Memoirs of the Faculty of Engineering, Okayama University, Vol.26, No. 2, pp.111-128, March 1992

Memory Search using Genetic Algorithms and a Neural Network Model

Shigetoshi NARA[1] and Wolfgang BANZHAF[2]

(Received January 18 , 1992)

SYNOPSIS

An information processing task which generates combinatorial explosion and program complexity when it is treated by a serial algorithm is investigated using both Genetic Algorithms (GA) and a neural network model (NN). The task in question is to find a target memory from a set of stored entries in the form of "attractors" in a high dimensional state space. The representation of entries in the memory is distributed ("an auto associative neural network" in this paper), and the problem is to find an attractor under a given access information where the uniqueness or even existence of a solution is not always guaranteed (an ill-posed problem). The GA is used as an algorithm for generating a search orbit to search effectively for a state which satisfies the access condition and belongs to the target attractor basin in state space. The NN is used to retrieve the corresponding entry from the network. The results of our computer simulation indicate that the present method is superior to a search method which uses random walk in state space. Our technique may prove useful in the realization of flexible and adaptive information processing, since pattern search in high dimensional state spaces is common in various kinds of parallel information processing.

[1]Department of Electrical and Electronic Engineering
[2]Central Research Laboratory, Mitsubishi Electric Corporation, Tsukaguchi Honmachi 8-1-1, Amagasaki, Hyogo 661

1 Introduction

Although quick and great progress has been made with modern LSI computers working based on the von Neumann type of serial algorithms, there has been growing interest in parallel or flexible (non von Neumann type) information processing such as those typically observed in biological systems. Serial algorithms run into problems with (1) combinatorial explosion and (2) program complexity in realizing flexible functions. One possibility to improve flexibility in information processing is the application of complex nonlinear dynamics including chaos, which attracts increasing interest not only in the field of physics but also in the interdisciplinary realm[1]. Complex dynamics is characterized as "emerging complex behaviour generation"[2]-[4]. The great variety of possible dynamical structures suggests to us that an application to complex information processing may avoid combinatorial explosion and/or growing control complexity [5], [6], [7]. The main problem of flexible information processing function is, generally speaking, due to the fact that there are too many degrees of freedom for sequential control. Therefore, in a practical sense, the internal complexity contained in nonlinear complex dynamics is actually useful to decrease the control complexity as well as help to avoid combinatorial explosion[8], [9].

In order to make the problem clearer, we shall concentrate on a particular information processing task. However, if we impose too severe restrictions for the proposed information processing context, it will lead us to a less or even poorly interesting result because of a very *ad hoc* solution for the problem, while such restriction brings us an easy modeling and formulation. Therefore, in setting the context, one needs to pay attention so as to keep the universality of application to wide field of processing. We propose pattern search in a high dimensional state space[8], [9], in which the stored information is represented as distributed memory entries (a neural network in the present paper) in the form of "attractors"[10]. In more detail, the task is to specify or retrieve, if

existent, one or more of the stored patterns under the condition that the given access information is not complete and not sufficient to reach the target pattern directly. This is a kind of ill-posed problems in which the uniqueness or even the existence of a solution is not guaranteed. It is a desirable function of the information processor to solve this complex problem efficiently, and there are two important points with respect to the search process in the state space[9]. They are

1. How can the processor quickly and efficiently find the target basin from the ambiguous acccess information ?
2. Provided there is no memory which satisfies the access information, can the search processor generate information close to the requested pattern ?

Several methods qualify for the envisioned function. (1) random walk in state space ([11], for an example), (2) chaotic dynamics [7]-[9], [12], (3) genetic algorithms - frequently called evolutionary strategies, (4) neural networks, (5) cellular automata [13], [14], (6) general nonlinear dynamics [15]. In this paper, we employ a method which uses both a neural network (NN) and Genetic Algorithm (GA). The former area has a long history and became an especially active field during the past decade as a powerful method for parallel processing [16]-[17]. Associated memory and classification of highly complicated signals [18] are two prominent applications. The latter also attracted a large number of workers in the past years especially in the field of optimization in high dimensions [19]-[23].

2 The Neural Network Model

To begin with, let us start with the description of the state space considered. Without loss of generality, we employ a state space of image patterns consisting of 20×20 pixels and introduce one neuron corresponding to each pixel. Using a Hopfield network, neural activity is restricted to two states, +1 (firing state) and -1 (non firing state), and we obtain a state space consisting of bit patterns the number of which is 2^{400} (the top points of hyper cube in 400-dimension), each pattern specified by a 400-dimensional state vectors, $\mathbf{v} = (v_1, v_2, \cdots, v_{400})$. In the present model, each neuron is assumed to be coupled with each other and their coupling strength is represented by a synaptic connection matrix the dimension of which is 400×400. In this state space, we embed 30 patterns as the stored memories which form "attractors".

