# Evolutionary optimization algorithm by entropic sampling

Chang-Yong Lee[1,*] and Seung Kee Han[2]
[1]*Research Department, Electronics and Telecommunications Research Institute, Yusong P.O. Box 106, Taejon, 305-600, Korea*
[2]*Department of Physics, Chungbuk National University, Cheongju Chungbuk 360-763, Korea*
(Received 16 June 1997)

A combinatorial optimization algorithm, genetic-entropic algorithm, is proposed. This optimization algorithm is based on the genetic algorithms and the natural selection via entropic sampling. With the entropic sampling, this algorithm helps to escape local optima in the complex optimization problems. To test the performance of the algorithm, we adopt the *NK* model (*N* is the number of bits in the string and *K* is the degree of epistasis) and compare the performances of the proposed algorithm with those of the canonical genetic algorithm. It is found that the higher the *K* value, the better this algorithm can escape local optima and search near global optimum. The characteristics of this algorithm in terms of the power spectrum analysis together with the difference between two algorithms are discussed. [S1063-651X(98)10103-4]

PACS number(s): 02.70.Lq, 02.60.Pn, 87.53.Tf

## I. INTRODUCTION

There has been research on the optimization algorithms that mimics the biological evolution and natural selection to study complex systems. Genetic algorithms [1], evolutionary programming [2], and evolutionary strategies [3] are currently in the mainstream of the investigation. They share many aspects of evolution in common: each maintains a given population of the trial solution to the problem, imposes genetic and/or random changes to the configuration of the solution, and makes selections for survival for the next generations. Although these algorithms use the concept of evolution as the underlying principle, there are somewhat different viewpoints as to what exactly is being evolved. In the genetic algorithms, for example, simulated evolution is emphasized on the genotypic level, that is, on the level of the string of the symbols (chromosome), via crossover and mutation. However, in the evolutionary programming and evolutionary strategies, evolution is realized by implementing the change in the adaptation and the diversity of the populations of the organisms.

In the usual evolutionary optimization algorithms, even though the selection scheme can be performed in any of several ways, selections almost always use the fitness (or energy) function as the measure of the survival for the next generation. (From now on, we use fitness and energy interchangeably). That is, selection is implemented by giving higher probability to the fitter individuals to survive for the next generation. Partially due to the selection rules and genetic operators that are designed to exploit better fitness in the population, one encounters problems of falling into one of many local optima in the optimization of the complex systems. In the canonical genetic algorithms, for example, once a configuration of a system reaches one of the local minima, it is apt for other configurations to be accumulated in the local minimum, which leads to a premature convergence. This premature convergence is manifested as a premature loss of diversity in the population and, as a consequence, it is hard to search as many configurations of the solution as possible. The mutation operator by itself is often not enough to prevent this premature convergence. To avoid this difficulty, it is necessary to maintain two basic strategies in the general optimization problems: ''exploring'' robustly as many configurations as possible while ''exploiting'' the best solutions to the given population.

Entropic sampling [4] provides a way to overcome this difficulty. This sampling directs the evolution to lower entropy of the system. In this way, the rate of rare configurations to be selected is higher than that of abundant configurations. As a result, if a configuration falls into a local minimum and many are piled up, the rate of the acceptance of configurations concentrated near the local minimum is highly suppressed, which leads the system to escape easily from the local minimum. Therefore, by utilizing the entropic sampling in the acceptance of the offspring, one can overcome the high barriers between local minima and continue to search for the global or nearly global minimum. Contributions were made to emphasize the effect of the entropic sampling in the genetic algorithms [5,6] and the simulated annealing [7].

This work, which is based on entropic sampling and genetic algorithms, attempts to provide an optimization algorithm based on the neutrally selective genetic change in the level of the chromosome and the entropy-based selection. From this one can view the evolution as an interplay between variation and selection. More specifically, variation is carried out by the random drift of the genotype by choosing two parents randomly and performing genetic operations (crossover and/or mutation) applied to the genotypic level and natural selection takes place via the entropic sampling. In particular, acceptance of the new offspring is determined by entropic sampling, which also provides a natural link between the behavior of the genotypic level and that of the phenotypic level.

