Chapter 1
Reconsidering Information and Communications Technology from Life

Hidefumi Sawai

Abstract When we consider the advanced Information and Communication Technology (ICT), it has appeared to be useful to know deeply about life by recent studies.

In this chapter (and the latter half part of Chap. 5), several cases for ICT researchers and practitioners to develop the better ICT will be explained. Here, these cases include:

- *Brain structure and functions* as one of the by-products of biological evolution, and various *information processing models* with the time and space structure derived from brain.
- Genetic algorithm as a model of biological evolution itself, and *evolutionary computation algorithm* as an extended form of it.
- Algorithm as a model of cell metabolism in the early stage of biological evolution.
- *Algorithm* based on a model of *sexual selection*.
- A useful guideline for constructing a future ICT society which can be obtained by the survey results of trend in recent *complex network science* (described in the latter half part of Chap. 5).

Keywords Life and ICT · Brain function · Evolutionary mechanism

H. Sawai (✉)
Kobe Research Laboratories, National Institute of Information and Communications Technology, 588-2, Iwaoka, Nishi-ku, Kobe, 651-2492, Japan
e-mail: sawai@nict.go.jp

Where the world ceases to be the scene of our personal hopes and wishes, where we face it as free beings admiring, asking and observing, there we enter the realm of Art and Science.

—Albert Einstein

1.1 Connection Between Life and ICT

1.1.1 Proximate Factor and Ultimate Factor

The reasons for drawing inspiration from life are self-evident. Take, for example, the brain, the seat of intelligence and awareness. The product of four billion years of sustained evolution, this unparalleled object is most definitely not an overnight creation.

To be "*inspired by life*" and to build intelligent information and communication systems, we must not only investigate the functions of the organisms currently found on Earth, but penetrate deep to discover how the diverse and ingenious functions observed today were acquired over the long course of evolution. In other words, we need to look back at the developmental stages in the evolution of living organisms—an *ultimate factor*—in addition to *proximate factors*, proximate factors being the adaptation of various organisms to their immediate environments. The relationship between proximate factors and ultimate factors is briefly discussed below. No understanding of the historical background and significance of modern life science or cognitive science is complete without an understanding of this relationship.

1.1.2 Nature's Hierarchy

The most beautiful thing we can experience is the mysterious. It is the source of all true art and science. He to whom this emotion is a stranger, who can no longer pause to wonder and stand rapt in awe, is as good as dead: his eyes are closed.

—Albert Einstein

It is believed that the universe was created about 15 billion years ago and the Solar System and the Earth approximately 4.6 billion years ago. Primitive life forms are believed to have appeared around 3.8–4 billion years ago. Chemical evolution during the first 600 million years after the formation of Earth eventually led to the origin of life.

Table 1.1 presents the *hierarchy of nature* and the disciplines and research topics associated with each level of this hierarchy. At the smallest scales are *elementary particles, atoms, and molecules*. The topics discussed in this book that

Table 1.1 Nature's hierarchy

Hierarchical levels	Fields and subjects of research
1. Elementary particles, atoms, and molecules	Nano-bio science, artificial chemistry, molecular communication
2. Genes	Molecular biology, neutral theory of evolution, virus evolution
3. Amino acids	Chemical genetic algorithm/programming (CGA/CGP)
4. Proteins	Protein engineering, molecular communication (motor proteins)
5. Cells	Neurons, molecular communication (Ca ion diffusion)
6. Tissues and organs	Tissue engineering, brain and mind, consciousness
7. Organism	Tissue engineering, brain and mind, consciousness
8. Population	Immune system, ESS[*], evolution of altruistic behavior, multi-agent system
9. Species	Darwinian theory of evolution, Neo-Darwinism, co-evolution
10. Ecosystem	Theory of habitat segregation, mimicry, migration strategy of population
11. Earth	The Gaia hypothesis, environmental problems
12. The solar system, the galaxy, and the universe	Origin of life

[*] *ESS* Evolutionary Stable Strategy

correspond to this level of the hierarchy are *artificial chemistry* in Chap. 3 and *molecular communications* in Chap. 2. The next level of the hierarchy is *genetic*. Section 1.3 introduces genetic algorithms and algorithms based on *genetic duplication*. Beyond this is the *amino-acid level* of this hierarchy; topics associated with this level include *chemical genetic algorithms* (CGAs) and *chemical genetic programming* (CGP), presented in Sect. 1.4. Chapter 2 discusses molecular communication and motor proteins as topics at the *protein level* of the hierarchy.

As issues at the *cellular level*, Sect. 1.2 discusses *neural network modeling* based on the model of neurons and synapses. Note that Ca ion diffusion, a topic discussed in Chap. 2, is also associated with this hierarchical level. At the level of *tissues and organs*, researchers are currently pursuing studies of brain machine interfaces (BMIs) to extract and apply neuronal activity outside the brain as part of efforts intended to investigate *consciousness*, the state emerging from the brain functions. *The Origin of Species* by Darwin is associated with the organism level of the hierarchy. Section 1.3.4 discusses algorithms inspired by the theory of *sexual selection*. At the level of populations and species, perhaps the most characteristic topic is *co-evolution* involving multiple agents; the acquisition of behavioral strategy by multiple agents in CGP discussed in Sect. 1.4.2 is indeed based on the mechanism of co-evolution. At the *ecosystem level*, the topic presented is parallel distributed processing for parameter-free genetic algorithms (presented in Sect. 1.3.2), a model inspired by the migrational strategies of various populations. At the *level of the Earth*, environmental problems such as the mitigation of global warming constitute the most pressing topics. At the level of the *universe*, when we consider that the origin of life is a lifeless molecule, we

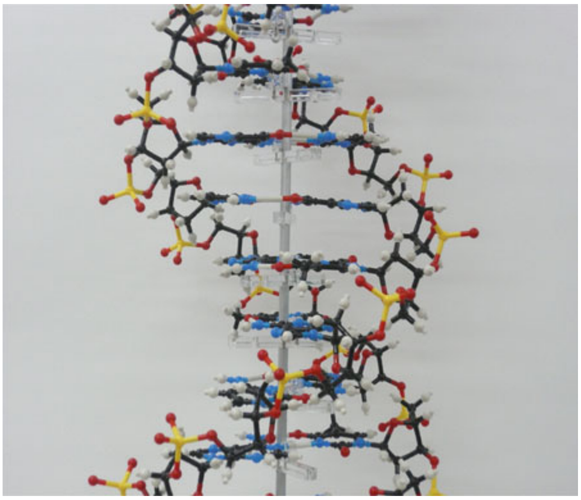


Fig. 1.1 Double helix structure of DNA

understand that the vast scale of the universe is closely linked to small-scale structures at the level of elementary particles, atoms, and molecules. We also consider how what we observe of the universe (*nature's hierarchy*) reflects a mode of observation unique to humans.

A satisfactory *Theory of Everything*, or TOE, that provides a unified explanation for all levels of the hierarchy presented in Table 1.1 has yet to be established. *Micro-scale* theories such as elementary particle theory and quantum mechanics and *macro-scale* theories such as electrodynamics, Newtonian dynamics, and Einstein's general theory of relativity have resolved many of nature's mysteries. But the handling of *mesoscale* problems remain a challenge, and the respective theories remain works in progress, despite the emergence of studies on complexity (complex science) through chaos theory. As is well known, the discovery of DNA's double helix structure (shown in Fig. 1.1) by Watson and Crick in 1953 triggered rapid progress in *molecular biology* and has pushed the discipline to its current prominence.

In the *information sciences*, research has declined in the area of artificial intelligence, applications of which include expert systems that seek to embody human expertise. However, studies continue on artificial neural networks, artificial life, and artificial chemistry, which have come to be regarded as fundamental areas of study. Current chapter and Chap. 3 of this book introduce recent research achievements in this area. While Descartes's formulation of the *mind–body duality* has sequestered physical and spiritual or mental issues as problems in totally distinct dimensions, recent progress in cognitive science and the development of technologies for non-invasive measurement of brain functions such as f-MRI (functional Magnetic Resonance Imaging), MEG (Magneto Encephalography), NIRS (Near-Infrared Spectroscopy), and EEG (Electro Encephalography) have transformed the study of human *consciousness* and subjective perception into subjects of natural science (cognitive science). This is a potentially epochal event in the history of scientific and technological development.

1.2 Hints from Brain Function

1.2.1 Brain Structures and Their Functions

This section presents a summary of the structure and functions of the brain. As shown in Fig. 1.2, the brain is divided into left and right hemispheres. The brain can also be divided into the following units: the cerebrum, consisting of frontal, parietal, temporal, and occipital lobes; the cerebellum; the brain stem; and the spinal cord. Each part has a modular structure consistent with its function. The brain constitutes a system of immense complexity, composed of some 100 billion *neurons* (counting both the cerebrum and cerebellum), with each neuron connected to other neurons by several thousand to tens of thousands of synapses. This complex organ is the product of evolution. Figures 1.3 and 1.4 present the structures of a neuron and a neural network, respectively. As shown in Fig. 1.5, a synapse in essence is the

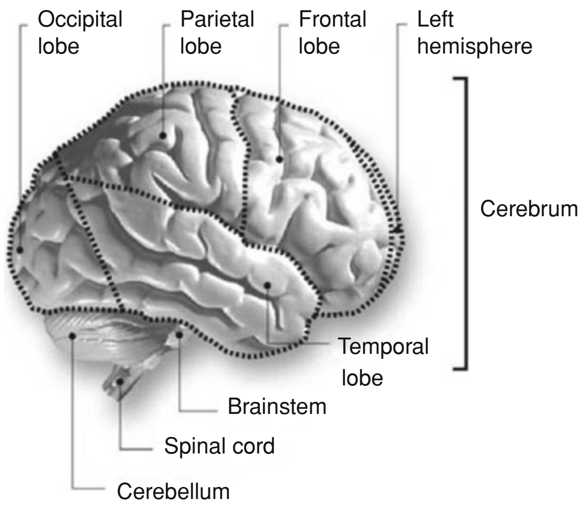


Fig. 1.2 Brain structure

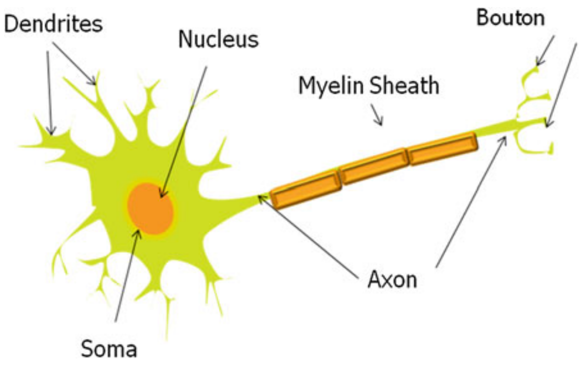


Fig. 1.3 Neuron (nerve cell)

Fig. 1.4 Neural network

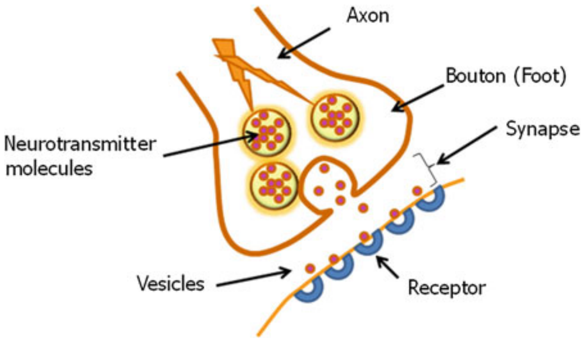


Fig. 1.5 Synapse

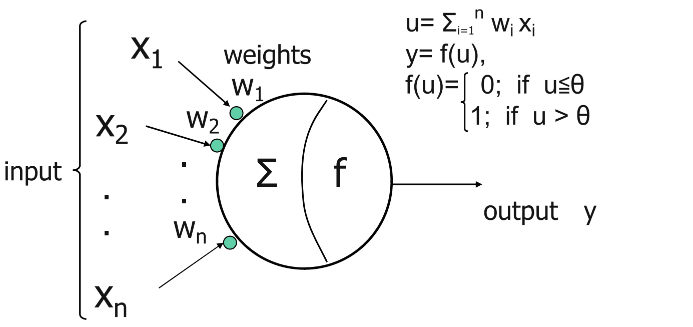
space between two neurons across which electric pulses traveling along an *axon* are relayed to an adjacent *dendrite* via chemical transmitters.

A neural network is an example of an information processing model based on brain function. Presented below are modeling and design methods for *neural network* architectures suitable for speech or image pattern recognition.

Special focus will be given to the feature extraction technique tailored to the temporal structure of speech and the spatial structure (information structure) of images.

1.2.2 Neural Networks Modeling Brain Function

Figure 1.6 is a model of the neuron proposed by McCulloch & Pitts in 1943. The input signal x_i ($i = 1, 2, \ldots, n$) is input to the neuron, with the links representing synapses weighted by w_i. The resulting internal potential u is the sum of these products, or $\Sigma w_i x_i$. The sum is input to the threshold function $y = f(u)$, which is $y = 0$ when the value of u is below a certain threshold value of θ and $y = 1$ when

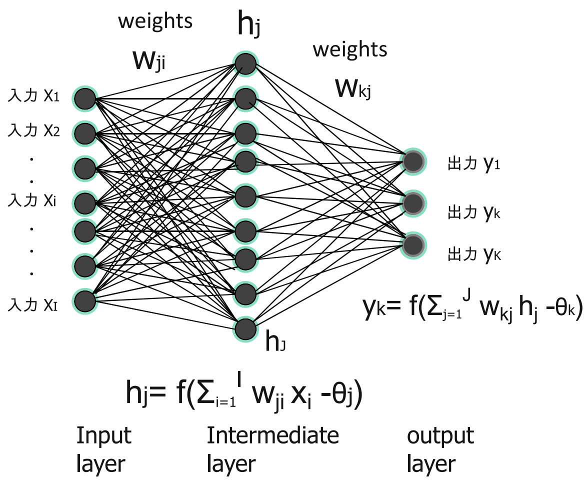


$u = \Sigma_{i=1}^{n} w_i x_i$
$y = f(u),$
$f(u) = \begin{cases} 0; & \text{if } u \leq \theta \\ 1; & \text{if } u > \theta \end{cases}$
input
x_1
x_2
x_n
weights
w_1
w_2
w_n
Σ
f
output y

Input signals x_i (i=1, 2, …, n) are fed into the neuron with the synapse's weights w_i. As a result, the inner potential u becomes the value ∑wi xi. Then, u is input to the threshold function y=f (u) , and the output y becomes 1 with firing when u is greater than a threshold θ, and 0 when u is less then the threshold.

Fig. 1.6 Formal neuron in McCulloch and Pitts [25]

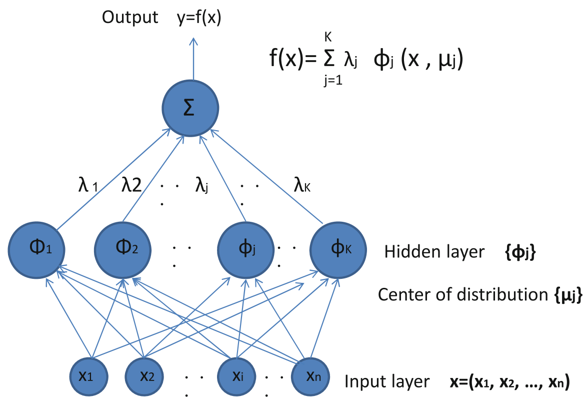
Fig. 1.7 Artificial neural network (ANN) [26]



u is greater than the threshold. Thus, this neuron model adopts a weighted majority logic based on threshold value. The threshold function $y = f(u)$ is generalized into a differentiable sigmoid function $y = 1/\{1 + \exp(-x + \theta)\}$ and applied to the learning rule by *error back-propagation* for the artificial neural network (ANN) shown in Fig. 1.7.

