Introducing Assignment Functions to Bayesian Optimization Algorithms

Masaharu Munetomo [a], Naoya Murao [b], Kiyoshi Akama [a]

[a] *Division of Large-scale Computing Systems, Information Initiative Center, Hokkaido University, Sapporo, Hokkaido 060-0811, JAPAN.*

[b] *Division of Systems and Information Engineering, Graduate School of Engineering, Hokkaido University, Sapporo, Hokkaido, 060-0811, JAPAN.*

Abstract

In this paper we improve Bayesian optimization algorithms by introducing proportionate and rank-based assignment functions. A Bayesian optimization algorithm builds a Bayesian network from a selected sub-population of promising solutions, and this probabilistic model is employed to generate the offspring of the next generation. Our method assigns each solution a relative significance based on its fitness, and this information is used in building the Bayesian network model. These assignment functions can improve the quality of the model without performing an explicit selection on the population. Numerical experiments demonstrate the effectiveness of this method compared to a conventional BOA.

Key words: evolutionary computation, Bayesian optimization algorithms, assignment functions

1 Introduction

In order to realize a truly robust search algorithm, one that is applicable to a wide spectrum of applications, a variety of evolutionary algorithms have been proposed. Genetic algorithms (GAs) maintain a population of solution candidates in the problem domain, each solution represented by a parameter string, and arrive at an optimal or near-optimal solution by applying selection, recombination, and mutation operators over repeated generations.

Email addresses: munetomo@iic.hokudai.ac.jp (Masaharu Munetomo), naoya.m@cims.hokudai.ac.jp (Naoya Murao), akama@cims.hokudai.ac.jp (Kiyoshi Akama).

It is essential to ensure tight linkage for effective genetic search through recombination operators such as crossovers. When a subset of loci in the parameter space is found to form a building block (BB) — a good candidate for a subsolution of the larger problem — these parameters should be "tightly" encoded on a string. In other words, they should be closely linked to each other in such a way that recombination operations will preserve the BB when a new generation of solutions is created. Conventional recombination operators such as one-point crossover will easily disrupt BBs if they are not tightly linked on a string.

In the early history of GA applications, it was common to employ problem-specific knowledge to ensure the tight linkage of associated parameters. This advance knowledge is not always available when starting a new genetic search, so it is sometimes difficult to ensure tight linkage by this approach.

More advanced GAs have recently been proposed that can learn or identify linkage groups, either directly or indirectly, without the need for problem-specific knowledge. The "Linkage Learning" GA, for example, applies two-point crossovers on circular strings to obtain indirect information on important linkages. The Linkage Identification with Nonlinearity Check (LINC) algorithm [12], on the other hand, pioneered a number of techniques for directly identifying linkage groups by detecting nonlinear behavior under pair-wise perturbations.

In addition to these linkage identification techniques, estimation of distribution algorithms (EDAs) have been introduced that obtain linkage information indirectly by estimating the probable distribution of alleles in a set of promising solutions.[13] Unlike classical GAs, EDAs do not apply genetic operators to individual candidate solutions. Instead they generate offspring based on the estimated probabilistic distribution of a population of promising solutions. Linkage information is implicit in these distributions, so we do not need to ensure tight linkage in advance.

In this paper, we propose a novel approach of EDAs employing assignment functions in the estimation process of Bayesian optimization algorithm (BOA)[15–17,19,20], one of the most sophisticated approach in EDAs. The BOA first selects a set of promising solutions from the original population, using the information in these candidates to estimate the probabilistic distribution of the solution space. Aside from this selection step, however, differences in fitness among the population are not taken into account — the chosen solutions are treated equally in building the model distribution. In order to improve the accuracy of the model, one can introduce a function that assigns different relative weights to individual solutions according to their significance. By employing these assignment functions, we can simplify the algorithm, making it unnecessary to perform explicit selections — the BOA simply repeats

model-building and generation of solution candidates based on the model.

This paper is organized as follows. First we review EDAs briefly, including the BOA proposed by previous studies. Second, we introduce our assignment functions and show how they can improve the model-building accuracy of the BOA. Finally we perform numerical experiments, and compare the effectiveness of the proposed method with conventional approaches.