To test the performance of the proposed algorithm, the *NK* model (*N* is the number of bits in the string and *K* is the degree of epistasis) [8], a generalization of the spin-glass model, was chosen with a tunable ruggedness parameter *K*. We demonstrate that the genetic search under the guidance

*Electronic address: clee@logos.etri.re.kr

 

of entropic sampling yields better results than the conventional genetic algorithms especially for rugged landscapes.

This contribution is organized as follows: In Sec. II, we discussed the idea of entropic sampling. This is followed by the procedure of the genetic-entropic algorithm and the role of the entropic sampling to the proposed algorithm. The simulation results and the characteristics of the algorithm using NK model are presented in Sec. IV. The last section is devoted to the conclusion.

## II. ENTROPIC SAMPLING

In the usual Monte Carlo simulation, the importance sampling comes in if one chooses $P_x$, the sampling probability of having configuration $x$, to be proportional to the Boltzmann factor, $\beta E(x)$,

$$P_x = Ce^{-\beta E(x)}, \tag{1}$$

where $C=\Sigma_x e^{-\beta E(x)}$ and $\beta=(kT)^{-1}$. The probability $P_B(E)$ that a system has an energy in a small range between $E$ and $E+\delta E$ can be obtained simply by adding the probabilities for all states whose energy lies in this range; i.e.,

$$P_B(E) = \sum_{(x:E<E_x<E+\delta E)} P_x. \tag{2}$$

Since all these states are equally probable, one needs to multiply the number of the state (or configuration) in this energy range. Therefore, the probability $P_B(E)$ becomes

$$P_B(E) = C\Omega(E)e^{-\beta E} = e^{S(E)/k-\beta E}. \tag{3}$$

Here, $\Omega(E)$ is the number of the possible states with the energy $E$ and entropy is defined as

$$S(E) = k\ln\Omega(E). \tag{4}$$

$k$ is the Boltzmann constant, which will be set to $k=1$ for convenience. Usually the $\Omega(E)$ is a rapidly increasing function of $E$. This together with the rapidly *decreasing* factor $e^{-\beta E}$ in the above equation results in a maximum probability when the free energy $F=E-TS$ is minimum. The larger the system, the shaper the maximum in $P_B(E)$. Thus the probability of the occurrence of the lowest-energy state is very small in this sampling method. This sampling method is generally a good method, but it fails to satisfy the ergodicity if there exist many high barriers between all the possible energy configurations. That is, it is hard to search the whole configuration space if the system has many deep local minima.

One way to overcome this difficulty is to relax the Boltzmann factor. By replacing $\beta E$ by some arbitrary function $J(E)$, we have a freedom to choose $J(E)$ in any way we want. In particular, if we set $J(E)=S(E)$, then the probability of the occurrence of a configuration with energy $E$ is inversely proportional to the exponential of the entropy:

$$P_x \propto e^{-S(E(x))} \tag{5}$$

or

$$P_E(E) \propto \Omega(E)e^{-S(E(x))}. \tag{6}$$

It should be noted that, in this way, the probability of having energy $E$ is the same irrespective of the energy. Hence this sampling method provides for a random walk through the system's energy space. In the energy space, this sampling method is designed to access all the fitness space with an equal probability.

One general way to produce random variables with a given probability distribution is known as the Metropolis algorithm [9]. In this algorithm, a trial configuration represented by $x'$ is accepted or rejected with respect to the test configuration $x$ according to the ratio $r$, which is known as the detailed balanced condition,

$$r \equiv \frac{W(x\rightarrow x')}{W(x'\rightarrow x)} = e^{\{S(E(x))-S(E(x'))\}}, \tag{7}$$

where $W$, transition probability, depends only on the change in fitness. If $r\geqslant 1$, then the trial configuration is accepted. If $r<1$, the configuration is accepted with a probability $r$. This latter can be accomplished by comparing $r$ with a random number generated from the uniform distribution in the interval between 0 and 1.

