International Journal of Innovative
Computing, Information and Control
Volume **8**, Number **9**, September **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 6371–6388

AN ENTROPY APPROACH TO EVALUATION RELAXATION FOR BAYESIAN OPTIMIZATION ALGORITHM

Hai Thi Thanh Nguyen[1], Hoang Ngoc Luong[2] and Chang Wook Ahn[1,*]

[1]Department of Computer Engineering
Sungkyunkwan University
2066 Seobu-ro, Suwon 440-746, Korea
*Corresponding author: cwan@skku.edu

[2]Centrum Wiskunde and Informatica
Amsterdam, NL-1090 GB, The Netherlands

Received April 2011; revised October 2011

Abstract. *Bayesian Optimization Algorithm (BOA), a multivariate estimation of distribution algorithm, needs incorporating with efficiency enhancement techniques to be capable of solving difficult large-scale problems in a reliable and scalable manner. In this paper, we present a novel evaluation relaxation method which is based on the conditional entropy measurement. The concept of conditional entropy is rigorously analyzed and then is used to investigate the stability of the population. Especially, we utilize the evaluation relaxation strategy (ERS) proposed herein to determine whether a candidate solution should be evaluated by actual functions or be estimated by surrogate models. BOA coupled with our entropy-based ERS, termed en-BOA, shows its superiority in significantly reducing the total number of expensive fitness evaluations until reliable convergence. Experimental results prove that the entropy-based ERS enhances the efficiency of BOA while not negatively affecting the scalability of the original algorithm. In addition, the effects of our efficiency enhancement technique on population sizing requirements are also discussed.*

Keywords: Bayesian optimization algorithm, Conditional entropy, Efficiency enhancement, Evaluation relaxation, Fitness evaluation

1. **Introduction.** Inspired by biological mechanisms in nature, such as natural selection or genetic inheritance, evolutionary algorithms (EAs) [1, 2] have been widely employed to solve optimization tasks in system design problems [3, 4]. For the last two decades, research in EAs has been diversified toward various directions, most notably Estimation of Distribution Algorithms (EDAs) [5, 6]. A key difference between traditional EAs and EDAs is that EDAs replace the conventional crossover and mutation operators of EAs by building and sampling probability distributions that model promising solutions found so far in order to generate new offspring for the next iteration. Categorized into three groups: Univariate EDAs [7], Bivariate EDAs [8] and Multivariate EDAs [9, 10], EDAs employ machine learning techniques to capture explicitly the underlying (in)dependencies among design variables of the problem at hand. Exploiting such information to evolve their populations, EDAs can avoid disruptions of building blocks, caused by random variation operators, such as crossover and mutation. The effectiveness of EDAs depends upon the accuracy of the underlying probabilistic models, i.e., discovering correctly relationships of different variables. Using Bayesian networks to model promising candidate solutions, Pelikan et al. [11] proposed the Bayesian optimization algorithm (BOA) to solve various classes of optimization problems efficiently and reliably. Furthermore, Ahn et al. [12, 13]

extended BOA into the continuous optimization domain: Real-coded Bayesian Optimization Algorithm (rBOA).

Optimization algorithms evolve their search efforts toward promising regions where individuals of high fitness values locate on the problem landscape. Fitness values of candidate solutions reflect their suitability for solving the problem at hand. EAs require adequate amounts of fitness evaluations before reaching their optimal solutions. In many real-world applications, fitness functions are complicated and expensive in terms of computational time and resources. Advanced branches of EAs, such as EDAs, are shown to be capable of solving bounded difficult problems reliably and scalably in polynomial time. However, polynomial complexity in evaluating whole populations when solving large-scale complex problems would still be impractical for both traditional GAs and advanced EDAs. Therefore, efficiency enhancement techniques (EETs) [14, 15, 16] are always crucial in developing competent EAs which are able to converge toward the optima in a timely practical manner. Basically, while EAs are effective in finding acceptable solutions for hard problems of extremely large search spaces, they need to be assisted with EETs to become applicable in real-world problems.