2 Bayesian Optimization Algorithms

In the field of evolutionary computations, Estimation of Distribution algorithms (EDAs) were first introduced by Mühlenbein et al. [13] EDAs, which are also called probabilistic model-building genetic algorithms (PMBGAs) [6,7], have been studied in the context of realizing a robust evolutionary search without *a priori* knowledge of linkage in the problem. EDAs are population-based search algorithms, and thus similar to GAs in many respects; they do not, however, employ explicit genetic operators such as crossovers and mutations. Instead they build probabilistic models from a collection of promising solutions — those strings with higher (for maximization problems) or lower (for minimization problems) fitness values — and employ this model to create the next generation of strings. In general, EDAs perform the following sequence repeatedly to obtain optimal solutions:

(1) randomly initialize a population of N strings.
(2) sort strings by fitness and select M ($M < N$) strings with high fitness (for maximization problems).
(3) create a probabilistic model base on the selected strings.
(4) create a new generation of N strings based on the model.
(5) if the specified termination condition is not met, go to (2).

The underlying idea of EDAs is that if we can detect some bias in the fitness distribution of the string population, then we can expect to generate relatively good solutions from a model based on an appropriately selected sub-population.

A variety of EDA methods have been proposed, such as PBIL (Population-Based Incremental Learning) [1], UMDA (Univariate Marginal Distribution Algorithm) [13], BMDA (Bivariate Marginal Distribution Algorithm) [14], and so on. CGA (compact GA)[6] and ECGA (extended compact GA) [7] have also been proposed as PMBGAs. Detailed reviews of EDAs and PMBGAs are available in a survey article by Pelikan [18] and a textbook by Larrañanga and Lozano [8].

The Bayesian optimization algorithm (BOA), proposed by Pelikan et al. [15,16], is considered one of the most sophisticated EDAs. BOA builds probabilistic models based on Bayesian networks. The Bayesian networks employed in BOA are composed of nodes and directed links. Each node X_i ($i = 0, \cdots, l-1$, where l is the string length) of the network represents a random variable: the probability that locus s_i has the value 1. A directed link is established from node i to node j in order to indicate that node i is a parent of node j. The probability distribution of each random variable X_i can be written as $p(X_i|\prod_{X_i})$, where $\prod_{X_i}$ is the set of parents of X_i: that is, the set of nodes connected to node i. The overall probabilistic distribution over the string X is thus determined by the joint distributions as follows:

$$p(X) = \prod_{i=0}^{n-1} p(X_i|\prod_{X_i}). \tag{1}$$

BOA performs its model-building and search processes according to the following sequence [16]:

(1) set $t \leftarrow 0$, randomly generate an initial population $P(0)$
(2) select a set of promising strings $S(t)$ from $P(t)$
(3) construct the network B using a chosen metric and constraints
(4) generate a set of new strings $O(t)$ according to the joint distribution encoded by B
(5) create a new population $P(t+1)$ by replacing some strings from $P(t)$ with strings from $O(t)$, and set $t \leftarrow t+1$
(6) if the termination criteria are not met, go to (2).

Similar to a GA, BOA initializes a population of randomly generated binary strings and selects a subset of the most promising solutions with relatively high (in maximization problems) fitness values. The Bayesian network B is constructed using these strings based on a specified metric and set of constraints. In its original form, BOA measures the quality of the network with the Bayesian Dirichlet (BD) metric [4] $p(D, B|\xi)$:

$$\begin{aligned} p(D, B|\xi) = p(B|\xi) \prod_{i=0}^{n-1} \prod_{\pi_{X_i}} \frac{\Gamma(m'(\pi_{X_i}))}{\Gamma((m'(\pi_{X_i}) + m(\pi_{X_i})))} \\ \times \prod_{x_i} \frac{\Gamma((m'(x_i, \pi_{X_i}) + m(x_i, \pi_{X_i})))}{\Gamma(m'(x_i, \pi_{X_i}))} \end{aligned} \tag{2}$$

where D is the data set which means a set of promising solutions, $p(B|\xi)$ is a prior probability of network B, Γ function is defined as $\Gamma(a) = (a-1)!$, $m(\pi_{X_i})$ represents the number of data where $\prod_{X_i}$ equals to π_{X_i} in D, $m(x_i, \pi_{X_i})$ is the number of data where X_i is x_i and $\prod_{X_i}$ is π_{X_i} in D. ($m(\pi_{X_i}) = \sum_{x_i} m(x_i, \pi_{X_i})$).

$m'(x_i, \pi_{X_i})$ is obtained with $p(B|\xi)$ from prior information of the network $m(x_i, \pi_{X_i})$ concerning the given problem.

As for the prior probability $p(B|\xi)$, Heckerman et al. [4] suggest a simple assignment function such as $p(B|\xi) = c\kappa^{\delta}$.[16] Here, c is a normalization constant, $\kappa \in (0, 1]$ is another constant parameter, and δ is the number of edges that are different in network B and in the network for prior probabilities.