Column 1: Artificial Neural Networks and Machine Learning

Artificial neural networks (ANNs) are information processing machines that model the structure and functions of a brain. However, the structure of an actual brain consists of vast numbers of neurons (a human brain is said



Output y=f(x)
$f(x)=\sum_{j=1}^{K} \lambda_j \phi_j(x, \mu_j)$
Σ
λ1
λ2
λj
λK
Φ1
Φ2
ϕj
ϕK
Hidden layer {ϕj}
Center of distribution {μj}
x1
x2
xi
xn
Input layer x=(x1, x2, ..., xn)

Fig. 1.8 Radial basis function (RBF) networks (reproduced from Ref. [1])

to contain more than 100 billion neurons) interlinked to form a vast neural network structure. A brain is an extraordinarily complex system—a super-complex system. An ANN seeks to abstract the brain's structure and functions to extract the essence and to generate the ability to learn.

ANNs can be classified into two categories, depending on learning method: supervised learning models and unsupervised learning models [1, 2]. Typical examples of supervised learning models include the following:

Rosenblatt's perceptron
Multi-layer perceptron (MLP)

These examples separate and recognize patterns in a data space using hyperplanes. In contrast, the two examples below clusterize patterns by combining basic functions (such as Gaussian functions) called kernel functions.

Radial-basis function (RBF) networks (Fig. 1.8 in this column)
Support vector machines (SVM)

Figure 1.9 shows the differences in pattern classification between MLP and RBF.

Other examples include statistical and probabilistic computational models. Two such models are listed below.

Boltzmann machine
Bayesian network

On the other hand, typical examples of the "unsupervised learning model" include the below.

Kohonen's self-organizing feature map (SOM). Although not an ANN, reinforcement learning models [3], inspired by learned behavior in animals, are machine learning models intermediate between supervised and unsupervised learning models. Two examples are given below.

Time difference (TD) learning
Q-learning

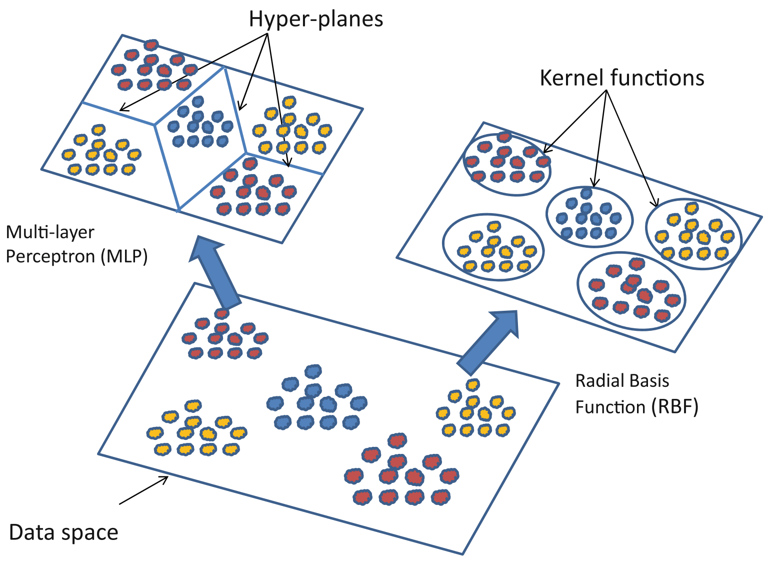


Fig. 1.9 Differences in pattern classification between MLP and RBF (Reproduced from Ref. [2])

1.2.3 Time-Delay Neural Networks Suitable for Processing Sequential Information and Their Expansion [4–8]

Figure 1.10 shows a unit used in a *time-delay neural network* (*TDNN*) [4]. The architecture shown here was proposed to process time series data—for example, speech. To the left is the input unit, which is linked to the host output unit via synapse W, which can be a non-time-delay concatenation or a time-delay concatenation with a delay of $D_1, D_2, \ldots, D_n$. This architecture is suitable for processing speech signal patterns that have temporal structures. Weighted summation ($\Sigma w_{ij}\, x_i$) is performed on the input signals, and the resulting value is sent to the sigmoid function $f(x) = 1/\{1 + \exp(-x)\}$ and the result output.

Figure 1.11 shows the architecture of a TDNN designed to distinguish the voiced plosives /b, d, g/ in the Japanese language [4]. From bottom to top are the input layer, hidden layer 1, hidden layer 2, and output layer. The input layer consists of a total of 240 units, or 15 and 16 units in each of the x- and y-axis directions.

The x- and y-axis correspond to the time- and frequency-axis, respectively. The time–frequency spectrum (sound spectrum) produced by frequency analysis performed every 10 ms is input to the input layer. This time–frequency spectrum is shifted by a time window of 30 ms (3 units) and after being multiplied by the weighting factors of synapses having time-delay, it is linked to the host unit in hidden layer 1. As in the input layer, the total of 40 units in hidden layer 1, five units in the x-axis (time-axis) direction, and the eight units in the y-axis (frequency-axis) direction are linked to hidden layer 2 with the time-delay. In hidden

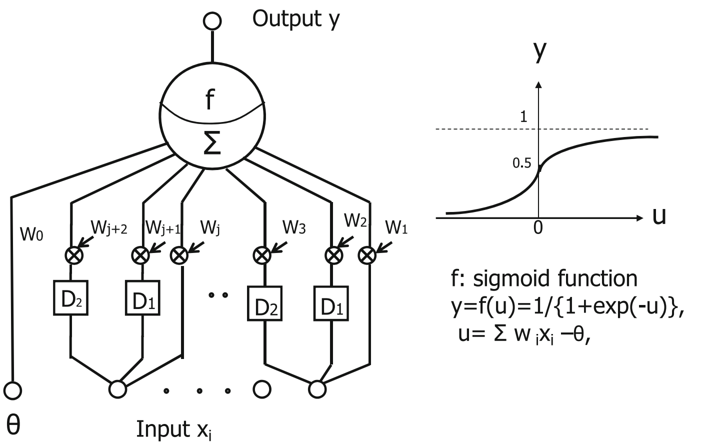
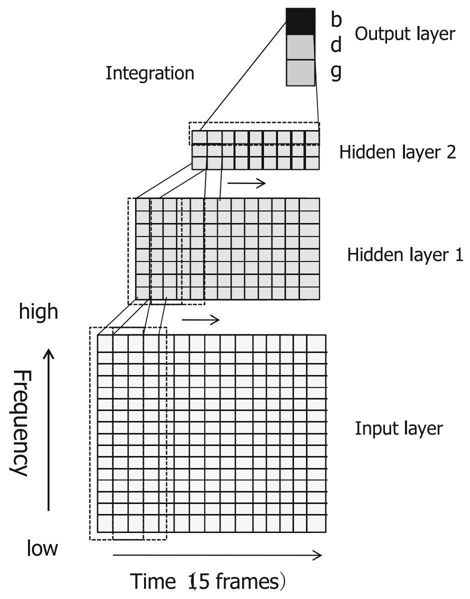


Fig. 1.10 Unit of time-delay neural network (TDNN)

Fig. 1.11 Time-delay neural network (TDNN) architecture



layer 2, each of the nine units in the *x*-axis direction is assigned to one of three categories /b, d, g/, and concatenated to the output unit with the time-delay. This TDNN is trained by error-propagation. The results of discrimination testing for input phonemes not used for training indicate that the TDNN achieves a speaker-dependent recognition rate of approximately 98–99%, representing a reduction in misrecognition rate to approximately 1/4 that of the conventional HMM (Hidden Markov Model) widely used for speech recognition applications. (HMM is associated with a recognition rate of 91–97%.) By expanding the phonemic category in

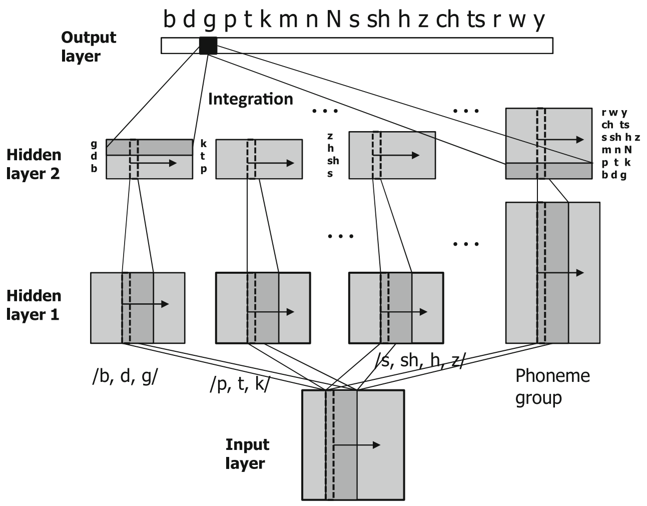


Fig. 1.12 Modular architecture of all consonant network

a similar manner, we should be able to build a TDNN for all Japanese phoneme groups.

Figure 1.12 shows the modular architecture of a TDNN capable of discriminating the 18 consonants of the Japanese language [5]. As the figure shows, the 18 consonants are divided into six groups—the voiced plosives /b, d, g/, unvoiced plosives /p, t, k/, nasals /m, n, N/, fricatives /s, sh, h, z/, affricates /ch, ts/, and liquids and semi-vowels /r, w, y/. A TDNN capable of distinguishing between the six phoneme groups is designed so that the results of discrimination processing within each phoneme group and those from group discrimination processing can be linked in the output layer.

Figure 1.13 shows a TDNN system with a TDNN capable of distinguishing between the Japanese language vowels /a, i, u, e, o/ and a TDNN capable of distinguishing between the six consonant groups and the vowel group added to the TDNN in Fig. 1.12, in addition to a speaker-dependent recognition TDNN expanded to operate as a speaker-independent recognition network [6]. As the figure shows, this TDNN is a large-scale network with a 3D structure. Its scale enables speech recognition of all Japanese consonants and vowels in speaker-independent mode. Adding a discrimination unit (Q) for the presence/lack of voice (voiced/unvoiced) makes it possible to automatically recognize Japanese phonemes (called phoneme spotting) simply by scanning vocalized speech along the temporal direction. Readers are referred to Ref. [6] for more information on the recognition performance of these networks.

Figure 1.14 outlines a different approach that also results in a speaker-adaptive neural network [7]. The three lowermost layers are neural networks that perform speech spectra mapping to adapt the speech of an unknown speaker to that of a standard speaker used in training. This approach—providing the mapping to the

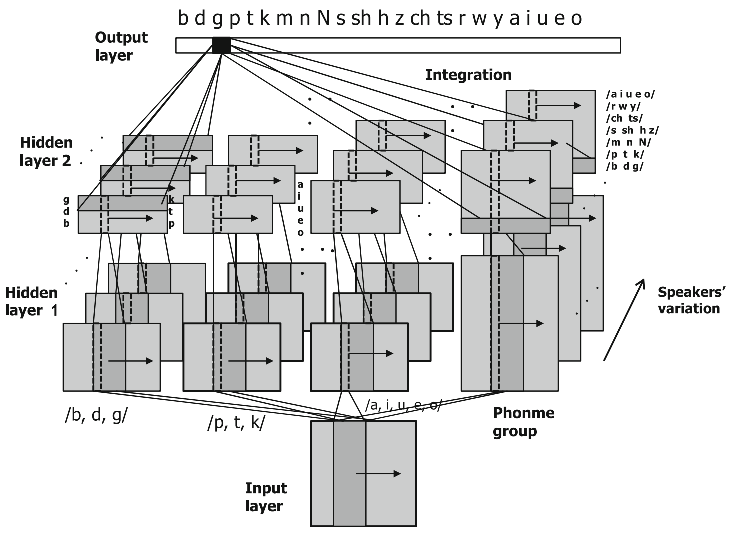


Fig. 1.13 Large-scale TDNN architecture for speaker-independent recognition

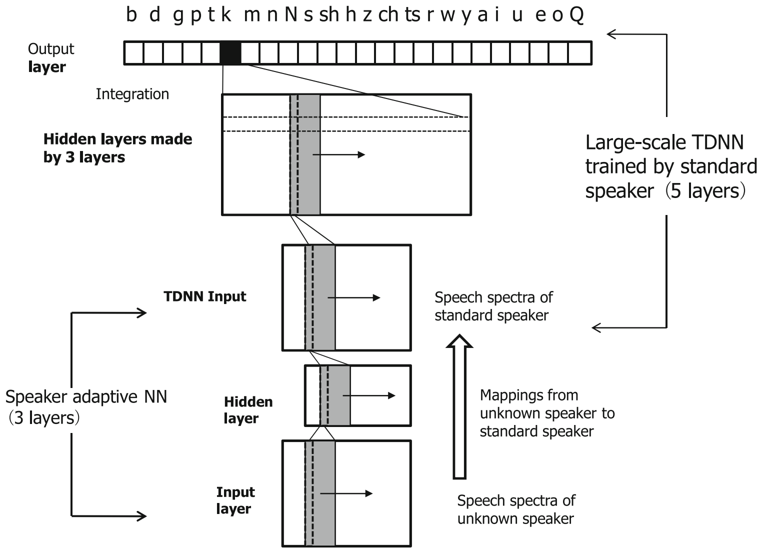

b d g p t k m n N s sh h z ch ts r w y a i u e o Q
Output layer
Integration
Hidden layers made by 3 layers
Large-scale TDNN trained by standard speaker (5 layers)
TDNN Input
Speech spectra of standard speaker
Speaker adaptive NN (3 layers)
Hidden layer
Mappings from unknown speaker to standard speaker
Input layer
Speech spectra of unknown speaker

Fig. 1.14 Large-scale TDNN architecture with speaker-adaptive neural network

TDNN in advance—may represent a highly effective strategy for applying a TDNN trained to a standard speaker to recognize the speech of unknown speakers.

Figure 1.15 shows an expanded architecture model of a new TDNN designed to absorb temporal and frequency fluctuations in speech. Time and frequency windows are installed in the input layer, and extractions of feature quantities (the feature quantity associated with temporal fluctuations and those associated with

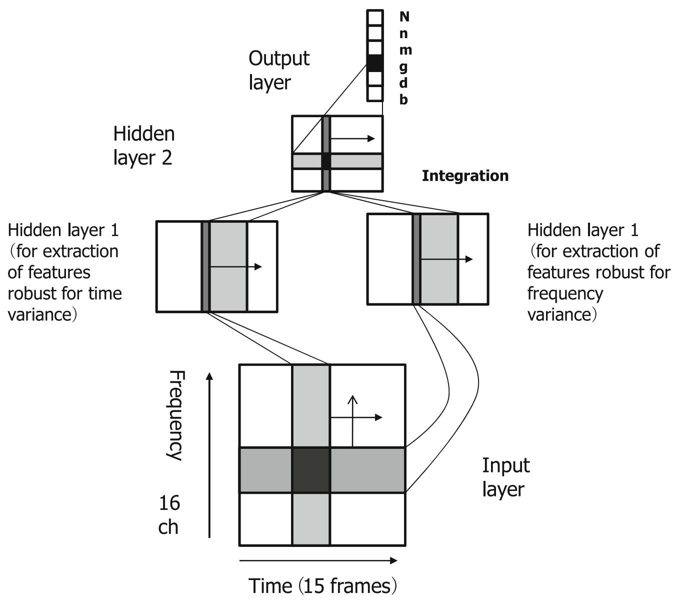

N
n
m
g
d
b
Output layer
Hidden layer 2
Integration
Hidden layer 1 (for extraction of features robust for time variance)
Hidden layer 1 (for extraction of features robust for frequency variance)
Frequency
16 ch
Input layer
Time (15 frames)

Fig. 1.15 Frequency-time-shift-invariant TDNN architecture

frequency fluctuations) are integrated in hidden layer 1. The signal is ready for output by the output layer after integration in hidden layer 2. The figure is an expanded TDNN designed to make precise distinctions within a category consisting of six phonemes associated with high misrecognition rates: the voiced plosives /b, d, g/ and nasals /m, n, N/.