To achieve the sampling with the entropy of the system, we need to know the entropy of the system. In general, the entropy $S(E)$ is not known *a priori*. To know the exact entropy is in fact equivalent to solve the problem in the first place. We, however, can estimate entropy by short simulation starting with an incorrect test entropy $S(E)=0$ for all energy and successively simulating with an ever better estimate of the true entropy. This optimization method is based on the feedback information about the entropy of the system, which can be obtained by the following scheme [4,7]: (i) start with a test entropy $J(E)=0$. (ii) From the local simulations with the probability distribution of Eq. (7) obtain the histogram $H(E)$ of the states with fitness between $E$ and $E+\delta E$. (iii) corrected entropy $S(E)$ is given as [4]

$$S(E) = \begin{cases} J(E) & \text{if } H(E)=0 \\ J(E)+\ln H(E) & \text{otherwise.} \end{cases} \tag{8}$$

## III. GENETIC-ENTROPIC ALGORITHM

In practice, the genetic-entropic algorithm consists of the following steps:

*Step 1*. Obtain a rough estimate of the entropy and prepare the initial test configurations: from the random configurations (or chromosome) of the system, calculate the energy of configurations and their histograms. Estimate the entropy according to the following formulas:

$$S(E) = \ln H(E), \tag{9}$$

where $H(E)$ is a histogram of the energy $E$. Or, we can start with $S(E)=0$ for all $E$.

Prepare $N$ random configurations $P_1, P_2, \ldots, P_N$. Choose randomly an initial test configuration $x$ from the initial population and calculate the energy $E(x)$ and the entropy $S(E(x))$ of the configuration.

*Step 2*. To obtain one offspring $x'$ represented by $C_{k+1}$ in Fig. 1, select parents $(P_i, P_j)$ randomly from the population and apply the crossover and mutation operators. One off-

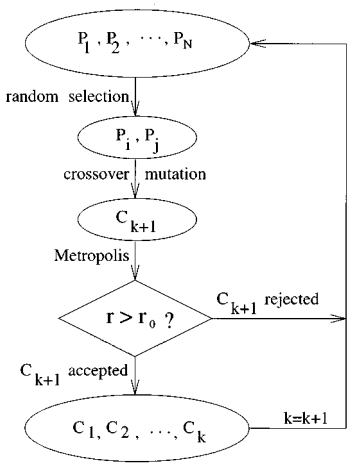


FIG. 1. Pictorial demonstration of the step 2 of the algorithm.

spring $x'$ is chosen randomly out of two offspring. Calculate the energy $E(x')$ of the new configuration $x'$ and its corresponding entropy $S(E(x'))$. If the energy is higher than the wall, the new configuration is rejected (we will discuss this below). To accept $x'$ as a new configuration, compare $S(E(x'))$ with $S(E(x))$ of the $x$, the latest accepted configuration represented by $C_k$ in Fig. 1. This can be done by evaluating the ratio $r$ from Eq. (7). If $r \geqslant 1$, then the new configuration $x'$ is accepted and if $r<1$, then $x'$ is accepted with the probability $r$. This can be done by comparing $r$ with $r_0$, a uniform random variable from 0 to 1. If the configuration $x'$ is accepted, then the corresponding histogram is added by 1 unit, that is, $H(E(x')) \rightarrow H(E(x'))+1$ and the trial configuration $x'$ becomes test configuration $x$. If rejected, select parents from the population randomly and repeat the above procedure until a new configuration is accepted.

Repeat this procedure until the number of the accepted configurations is the same as that of the population. Now, the set of the accepted configurations $(C_1, C_2, \ldots, C_N)$ becomes the new population for the next generation. Figure 1 shows the pictorial explanation of this step.

*Step 3*. Repeat step 2 until one reaches the desired generations for the entropy update. This completes one entropy evolution unit, that is, the estimated entropy is not changed during this step. In this step, the number of generations for the entropy update is a parameter that needs to be chosen. After one completes one entropy evolution unit, one needs to update the entropy.

*Step 4*. With the updated histograms, estimate *new* entropies $S(E)$ for each $E$ as follows:

$$S(E)=\begin{cases} J(E) & \text{if } H(E)=0 \\ J(E)+\ln H(E) & \text{otherwise} \end{cases} \tag{10}$$

where $H(E)$ is the unnormalized histogram of the sampling. If $E_{\min}$ is the minimum fitness obtained so far, then for $E<E_{\min}$ we extrapolate the entropy $S(E)$ with the slope

$$\frac{S(E_{\max})-S(E_{\min})}{E_{\max}-E_{\min}} \tag{11}$$

This extrapolation expedites the evolution of the system to the desired direction. One can of course extrapolate other ways for example, a more or less steeper slope. This may relate to the faster or slower annealing rate.

*Step 5*. With the new estimate of $S(E)$, repeat steps 2 through 4 to the desired number of the entropy evolution.