The maximum degree of the network (i.e., the maximum number of links connected to a node) is constrained by the value k. In order to search for the selected strings, BOA performs a greedy search to minimize the metric under this constraint. The greedy algorithm is outlined below.[16]

(1) initialize the network B (e.g., to an empty network)
(2) choose all simple graph operations that can be performed on the network without violating the constraints
(3) pick the operation that increases the score of the network the most
(4) perform the operation picked in the previous step
(5) if the network can no longer be improved without violating the given constraints on its complexity, or when a maximal number of interactions has been reached, then finish
(6) go to (2)

To generate new instances for the next generation, BOA performs the following algorithm[16].

(1) mark all variables as unprocessed
(2) pick up an unprocessed variable X_i with all parents already processed
(3) set X_i to x_i with probability $p(X_i = x_i | \prod_{X_i} = \pi_{X_i})$
(4) mark X_i as already processed
(5) if there are unprocessed variables left, go to (2)

As for the computational complexity of a BOA, network construction — searching for the optimal network that minimizes the BD metric — tends to dominate the computational overhead. The computational complexity of a greedy search of network structures is $O(k2^k l^2 N + kl^3)$, where l is the string length, k is the maximum degree, and N is the number of strings. Assuming that k is constant, we have an overall time complexity of $O(n^2 N + n^3)$ for the network construction performed in each generation.[16]

3 Introducing assignment functions

In this paper, we introduce a method of improving the accuracy of probabilistic model-building by drawing more information from the strings' fitness values. In genetic algorithms, a wide variety of selection methods and operators have been proposed that employ fitness values. In EDAs such as the BOA, however, selection is based on a simple truncation. Thus, while fitness values are indirectly employed in the selection process, they are not directly utilized in the model-building process. Even in the most sophisticated EDA methods, fitness values are not considered in the metric and therefore are not used effectively.

In our approach, fitness values are utilized in the probabilistic model-building process. We employ an "assignment function", which assigns a weight to each string reflecting its relative significance in the population. Probabilistic models are generated based not only on the information in the selected strings, but also on their relative significance. We thus expect to improve the accuracy of the probabilistic models, which will reflect differences in the fitness of the promising solutions. This should in turn improve the quality of the offspring generated by the model.

An EDA with an assignment function is performed according to the following sequence:

(1) randomly initialize a population of strings.
(2) calculate a relative significance for each string based on the assignment function
(3) create a probabilistic model based on the strings and their significance.
(4) create a population of strings for the next generation based on the model.
(5) if a specified termination condition is not met, go to (2).

The assignment functions used here are similar to the roulette-wheel (proportionate) selections or ranking selections used in GAs [3]. We propose two types of assignment functions, each defined on a single string s of a given population.

Rank-based assignment function: $a_r(s) = g(r(s))$, where $r(s)$ is the rank of s found by sorting the population of strings according to their fitness values. (Strings with exactly the same fitness value are sorted by their order in the population, and their ranks are assigned consecutive values.) The significance $g()$ can be any non-negative, integer-valued, monotonically decreasing function.

Proportionate assignment function: $a_p(s) = [c \times f(s)/\sum_s f(s)]$, where $f(s)$ is the fitness value of string s and c is a constant. We may employ scaling methods in this approach to control fitness differences.

These assignment functions provide the model-building process with more accurate and reliable information on the fitness of the strings. Conventional EDAs can be considered as a special case of the rank-based assignment function, where $g(r(s)) = 1$ for $1 \leq r(s) \leq M$ and $g(r(s)) = 0$ otherwise. By introducing assignment functions, we expect to improve the accuracy of model-building in a BOA with less computational overhead.

The relative significance $a(s)$ is used to calculate the BD-metric terms $m(\pi_{X_i})$ and $m(x_i, \pi_{X_i})$:

$$m(\pi_{X_i}) = \sum_{s \in P(\pi_{X_i})} a(s) \tag{3}$$

$$m(x_i, \pi_{X_i}) = \sum_{s \in P(x_i, \pi_{X_i})} a(s) \tag{4}$$

$P(\pi_{X_i})$ represents the set of strings where $\prod_{X_i}$ equals π_{X_i}, and $P(x_i, \pi_{X_i})$ represents the set of strings where $X_i = x_i$ and $\prod_{X_i} = \pi_{X_i}$ in the current population. The assignment function $a(s)$ is the significance of string s in the population.

4 Empirical results

We performed numerical experiments comparing the effectiveness of the BOA with various assignment functions to the conventional BOA method.