Efficiency enhancement techniques can be classified into four categories: Parallelization, Hybridization, Time continuation/utilization, and Evaluation relaxation [17]. In parallelization [18, 19], multiple processors are employed at the same time to divide and allocate computational resources according to some topologies, such as master-slave, fine-grained, coarse-grained, or hierarchical. Hybridization techniques [20, 21] combine global searchers (i.e., evolutionary algorithms) with local searchers (e.g., greedy search or hill-climbing) to accelerate the convergence speed toward optimal solutions. Time continuation/utilization [22, 23] enables practitioners to choose between an EA that has a small population, but performs for many generations, and an EA that has a large population, but performs for fewer generations. Finally, evaluation relaxation [24, 25] tries to replace accurate and costly fitness functions with other inaccurate but more economical estimation models when possible.

Similar to other EDAs, the performance of BOA is strongly influenced by two factors: the probabilistic model construction (i.e., the Bayesian network) and the computational cost of fitness evaluations. In industrial optimization tasks, the time complexity of BOA is mainly associated with the latter factor. Thus, research has been conducted to reduce the number of expensive fitness evaluations. Beside the abovementioned EETs, other techniques have also been proposed specifically for BOA. Pelikan et al. [26] applied fitness inheritance as an evaluation relaxation strategy (ERS) to improve the performance of BOA. This ERS helps BOA reduce significantly the number of actual fitness evaluations by allowing new offspring to inherit fitness values of individuals from previous generations. Lima et al. [27] designed a substructural local search to investigate candidate solutions' neighborhoods whose topologies are defined by dependencies encoded in the Bayesian networks. As a result, BOA incorporated with the local search explores search spaces more efficiently and thus uses smaller amounts of costly fitness evaluations.

To improve the performance of the standard BOA, this paper proposes a mechanism to identify when and which individuals should be estimated or evaluated by using the concept of entropy [28]. Luong et al. [29, 30] proposed a similar entropy-based ERS, but the research focused on the effects of a small promising portion of the population, called the elite set. In this paper, we consider and investigate the effects of both the selected set and the unselected set (i.e., whole population) on entropy computation. The remaining of this paper is organized as follows. In Section 2, we briefly describe the standard BOA procedures and some of its related evaluation relaxation methods. Section 3 reviews the formula for entropy computation of populations in BOA. Section 4 proposes our approach

using the entropy concept for evaluation relaxation. We demonstrate experiments and results in Section 5. Finally, Section 6 concludes this paper, and the prospects for future work are also discussed.

2. Related Work.

2.1. Original BOA framework.

Bayesian optimization algorithm (BOA) belongs to the class of multivariate EDAs. Instead of using traditional variation operators (i.e., crossover, mutation), we build Bayesian networks [31] to model high-order interactions among variables and consequently sample the constructed model to generate new offspring. It combines the prior information about the structure of the problem and the set of promising solutions found so far to estimate their distribution. The underlying probability distribution is estimated as the product of conditional probability distributions of each variable X_i given its parents Π_i.

$$p(X_0, X_1, \ldots, X_{n-1}) = \prod_{i=0}^{n-1} p(X_i|\Pi_i) \tag{1}$$

where $(X_0, X_1, \ldots, X_{n-1})$ is the vector of random variables, Π_i is the set of parent nodes of X_i, and $p(X_i|\Pi_i)$ is the conditional probability of X_i given its parents Π_i [11]. Moreover, n denotes the problem size. The framework of standard BOA is described in Algorithm 1.

Algorithm 1 Bayesian optimization algorithm

1: Set t:= 0.
2: Generate the first population $\mathcal{P}(0)$ at random.
3: Evaluate $\mathcal{P}(0)$.
4: **while** termination criterion is not met **do**
5: Select a parents set $\mathcal{S}(t)$ from $\mathcal{P}(t)$.
6: Construct the Bayesian network $\mathcal{B}(t)$ to model the selected set of parents $\mathcal{S}(t)$.
7: Generate offspring $\mathcal{O}(t)$ by sampling from $\mathcal{B}(t)$.
8: Evaluate $\mathcal{O}(t)$.
9: Replace some solutions of $\mathcal{P}(t)$ with $\mathcal{O}(t)$ to create new population $\mathcal{P}(t+1)$.
10: t:= $t+1$.
11: **end while**

Constructing Bayesian networks from the selected promising individuals requires two procedures: learning the structure (i.e., conditional (in)dependencies) and learning the parameters (i.e., conditional probabilities). Usually, a greedy search algorithm is employed to build an acceptable Bayesian network for BOA [32]. After that, parameters for the constructed structure can be computed as the relative frequencies of all the possible building blocks described by the decomposition of the networks [32]. Each node of the Bayesian network stores its corresponding conditional probability table having information about the relationship of that node with its parent nodes.