One more ingredient in this algorithm is the placement of the hard wall. We particularly define $E_{\max}$ as the location of the wall and $E_{\min}$ as the minimal energy obtained so far. By fixing the interval between $E_{\max}$ and $E_{\min}$, annealing can be achieved as a better $E_{\min}$ is found. We also choose the size of the sampling interval to be fixed and large enough to allow fluctuations throughout the annealing process. The wall energy is the maximum energy that each chromosome can have, beyond which all attempts are rejected. This wall-energy mimics a slow lowering of the temperature in the simulated annealing and the annealing is achieved by reducing the value of wall energy in the annealing direction. The exact value of the wall energy is not important as long as it is large enough.

The main differences between the conventional genetic algorithms and this algorithm are twofold: Firstly, in the conventional genetic algorithms, the better fitness a chromosome has, the higher its probability of being selected in the reproduction process. This implies that the chromosome of better fitness function *always* has higher probability to survive. In the proposed algorithm, however, it is not the fitness but the entropy that is the measure of the survival for the next generation. Thus, even if a chromosome has a worse fitness, it could have a higher probability of being selected if it falls in rare configurations. Secondly, in the conventional genetic algorithms, competitions are made in the course of the reproductions, that is, competitions occur in the selection procedure among the parents. Once the reproduction is carried out, the offsprings are produced in parallel and there is no direct correlation among offspring except through their parents. Whereas in the proposed algorithm, parents are selected randomly and the competitions are made in the course of the acceptance of the new configurations rather than in the level of parents. Each produced offspring is tested (to be accepted or rejected) against the previously accepted offspring through the Metropolis algorithm. That is, there is a probability competition between the two subsequent offspring. In this way, the next generation consists of offspring that are random samples of the desired probability distribution.

With this sampling method, the system can escape from local minima with relative ease. In the genetic algorithm, once a population falls in a local minimum, it tends to stay in the local minima. This leads to premature convergence. The main operation that prevents this premature convergence is mutation. Mutation alone, however, is sometimes not sufficient. When mutation probability is small, it is hard to get out of the local minimum, especially for a very rugged fitness landscape. When the mutation probability is large enough to escape the local minimum, it is apt to yield a random search that is far from the optimization. Whereas in the genetic-entropic algorithm, the probability of finding the system in the configuration $x$ is proportional to the entropy of the system,

$$P_E(x) \propto e^{-S(E(x))}, \tag{12}$$

that is, the offspring are encouraged to decrease the entropy. Thus once the population near a local minimum increases, the corresponding entropy near the local minimum becomes large. Consequently, the rate of acceptance of a configuration near the local minimum becomes smaller, which helps to escape from the local minimum.

The selection rules of the proposed algorithm for the offspring help one to search and maintain good schemata. Once an offspring finds a good schema, the next offspring tend to maintain the schema because of the Metropolis algorithm. Thus, the new schema remains robust against the crossover or mutation operators in the following generations. As a result, there is better chance of finding the global minimum.

There is one drawback to this algorithm. As we need to update the entropy after certain generations, it is difficult to determine when we need to update the entropy. This certainly depends on the problem at hand and some trial and error have to be carried out. It is, however, found that, at least in NK model, the performance is not very sensitive to the choice of the generation of the entropy update.

## IV. EXPERIMENTAL RESULTS AND DISCUSSIONS USING THE NK MODEL

The proposed algorithm is tested against the *NK* model. There are two major reasons to choose the *NK* model to demonstrate the effectiveness of the proposed algorithm. Firstly, this model provides a natural means of the coding. It is free of the representation problem and there is no ambiguity of the definition of the fitness. In this way, one can compare the proposed algorithm with the canonical genetic algorithms without having dependence of the coding and the ordering problems. (Coding and ordering problems in the genetic algorithms do occur in other problems, such as traveling salesman problem (TSF). See Ref. [10] for details.) Secondly, in this model, the ruggedness can be controlled [8]. That is, by adjusting $K$, one can tune the ruggedness of the fitness landscape of the model so that one can have a tunable number of local minima and as a result can test the performance of the escaping from local minima.