4.1 Assignment functions

For rank-based assignment functions, we employ the three functions $AR1 \sim AR3$ shown in figure 1. The x-axis shows the rank of a string in the population, and the y-axis shows its assigned relative significance $a(s)$. The current best solutions are strongly weighted in function $AR2$, which places a relatively high selective pressure on the estimated probabilistic distributions.

We also employ the following proportionate assignment function based on the fitness $f(s)$ of string s:

$$APc(s) = [c \times \frac{\hat{f}(s)}{\sum_s \hat{f}(s)}] \tag{5}$$

$$\hat{f}(s) = \frac{f(s) - f_{min}}{f_{max} - f_{min}}, \tag{6}$$

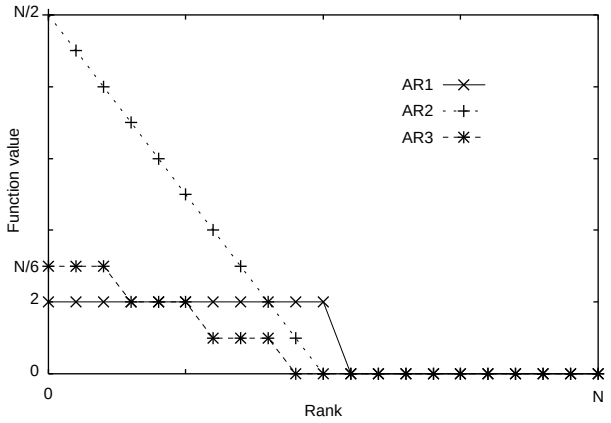

Fig. 1. Rank-based assignment functions employed in the experiments

where f_{max}, f_{min} are the maximum and minimum fitness values in the current population of strings. The constant c controls the selective pressure of the model-building process, and is set to either 1, 3, or 5 for a total of three proportionate assignment functions.

4.2 Solutions to conventional test functions

We compare the quality of various solutions to the following conventional numerical test functions where $n = 30$. Each x_i is represented by a signed, fixed-point, binary encoded variable. The optima of all functions are zero; these are therefore minimization problems.

$$\text{Sphere: } f(x) = \sum_{i=1}^{n} x_i^2,$$
$$x_i \in [-5.12, 5.12] \text{ (10bit)}$$

$$\text{Rosenbrock: } f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)] + (x_i - 1)^2,$$
$$x_i \in [-2.048, 2.048] \text{ (12bit)}$$

$$\text{Rastrigin: } f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10],$$
$$x_i \in [-5.12, 5.12] \text{ (10bit)}$$

$$\text{Ackley: } f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\cos 2\pi x_i) + 20 + e,$$
$$x_i \in [-32.768, 32.768] \text{ (16bit)}$$

$$\text{Griewank: } f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}} + 1),$$
$$x_i \in [-512, 512] \text{ (10bit)}$$

For each function we observed the best fitness values in a population consisting of 10,000 strings, after a total of 100,000 fitness evaluations in each experiment. Tables 1 through 5 show the average (AVE), standard deviation (STD), minimum (MIN), and maximum (MAX) best fitness value over 10 experiments. Results are shown for the original BOA and for the various assignment functions.

Table 1
Results for the Sphere function

	AVE	STD	MIN	MAX
BOA	9.35	1.47	7.46	12.56
AR1	9.35	1.32	6.98	11.03
AR2	7.95	1.08	6.02	9.73
AR3	7.95	1.05	6.32	9.39
AP1	9.63	1.21	7.92	11.66
AP3	10.11	1.07	8.07	11.61
AP5	10.19	1.06	7.66	11.47

Table 2
Results for the Rosenbrock function

	AVE	STD	MIN	MAX
BOA	72.47	10.18	54.96	86.50
AR1	99.44	16.08	77.85	127.27
AR2	35.20	6.31	28.49	51.52
AR3	107.88	29.12	66.03	159.74
AP1	88.03	12.46	60.11	104.84
AP3	94.80	19.69	61.95	122.00
AP5	108.60	18.07	84.34	138.29

Table 3
Results for the Rastrigin function

	AVE	STD	MIN	MAX
BOA	140.70	6.46	125.79	148.87
AR1	138.59	8.86	123.56	151.91
AR2	114.01	8.97	97.19	127.67
AR3	121.71	9.17	109.62	133.47
AP1	149.24	5.66	140.71	160.68
AP3	148.45	9.57	130.01	163.59
AP5	134.59	8.18	119.83	146.37