After modeling the Bayesian network, we would sample it to produce new candidate solutions having similar characteristics of the learned data. Variables of an offspring are generated following the *Probabilistic Logic Sampling* mechanism that the values of parent variables Π_i are calculated before generating children variables X_i [5].

2.2. **Evaluation relaxation in BOA.** This paper handles the problem of efficiency enhancement in EDAs. We concentrate on evaluation relaxation methods. In evaluation relaxation techniques [33], an accurate, but expensive fitness function is replaced by a less accurate, but inexpensive surrogate function. Therefore, we can reduce the total number of costly fitness evaluations.

In this paper, we utilize the fitness model proposed by Pelikan et al. [26]. Their methodology uses the probabilistic model to construct a surrogate fitness model. The fitness values of some individuals are estimated based on the surrogate model. Only a certain proportion of new offspring are evaluated by the actual evaluation function in each iteration. All the individuals evaluated by the actual fitness function are used to estimate the coefficients of the surrogate model. The fitness values of the individual in BOA can be estimated as

$$f_{\text{est}}(X_0, X_1, \ldots, X_{n-1}) = \bar{f} + \sum_{i=0}^{n-1} \left(\bar{f}(X_i|\Pi_i) - \bar{f}(\Pi_i) \right), \tag{2}$$

where $\bar{f}$ denotes the average fitness of all individuals used to construct model, $\bar{f}(X_i|\Pi_i)$ denotes the average fitness value of solutions with X_i and Π_i, and $\bar{f}(\Pi_i)$ is the average of all solutions with Π_i [26].

The above surrogate model has a substantial speedup on several additively separable problems of bounded difficulty. However, it requires a larger population size; when the problem size increases, the speedup slows down. In this paper, we also employ this surrogate model to estimate the fitness values of candidate solutions. But our contribution lies in designing a mechanism to determine whether the fitness value of an individual should be evaluated (by the actual function) or be estimated (by the surrogate model). In the decision-making process, we use a theory of conditional entropy measurement presented in Section 3.

3. Conditional Entropy Measurement.

In simple case, the classical entropy [28] of an observation with discrete probability distribution p_i is defined as:

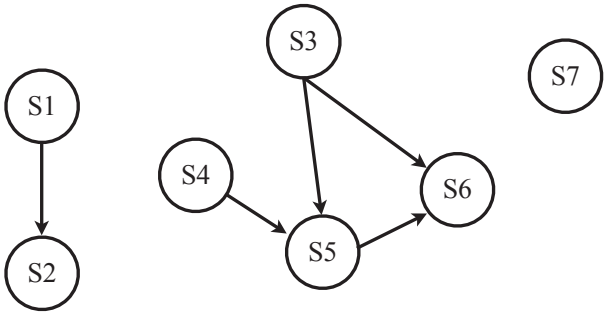
$$\mathcal{H}(X) = -\sum_i p_i \log(p_i).$$

The population entropy is a measure of the diversity of the evolutionary population. Thus, the entropy can determine the rate of convergence of the current population. Extending from the original entropy measure, Ocenasek [34] derived the entropy $\mathcal{H}(X)$ of a certain population in BOA as the sum of local conditional entropies according to the factorization of the probability distribution $p(X)$ in Equation (1):