Secondly, let us employ

$$v_i(t+1) = \Theta(\sum_{j=1}^{400} T_{ij} v_j(t) - h_i) \tag{1}$$

as the time developing rule of this neural network model, where each neuron is represented by discrete variables $v_i = \pm 1$ and T_{ij}, the synaptic connection matrix. Θ is a step function and h_i is a threshold of neurons which is taken to be zero in this paper for convenience. With respect to the 30 patterns we used for actual simulations, the overlap between them is relatively large as can be observed from Fig.1. Thus, retrieval performance would become very bad if we applied an auto correlation memory. Therefore, we used orhogonalization between patterns [24], [25] by introducing adjoint state vectors $\mathbf{v}_\alpha^\dagger$ ($\alpha = 1, 2, \cdots, 30$) which are defined by

$$\mathbf{v}_\alpha^\dagger \bullet \mathbf{v}_\beta = \delta_{\alpha\beta}, \quad \mathbf{v}_\alpha^\dagger = \sum_\gamma a_{\alpha\gamma} \mathbf{v}_\gamma, \quad a = o^{-1}, \quad o_{\alpha\beta} = \mathbf{v}_\alpha \bullet \mathbf{v}_\beta \tag{2}$$

a is the inverse matrix of 30×30 correlation matrix the elements of which are defined by the scalar products between two pattern vectors. The synaptic

connection matrix is now defined as

$$\mathsf{T} = \sum_{\alpha=1}^{30} \mathbf{v}_\alpha \otimes \mathbf{v}_\alpha^\dagger \tag{3}$$

which is known as the pseudo-inverse of the pattern matrix. T_{ij} is still symmetric, so that the energy function, $E = -\sum_{ij} v_i T_{ij} v_j$, is a Lyapunov function for this system. The resulting basins of attraction in state space become quite large now and it is expected that in these basins much information is distributed which we call "seeds" for the access. With respect to the "seed", we mean that it contains many compressed or partial features of target patterns (attractors). Namely, once some pattern (seed) containing the partial feature of an attractor is given to the neural network, it will be entrained into the attractors, as if one retrieve the specified person from glancing of the shape of face.

By introducing the state space of the NN, our ill-posed problem of pattern search becomes clearer. Suppose that there is an accessor who wants to find a specified face (not always one) from data base of various faces with use of ambiguous feature of target faces, in the present simulation, the feature of eye shape. This type of information task is exemplified in the task of tracking a criminal suspect using a large amount of face photographs under the ambiguous evidence of an accidental witness. Therefore, our simulation was set to search target face patterns which have the eye-shape feature consisting of 40 bit. The state space is the 400-dimensional pattern space, but note that the existence of a solution is not assumed. We gave eye-shape data (40 bit) of the fourth face pattern in Fig.1 as access information. The search algorithm is described in the next section.

3 The Genetic Algorithm

In this section, we consider the application of a Genetic Algorithm [26] to generate an effective search orbit in the state space.

Let us define a gene string as consisting of 401 elements, where the first element x_0 is the header with the evaluation value of the gene and the other 400 elements x_i are a state vector. We introduce a certain number of these genes and call them together the gene pool. Here as in most GAs, two different kinds of operations are performed, "mutations" and "recombinations". The former are operations randomly changing a certain number of components of a gene. The latter are operations where two or more genes cooperate to generate a new gene string with a certain number of elements coming from each parent string.

In both operations, the elements subject to the operations are chosen using random generators.

In the following, we give a brief description of the applied operations.

a) Mutation :

Mutations in the present paper consist of the following three different operations. They are

(1) Single flip: choose a gene $\Rightarrow$ generate a random number $1 \leq n_1 \leq 400 \Rightarrow$ flip one component n_1

(2) Complement: choose a gene $\Rightarrow$ generate two random numbers $1 \leq n_1, n_2 \leq 400 \Rightarrow$ flip all components between n_1 and n_2

(3) Inversion: choose a gene $\Rightarrow$ generate two random numners $1 \leq n_1, n_2 \leq 400 \Rightarrow$ invert the order of components between n_1 and n_2

b) Recombination (Cross over) :

choose two genes, say gene α and $\beta \Rightarrow$ generate two random numbers between $1 \leq n_1, n_2 \leq 400$ and take $n_3 = \text{Min}\ [\ n_1 + n_2,\ 400\ ] \Rightarrow$ exchange the components of gene α and β between n_1 and n_3 with each other.