Table 4
Results for the Ackley function

	AVE	STD	MIN	MAX
BOA	11.20	0.38	10.69	11.85
AR1	11.15	0.17	10.90	11.52
AR2	9.77	0.28	9.42	10.21
AR3	9.80	0.35	9.17	10.24
AP1	11.16	0.42	10.23	11.63
AP3	11.48	0.40	10.55	12.00
AP5	11.79	0.44	11.09	12.42

Table 5
Results for the Griewank function

	AVE	STD	MIN	MAX
BOA	25.43	3.38	20.30	32.25
AR1	25.25	1.41	22.89	27.70
AR2	21.66	2.24	17.66	26.45
AR3	23.55	1.74	20.66	26.40
AP1	24.66	2.16	20.02	26.90
AP3	25.31	1.98	21.53	27.60
AP5	26.64	2.75	20.38	30.53

Assignment function AR2, which assigns a large relative significance to high-ranked solutions, achieves the best results for all test functions. In addition, BOA with AR2 apparently gives better results than BOA without an assign-

ment function for all test problems except the sphere (the simplest of all). Proportionate assignment functions, which employ relative fitness values instead of rankings, did not improve the quality of any solutions. This point is discussed in the next experiment, which employs test functions with uniformly or exponentially scaled traps.

4.3 The number of fitness evaluations required for K-bit trap functions

We now describe numerical experiments which compare the number of fitness evaluations needed to obtain optimal solutions. These experiments employ test functions with uniformly and exponentially scaled traps, which are also used to test linkage learning GAs.[5,9] The weighted K-bit trap functions $f(s)$ are defined as follows.

$$f(s) = \sum_{i=1}^{L} w_i f_i(u_i), \tag{7}$$

where w_i is the weight of the i-th K-bit trap sub-function f_i:

$$f_i(u_i) = \begin{cases} K - u_i - 1 \text{ if } 0 \leq u_i \leq K-1 \\ K \qquad\qquad \text{if } u_i = K \end{cases} \tag{8}$$

The function u_i represents the unitation, or the number of 1's occurring in the i-th K-bit substring s_i ($s = s_1, s_2 \cdots s_L$). In this function, a building block sub-solution must be obtained for each trap sub-function. The building blocks need to be mixed and tested through recombination to obtain a global optimal solution. This function is therefore frequently employed to test genetic and evolutionary computations, which employ crossover operations to recombine the building blocks.

The BOA is not dependent on the order of the loci in encoding strings, so we can obtain the same results even when the strings are randomly encoded. This is one of the principal merits of the BOA. Conventional GAs must use prior knowledge to preserve the linkage of their building blocks, so may fail to obtain solutions when the information is presented in a random order.

The weight w_i of sub-function f_i reflects the importance of this building block to the overall fitness, or the "signal difference". To control the difficulty of the test function, we consider the following two extreme cases in assigning signal differences.

uniform case: $w_i = 1.0$ for all i

exponential case: $w_0 = 1.0$, $w_i = 0.5 \times w_{i-1}$ for $i = 1, 2, \cdots$

In the first case all sub-functions are weighted equally, and the BOA detects all building blocks simultaneously early in the model-building process. In the exponential case, on the other hand, the most significant sub-function dominates all the other sub-functions. Only the building blocks for the most significant sub-functions will be searched in the early stages of model-building, which leads to a kind of *domino convergence*. Building blocks are obtained sequentially, from larger to smaller weights.

In the following experiments we employ the sum of 5-bit trap functions ($K = 5$), with string lengths $l = 30, 60, 90, 120$. For $l = 120$ this function has $2^{24}-1 = 16777215$ local optima, and is thus a difficult problem for conventional local search techniques. We expect that Bayesian networks can identify the sub-solutions of each trap sub-function separately.

Other conditions for the experiments are set as follows:

- The population size is optimized in each experiment to the value shown in table 6.
- The maximum degree k of nodes in the Bayesian network is set to 4.
- All data plotted in the figures represent the average result of 10 experiments.

Table 6
Population sizes for the experiments

string length	30	60	90	120
uniform case	1300	3300	5200	7400
exponential case	3000	8500	15000	35000

Figures 2 and 3 compare the overall number of fitness evaluations required to obtain optimal solutions, under uniform and exponential weighting of the sub-functions respectively. The x-axis shows the string length (problem size), and the y-axis shows the number of fitness evaluations (computational cost). The points plot the average result of 10 experiments, and the vertical bars indicate a range of one standard deviation ($\pm\sigma$).