The *NK* model defines a class of parametrizable fitness landscape whose search space is composed of all possible configurations of $N$ loci, denoted by $\mathbf{s}=\{s_i\}, i=1,\ldots,N$. It is a simple class of landscapes whose ruggedness can be controlled by the parameter $K$, the degree of epistasis. $K$ is the number of other loci on which the fitness contribution of each locus $i$ depends. For each locus $i$, the $K$ other loci can be chosen either randomly or according to some topology. If we restrict possible values of each locus $s_i$ to be 0 and 1, then the size of the search space is $2^N$. If we denote $\vec{n_i}=(n_1,n_2,\ldots,n_K)$ as the set of the $K$ loci on which the fitness of the locus $i$ depends, then the interaction space $(s_i,\vec{n_i})$ consists of $2^{K+1}$ possible configurations of $K+1$ alleles and assign a randomly chosen fitness $E_i$ between 0 and 1 to each such configuration. The fitness of the each chromosome is then defined as the average of the fitness for each locus:

$$E(\mathbf{s})=\frac{1}{N}\sum_{i=1}^{N} E_i(\mathbf{s}). \tag{13}$$

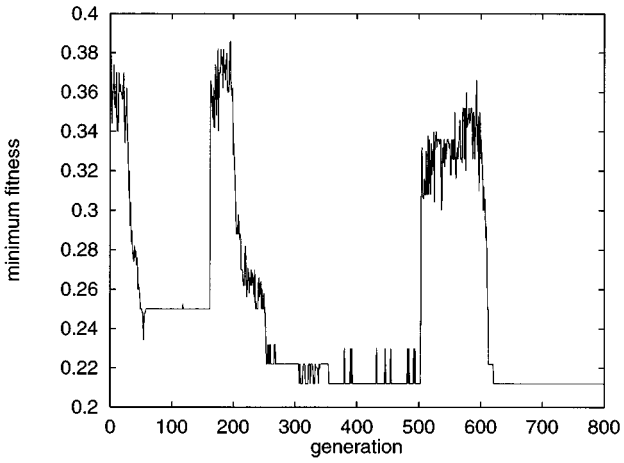


FIG. 2. Minimum fitness as a function of generation with $N=50$ and $K=8$. We plot the minimum fitness up to the 800th generation. The unit of the fitness in the *NK* model is dimensionless.

The advantage of using the *NK* model lies in its tunable ruggedness. $K=0$, for example, yields that all loci are independent among others and there is no ruggedness in the fitness landscape. As $K$ increases, each locus is more dependent on other loci, which, in turn, gives more ruggedness in the fitness landscape. At $K=N-1$, the maximum possible value of $K$, every locus depends on every other locus so maximum ruggedness occurs [11].

The proposed and the conventional genetic algorithms are tested against the NK model with various $K$. The $K$ other loci for each locus are chosen randomly. The parameters that we have used for both algorithms are the following: (i) number of the population: 500; (ii) crossover operator: 1 point crossover operator; (iii) mutation probability: 0.02; (iv) selection: tournament selection with tournament size 5, for the genetic algorithm only; (v) entropy update: every 50 generations, for the genetic-entropic algorithm only; (vi) number of generation: 2000 generations.

Figure 2 shows a typical behavior of the minimum fitness as a function of generations. As can be seen from Fig. 2, the system reaches a local minimum and stays near it until we reach approximately the 150th generation. During the period when the system stays near the local minimum, the histograms in the vicinity of the local minimum pile up. As a result, when the entropy is updated at the 150th generation, the entropy in the vicinity of the local minimum is large enough so that entropy in the lower fitness configuration becomes larger than that of higher fitness. This causes the higher fitness configurations to have higher probabilities to be accepted from Eq. (8). Therefore, selection of the offspring tends toward the higher fitness state, which is manifested as the first bump in Fig. 2. After producing configurations of higher fitness for some generations, the entropy in the higher fitness area becomes large and the updated entropy forces the system to produce the offspring of lower fitness. In this way, the population can get out of the local minimum and look for the global minimum. As the system repeats this process, it reaches a lower fitness than reached by previous systems.