By these operations constantly varying strings are supplied which are selected according to their quality as follows.

c) Selection :

Each trial string which is the result of an application of one of the above operations onto a string from the gene pool is put into the neural network as an initial pattern and is updated until the neural network gives the converged pattern. Then, the converged pattern is evaluated in comparison to the given access information and will substitute its predecessors from which it was generated if it possesses higher or equal quality. It will be discarded if its quality is lower than that of the predecessor. Since every string carries its quality signal, this is a totally local selection method. It was already successfully applied in the case of the travelling salesman problem [26]. Fig.2 shows the overall process.

4 Results of Simulations

A simulation was done by employing nine genes in the gene pool and both mutation and recombination processes. A comparison was done with the results of random search which used random walk in state space. In both cases, we started with randomly chosen patterns. Fig.3 shows intermediate patterns along the time development of the search using the GA defined in the previous section. Note that all the patterns are corresponding to local minima of the energy landscape in state space since the trial patterns created by GA operations are made to converge to the definite patterns after many updates in the neural network.

In Fig.3 there appears a pattern worth to note because it is not the memory

patterns. One can interpret this new pattern to be a spurious pattern which is a partial superposition of stored patterns and notice that the network produced a new meaningful pattern from embedded memories in the sense of "spontaneous synthesis". This demonstrates us the possibility of information generation by the processor when the system can not find a requested pattern in memory which satisfies the access condition completely.

Now, let us turn to an evaluation of the search performance. We define $f(N)$ as the ratio between the number of successful trials and the total number of trials. One trial means a search process using a gene pool with nine genes with random patterns as initial state. A trial is successful if the access information is satisfied within the given iteration number N of operations. If there is no appearance of the target pattern within the given iteration number N, we regard the trial to be unsuccessful. The same quantity is calculated for the random search and shown for comparison.

In the case of a random search, it is easy to calculate $f(N)$. Let us note the fact that it is posssible to define the one step success probability in random search. It can be expressed by the ratio of the basin volume to the total volume of state space because each step can be regarded as completely random so that the process is considered to be Marcovian and there is no correlation or memory effect between two succeeding steps. Therefore, $f(N)$ has the form

$$f(N) = \sum_{r=0}^{N-1} (1-p)^r p = p\frac{1-(1-p)^N}{1-(1-p)} = 1-(1-p)^N \tag{4}$$

In order to understand the N-dependence of $f(N)$ in more detail, one can regard $f(N)$ as a continuous variable with respect to N. Then, a differentiation yields

$$\frac{d}{dN}f(N) = (1-p)^N \log\frac{1}{1-p} = \exp(-N\log\frac{1}{1-p})\log\frac{1}{1-p} \tag{5}$$

This indicates that $df(N)/dN$ should depends on N with an exponential damping in random search. We show the statistical result of our simulation in Fig.4

and Fig.5 both in the case of random search and the genetic search with respect to $df(N)/dN$, where the differentiation was done numerically in the discrete and finite intervals. The result of random search indicates an exponential damping, which is quite plausible as noted above. On the other hand, as shown in Fig.5 the case of genetic search indicates a very characteristic distribution of the differential success rate as a function of N. This tendency is not accidental or due to fluctuation of statistics because a more accurate result gives the same dependency as shown in Fig.6, which is obtained from the averaged success rate of 10000 samples whereas the former was obtained from averaging over 1000 samples. It is an interesting question why the distribution function indicates this very different behaviour in genetic search as compared to a random search.

However, it is sure that the genetic search is greatly superior to the method of random search by almost an order of magnitude, as indicated in Fig.7. The same simulation was done in for different numbers of genes in the gene pool. Results are shown in Fig.8 and a considerable improvement can be observed if we increase the population size n. Note that in order to obtain Fig.8 we have divided the total number of trials by the size of the population.

A further imporovment can be obtained by adjusting the recombination frequency carefully, see Fig.9.

5 Concluding Remarks

1. Genetic search algorithms are superior to random search by almost one order of magnitude

2. The employment of both mutation and recombination operations improve the search performance. An optimization of mutual freqency is beneficial.

3. The search performance is improved considerably when we increase the number of genes in the gene pool. This indicates that the present method is especially suited for parallel processing.