Figure 1.16 is a neural network with block windows inspired by the "neocognitron", extensively investigated for applications in handwritten letter recognition. As with handwritten letter recognition, it is necessary to absorb fluctuations in the absorption along the time (*x*-axis) and frequency (*y*-axis) directions in speech pattern recognition. Thus, we can build an architecture that promotes such absorption by installing block-shaped windows in the lower layers so that the feature quantities can be integrated in succession as connections are made upward. The recognition performance of these neural networks is discussed in Ref. [8].

1.2.4 Expansion of Time-Delay Neural Networks to Rotation-Invariant Pattern Recognition [9]

The architecture in Fig. 1.17 is an expanded NN having *axial symmetry* created by expanding the translation invariance of TDNN to rotational invariance. The synapse weighting factors having parallel assignments from the bottom to upper layers retain the original values acquired through training. This architecture means that if

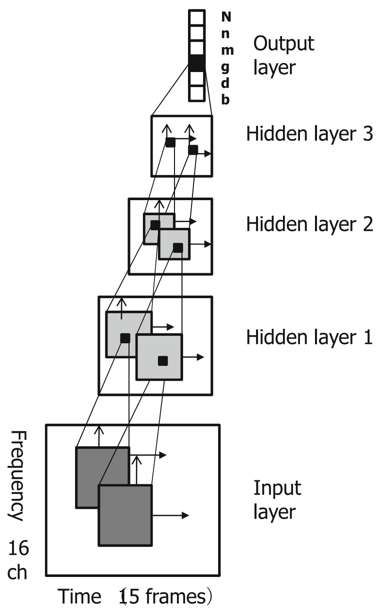


Fig. 1.16 Block-windowed neural network (BWNN) architecture

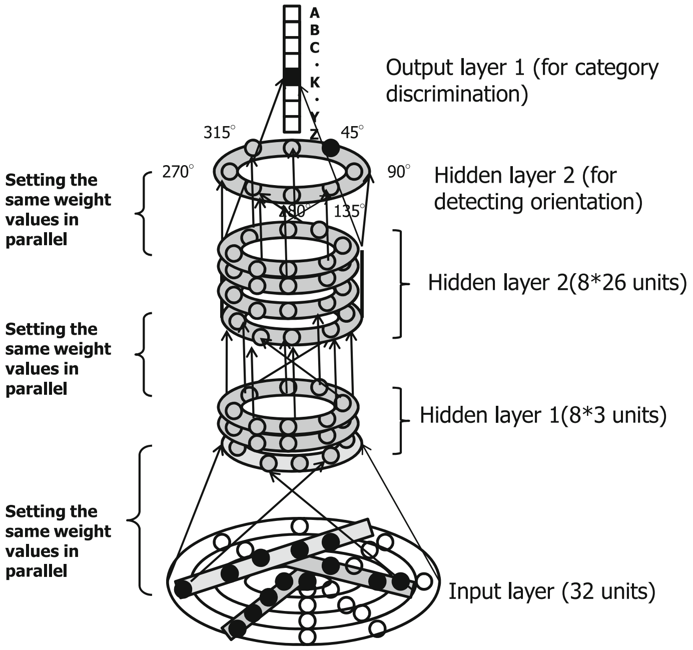

A
B
C
K
Y
Z
Output layer 1 (for category discrimination)
315°
45°
270°
90°
180°
135°
Hidden layer 2 (for detecting orientation)
Setting the same weight values in parallel
Hidden layer 2(8*26 units)
Setting the same weight values in parallel
Hidden layer 1(8*3 units)
Setting the same weight values in parallel
Input layer (32 units)

Fig. 1.17 Axially symmetric neural network architecture

the synapse weighting factors can be learned using error back-propagation for every pattern of the letter pattern class categories (A–Z) input at a given rotational position (0 deg., for example), the neural network can correctly recognize the class category and rotation angle when an arbitrary class category is input at any angle. The figure shows a neural network architecture capable of recognizing the 26 letters of the alphabet at rotation angle intervals of 45°. From bottom to top are the input layer, hidden layer 1, hidden layer 2, output layer 2 (for rotation angle recognition), and output layer 1 (for class category recognition). As this neural network has an axially symmetric structure, proper execution of the recognition function requires careful alignment of the center of the pattern of the image fed to the input layer.

1.3 Theory of Evolution and Information Processing Model

An information processing model inspired by biological evolution is a computational algorithm based on genetic algorithms and evolutionary computation. Table 1.2 is a compilation of the evolutionary computation algorithms introduced in this and subsequent sections.

In 1859, Charles Darwin released his book, *The Origin of Species* [10], which dealt with the genetics and evolution of organisms. Prompted by the book, John Holland advanced the idea of genetic algorithms (GA) in the 1960s [11]. Genetic algorithms have been applied to problems in numerous fields, including functional optimization, combinatorial optimization, and parameter optimization in machine design. In recent years, genetic algorithms have been integrated with other techniques such as Evolutionary Strategies (ES) by Rechenberg and Evolutionary Programming (EP) by L. Fogel to form a research discipline known as Evolutionary Computation (EC).

Column 2: Genetic Algorithms [11, 28, 29]

Genetic algorithms offer certain advantages over the algorithms used in other information processing systems: they do not need to assume differentiability

Table 1.2 Some examples of evolutionary computation algorithms

Life phenomenon, theory of evolution (from micro to macro)	Evolutionary computation algorithm
Disparity theory of evolution (Furusawa) [13]	Parameter-free GA [12]
Theory of gene-duplication (Ohno) [16]	Gene duplicating GA [15]
Codon to amino acid translation	Chemical GA [22], Chemical GP [23]
Sexual selection (Darwin) [10]	Evolutionary computation based on sexual selection [19]
Ecosystem	Hierarchical parallel distributed GA [14]

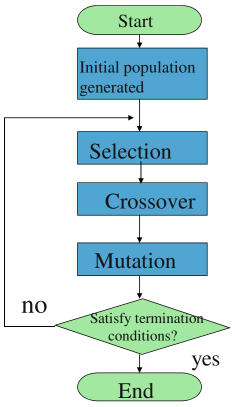


Fig. 1.18 Flow chart of GA (SGA)

of evaluation functions (functions can be indifferentiable); genetic manipulations such as crossovers and mutations readily circumvent the tendency to become mired in local solutions; and they offer a superior capacity for global and local search. Here, we describe the most basic of all such algorithms: the Simple Genetic Algorithm (SGA). Figure 1.18 shows a flowchart for SGA.

We begin by preparing an initial population consisting of *n* individuals. Based o n roulette-wheel selection, we select *n* individuals at a probability proportional to the fitness *fi* of the individual *i*. Two individuals randomly selected from the *n* individuals undergo "crossover" at an arbitrary crossove r-point on a "chromosome." This is called a one-point crossover. The ratio of individuals experiencing crossover to the overall population is known as the crossover rate. In subsequent "mutations," the genes of randomly selected individuals are modified at a specified mutation rate. In a binary coded genotype, the bit-flip from 1 to 0 or from 0 to 1 is performed at the probability corresponding to this mutation rate. The operations up to "selection," "crossover," and "mutation" correspond to a single GA generation. If the fitness of the population is sufficient for environmental evaluations (that is, if the quality of a given solution for a given problem is adequate), the process ends. If not, the process returns to "selection," repeats the series of genetic manipulations described above for several generations, in which process the population evolves, possibly leading to a satisfactory solution population.

In general, SGA is used as an evolutionary method to acquire solutions for relatively easy problems. As the problems become more difficult, SGA has increasing difficulty in efficiently acquiring optimal solutions. For this reason, various other evolutionary computational methods have been proposed. Examples include the parameter-free genetic algorithm (PfGA) [30] and the chemical genetic algorithm (CGA) [22] described in this document. Figures 1.19 and 1.20 in this column show, respectively, the multi-point crossover and block-wise mutation used in PfGA.

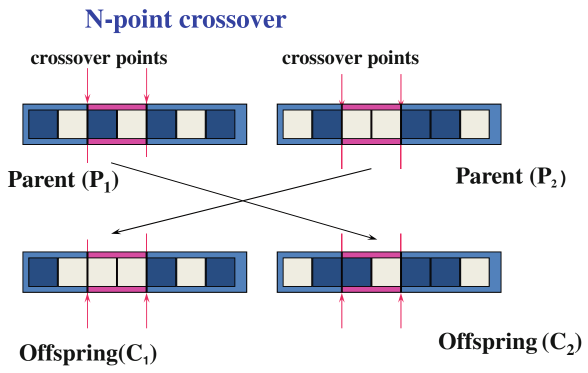


Fig. 1.19 Multi-point crossover (n-point crossover) in the parameter-free genetic algorithm (PfGA)

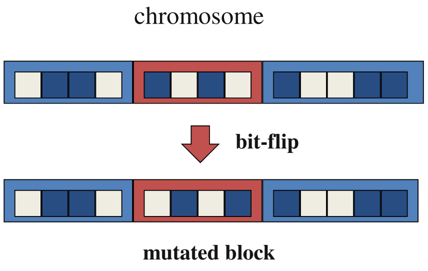


Fig. 1.20 Block-wise mutation in the parameter-free genetic algorithm (PfGA)

The migration of individuals from one local population to another makes it possible to obtain an optimum solution fairly rapidly by parallel distributed processing while expanding population diversity. This document describes several methods of parallel distributed processing based on the PfGA.

1.3.1 *Parameter-Free Genetic Algorithms [12] Based on Disparity Theory of Evolution [13]*

This section discusses a new algorithm, the parameter-free genetic algorithm (PfGA), which requires no initial setting of genetic parameters such as initial population size, crossover rate, or mutation rate. The PfGA was inspired by and builds on the *disparity theory of evolution* proposed by Furusawa et al. [13], which itself is based on mutations in the double strands of DNA (Figs. 1.21, 1.22). According to disparity theory, when the double strands of DNA unwind and a copy of each is created, a difference emerges in the rate of mutation between leading and lagging strands. This is because the direction of replication in the former is the same as the direction of unwinding, while the direction of replication in the latter is in the opposite direction. While mutations are generally rare in the leading

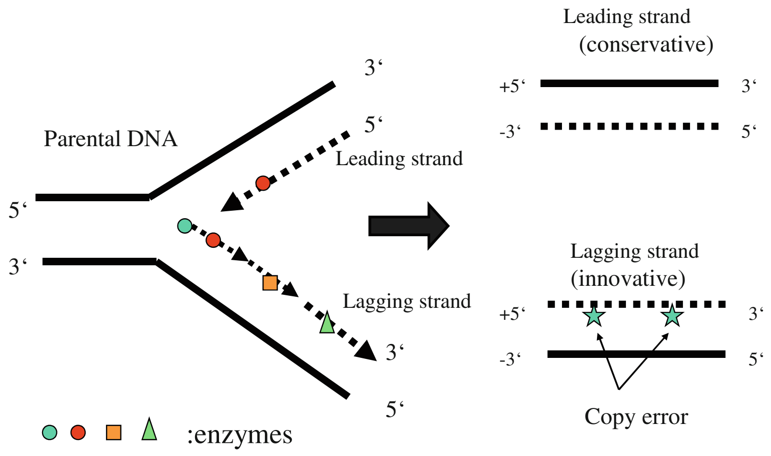


Fig. 1.21 Hypothesis based on the disparity theory of evolution (Reprinted from K. Wada, “Evolution of Digital Life”, Fig. 7., page 73, Iwanami Science Library 11, Iwanami Shoten, 1994.)

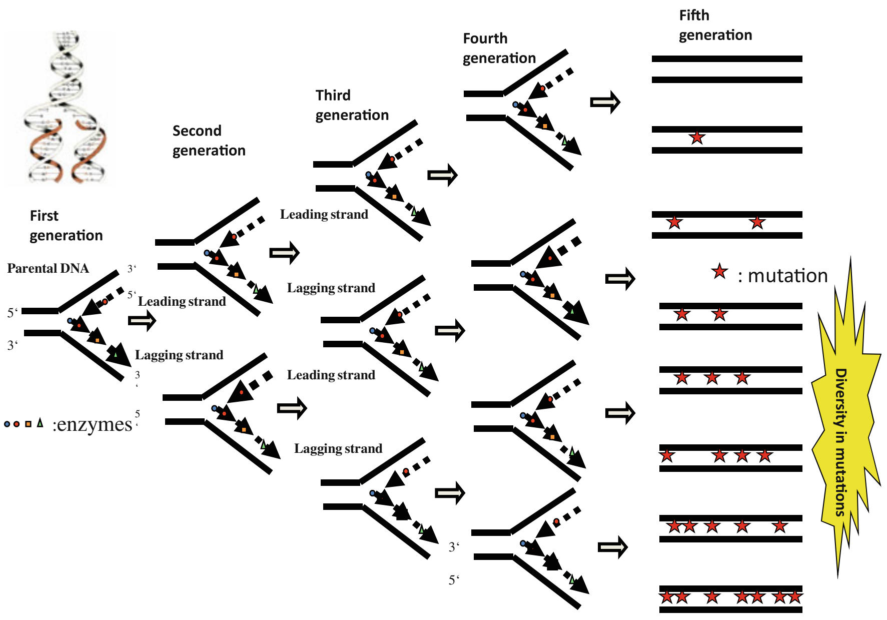


Fig. 1.22 Emergence of diversity according to the disparity theory of evolution (recreated by the author based on Fig. 7 from Ref. [27])

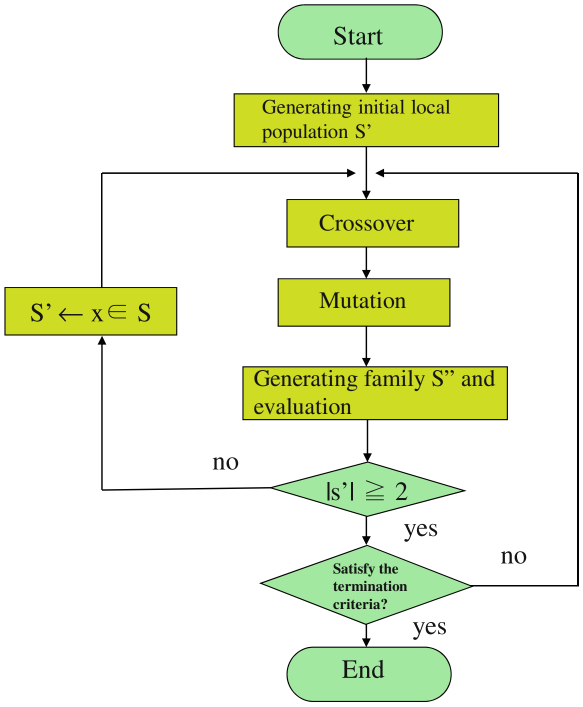
strand (conservative), the lagging strand displays comparatively high mutation rates (innovative). Disparities in replication errors accumulate through crossovers and mutations over generations, creating DNA diversity within a single population via DNA strands experiencing little or no mutation and strands with accumulated mutations. The former type promotes the stability of the population, while the latter promotes flexibility. In the case of PfGA, the former corresponds to the optimized individual at a specific point in time, while the latter corresponds to the offspring produced by crossover and mutation. Thus, under the disparity theory of evolution, the mechanism by which diversity is retained while maintaining a balance between exploitation (localized search) and exploration (global search) can be understood as a balance between genetic *stability* and *flexibility*.

In PfGA, the population is regarded as a set of all possible solutions in the whole search space. In this whole search space S, a local sub-population S' is set. Two individuals are selected from this sub-population S' as parents. The parents are subjected to crossover and mutation to generate a family (S'') of four, with two offspring. The fitness of the four individuals within this family is then evaluated to select or eliminate individuals to evolve local population S' and to execute a search for the solution.

Below are the steps in the basic algorithm for the PfGA (See Fig. 1.23).