In Figs. 3 and 4, we compare the performance of the proposed algorithm with the conventional genetic algo-

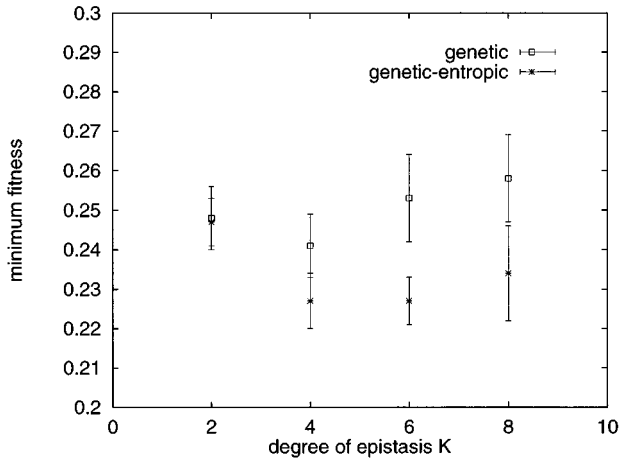


FIG. 3. Comparison of the performance of the genetic algorithms and that of the genetic-entropic algorithm. The minimum fitness (dimensionless quantities) after the 2000th generation is plotted as a function of $K$. The top data are from the genetic algorithms and the bottom are from the genetic-entropic algorithm. The data are obtained from $N=50$ and for each $K=2$, 4, 6, 8. For each $K$, 10 independent trials are averaged and the error bars denote corresponding standard deviations.

rithms. We especially plot the minimum fitness of the $NK$ model obtained by the proposed algorithm and the conventional genetic algorithm as a function of the epistasis $K$. As $K$ becomes larger, we can see the clear difference, that is, the proposed algorithm performs better as the system becomes more rugged. This supports the proposition that the proposed algorithm can search the global minimum by escaping many local minima.

To see the population effect of these algorithms, we performed another experiment. In this experiment, we fixed all parameters except the number of the population. Table I shows the result of the experiment with $N=30$, $K=7$. As we notice, the proposed algorithm shows better results over all populations. From this experiment, one can conclude that the performance of the algorithm is independent of the number of the population.

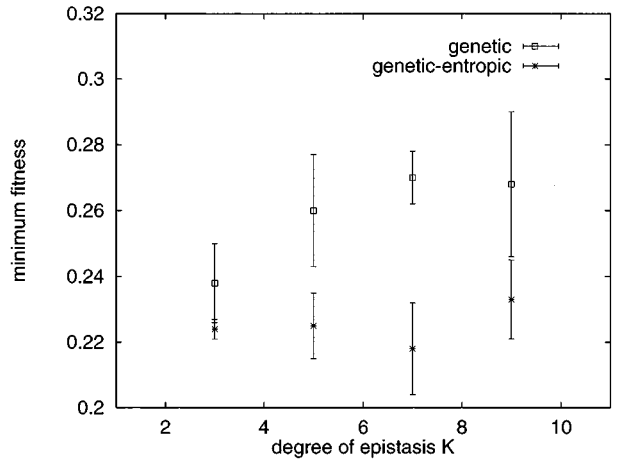


FIG. 4. Same as Fig. 3. The data are obtained from $N=30$ and for each $K=3$, 5, 7, 9. For each $K$, 10 independent trials are averaged and the error bars denote corresponding standard deviations.

TABLE I. Performance comparison of the both algorithms with various size of populations. The data are obtained for $N=30$ and $K=7$. Top data are the minimum fitness (units are dimensionless) obtained from the genetic-entropic (GE) algorithm and bottom from the conventional genetic algorithms (GA). For each set, 10 independent trials are averaged and corresponding standard deviations are in the parentheses.

| Population | 10 | 30 | 50 | 100 | 300 | 500 |
|---|---|---|---|---|---|---|
| GE | 0.23 | 0.22 | 0.23 | 0.21 | 0.21 | 0.21 |
| | (0.02) | (0.02) | (0.01) | (0.01) | (0.02) | (0.02) |
| GA | 0.27 | 0.27 | 0.27 | 0.25 | 0.24 | 0.24 |
| | (0.01) | (0.02) | (0.01) | (0.02) | (0.02) | (0.02) |

To study the characteristics of the algorithm, we evaluated the power spectrum $F(\omega)$ of the fitness. Since we accept offspring sequentially via the Metropolis algorithm, the set of fitness $E(x)$ of the subsequent offspring can be regarded as a time series. To see the effect of the entropic sampling, we carried out the following experiment with many generations: (i) $N=50$, $K=4$; (ii) number of the population: 500; (iii) number of the sample: 5 equally spaced samples for each generation; (iv) entropy update: every 200 generations; (v) number of generations: $2\times10^5$ generations; (vi) total number of data: $1\times10^6$.