4. In performing the simulation, many spurious attractors were found and there were many useful spurious attractors in the sense of synthesized patterns (information generation) from the stored patterns.

Our method may prove useful in realizing flexible and adaptive information processing because pattern search in high dimensional state space appears to be common in various kind of parallel information processing.

The authors deeply thank to Dr. Peter Davis for his valuable comments.

References

[1] Dynamic Patterns in Complex Systems, ed. J. A. Kelso, A. J. Mandell and M. F. Shlesinger, World Scientific (1988)

[2] H. G. Schuster : Deterministic Chaos, VCH Weinheim, 1985

[3] R. L. Devaney : An Introduction to Chaotic Dynamical Systems, Benjamin Cummings, 1986

[4] K. Kaneko : Physica D, vol.41, 137 (1990)

[5] J. S. Nicolis : Rep. Prog. Phys. vol. 49, p. 80 (1986)

[6] Neural and Synergetic Computers, vol. 42, (Springer Series in Synergetics edited by H. Haken), 1989

[7] I. Tsuda, E. Koerner and H. Shimizu : Prog. Theor. Phys. vol. 78, p.51 (1987)

[8] P. Davis and S. Nara : Tech. Dig. of Int. Conf. on Fuzzy Logic and Neural Networks, Iizuka, (1990)

[9] P. Davis and S. Nara : Proceedings of The First Symposium on Nonlinear Theory and Its Applications, p. 97 (1990)

[10] Y. Mori, P. Davis and S. Nara : Journal of Physics A, vol. 22 (1989) L525

[11] S. Kirkpatrick, C. D. Gellat and M. P. Vecci : Science, vol. 220, p.671 (1983)

[12] S. Nara and P. Davis : preprint

[13] J. E. Hopcroft and J. D. Ullman : Introduction to Automata Theory, Language and Computation, Addison-Wisley Publishing

[14] S. Wolfram : Theory and Application of Cellular Automata, World Scientific, Singapore, 1986

[15] H. Haken : Information and Self-Organization, Springer-Verlag (1988)

[16] Neurocomputing, edited by J. A. Anderson and E. Rosenfeld, The MIT Press (1988)

[17] Neurocomputing 2, edited by J. A. Anderson, A. P. Pellionisz and E. R. Rosenfeld, The MIT Press (1991)

[18] D. Rumelhart et. al : Parallel Distributed Processing, ed. J. L. McClelland, D. E. Rumelhart and PDP Research Group, MIT Press (1986)

[19] D. E. Goldberg : Genetic Algorithms in search, optimization and machine learning, Addison-Wesley Publishing Company, (1989)

[20] Genetic Algorithms and Simulated Annealing, edited by L. Davis, Pitman, London, (1989)

[21] Proceedings of The Third International Conference on Genetic Algorithms, ed. J. D. Schaffer, Morgan Kaufman Publishers (1989)

[22] see, Proc. 4th Int. Conf. on Genetic Algorithms, San Diego 1991, Eds.; R. K. Belew and L. B. Booker, Morgan Kaufmann San Mateo, CA, 1991

[23] see, Proc. 1st Int. Conf. on Parallel Problem Solving from Nature, Dortmund 1990, Eds.; H. P. Schwefel and R. Manner, Springer Berlin, 1990

[24] L. Rersonnaz, I. Guyon and G. Dreyfus : Phys. Rev. A34, p. 4217 (1986)

[25] Fuchs and Haken : Dynamic Patterns in Complex Systems, ed. J. A. Kelso, A. J. Mandell and M. F. Shlesinger, World Scientific, p. 33 (1988)

[26] W. Banzhaf : Biological Cybernetics, vol. 64, p.7 (1990)

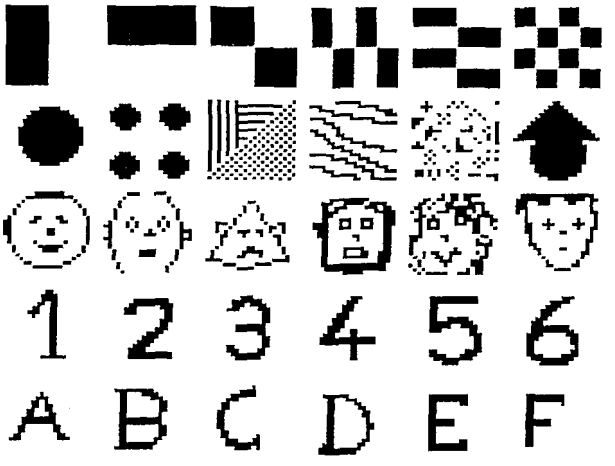


Figure 1: 30 memory patterns consisting of 20 × 20 pixels (neurons). Suppose the fourth pattern among the face patterns (third row) to be the target of memory search in our simulation of an ill-posed problem.