1. An individual is randomly extracted from S and is regarded to be initial local population S'.
2. An individual is randomly extracted from S and added to the local population S'.
3. Two individuals are randomly extracted from local population S' for use as parent 1 (P_1) and parent 2 (P_2) in the multi-point crossover.

Fig. 1.23 Flowchart of parameter-free GA (PfGA)


Start
Generating initial local population S'
Crossover
Mutation
Generating family S" and evaluation
|s'| ≧ 2
no
yes
S' ← x∈ S
Satisfy the termination criteria?
no
yes
End

4. Of the two individuals generated by the crossover, one is randomly selected and inverse mutation is applied at a random number of points at random positions.
5. Selection and elimination is performed for a total of four individuals (referred to as a family) consisting of the two generated offspring (C_1 and C_2) and the two parents (P_1 and P_2), by selecting either one or three members of the family to be returned to local population S', based on the calculated fitness.
6. If the local population size $|S'| \geq 2$, then return to step 3; if $|S'| = 1$, then return to step 2 and repeat the cycle.

Multi-point crossover is used as the crossover mode in PfGA. In multi-point crossover, both the number and positions of crossover points are determined randomly, and crossover is executed between chromosomes of two different individuals. Mutations use the erroneous copy of chromosomes generated during the crossover. Thus, of the two offspring produced, one is randomly selected and a partial inversion of the gene sequence carried out to create mutations at a random number of points at random positions. Here, of the two offspring produced by the crossover, one is left untouched by mutation to allow one offspring to retain at least a portion of the parents' traits. In this manner, the PfGA is implemented to execute genetic manipulation based on random numbers and to minimize *ad hoc* choices.

For selection and elimination, to retain the diversity of the local population while maintaining a balance between global search (exploration) and local search (exploitation), the fitness of the four members of the family (S'') are compared and selections made in the manner presented in the following four cases as shown in the left plot of Fig. 1.24. Local population S' is evolved while dynamically maintaining a balance between global and local search and concurrently changing

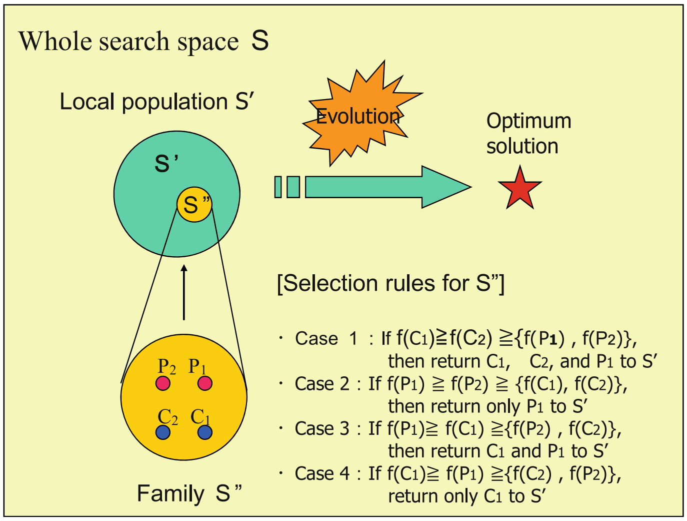

Whole search space S
Local population S'
Evolution
Optimum solution
S'
S"
P2 P1
C2 C1
Family S"
[Selection rules for S"]
· Case 1 : If f(C1)≧f(C2) ≧{f(P1) , f(P2)}, then return C1, C2, and P1 to S'
· Case 2 : If f(P1) ≧ f(P2) ≧ {f(C1), f(C2)}, then return only P1 to S'
· Case 3 : If f(P1)≧ f(C1) ≧{f(P2) , f(C2)}, then return C1 and P1 to S'
· Case 4 : If f(C1)≧ f(P1) ≧{f(C2) , f(P2)}, return only C1 to S'

Fig. 1.24 Population and selection rule in PfGA

the size of the local population S' based on an implicit set of rules for switching between cases 1–4, depending on the relative superiority of the fitness of family members. This feature improved the search efficiency of PfGA over other GAs of fixed population size, since it eliminates the need to perform unproductive searches. In addition, the best individual among the four family members is always returned to the local population S'. Thus, the family may be said to be adopting an elite-preserving strategy: The algorithm guarantees the retention of the best individual at a given point in time while simultaneously performing an active search over a very wide (neighborhood) space. If a better individual than the currently best individual is generated, the center of the search transfers to that offspring; if not, the current best individual is retained. This avoids fitness degradations during the course of evolution.

Column 3: The Disparity Theory of Evolution [13, 31 – 33]

Mutation is the driving force of evolution. DNA mutations are now known to be distributed unevenly—to exhibit disparity. The two strands forming the double helix of the DNA replicate in opposite directions. While one strand (the leading strand) is continuously replicated by an enzyme known as polymerase, the other strand (the lagging strand) is replicated by several enzymes in a complex process of connecting partially replicated fragments (called Okazaki fragments), resulting in a relatively high error (mutation) rate during replication. The error rate for the lagging strand is ten to a hundred times (or even more) than that of the leading strand. In other words, a disparity results wherein the rates of mutations in the replicated double helix strands are unequal. One strand may have many mutations, the other very few.

Based on what is known as the parity model, mutations were previously believed to be distributed evenly in a population. However, an excessive mutation rate will tend to lead to the death of the entire population. The parity model distributes mutations evenly in the DNA of the offspring. In contrast, the disparity model produces both DNA without mutations (or wildtype) and DNA with more mutations than predicted by the parity model. In short, populations with diverse mutations appear, while the wildtype is maintained. Figure 1.22 shows the emergence of diversity according to the disparity theory of evolution. It shows expanding diversity in a population based on the accumulation of mutations generation after generation.

These patterns hold not just for lower organisms such as *Escherichia coli*, but for more complex organisms. Therefore, it is possible to accelerate the rate of evolution by increasing the mutation rate while avoiding the risk of extinction. Dr. Mitsuru Furusawa, who proposed the disparity theory of evolution, calls it the “creation of diversity with guaranteed capital." The disparity theory of evolution features both "conservation," which reduces the risk of extinction by guaranteeing the wildtype after generations, and "innovation," which generates the diversity of individuals needed for evolution. The authors proposed the parameter-free genetic algorithm (PfGA) [30] as an information processing model that accelerates evolution while incorporating a dynamic balance between conservation and innovation.

1.3.2 Expansion of Parameter-Free Genetic Algorithm to Parallel Distributed Processing Techniques [14]

This section describes techniques for *parallel distributed processing* related to the parameter-free genetic algorithm (PfGA) inspired by ecosystems. In general, the main objective of parallel processing in any processing, including GAs, is to increase processing speed. However, we can dramatically enhance the efficiency of search problem processing based on a GA by introducing interactions between individuals by migration, rather than simply dividing up the task. In the case of a coarse-grained parallel GA, the local population is treated as the processing unit, and individuals are migrated between local populations at appropriate frequencies. In a fine-grained GA, the neighborhood of a given individual is treated as the processing unit, and overlaps are set among neighbors. The former is frequently referred to as the island model, wherein a single local population constitutes the deme of a single species. We use this as the model.

1.3.2.1 Two Parallel Processing Architectures

Two types of parallel processing architectures are used: the *uniformly-distributed type* and the *master–slave type*. The uniformly-distributed type corresponds to a situation in which all local populations have the same role and local population monitoring functions are absent.

On the other hand, in the master–slave type, a master local population is equipped with a function for monitoring the processing of other local populations, called slaves. Figure 1.25 shows the uniformly-distributed-type PfGA architecture. From the whole search space S, an N number of local populations $S_i^{'}$ $(i = 1, \ldots, N)$ is derived; in each local population $S_i^{'}$, there exists a family $(\mathrm{S}_i^{''})$ that performs PfGA crossovers, mutations, and selection. Migration of individuals may occur between any of the local populations.

Figure 1.26 shows the master–slave architecture. $S_0^{'}$ is the master local population, while $S_i^{'}$ $(i = 1, \ldots, N)$ are the slave local populations. The master $S_0^{'}$ consistently (or at constant intervals) seeks to identify the best individual in all slave populations.

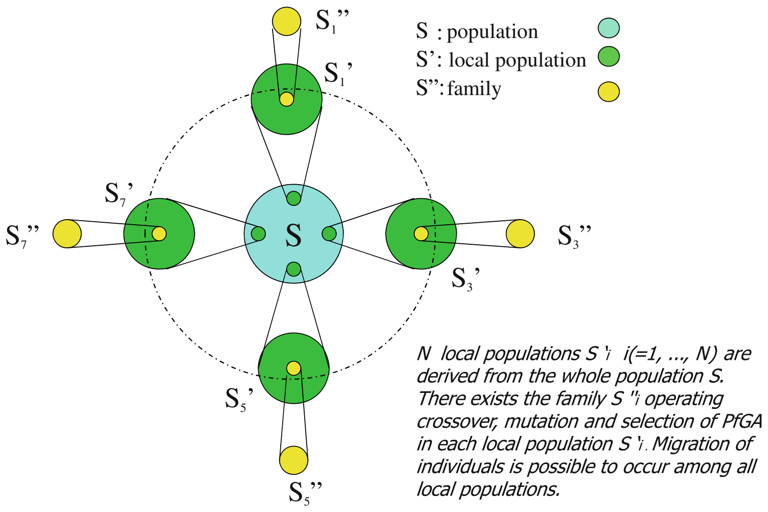

S₁''
S₁'
S : population
S': local population
S'': family
S₇'
S₇''
S
S₃''
S₃'
S₅'
S₅''
N local populations S'ᵢ i(=1, ..., N) are derived from the whole population S. There exists the family S''ᵢ operating crossover, mutation and selection of PfGA in each local population S'ᵢ. Migration of individuals is possible to occur among all local populations.

Fig. 1.25 Uniformly-distributed (UD) type architecture

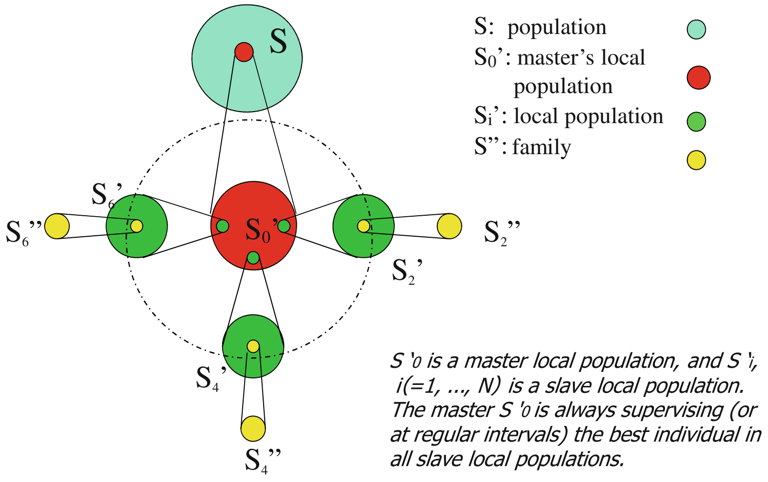
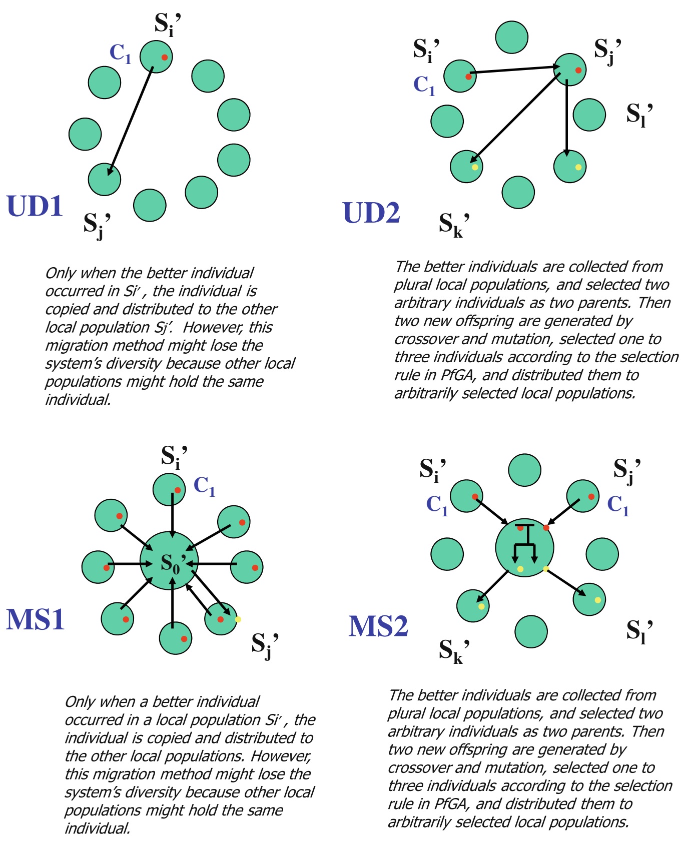

S
S: population
S₀': master's local population
Sᵢ': local population
S'': family
S₆'
S₆''
S₀'
S₂''
S₂'
S₄'
S₄''
S'₀ is a master local population, and S'ᵢ, i(=1, ..., N) is a slave local population. The master S'₀ is always supervising (or at regular intervals) the best individual in all slave local populations.

Fig. 1.26 Mater–slave (MS) type architecture

1.3.2.2 Two Migration Methods

Several migration strategies may come to mind, but here we adopt the following two. In the first, an individual in a given local population is copied and distributed to other local populations only when a good individual emerges. This is called the direct migration type. The disadvantage of this method is that the same individual is retained by other local populations after migration, threatening system diversity.



Only when the better individual occurred in Si′, the individual is copied and distributed to the other local population Sj′. However, this migration method might lose the system's diversity because other local populations might hold the same individual.

The better individuals are collected from plural local populations, and selected two arbitrary individuals as two parents. Then two new offspring are generated by crossover and mutation, selected one to three individuals according to the selection rule in PfGA, and distributed them to arbitrarily selected local populations.

Only when a better individual occurred in a local population Si′, the individual is copied and distributed to the other local populations. However, this migration method might lose the system's diversity because other local populations might hold the same individual.

The better individuals are collected from plural local populations, and selected two arbitrary individuals as two parents. Then two new offspring are generated by crossover and mutation, selected one to three individuals according to the selection rule in PfGA, and distributed them to arbitrarily selected local populations.

Fig. 1.27 Migration strategy selection method in UD: direct migration type(UD1) and hierarchical migration type (UD2). Migration strategy selection method in MS: Direct migration type (MS1) and hierarchical migration type (MS2)

We adopt the second strategy, in which good individuals are gathered from multiple local populations and two individuals are arbitrarily selected to be the new parents. They bear two offspring (by crossover and mutation), and one to three members of the family are distributed according to the selection rules in PfGA to arbitrarily selected local populations. Since this migration method implements meta-level PfGA operations from the perspective of the local population, it is called the hierarchical migration type (Fig. 1.27).

1.3.2.3 Performance Results

We performed parallel processing for the four different combinations of parallel architecture (uniformly-distributed/master–slave) and migration type (direct/hierarchical) to investigate the effects of migration.

An evaluation of search performance showed that increasing the number of local populations *reduced* the number of evaluations required before success by *a ratio of 1/N* (N: the number of local populations). Among the four types of architecture/migration method examined, search performance, from high to low, fell into the following sequence: UD1, MS1, MS2, UD2. We confirmed that increasing the number of local populations increases the chance of success through the effects of migration relative to serial processing.

1.3.3 Information Processing Model Based on Gene Duplication [15]

This section discusses the gene-duplicating GA (GDGA) inspired by the theory of *gene-duplication* proposed by Susumu Ohno in the 1970s. The theory of gene-duplication claims that the replication and reuse of gene fragments in the evolution of all organisms, from viruses and plants to animals, fuels a drive toward life-forms of ever-growing sophistication.