Compared to the original BOA, algorithms incorporating assignment functions $AR2$ and $AR3$ obtain an optimal solution with less computational overhead. The results obtained by using $AR1$ are similar to those of the original BOA. These two algorithms exert essentially the same weighting pressures on the model-building process, but there should be some minor differences evident in other performance measures. This figure illustrates that the accuracy of model-building was improved by assigning a relatively large significance to the best solutions in the population. Since there is little difference between the results of $AR2$ and $AR3$, we see that the precise value of the significance assigned is not very important.

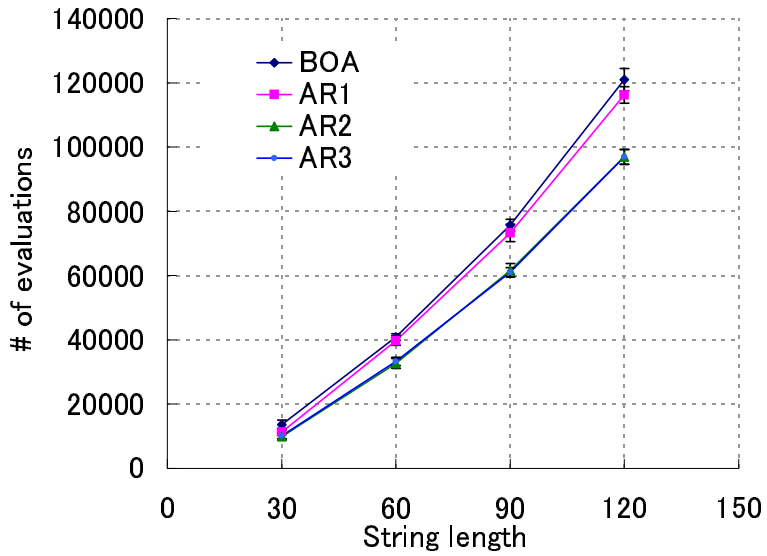

Fig. 2. Results for rank-based assignment functions (uniform case)

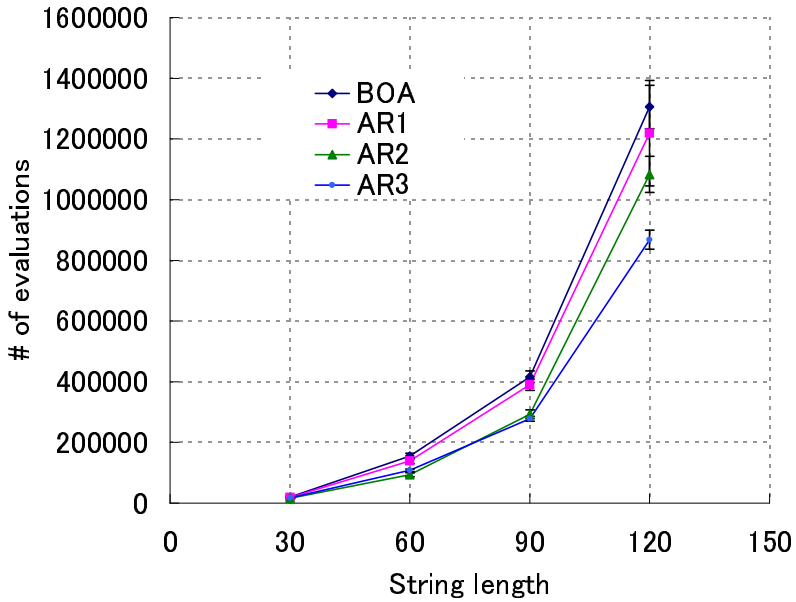

Fig. 3. Results for rank-based assignment functions (exponential case)

Figures 4 and 5 are similar, comparing the number of fitness evaluations required by the algorithm under the three proportionate assignment functions APc.

In contrast with the conventional test functions of section 4.2, here proportionate assignment functions significantly improve the performance of the BOA. For both $c = 3$ and $c = 5$, we obtain optima at a lower computational cost. For the uniformly scaled test function, the proportionate scheme even performs better than the rank-based assignment functions. This is partly because