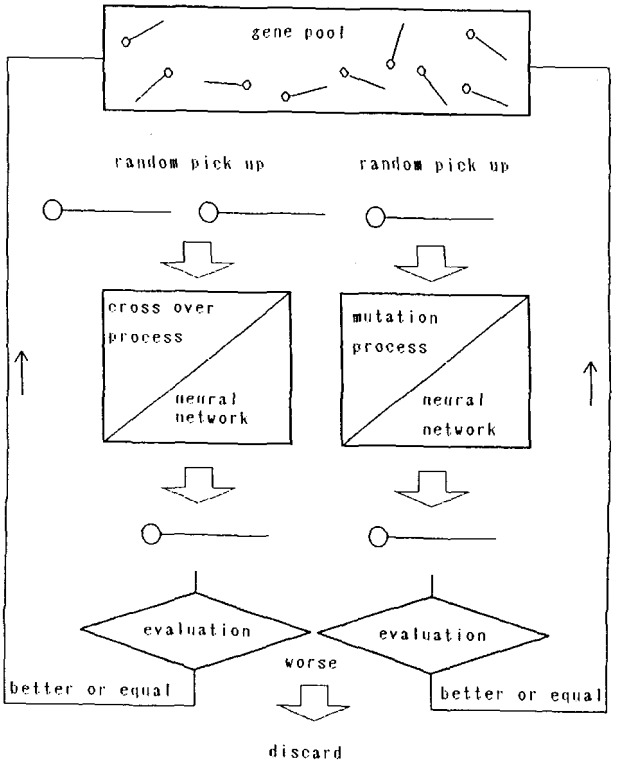
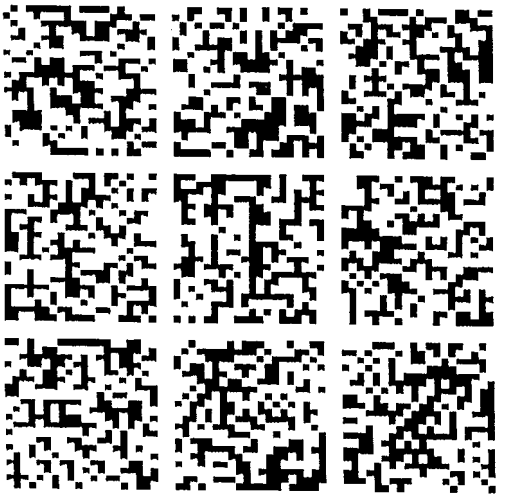
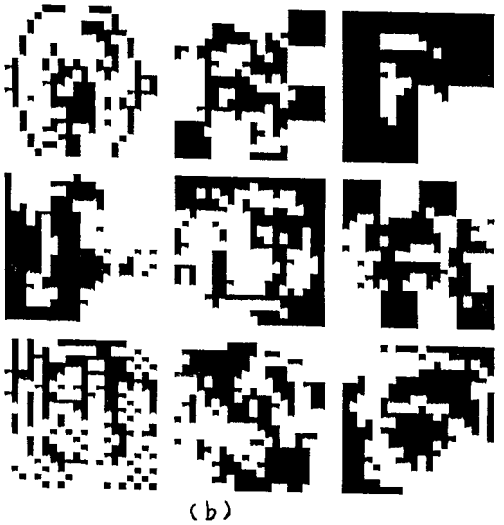


Figure 2: Over all algorithm of the memory search simulation. Note following points in the algorithm: in the mutation process, [1]when we operate one of the mutation operations defined in Section 3-a, the sequence of operation in the repeating processes is taken as cyclic, i.e.[⇒ mutation.1 ⇒ mutation.2 ⇒ mutation.3 ⇒] [2]the generated pattern (gene) is given to the neural network as an initial condition and the recurrent updating of firing patterns is done until it converges. [3]if any converged pattern has the given feature (feature of target), the search process stops and is regarded as successfully finished. If not, the evaluation value is compared with the predecessor (gene). In the recombination process, the evaluation procedure is the same with mutation process.