Ohno distills this phenomenon into a succinct formula: "*a single creation and one hundred plagiarisms*." [16] Gene duplication is assumed to occur by unequal crossover between chromatids on a single chromosome, unequal crossover between homologous chromosomes during the meiotic process, and partial repetitive duplication of DNA. Inspired by this gene duplication mechanism, we have proposed four gene duplication models: gene concatenating (Fig. 1.28), gene-prolonging (Fig. 1.29), gene coupling (Fig. 1.30), and extended gene coupling (Fig. 1.31).

This computational method is based on a *divide-and-conquer* GA in which a given problem is broken down into sub-problems, which are then combined to obtain the solution for the original problem. Each individual concatenates the partial solution that each had accumulated up to that point in time, then the

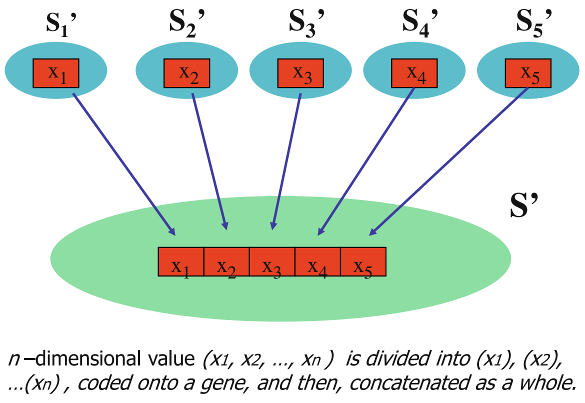
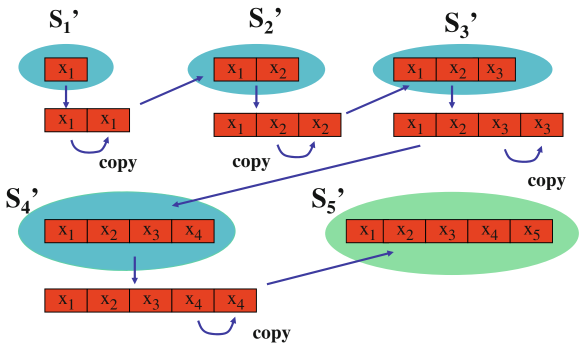

S1'
S2'
S3'
S4'
S5'
x1
x2
x3
x4
x5
S'
x1 x2 x3 x4 x5
n–dimensional value (x1, x2, ..., xn) is divided into (x1), (x2), ...(xn), coded onto a gene, and then, concatenated as a whole.

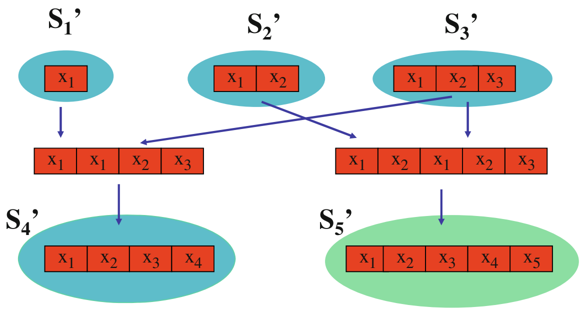
Fig. 1.28 Gene duplication in a gene-concatenating model

Fig. 1.29 Gene duplication in a gene-prolonging model



Low dimensional genes are copied step by step such as (x1), (x1, x2), ...(x1, x2, ..., xn), migrated to other sub-populations, and then prolonged by one dimension by one dimension.

Fig. 1.30 Gene duplication in a gene-coupling model

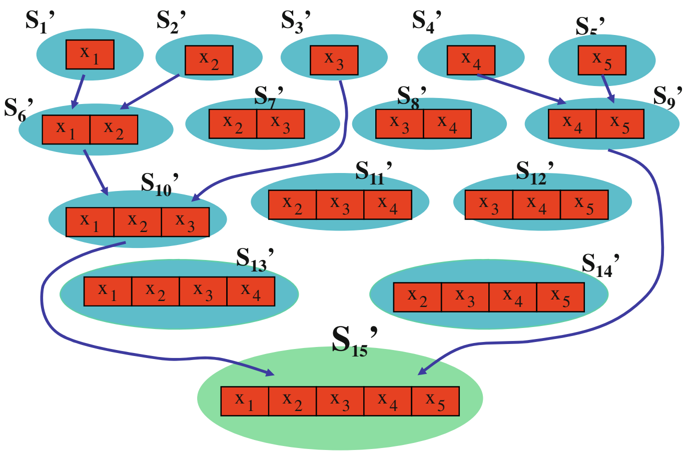


Coupling (x1) with (x1, x2, x3) becomes (x1, x2, x3, x4), and coupling (x1, x2) with (x1, x2, x3) becomes (x1, x2, x3, x4, x5).

individual migrates between the local populations. This strategy makes it possible to obtain the solution more efficiently and quickly.

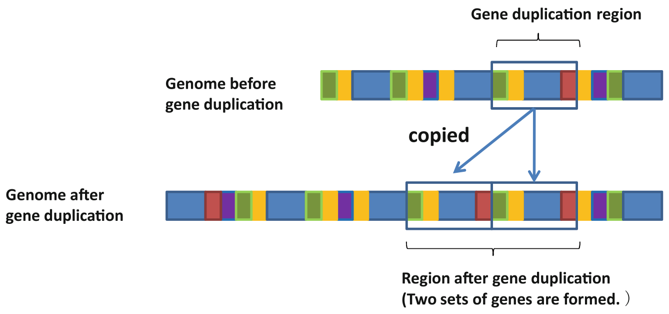
Gene duplication, a powerful tool in solving multi-dimensional functional optimization problems, is a genetic operator applicable to individuals of different gene lengths. It is applied by first coding variables to genes for each sub-dimension, then setting the fitness function for each subspace and running the GA to obtain the (quasi) optimum solution. By concatenating individuals owning the gene corresponding to this (quasi) optimum solution, we can solve an optimization problem in higher dimensions. This algorithm is implemented by having individuals with differing gene lengths migrate between local populations. Overall, the algorithm performs crossover, mutation, and selection within a local population; between separate local populations, it performs duplication and migration, in that sequence.

In a simulation evaluating the search performance of the four types, we used a functional optimization benchmark problem to compare success rates, probabilities, and convergence performance in obtaining the optimum solution. We found that increasing the number of migrating individuals increases population diversity,



By considering the locus of each gene corresponding to each dimension, coupling (x1) with (x2) becomes (x1, x2), and coupling (x1, x2) with (x3) becomes (x1, x2, x3).

Fig. 1.31 Gene duplication in an extended gene-coupling model



Gene duplication region
Genome before gene duplication
copied
Genome after gene duplication
Region after gene duplication (Two sets of genes are formed.)

The gene duplication region is the region created when living creatures copy genes during the process of evolution. This is believed to be central to the evolution of higher organisms from lower organisms.

Fig. 1.32 Gene duplication

thereby confirming improvements in convergence performance and the effectiveness of this computation method.

Column 4: Theory of Gene Duplication [16–18, 34]

Gene duplication refers to the presence of two or more identical genes in the genome or set of genes (Fig. 1.32). Even if one set of duplicate genes undergoes a debilitating mutation that seriously affects its functions, the functions of the living creature proceed unimpaired if the other continues to function normally.

According to “Seimei no Tanjo to Shinka (The Birth and Evolution of Life)” by Susumu Ohno [17], “evolution by gene duplication” [16] can be summarized as follows:

Since most existing genes have already evolved to a state of near-perfection with regard to function, evolution is impossible if natural selection monitors all genes. True evolution can occur only by acquiring genes with unprecedented new functions. For this to occur, the existing active sites must change. This raises a problem: Natural selection will not permit this. To evade monitoring by natural selection, one must make copies of oneself and make the monitoring focus on one of the copies. This is the only way to acquire such unprecedented new functions.

Gene duplication is one of evolution’s mechanisms. The relationship between the original gene and the replicated gene corresponds to the relationship between the leading strand and the lagging strand described in Column 3: The Disparity Theory of Evolution. It also demonstrates the importance of the exquisite balance between the conservation strategy based on guaranteed capital (i.e., the status quo) and an innovation strategy entailing radical risks (since mutations very rarely generate a preferable characteristic) in an overall evolution strategy that avoids extinction. The gene duplicated GA (GDGA) described in the main text, an information processing model modeling the gene replication process of the theory of gene duplication, implements various duplication processes while migrating them between local populations.

1.3.4 Information Processing Model Based on Sexual Selection [19]

The theory of *sexual selection* seeks in part to explain the extensive differences between the phenotype and behavior of the two sexes in certain sexually reproducing organisms. Certain well-known examples include competition between males (generating antlers) and female preferences (generating peacock plumage) [20, 21]. Numerous hypotheses based on female preference seek to explain the evolution of traits that appear disadvantageous from the perspective of natural selection.

Two famous hypotheses are the *runaway hypothesis* and the *excellent gene hypothesis*. The former assumes that female preferences within a population are always biased and that male traits are always variable due to random mutations. In such cases, males with the preferred traits are more likely to father large numbers of offspring, regardless of fitness in terms of natural selection, thereby conferring an indirect advantage to the gene manifesting that trait and resulting in the rise of the male trait preferred by females. The latter hypothesis claims that the male shows off the quality of his genes even if doing so comes at some cost, with the result that the trait and the female preference for that trait grow more common. However, the actual processes at work in sexual selection remain unclear.

The genetic assignment of sex to individuals has led to a state in which an individual of one of the sexes effectively observes and chooses the phenotype of the other, and we will focus on the effect of such asymmetric roles of males and females on the process of evolution.

We will also examine the role of mutation (the simplest transition rule) as a genetic operator that drives organisms toward the direction of evolution based on the fitness landscape. A model is assumed in which a mutation rate is coded in the gene as a parameter for mutation and in which fitness varies.

We will focus in particular on the interaction between sexual selection and mutation.

Working from these perspectives, we introduce an evolutionary computational model in which mutation rates are encoded in the gene and also account for sex and sexual selection. Based on this model, we investigate how mutation rates become self-adaptive within a population and how the direction of evolution is determined in phenotypic space.

Column 5: Sexual Selection [35 – 38]

Male and female individuals of the same species inhabiting a similar natural environment can take significantly different forms. This cannot be explained by natural selection and presented a riddle that puzzled Darwin himself. He focused on the competition between males to win females. In nature, males must compete to gain access to females, a major factor leading to the evolution of characteristics like the splendid horns worn by stags. On the other hand, the beautiful plumage of a peacock (male) is clearly unlikely to have developed to aid in combat between males. This characteristic is believed to help females choose a mate in assessments of courtship displays. Darwin believed females selected males with more beautiful plumage in the male population, resulting in males in the next generation with even more striking plumage. In this manner, in a process called sexual selection, female preference and competition between males would eventually lead to sexual dimorphism, or differences in characteristic between males and females. Although the process whereby sexual selection leads to sexual dimorphism is similar to natural selection, additional energy and resources are needed to create and maintain the horns of the stag or the plumage of the peacock. These characteristics may confer reproductive advantages that offset their disadvantages with regard to natural selection.

1.3.4.1 Constructing a Computational Model

How does an asymmetric relationship between the sexes whereby one sex observes and selects the phenotype of the other affect mutation rates? We propose a model based on sexual reproduction with its own mutation rate encoded into the genes. For the sake of convenience, the observing and observed sex, respectively, are regarded as female and male.

A real-valued genetic algorithm (real-valued GA) is used as the evolutionary computational model. The methods of genetic recombination are chromosomal exchange between individuals and isotropic mutation of each gene.

Sexual selection will focus on relative phenotypic value. (Example: a strong preference for taller individuals or individuals of a certain stronger coloring [e.g., bluer].) In such modeling schemes, the direction of the transition of next-generation males in phenotypic space is determined by the direction of female preferences.

1.3.4.2 Individual Phenotype

Each individual is assigned a sex and with two broad phenotypic categories, *trait* and *preference*. Traits are expressed in both sexes and determine the individual's fitness. In contrast, preferences are expressed only by females and act as a mechanism by which males are assessed and selected. Preferences do not affect natural selection. Traits and preferences are represented in phenotypic space by trait vector $t = \left(x_t^1, x_t^2, \ldots, x_t^n\right)$ and preference vector $p = \left(x_p^1, x_p^2, \ldots, x_p^n\right)$ in n-dimension Euclidean space.

Sexually reproducing organisms are diploid in nature, but here, given the emphasis on interactions due to preference rather than the mode of reproduction itself, the model assumes that the genotype of each individual is haploid for the sake of simplicity, and an individual will have three kinds of chromosomes: sex chromosome, preference chromosome, and a trait chromosome (Fig. 1.33).

The sex gene is coded in a single bit and the other genes are encoded as real values. However, the preference gene in the male is considered not expressed, creating a buffer for male preference and sustaining diversity in preference. Furthermore, we place the gene encoding for the mutation rate (σ) for the trait and

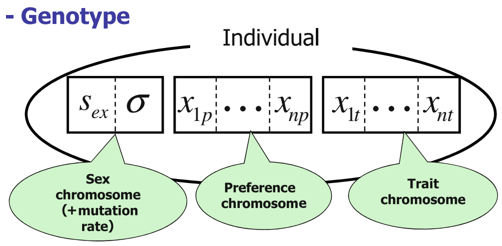


Fig. 1.33 Proposed model of sexual selection

preference genes on the sex chromosome, making the expressed mutation rate dependent on sex.

1.3.4.3 Natural Selection

A known hummingbird species displays sexual dimorphism in beak morphology, with the males and females of the species feeding on flowers of different shapes. This constitutes habitat segregation in the form of resource partitioning. The male and female members of this species can be considered to have followed different paths in natural selection.

Natural selection is posited to operate separately on the sexes. This renders a constant sex ratio while allowing the sexes to generate different traits (sex difference).

1.3.4.4 Sexual Selection

Under sexual selection, a female chooses a male to form a pair, and the offspring produced are one male and one female to maintain the constant sex ratio. All females will always be part of a pair at least once, while males are allowed to pair only when selected by a female as a preferred male. In short, this population is polygamous.

(1) The female i observes M numbers of males at random as potential mates. The average trait vector t_i^0 of the observed male population is calculated, and the relative trait vector $t_{ij} = t_j - t_i^0$ of male j $(j = 1, 2, \ldots M)$ is used as the selection target.
(2) The difference in direction θ_{ij} between t_{ij} and the female preference vector p_i is calculated. The strength of preference is defined as given by $\cos(\theta_{ij})$. The male having a trait vector in the direction closer to the direction of the female preference vector will be strongly favored.
(3) The most preferred male is selected deterministically.

By using the relative trait vector as the selection target, we can search for the direction toward which the male population should shift in phenotypic space—or the direction of evolution—based on the direction of the female preference vector.

In sexual selection, males preferred by more females (i.e., attractive males) gain the advantage, and selection works directly on the male, manifesting as the number of offspring in the next generation. While females in this model are exempt from the direct operation of sexual selection, through genetic exchange with preferred males, the offspring of females having a genetic preference for males with advantages in terms of natural and sexual selection is likely to have the advantage in the next generation. This means sexual selection works indirectly to the advantage of females having genes for such preferences.

1.3.4.5 Genetic Manipulation

Two types of chromosomal exchange and mutation are used as methods of genetic manipulation. With respect to interactions between mutation rate and sexual selection, parent chromosomes are exchanged at a constant probability when parent genes are copied to produce offspring.

Mutations are accomplished by imparting perturbations that follow the normal distribution function $N(0,\ \sigma)$ on the offspring trait and preference genes $x_t^i, x_p^i (\imath = 1, 2, \ldots, n)$, respectively.

Here, the standard deviation σ corresponds to the mutation rate, which is varied adaptively by gene encoding.

$$\sigma = \sigma + \delta \quad \delta \sim \mathrm{N}(0, \sigma_0)$$

Note here that the standard deviation σ_0 of the normal distribution function of the perturbation imparted to σ is constant.

