogel, editor, *Proceedings of the 2nd IEEE Conference on Evolutionary Computation*, pages 188–192, Piscataway, NJ, 1995. IEEE Press.

J. Sarma and K. A. D. Jong. The behavior of spatially distributed evolutionary algorithms in non-stationary environments. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 572–578, San Francisco, CA, 1999. Morgan Kaufmann.