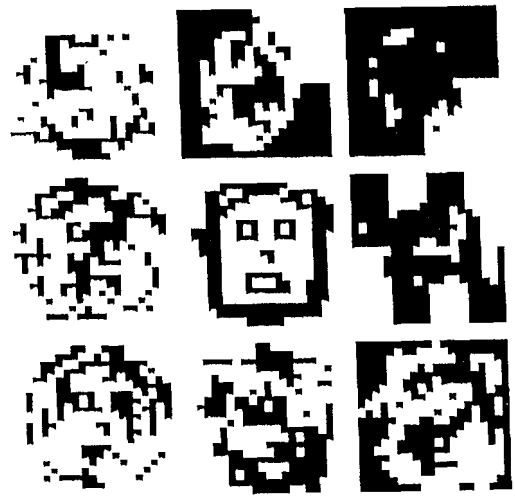


(a)

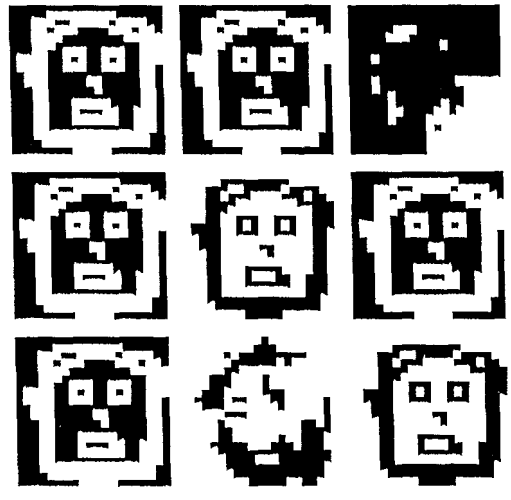
Figure 3: (a)Initial random patterns as given to nine genes (b)Nine patterns possessed by nine genes after first generation (c)Nine patterns of nine genes after 42-th generation. Note that the pattern in the left-lower corner is a superimposed pattern between the shape of the first face and the eye, nose and mouth of the fourth face in Fig.1 (d)Patterns of nine genes after 140-th generation. Most patterns have coverged to the target pattern.



(b)



(c)



(d)

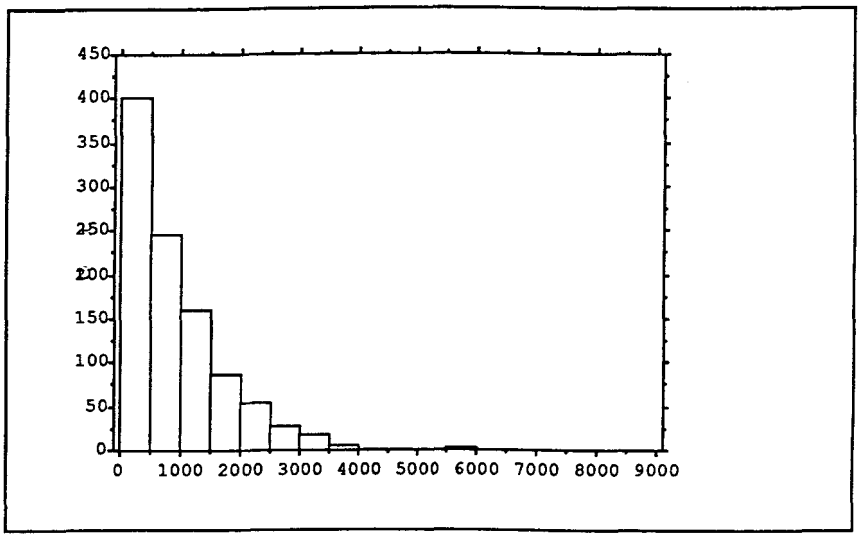


Figure 4: Differential success rate versus given upper limit of search step number in random search (1000 samples