1.3.4.6 Steps in Simulation

(1) Population is initialized at a sex ratio of 1:1.
(2) The following procedure is repeated until the termination condition is satisfied:
    (a) Natural selection
    (b) Sexual selection
    (c) Genetic manipulation

1.3.4.7 Problem

Traits and preferences, respectively, are represented using two-dimensional vectors $t = (x_t,\ y_t)$ and $p = (x_p,\ y_p)$ in examining the following functional maximization problem.

$$\mathrm{Max}\,\mathrm{f}(t) = x_t - 2\sin(\pi x_t) - 0.001 y_t^2 \exp(x_t)$$

The initial population is positioned near the point of origin, while each individual's fitness is determined by trait alone. In the above equation, local optimum solutions may be found on the line $y_t = 0$ with period 2. The problem, then, is to find a way to balance the search for the local optimum solution and the escape from the local optimum solution.

The effects of the third term become large with increasing x_t as the search progresses, and a slight transition of y_t from 0 results in a precipitous decline in fitness, rendering the search more difficult. Although there is no upper limit to the function, GA restricted to isotropic mutation alone places a practical limit on the search. This extremely simple model is an example of how adaptive acquisition of

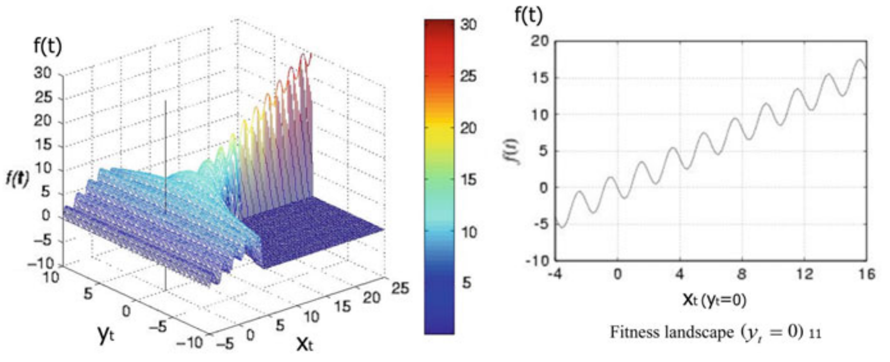
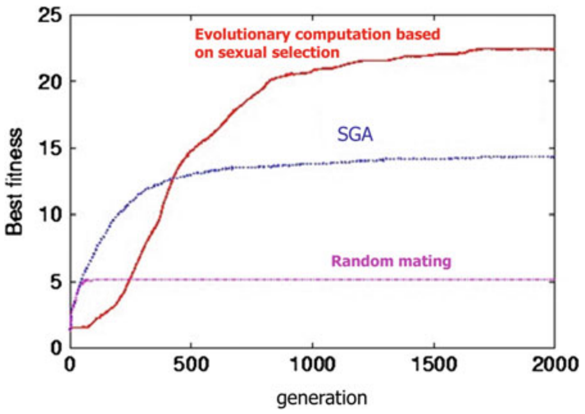


Fig. 1.34 Validation of the proposed model with an illustrative problem

Fig. 1.35 Experimental results



a direction advantageous to evolution (positive x_t direction) concurrent with a search for the optimum solution makes it possible to search as efficiently as possible. (Fig. 1.34)

Natural selection is executed by performing a roulette-wheel selection based on g, where $g = \exp(\alpha f\ (t))$ (α: scaling rate) when $f(t)$ is fitness. This is selected to make the search progress more smoothly when the exponential term in the above equation starts to have a strong effect on the latter part of the search and when the average fitness of the population has decreased exponentially.

We use preference vector $p = (x_p, y_p)$ as the unit vector to normalize the results after each mutation. This means that the preference selection of a male by a female is based solely on the direction of the trait.

1.3.4.8 Results of Experiment

As we can see from Fig. 1.35, the search in the proposed method progressed the most among the three types of strategies (random mating, SGA, and proposed method), indicating the effects of sexual selection on the search. We see no apparent differences in mutation rates between the sexes for the results of random mating. Under the proposed method, as the search proceeded, the male mutation rate surpassed the mutation rate for females, confirming that female preference triggers unequal mutation rates.

1.3.4.9 Search Process

Under the present method, when the search process becomes trapped in a local optimum solution, a runaway situation between the male trait and female preference results that intermittently drives explosive evolution out of the equilibrium state. As the search progresses, differences appear in average mutation rates between male and female populations and a division of roles arises. The sex exercising choice (female) carries out a conservative search with low mutation rates and the chosen sex (male) carries out an innovative search. This results in only the male population performing the search when escaping from the local optimum solution, while females dedicate themselves to maintaining present conditions and standing by to move on to better solutions through chromosomal exchange only after they have been found by the male population. In many sexually reproducing organisms, the production processes differ for reproductive cells between the sexes, and sperm cells have higher mutation rates than ova. The similarity between the characteristics of organisms and the proposed method is quite interesting. The advantages of balancing an innovative and conservative search during the search process through the acquisition of various mutation rates have been discussed in relation to the Neo-Darwinian algorithm by Wada et al. [13]. Since the error frequency varies between the two strands in DNA, they proposed a model in which the mutation rate varies within a single individual, thereby permitting a wide range of searches. This realizes a broad range of mutation rates within the population, allowing them to perform an extensive search in problems involving a high level of risk while maintaining present conditions.

1.4 Information Processing Based on the Modeling of Cells in Early Stage of Evolution

1.4.1 Chemical Genetic Algorithm (CGA) [22]

The mechanism of cell metabolism emerged over an astoundingly lengthy evolutionary process. Modeling this process should make it possible to search efficiently for an optimum solution by techniques totally different from conventional methods. This section discusses the *chemical genetic algorithm* (*CGA*), used for solving difficult problems by dynamically converting them into simpler

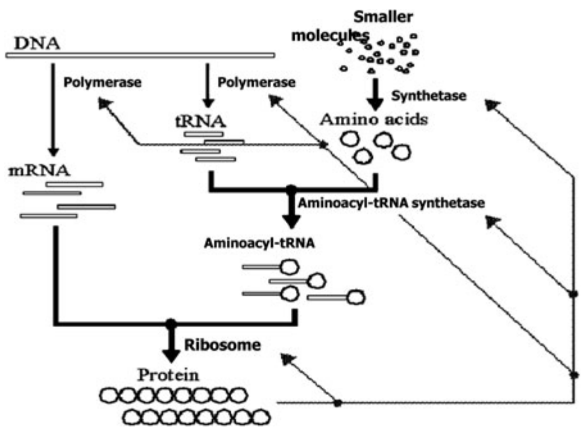


Fig. 1.36 Biochemical reaction for translating genetic information in a cell

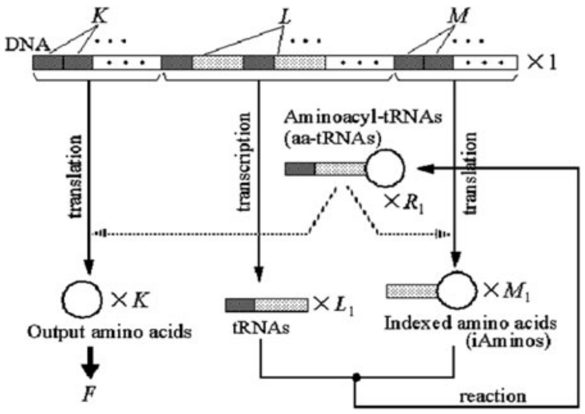


Fig. 1.37 A cell structure used in the CGA

problems—in short, by dynamically changing mapping from the genotype to the phenotype, as inspired by the mechanism of cell metabolism in the early stages of the evolutionary process. The section will also discuss *chemical genetic programming* (*CGP*), in which this solution method is applied to the evolution of programming, and introduce the problem of symbolic regression in artificial intelligence and describe the results of its application to the acquisition of multi-agent behavioral strategy.

1.4.1.1 Mechanisms of Cell Metabolism Generated in the Early Stages of the Evolutionary Process

In the early stages of cell evolution, cells are believed to have acquired their present metabolic processes by dynamically changing the mapping behavior from genotype to phenotype mapping (Fig. 1.36). Figure 1.37 presents a model based on this mechanism.

1.4.1.2 CGA Generation Cycle (See Fig. 1.38)

The steps in the CGA generation cycle are given below:

1. Initialization: First, we prepare a number, *N*, of cells having the structure presented in Fig. 1.37. In the initial state, no cell possesses aminoacyl tRNA (aa-tRNA), tRNA, or outputs amino acids. However, they do have random DNA strands and amino values.
2. Chemical reaction: The following 4-step reaction takes place in all cells: transcription, tRNA-amino acid reaction, translation into internal amino acid, and translation into output amino acid. In the several generations of the early stage, this reaction produces new tRNA and aa-tRNA, and their sizes grow. Within the next few generations, we exceed the size of the molecular pool size.
3. Selection: The fitness of the cell is calculated based on the output amino acid, and cells marking high fitness are selected by roulette-wheel selection. The selected cells are regenerated, and the complete internal information (DNA, 3 molecular pools) of each cell is copied to the daughter cell.
4. DNA mutation: As in normal GA, point mutations of genes are performed.
5. DNA crossover and molecular exchange between cells: Gene crossovers occur as in normal GA, and half the molecules are exchanged between two cells.
6. Calculation of fitness of the cell population: If the termination conditions are satisfied, the computation is complete. If not, return to step 2.

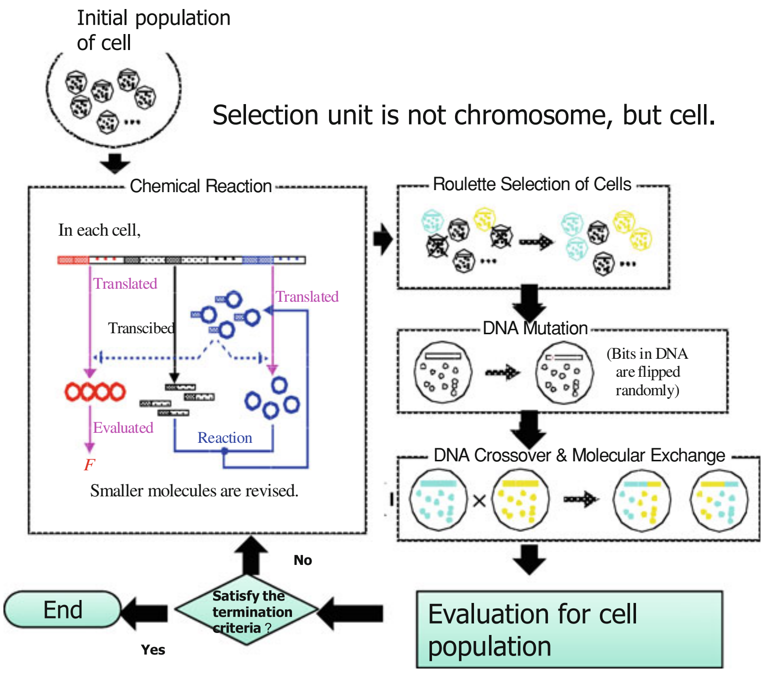

Initial population of cell
Selection unit is not chromosome, but cell.
Chemical Reaction
In each cell,
Translated
Transcibed
Translated
Evaluated
F
Reaction
Smaller molecules are revised.
Roulette Selection of Cells
DNA Mutation
(Bits in DNA are flipped randomly)
DNA Crossover & Molecular Exchange
No
Satisfy the termination criteria ?
End
Yes
Evaluation for cell population

Fig. 1.38 Entire algorithm of CGA

1.4.1.3 GA Evolvability (See Fig. 1.39)

By converting the "ragged fitness landscape" as seen in the genotype space presented on the left side of Fig. 1.39a into a smoothed landscape on the right side (Fig. 1.39b), we can improve *evolvability*.

Three types of deception problems (Types I, II, and III) and 2 benchmark functions were used to validate the search performance of CGA. SGA (simple GA) and PfGA (Parameter-free GA) are used for comparisons. In the simple Type I, $F(x)$ assumes the maximum value (optimum value) of 1 when $x_k = 1$ in all dimensions of $k(k = 1, \ldots, K)$. Type II is an intermediate complex type, and $F(x)$ assumes the maximum value of 1 only when, in each dimension k, $f_k(x)$ randomly takes a maximum at $x_k = 0$ or $x_k = 1$. In Type III, $F(x)$ assumes the maximum value of 1 only when $f_k(x)$ assumes a maximum of 1 at $x_k = \alpha_k$ (α_k is a uniformly random number between 0 and 1) in each dimension of k. As can be seen from Fig. 1.40, the ratio of the probability of $f(x)$ assuming a maximum (optimum value) of 1 in each dimension to the probability of taking the localized optimum value of 0.8 is 1:4. Thus, the probability of $f(x)$ taking optimum values at all dimensions $(k = 1, \ldots, K)$ is $(1/5)^K$. Types I and II constitute special cases of Type III (Table. 1.3).

1.4.1.4 Results of Analysis

Figures 1.41 and 1.42 present the evolution of the function $F(x)$ of CGA and SGA, respectively. The dispersion of the function values is large for CGA. In contrast, the dispersion is small for SGA. The optimum value of CGA (CGA best) is 0.8 or

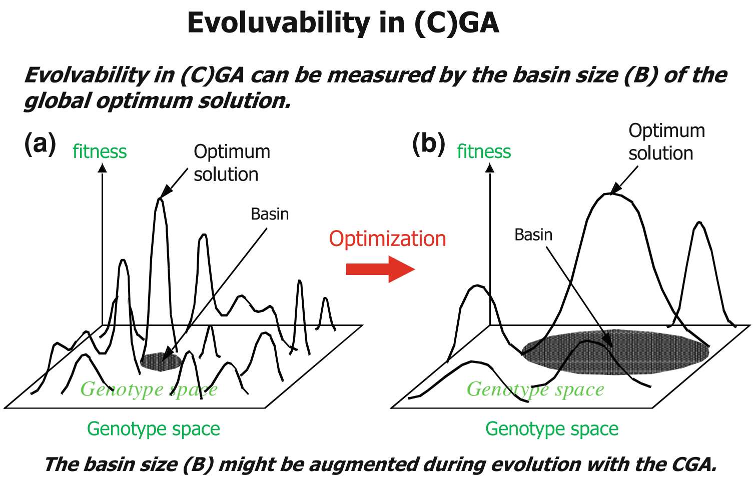
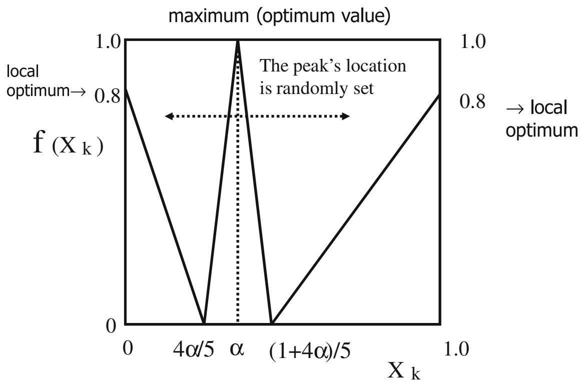

Evoluvability in (C)GA
Evolvability in (C)GA can be measured by the basin size (B) of the global optimum solution.
(a)
fitness
Optimum solution
Basin
Genotype space
Genotype space
Optimization
(b)
fitness
Optimum solution
Basin
Genotype space
Genotype space
The basin size (B) might be augmented during evolution with the CGA.

Fig. 1.39 Evolvability in C(GA)

Table 1.3 Three deceptive Problems

Type	Name	Features
I	Simple type	*f*(x) becomes maximum (optimum) when $x_k = 1$ for all dimensions $k(k = 1, 2, \ldots, K)$. This type is a special case of type III.
II	Intermediate type	*f*(x) becomes maximum (optimum) only when $f_k(x)$ becomes 1 at x_k = 1 or 0 randomly for all dimensions $k(k = 1, 2, \ldots, K)$. This type is a special case of type III.
III	Complex type	*f*(x) becomes maximum (optimum) only when all $f_k(x)$ become 1 at $x_k = \alpha_k$ (where α_k are different uniform random values between 1 or 0) for all dimensions $k(k = 1, 2, \ldots, K)$. As the ratio that $f_k(x)$ takes the maximum value of 1 and the value of 0.8 for each dimension, is 1–4, the probability that takes the optimum value 1 is $(1/5)^K$. This type includes the type I and II as its special cases.