A typical power spectrum outcome is shown in Fig. 5. The power spectrum of the statistical time series approximately forms a single power law in the form $F(\omega)\propto\omega^{-\alpha}$ as functions of the frequency $\omega$. It is found that the scaling exponent $\alpha$ is remarkably different for the low, middle, and high frequency states: for the low and high frequency ranges, $\alpha$ is close to 0, while $\alpha$ is far from 0 for the middle frequency range.

To analyze this further, we used the following relation: $\Delta\omega=2\pi/N\Delta t$, where $N$ is the total number of the data and

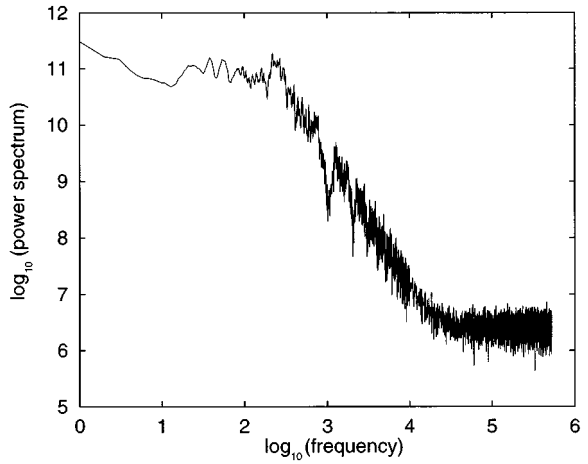


FIG. 5. Full power spectrum of the times series of the fitness. The power spectrum $[F(\omega)]$ versus frequency $(\omega)$ is plotted logarithmically on the base of 10. The unit of frequency is $\Delta t^{-1}$, where $\Delta t$ is the sampling interval. The power spectrum can be divided into three regions: low frequency region, from $\omega=1$ to $\omega\approx500$, which corresponds to the twice entropy update, middle frequency region, from $\omega\approx500$ to $\omega\approx50\,000$, which corresponds to about 4 generations, high frequency region, from $\omega\approx50\,000$ to $\omega\approx5\times10^5$, which corresponds to each fitness unit.

$\Delta\omega$ and $\Delta t$ are the frequency and time intervals, respectively. By setting $\Delta\omega=1$ for convenience, we have $\Delta t=2\pi/N$. Since the set of the time series consists of the real part only, the frequency $\omega_{\max}$ is given as

$$\omega_{\max}=\frac{1}{2}\frac{2\pi}{\Delta t}=\frac{N}{2}. \tag{14}$$

From this, we can express any frequency $\omega$ in terms of $\omega_{\max}$:

$$\omega=\frac{2}{n}\omega_{\max}, \tag{15}$$

where $n$ is the number of data corresponding to the frequency $\omega$.

At high $\omega$, $F(\omega)$ tends to become constant. Note that $\alpha=0$ corresponds to a random walk, thus in the high frequency range, the time series of the fitness function can be interpreted as a stochastic process. This implies that there is no correlation in this frequency range. Since this high frequency range corresponds to the time series of a few generation units, within a few generations the probability of subsequent offspring to be accepted is independent of the fitness. This is in accordance with the entropic sampling nature of the algorithm. At low $\omega$, $F(\omega)$ again tends to become constant up to a few entropy update units. This implies that after the entropy is updated, the algorithm again searches the fitness space with an equal probability, indicating that for the long time scale the entropic sampling is also achieved.

In the middle frequency range, however, we have nontrivial power behavior. Notice that the total time series contains the effect of the entropy updating, which we set by hand as a parameter. Thus, in order to see the correlation between generations within each entropy update unit, we calculate the power spectrum of the time series up to the entropy update instead of having full time series. This is shown in Fig. 6. After eliminating the entropy updating effect, we still see a nonzero exponent indicating a long range correlation. They are correlated such that low fitness is likely to be close (in the time series sense) to each other and the same is true for high fitness. This implies that once a good fitness is found, next fitness tends to be good or better than the previous one.