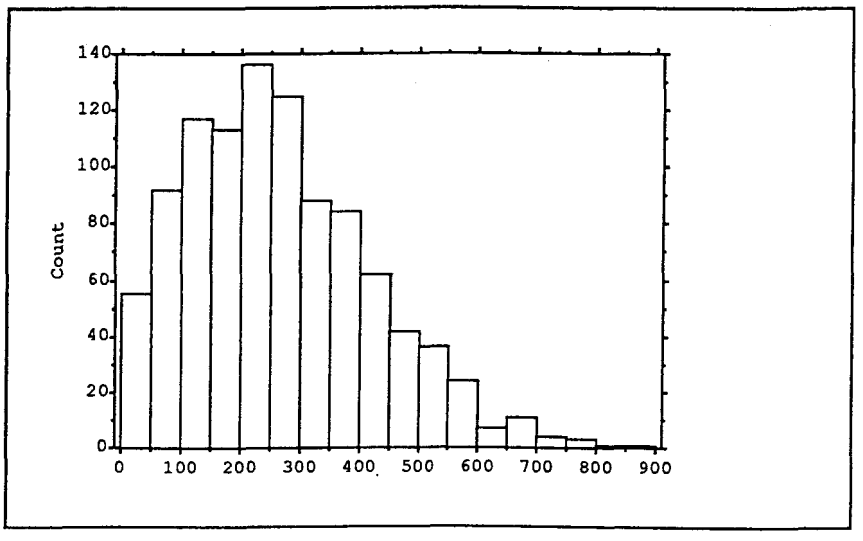


Figure 5: Differential success rate versus given upper limit of search step number in genetic search (1000 samples

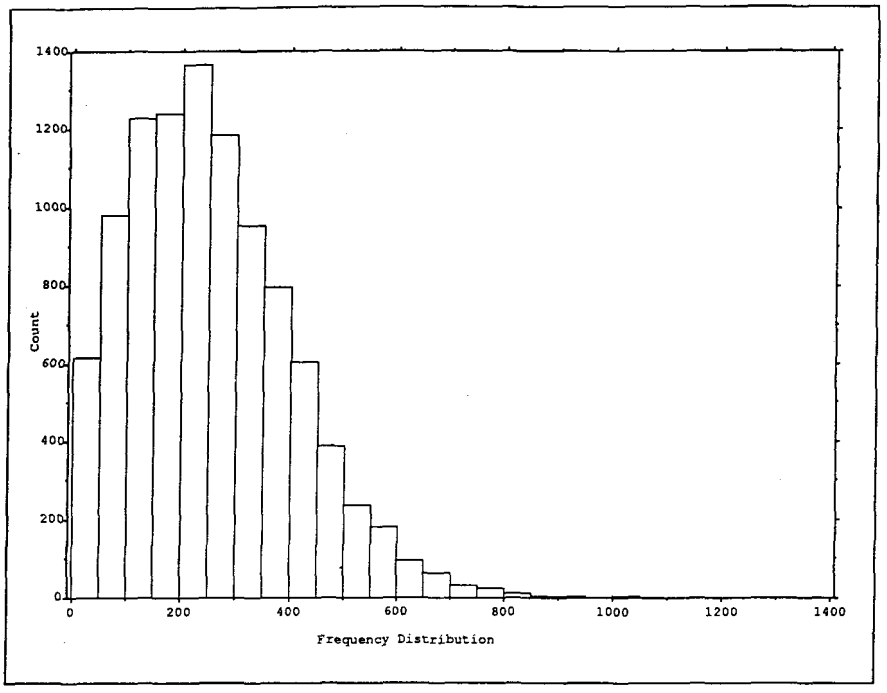


Figure 6: Differential success rate versus given upper limit of search step number in genetic search (10000 samples

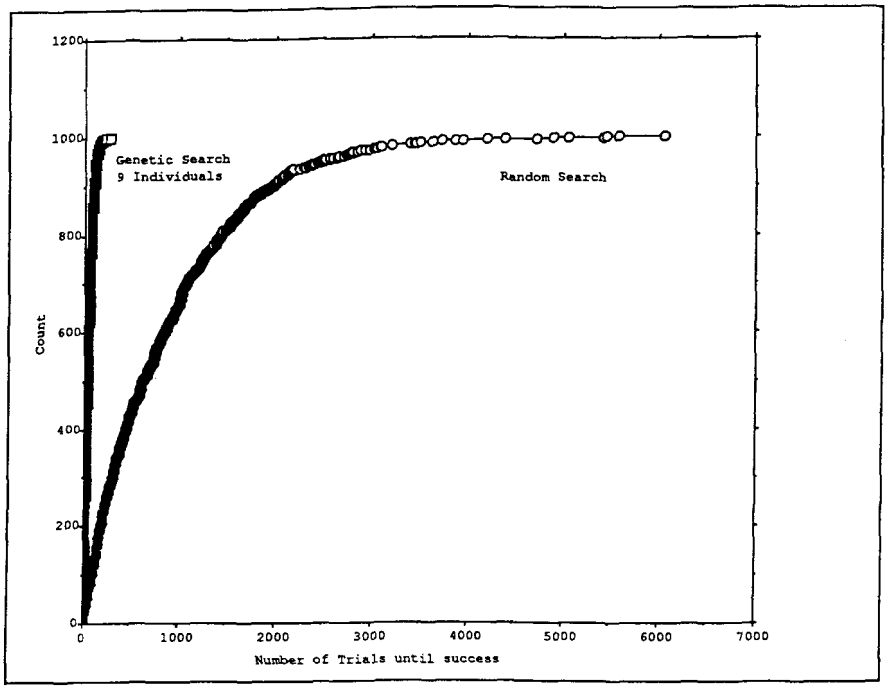


Figure 7: Comparison of success rate between random search and genetic search.