The "deceptive problem" is easily trapped by a local optimum, and cannot reach the global optimum on purpose using a usual naïve search algorithm. In type III, the peak value 1 at *Xk=αk* ,(*k*=1, ..., *K)* is global optimum, while the value 0.8 at *Xk=0,1,* (*k*=1, ..., *K)* is local optimum *(i.e., it is deceived as the optimum value).* The function F(X) to be optimized is defined as the following equation:

$$F(X)=\{(1/K)\,\Sigma\, f_k(x)\}^{\beta} \quad , \text{where} \quad K=5, \text{ or } 10, \beta=5 .$$

Fig. 1.40 Complex deceptive problem (type III)

higher and attains the status of the optimum solution. On the other hand, the optimum value of SGA fails to surpass 0.8 and is trapped in a local optimum solution.

Figures 1.43 and 1.44 present the time series of amino value histograms for CGA and SGA, respectively. The results are for a 5-dimensional Type III deception problem. We see that for all α_k assuming maximum values at each dimension of k, the amino value for CGA exceeds a certain value. In contrast, for SGA, the amino value exceeds a certain value in certain dimensions but not in others. This is a typical example of a search that has fallen into a local optimum solution.

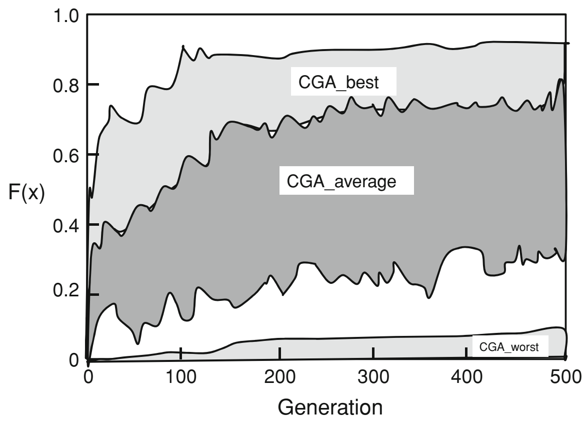


Fig. 1.41 Evolution of CGA for deceptive problem (type III)

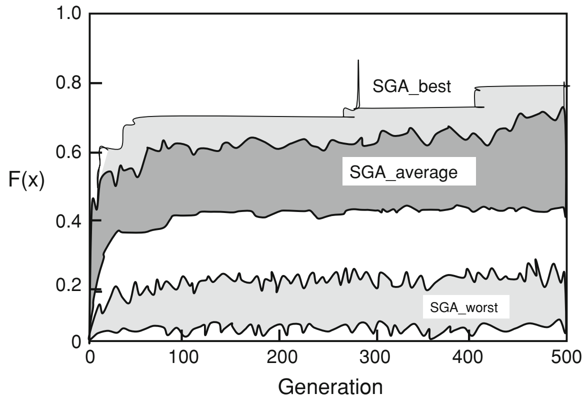


Fig. 1.42 Evolution of SGA for deceptive problem (type III)

1.4.1.5 Performance Comparison Among SGA, CGA and PfGA

Tables 1.4 and 1.5 present comparisons of performance evaluation results for SGA, CGA, and PfGA.

Table 1.4 gives the results for deception problems. A comparison of the success rates of CGA and SGA shows that the performance of CGA far outpaces SGA. Furthermore, PfGA is 100% successful for all types of deception problems. Table 1.5 gives the results for benchmark problems (Shekel's foxhole problem, Langerman function), and we see that CGA gives success rates comparable to PfGA.

Figure 1.45 shows the changes in basin size in the case of CGA. The figure shows a sudden increase in basin size at a certain point in the evolution (100 generations), considered to reflect the achievement of punctuated equilibrium associated with the transition stage in the evolution, mapping from genotype (binary value) to phenotype (function value).

Figure 1.46 presents the values of the codon-amino acid translation table. Under the initial conditions of evolution, variations in the values in the table

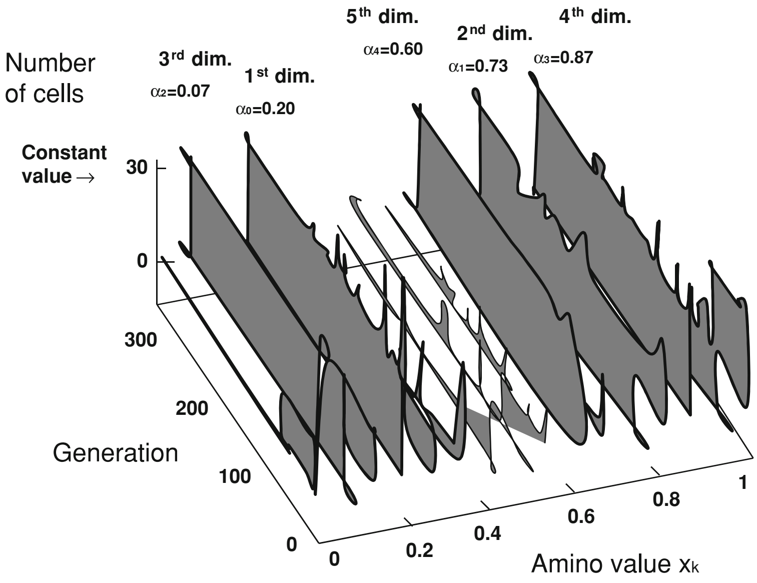


Fig. 1.43 Time series of amino value histogram for CGA

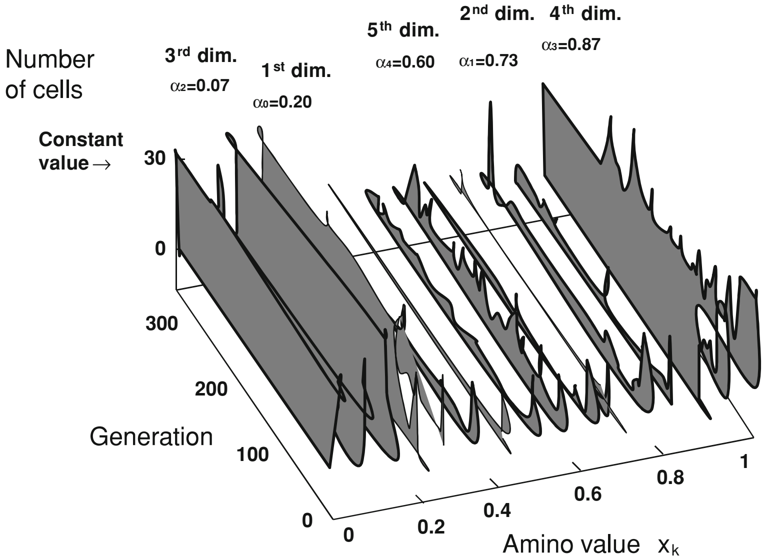


Fig. 1.44 Time series of amino value histogram for SGA

appear to be large changes in amino value per single bit of change in the codon (lower right). But as evolution progresses, the amino value changes only gradually relative to the change in codon value. This corresponds to the evolution (transition)

Table 1.4 Success rate for deceptive problems

GA	SGA	SGA*	CGA	PfGA	SGA	SGA*	CGA	PfGA
Dimension	5	5	5	5	10	10	10	10
Codon length	6	6	6	20	6	6	6	20
Scaling	Linear	Linear	Exponential	None	Linear	Linear	Exponential	None
Type I	2%	9%	100%	100%	0%	3%	100%	100%
Type II	12%	20%	100%	100%	0%	5%	100%	100%
Type III	47%	85%	95%	100%	14%	79%	43%	100%

Table 1.5 Success rate for benchmark problems

GA	SGA	SGA*	CGA (WF)	CGA (NF)	PfGA	SGA	SGA*	CGA (WF)	CGA (NF)	PfGA
Dimension	5	5	5	5	5	10	10	10	10	10
Codon length	6	6	6	6	20	6	6	6	6	20
Scaling	Linear	Linear	Exponential	Exponential	None	Linear	Linear	Exponential	Exponential	None
Shekel	5%	5%	5%	50%	37%	0%	0%	0%	0%	1.3%
Langerman	41%	47%	13%	35%	83%	0%	0%	0%	3%	1.7%

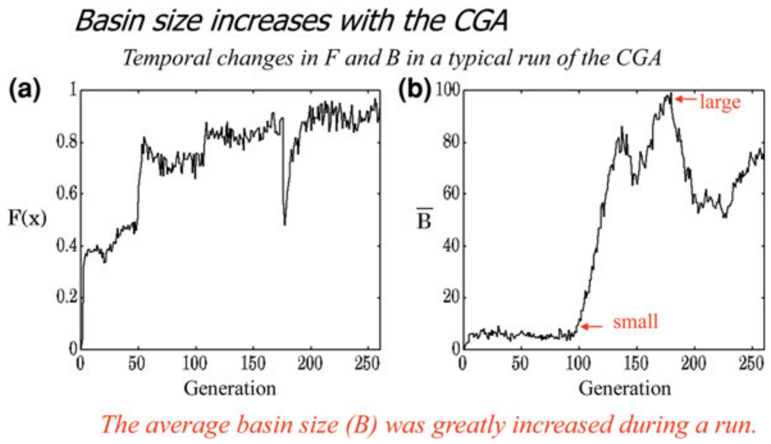


Fig. 1.45 Increase in basin size for CGA

of the landscape represented by the left plot of Fig. 1.39 into the smooth landscape represented by the right plot. The evolvability of CGA has greatly increased, indicating that the algorithm generates a solution method for difficult problems in an evolutionary manner, while automatically (in evolutionary fashion) converting difficult problems into easier problems.

The method for improving mapping techniques from genotype to phenotype through the evolutionary process is a highly generalizable optimization technique. The following section discusses a method for expanding CGA to genetic programming (CGP).

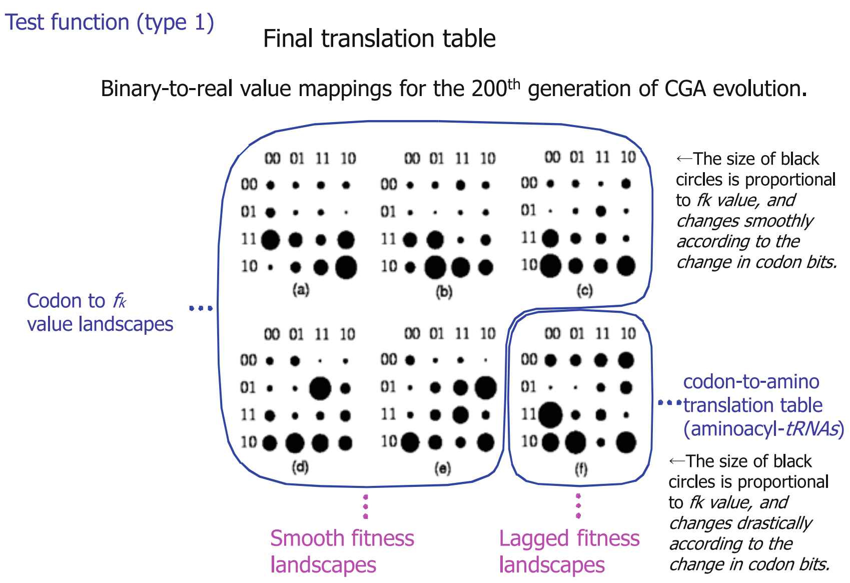

Test function (type 1)
Final translation table
Binary-to-real value mappings for the 200th generation of CGA evolution.
00 01 11 10
(a)
(b)
(c)
(d)
(e)
(f)
←The size of black circles is proportional to fk value, and changes smoothly according to the change in codon bits.
Codon to fk value landscapes
codon-to-amino translation table (aminoacyl-tRNAs)
←The size of black circles is proportional to fk value, and changes drastically according to the change in codon bits.
Smooth fitness landscapes
Lagged fitness landscapes

Fig. 1.46 Final translation table in binary-to-real value mapping for CGA

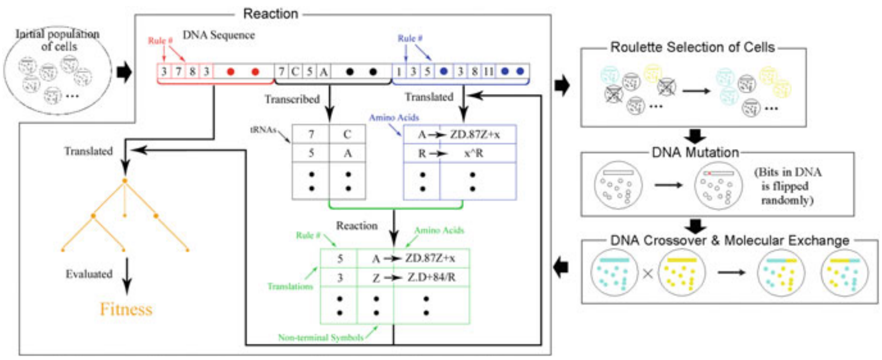


Fig. 1.47 CGP algorithm

1.4.2 Chemical Genetic Programming (CGP) [23]

Figure 1.47 shows how CGA is expanded to CGP. A comparison of Figs. 1.38 and 1.47 shows how the genes (DNA sequence) in CGA are converted into a combination of the rewriting rule numbers and the left sides of the rewriting rules. DNA is translated into protein synthesized by concatenating amino acids, after which fitness is calculated. At this point, the other portions of the DNA have been transcribed into tRNA or translated into amino acids. Aminoacyl tRNA, produced by their reaction, acts as a catalyst in this translation process. Modeling these metabolic processes inside the cell results in the evolutionary generation of the rewriting rule itself, and generates rules completely different from those in the initial state to make it possible to acquire rules with higher fitness scores.

Column 6: Genetic Programming

The procedures for genetic programming (GP) are basically the same as those for genetic algorithms (GA). However, while the genotype of the candidate solution in GA is a bit sequence or a real value sequence, it is a tree structure in GP. The tree structure in GP consists of a terminal set (a set of terminal symbols) containing independent variables and constants and a non-terminal set (also called a function set; a set of non-terminal symbols) containing functions and the four arithmetic operations of addition, subtraction, multiplication, and division. GP randomly generates an initial population, then evaluates the fitness of the population to select individuals. Next, as shown in Fig. 1.48, crossovers occur at probability *Pc* between a pair of individuals arbitrarily selected from the population. In the example shown in this figure, the terminal set is $\{x, y, z, a, b, c\}$, and the function set is {+, −, *, %, sin, cos, exp}. The two parents—Parent 1: $(x - a)/z + \sin(b*y)$ and Parent 2: $\cos(y + c)*(x - \exp(z))$—are each represented by a tree structure, and subtrees $b*y$ and $x - \exp(z)$ are stochastically selected for the crossover. The end result are the two offspring Offspring 1: $(x - a)/z + \sin(x - \exp(z))$ and Offspring 2: $\cos(y + c)*(b*y)$.