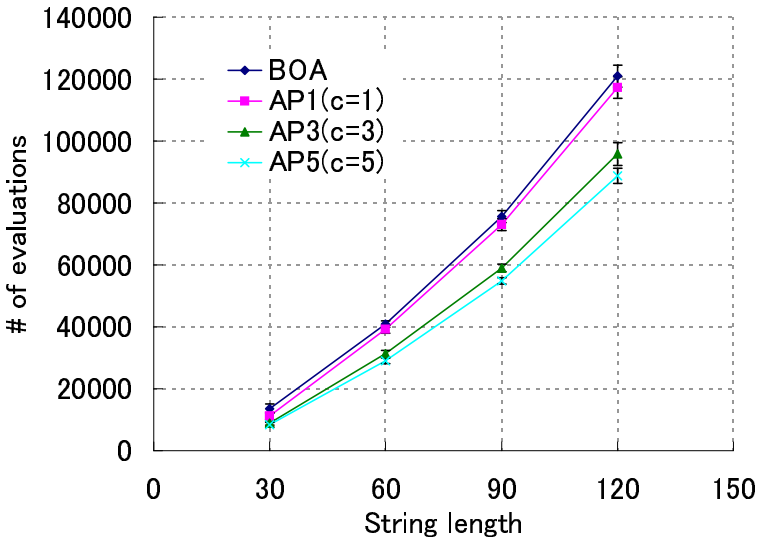

Fig. 4. Results for proportionate assignment functions (uniform case)

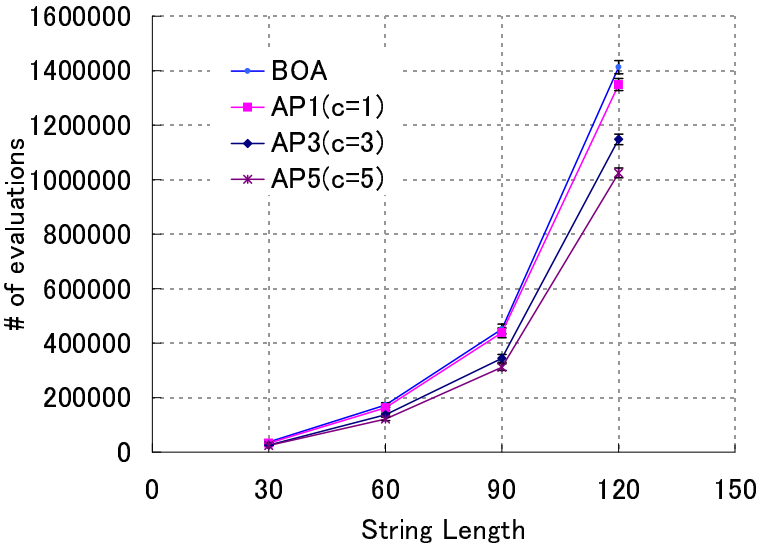

Fig. 5. Results for proportionate assignment functions (exponential case)

it is more favorable to employ function values directly when the test function does not cause any scaling difficulties. In Figure 5 results are more sensitive to the weighting pressure parameter c, because the sub-functions introduce some scaling difficulties in the building blocks. This also explains why the proportionate assignment functions could not improve performance on the conventional numerical test functions, where each variable is also encoded in an exponential manner (fixed-point binary encoding).

We also compared the execution times required to obtain optimal solutions, in order to demonstrate the effectiveness of our approach in an actual computing environment: an SGI Onyx300 consisting of MIPS R1400/500MHz processors. In this experiment, we employ the sum of 5-bit trap functions for a string length of $l = 90$. For the uniform case, Table 7 shows the execution times (in seconds) required for selection and calculation of the assignment functions (AFs), model-building, and fitness evaluation. Table 8 lists the same information for the exponential case. The results presented are the average of 10 experiments. We succeeded in obtaining optimal solutions for all experiments.

Table 7
Execution times (uniform case)

	Original BOA	proportionate (AP5)	rank-based (AR2)
Population size	5200	5200	5200
# Opt/Exp	10/10	10/10	10/10
Time for selections (s)	0.532	–	–
Time to calculate AFs (s)	–	0.002	0.572
Model-building time (s)	31.20	22.08	26.41
Fitness evaluations time (s)	0.758	0.528	0.598
# of fitness evaluations	76180	55120	60840
Average # of generations	27.3	19.2	21.4
Overall execution time (s)	32.77	22.59	28.27

In both cases we obtain optimal solutions with less computational overhead by introducing assignment functions. This result can be attributed to an improvement in the accuracy of the model-building process. Another source of improvement is the fact that these assignment functions obviate the need for a separate selection/sorting phase. A sorting algorithm such as quicksort, for example, makes $O(n \log n)$ comparisons where the proportionate scheme needs only $O(n)$ assignment function evaluations. Such an improvement may seem minor in the context of this problem. When the model-building is not costly and a huge number of strings are involved, however, the difference in sorting time should affect the overall efficiency.