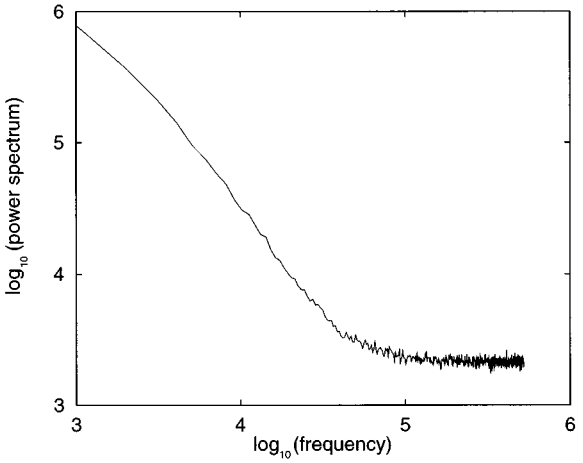


FIG. 6. The power spectrum $[F(\omega)]$ of the fitness versus frequency $(\omega)$ up to the entropy update generation. The low frequency region again starts at about $\omega\approx 50\,000$. Again, the power spectrum $[F(\omega)]$ versus frequency $(\omega)$ is plotted logarithmically on the base of 10 and the unit of the frequency is $\Delta t^{-1}$, where $\Delta t$ is the sampling interval.

## V. CONCLUSION

We propose an optimization algorithm based on the genetic algorithms and entropic sampling. This algorithm exhibits merits of the crossover operator in the genetic algorithms and utilize the idea of the entropic sampling. We apply this algorithm particularly to the NK model and compare the performances of the algorithm with that of the conventional genetic algorithms. By using this algorithm, we obtain better performance than the conventional genetic algorithms. As can be seen from the above section, this algorithm performs better especially for the higher $K$, rugged landscape.

In the conventional simulated annealing [12], configurations are updated locally, that is, a trial configuration does not differ much from the configuration against which the trial configuration is tested. Consequently, an evenly scattered search in the configuration space is hard to accomplish. Whereas, in the genetic algorithms, offspring can be updated globally, namely, offspring can have quite different configurations from their parents due to the crossover operator. However, in the selection scheme, configurations of higher fitness always have a higher probability of being selected, thus it is hard to escape from the local minima. The proposed algorithm tends to overcome these shortcomings from the simulated annealing and the genetic algorithms.

Since the entropic sampling is a general selection scheme, it will be also interesting to apply the entropic sampling method to other evolutionary optimization algorithms, such as evolutionary programming and evolutionary strategies. As we have mentioned in Sec. III, we have an additional parameter, namely, entropy update generation, that has to be determined. It is desirable to have some criteria in which this parameter can be fixed. The global convergence property [13] of this algorithm is also important and this problem is under investigation. The mechanisms behind the algorithm remain to be investigated further together with applications to other optimization problems. The comparison of the proposed algorithm with the simulated annealing and other algorithms is also under way.

## ACKNOWLEDGMENTS

The authors are very grateful to Dr. E. H. Lee for his encouragement and support. This research was supported by the Ministry of Information and Communications, Korea. S. K. Han was also partially supported by SRC program of Seoul National University, Korea.

[1] D. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning* (Addison Wesley, New York, 1989).
[2] D. Fogel, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence* (IEEE Press, New York, 1995).
[3] T. Bäck and H.-P. Schwefel, Evolutionary Comput. **1**, 1 (1993).
[4] J. Lee, Phys. Rev. Lett. **71**, 211 (1993).
[5] C.-Y. Lee and S. K. Han, in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, edited by T. Bäck, Z. Michalewicz, and X. Yao (IEEE, New York, 1997), p. 31.
[6] K. K. Lee and S. K. Han (unpublished).
[7] J. Lee and M. Y. Choi, Phys. Rev. E **50**, R651 (1994).
[8] S. A. Kauffman, *Lectures in the Science of Complexity*, edited by D. Stein, Santa Fe Institute Studies in the Science of Complexity, Lecture Vol. I (Addison Wesley, New York, 1989), p. 527.
[9] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).
[10] Z. Michalewicz, *Genetic Algorithms+Data Structure =Evolution Programs* (Springer-Verlag, Berlin, 1992).
[11] E. D. Weinberger, Phys. Rev. A **44**, 6399 (1991).
[12] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Science **220**, 671 (1983).
[13] G. Rudolph, IEEE Trans. Neural Netw. **5**, 96 (1994).