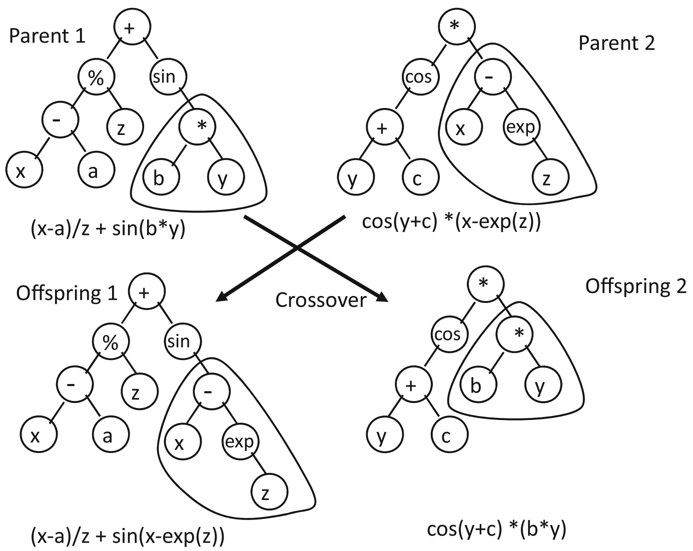


Fig. 1.48 Crossover in genetic programming

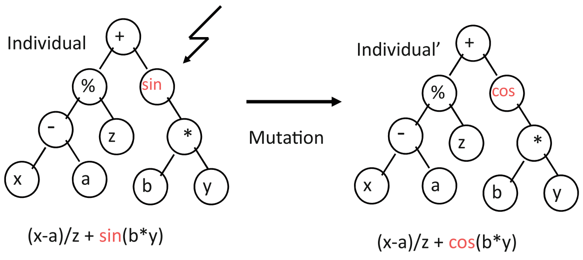


Fig. 1.49 Mutation in genetic programming

Next, as shown in Fig. 1.49 in this column, a mutation occurs with probability Pm in the non-terminal node representing sin. The function in the node changes from sin to cos, and the phenotype of the individual also changes from $(x - a)/z + \sin(b{*}y)$ to $(x - a)/z + \cos(b{*}y)$. Later, the population is evaluated. If the solution obtained is satisfactory, the optimum solution is output, and processing ends. If not, GP repeats the series of genetic manipulations above while advancing from generation to generation.

Genetic programming is widely applied as an approach to solving real-world problems in various areas of artificial intelligence, including symbolic regression problems, system identification, optimization control, planning, time series predictions, automatic programming, discovery of game strategies, solutions for inverse problems, rules discovery, pattern classification, evolution of emergent behavior, automatic programming of cellular automata, and evolvable hardware (EHW).

Chemical genetic programming (CGP), described in the main text, has expanded and developed from GP and can adapt to various environments by dynamically changing the mapping from the genotype to the phenotype. In other words, since feedback from the phenotype to the genotype makes it possible to maintain diversity in a population with a small number of individuals, it requires smaller populations than ordinary GP. This reduces the computational loads and memory required and increases the range of evolutionary possibilities.

1.4.2.1 Example of Application 1: Symbolic Regression Problems [23]

Figure 1.50 compares the fitness evolution curve in CGP and conventional GE (grammatical evolution). We see that evolution proceeds faster in CGP than GE, resulting in a good solution after 140 generations. Furthermore, even though the best solution appears faster, the average fitness of the population consistently remains below the best value, indicating that population diversity is sustained.

Figure 1.51 compares the solution generated by CGP and GE. For the target function $2x^6 + 3x^4 + 4x^2 + 100$, CGP gives $2x^6 + 501$, while GE gives $1.9x^6$. Since the normalized fitness values are 0.95 and 0.81 for CGP and GE, respectively, we may conclude that CGP is superior.

- *fitness= exp (a*exp(-d/b)),* $a=8$, $b=5*10^7$
- Good solution found after 140 generations.
- Population average remains significantly lower than best individual.
  i.e., population diversity is maintained.

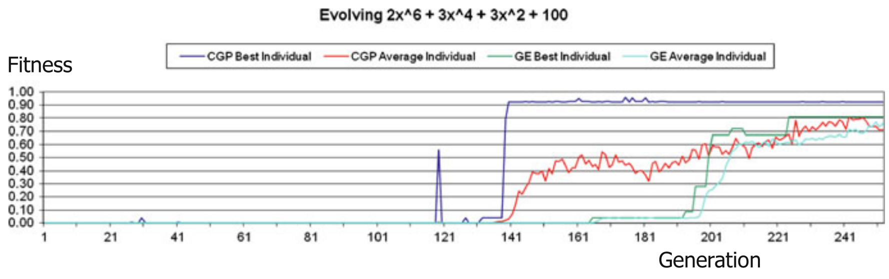


Fig. 1.50 Evolution curve of fitness

Generated formulas

- Target: $2x^6+3x^4+4x^2+100$
- CGP: $2x^6 + 501$ normalized fitness=0.95
- GE: $1.9*x^6$ normalized fitness=0.81

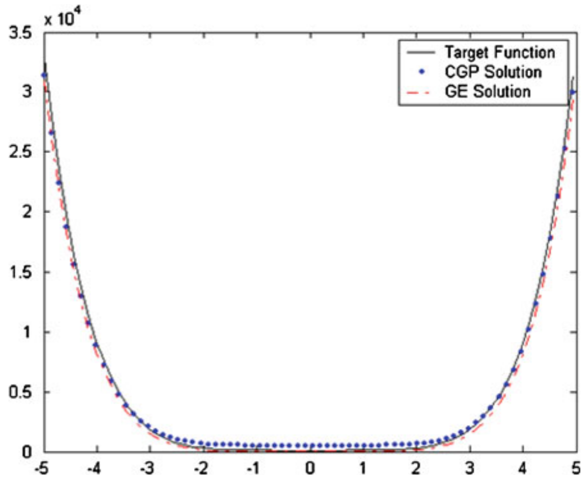


Fig. 1.51 Generated function: CGP vs. GE

1.4.2.2 Example of Application 2: Behavioral Strategy of Agents [24]

Figure 1.52 applies CGP to a multi (2)-agent problem, the game of “tag.” Two agents use CGP to generate in *co-evolutionary* fashion a behavioral strategy for catching the other as quickly as possible, or to evade the other for as long as possible.

Table 1.6 is a list of basic functions used in CGP.

Figure 1.53 shows the two agents in motion. S denotes the starting point. The objects numbered 1 and 2 in the center are obstacles. We see how the two agents (pursuer and evader) skillfully avoid the obstacle in the chase. Table 1.7 presents

Fig. 1.52 The game of tag

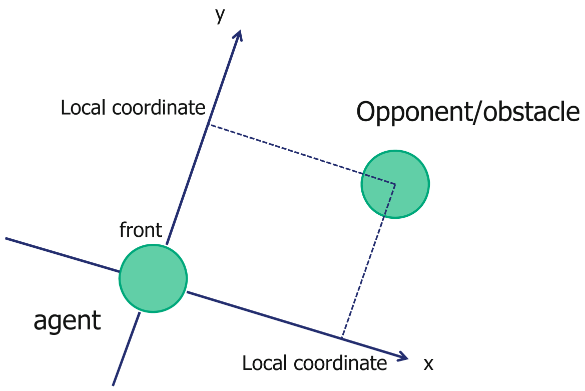


Table 1.6 List of basic functions

Function	Usage	Description
+	(+ a b)	Addition: $a + b$
−	(− a b)	Subtraction: $a - b$
*	(* a b)	Multiplication: $a * b$
%	(% a b)	If $b = 0$, then 1, else $a \div b$
Min	(min a b)	If $a < b$, then a, else b
Max	(max a b)	If a $> b$, then a, else b
Abs	(abs a)	Absolute value of a: $\|a\|$
Neg	(neg a)	Negative value of a: $-a$
Iflte	(iflte a b c d)	If $a \leqq b$, then c, else d

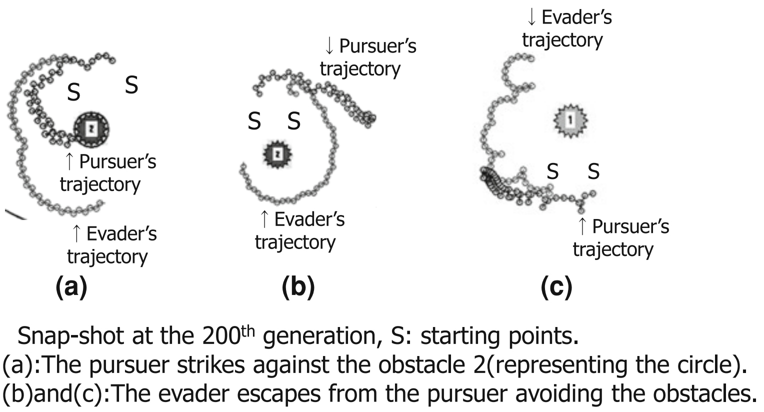


Fig. 1.53 Results of agent behavior

the behavioral strategies generated by the agents. Although the details are not given here, we can appreciate how both agents acquire various behavioral strategies through *co-evolutionary* generation.

Table 1.7 Emergent strategies

Emergent strategy	Generation	Player
Tagging	200–450	Pursuer
Weaving	200–450	Pursuer
Safe haven	200–450	Evader
Obstacle hugging	200–450	Evader
Boundary diversion	~450	Evader
Focused tagging	~450	Pursuer
Point circling	~500	Evader
Arcing	~1000	Evader
Shorting	~1000	Pursuer

References

1. H. Simon, *Neural Networks and Leaning Machines*, 3rd edn. (Prentice Hall, Upper Saddle River, 2008)
2. M.A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks* (MIT Press, Cambridge, 2006)
3. R.S. Sutton, G.B. Andrew, *Reinforcement Learning—An Introduction (Adaptive Computation and Machine Learning)* (MIT Press, Cambridge, 1998)
4. A. Waibel, in *Phoneme Recognition Using Time-Delay Neural Networks,Report of Speech Committee SP87-100* (December 1987), pp. 19–24
5. A. Waibel, H. Sawai, K. Shikano, Modularity and scaling in large phonemic neural networks. IEEE Trans. ASSP **37**(12), 1888–1898 (1989)
6. S. Nakamura, H. Sawai, Performance comparison of neural network architecture for speaker-independent phoneme recognition. Trans. IEICE D-II **J74-D-II**(10), 1361–1369 (1991)
7. K. Fukuzawa, Y. Komori, H. Sawai, M. Sugiyama, A Segment-Based Speaker Adaptation Neural Network Applied to Continuous Speech Recognition. in *Proceeding of the ICASSP'92*, San Francisco, vol. 1 (March, 1992), pp. 433–436
8. H. Sawai, Frequency-Time-Shift-Invariant Time-Delay Neural Networks for Robust Continuous Speech Recognition. in *Proceedings of the ICASSP'91, Toronto, S2.1*, vol. 1 (May 1991), pp. 45–48
9. H. Sawai, Axially Symmetric Neural Network Architecture for Rotation-Invariant Pattern Recognition. in *IEEE, WCCI (World Congress on Computational Intelligence)*, *Proceedings of the IEEE ICNN'94*, vol. 7 (June–July 1994), pp. 4253–4258
10. C. Darwin, *On the Origin of Species by Means of Natural Selection or the Preservation of Favored Races in the Struggle for Life* (John Murray, London, 1859)
11. J.H. Holland, *Adaptation in Natural and Artificial System* (The University of Michigan Press, Ann Arbor, 1975)
12. H. Sawai, S. Kizu, T. Endo, Parameter-free genetic algorithm (PfGA), Tech. Rep. IEICE **J81-D-II**(2), 450–452 (1998)
13. K. Wada, H. Doi, S. Tanaka, Y. Wada, M. Fu-rusawa, A neo-Darwinian algorithm: asymmetrical mutations due to semi-conservative DNA-type replication promote evolution. Proc. Natl. Acad. Sci. USA **90**, 11934–11938 (1993)
14. S. Adachi, H. Sawai, Effects of migration methods in parallel distributed parameter-free genetic algorithm. Trans. IEICE D-I **J83-D-I**(8) 834–843 (2000)
15. S. Adachi, H. Sawai, Evolutionary computation method inspired by gene-duplication: application to functional optimization. J. IPSJ **42**(11), 2663–2671 (2001)
16. S. Ohno, *Evolution by Gene Duplication* (Springer, New York, 1970)
17. S. Ohno, *Seimei no Tanjo to Shinka (The Birth and Evolution of Life)* (University of Tokyo Press, Tokyo, 1988)

18. S. Ohno, *Ooinaru Kasetsu—DNA karano Messeji (The Great Hypothesis: Messages from DNA)* (Yodosha, Tokyo, 1991)
19. K. Omori, Y. Fujiwara, S. Maekawa, H. Sawai, S. Kitamura, Evolutionary computation based on sexual selections driving asymmetrical mutation. J. Inst. Syst. Control Inf. Eng. **15**(8), 422–429 (2002)
20. M. Hasegara, *Why is the Male Peacock Beautiful?* (Kinokuniya Shoten, Tokyo, 1992)
21. Y. Iwasa, *Introduction to Mathematical Biology—Investigating the Dynamics in Biological Society* (Kyoritsu Shuppan, Tokyo, 1998)
22. H. Suzuki, H. Sawai, W. Piaseczny, Chemical genetic algorithms—evolutionary optimization of binary-to-real-value translation in GAs. J. Artif. Life **12**, 1–27, (2006)
23. W. Piaseczny, H. Suzuki, H. Sawai, Chemical genetic programming—evolutionary optimization of the genotype-to-phenotype translation set. J. Artif. Life Robot. **9**, 202–208 (2006)
24. C.S. Chan, J.C. Tay, H. Sawai, Discovering Pursuit and Evasion Strategies for a Bounded Environment with Two Obstacles. in *Proceedings of the Third International Conference for Computational Intelligence*, *Robotics and Autonomous systems* (*CIRAS 2005*) (Singapore, 13–16th December, 2005)
25. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **7**, 115–133 (1943)
26. D. Rumelhart, J. L. McCleland, et al., *Parallel Distributed Processing*, vol. 1 (MIT Press, Cambridge, 1986)
27. K. Wada, Dejitaru Seimei no Shinka (The Evolution of Digital Life). in *Iwanami Kagaku Raiburari 11 (Iwanami Chemistry Library 11)* (Iwanami Shoten, Tokyo, 1994)
28. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley Professional, New York, 1989)
29. D.E. Goldberg, *Genetic Algorithms: The Design of Innovation* (Springer, New York, 2009)
30. H. Sawai, S. Kizu, in *Parameter-free Genetic Algorithm Inspired by Disparity Theory of Evolution*. Lecture Notes in Computer Science, vol. 1498 (Springer, New York, 1998)
31. M. Furusawa, H. Doi, Promotion of evolution: disparity in the frequency of strand-specific misreading between the lagging and leading DNA strands enhances disproportionate accumulation of mutation. J. Theo. Biol. **157**, 127–133 (1992)
32. Chiba Bio Network, http://www.chibabio.net/comp/comp_07.html
33. M. Furusawa, Shinka wa Isshun de Okiru—Tainai ni Yadoru Shinka no Chikara (Evolution Occurs in an Instant—Evolutional Power inside the Body), *Seimeishi* (*Biohistory*), no. 31 (2001). http://www.brh.co.jp/seimeishi/1993-2002/31/ss_5.html
34. Seibutsugaku Jiten (Dai 4 Han) Idenshi Chofuku (*Dictionary of Biology*, 4th edn.), Gene Duplication), (Iwanami Shoten, Tokyo, 2000), p. 77
35. C. Darwin, *The Descent of Man and Selection in Relation to Sex*, Volume 1 of the Japanese translation by Mariko Hasegawa, Ningen no Shinka to Sei Tota I (Bun-ichi Co. Ltd., 1999, Japan)
36. C. Darwin, *The Descent of Man and Selection in Relation to Sex*, Volume 2 of the Japanese translation by Mariko Hasegawa, Ningen no Shinka to Sei Tota II (Bun-ichi Co. Ltd., Japan, 2000)
37. M. Hasegawa, *Osu to Mesu no Kazu wo Meguru Fushigi (Wonders about Numbers between Males and Females*) (NTT Publishing Co. Ltd., Tokyo, 1996)
38. M. Hasegawa, *Osu to Mesu—Sei wa Naze Aruka* (*Males and Females—Why Sex Exists*) (NHK Human University, Japan, 1997)