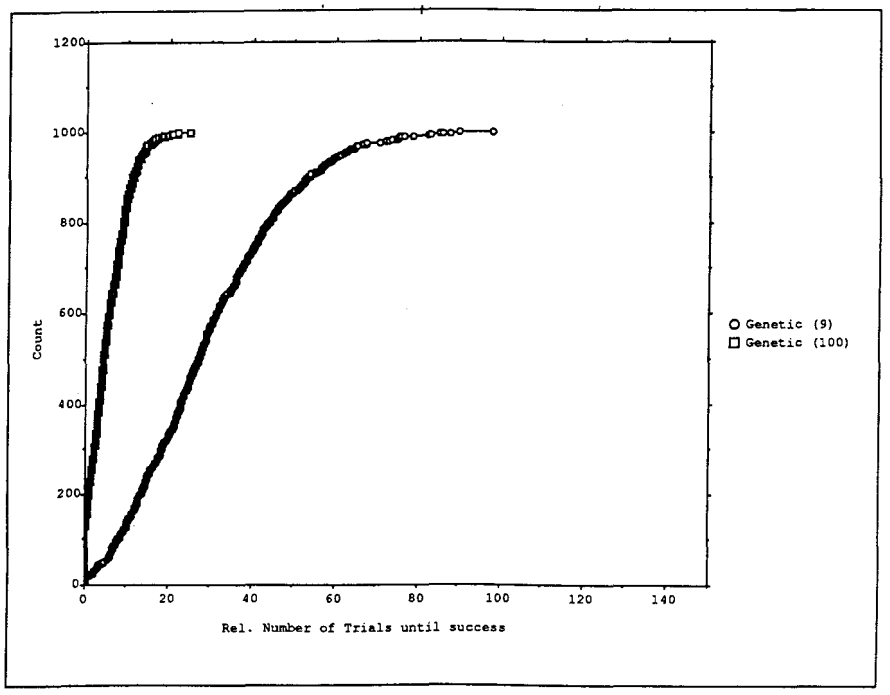


Figure 8: Comparison of success rate between different numbers of genes in the gene pool.

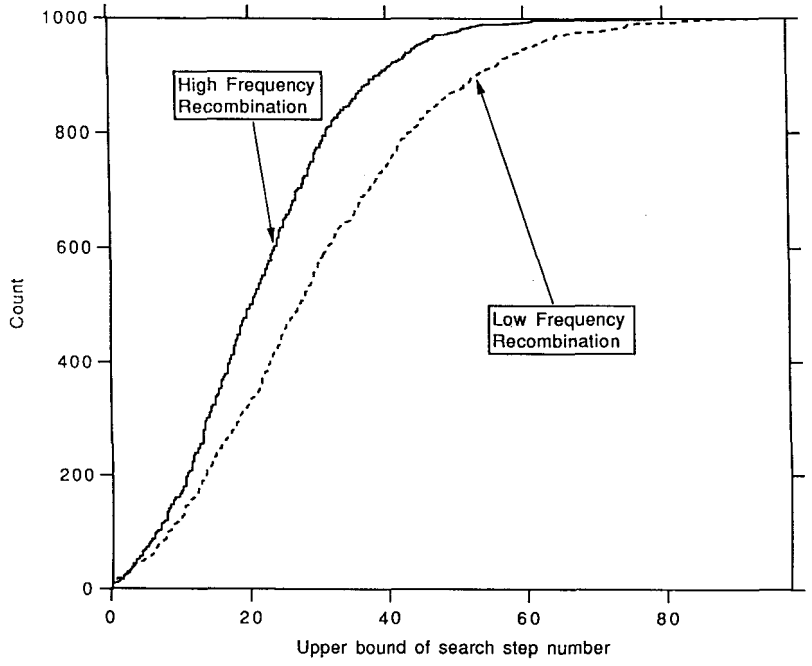


Figure 9: Comparison of success rate between different frequency of mutation and recombination in genetic search. (High: upper curve, Low: lower curve)