Table 8
Execution times (exponential case)

	Original BOA	proportionate (AP5)	rank-based (AR2)
Population size	15000	15000	15000
# Opt/Exp	10/10	10/10	10/10
Time for selections (s)	1.48	–	–
Time to calculate AFs (s)	–	0.036	5.43
Model-building time (s)	160.18	106.98	119.54
Fitness evaluations time (s)	4.17	3.40	2.93
# of fitness evaluations	416250	334500	293250
Average # of generations	53.5	42.6	37.1
Overall execution time (s)	189.4	110.5	149.5

5 Conclusions

In this paper, we show that assignment functions can result in more accurate probabilistic model-building and improved overall performance for the Bayesian Optimization Algorithm (BOA). The proposed assignment functions are inspired by the processes of proportionate and rank-based reproduction in conventional genetic algorithms. By assigning a greater significance to the best solutions of the current population, we improve the accuracy of the probabilistic models. Our approach also simplifies the conventional BOA by removing the explicit selection process; it simply alternates between building probabilistic models and generating solution candidates based on the models.

Through empirical comparisons, we demonstrate that the assignment function approach is more effective than the conventional BOA. Rank-based assignment functions are a robust choice, improving performance for all the test problems examined here. Proportionate functions achieve better results than rank-based functions for certain uniformly scaled test functions, but achieve no significant improvement for exponentially scaled functions. Parameter optimization is an essential topic for future work, especially in the case of proportionate assignment functions, to improve the robustness of this technique.

Acknowledgement

This research was partially supported by the Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Young Scientists (B), 15700175, 2003-2005.

References

[1] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning, Technical Report No.CMU-CS-94-163, Carnegie Mellon University.

[2] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):415–444, 1989.

[3] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[4] D. Heckerman and M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report Technical Report MSR-TR-94-09, Microsoft Research, 1994.

[5] G. Harik and D. E. Goldberg. Learning linkage. Technical Report IlliGAL Report No.96006, University of Illinois at Urbana-Champaign, 1996.

[6] G. Harik, F. Lobo and D. E. Goldberg. The compact genetic algorithm Technical Report IlliGAL Report No.97006, University of Illinois at Urbana-Champaign, 1997.

[7] G. Harik. Linkage learning via probabilistic modeling in the ECGA. Technical Report IlliGAL Report No.99010, University of Illinois at Urbana-Champaign, 1999.

[8] P. Larrañanga and J. A. Lozano. *Estimation of Distribution Algorithms — A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2002.

[9] F. Lobo, K. Deb, D. E. Goldberg, G. Harik and L. Wang. Compressed Introns in a Linkage Learning Genetic Algorithm. Technical Report IlliGAL Report No.97010, University of Illinois at Urbana-Champaign, 1997.

[10] M. Munetomo. Linkage identification based on epistasis measure considering monotonicity. In *Proceedings of the 4-th Asia-Pacific Conference on Simulated Evolution and Learning*, 550–554, 2002.

[11] M. Munetomo and D. E. Goldberg. Designing a genetic algorithm using the linkage identification by nonlinearity check. Technical Report IlliGAL Report No.98014, University of Illinois at Urbana-Champaign, 1998.

[12] M. Munetomo and D. E. Goldberg. Identifying linkage by nonlinearity check. Technical Report IlliGAL Report No.98012, University of Illinois at Urbana-Champaign, 1998.

[13] H. Mühlenbein, J. Bendisch, and H.-M. Voigt. From recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature IV*, 188–197, Springer-Verlag, 1996.

[14] M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, and P. K. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*, 521–535, London, Springer-Verlag, 1999.

[15] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. IlliGAL Report No. 99003, University of Illinois at Urbana-Champaign, 1999.

[16] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. Linkage problem, distribution estimation, and Bayesian networks. *Evolutionary Computation*, 8(3): 311–340, MIT Press, 2000.

[17] M. Pelikan, Tz-Kai Lin. Parameter-less hierarchical BOA. *Proceedings of the Genetic and Evolutionary Computation Conference 2004 (GECCO-2004)*, 24–35, Springer-Verlag, 2004.

[18] M. Pelikan, D. E. Goldberg, and F. G. Lobo. A Survey of optimization by building and using probabilistic models, *Computational Optimization and Applications*, 21(1):5–20, Kluwer Academic Publishers, 2002.

[19] M. Pelikan, D. E. Goldberg, and S. Tsutsui. Getting the best of both worlds: Discrete and continuous genetic and evolutionary algorithms in concert, *Information Sciences*, 156:147–171, Elsevier, 2003.

[20] K. Sastry, D. E. Goldberg, and M. Pelikan. Efficiency enhancement of probabilistic model building genetic algorithms IlliGAL Report No. 2004020, University of Illinois at Urbana-Champaign, 2004.