$$\begin{aligned}
\mathcal{H}(\mathbf{X}) &= \sum_{i=0}^{n-1} \mathcal{H}(X_i|\Pi_i) = \sum_{i=0}^{n-1} \sum_{\pi_i \in \mathcal{P}_i} p(\pi_i) \mathcal{H}(X_i|\Pi_i = \pi_i) \\
&= -\sum_{i=0}^{n-1} \sum_{\pi_i \in \mathcal{P}_i} p(\pi_i) \sum_{x_i \in \mathcal{X}_i} p(x_i|\pi_i) \log_2 p(x_i|\pi_i) \\
&= -\sum_{i=0}^{n-1} \sum_{\pi_i \in \mathcal{P}_i} \sum_{x_i \in \mathcal{X}_i} p(x_i, \pi_i) \log_2 p(x_i|\pi_i) \\
&= -\sum_{i=0}^{n-1} \sum_{\pi_i \in \mathcal{P}_i} \sum_{x_i \in \mathcal{X}_i} \left(\frac{m(x_i, \pi_i)}{N} \right) \log_2 \left(\frac{m(x_i, \pi_i)}{m(\pi_i)} \right)
\end{aligned} \tag{3}$$

where $\mathcal{P}_i$ denotes the set of possible vectors that can be assigned to Π_i, $\mathcal{X}_i$ is the set of all possible values of X_i, $m(x_i, \pi_i)$ is the number of solutions having parameter X_i set to x_i and parameters Π_i set to π_i, $m(\pi_i)$ counts the solutions with Π_i set to π_i, and N is the number of individuals in the population. We can compute the entropy value of some particular portions of the population with respect to the current network by using Equation (3). Figure 1 demonstrates an example of a Bayesian network and its corresponding entropy computation.

The research in [34] showed that the entropy values of the population can be used to determine when to terminate BOA. In this paper, we use Equation (3) to compute the entropy values of some particular portions of the BOA population. We compute the entropy value of parents set (i.e., the set of selected individuals on which the Bayesian networks are built) and unselected set (i.e., the set of individuals which would be discarded due to low fitness values) in each iteration. Figure 2 shows how the entropy values of the unselected set and the selected set (parents set) vary during the optimization progress. We perform the experiment with a BOA solving an Order-5 separable deceptive problem ($\gamma = 1.0$) of 90 bits (see Section 5.1). The entropy of the unselected set monotonically reduces except for the last generation. The entropy of the parents set decreases after every generation and then reaches its minimum value '0'. Moreover, the parents set has



$$\mathcal{H}(\mathbf{X}) = \mathbf{H}(\mathbf{S_1}) + \mathbf{H}(\mathbf{S_2}|\mathbf{S_1}) + \mathbf{H}(\mathbf{S_3}) + \mathbf{H}(\mathbf{S_4}) + \mathbf{H}(\mathbf{S_5}|\mathbf{S_3}, \mathbf{S_4}) + \mathbf{H}(\mathbf{S_6}|\mathbf{S_3}, \mathbf{S_5}) + \mathbf{H}(\mathbf{S_7})$$

FIGURE 1. A Bayesian network and conditional entropy measurement

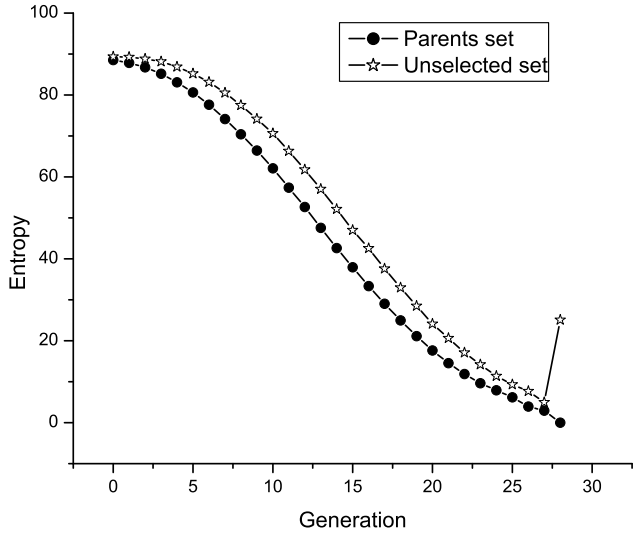


FIGURE 2. Entropies in a BOA solving an order-5 separable deceptive problem ($\gamma = 1.0$) of 90 bits

entropy values smaller than those of the unselected set at all the generations. This proves that the parents set is more stable than the unselected set in terms of entropy.

Pelikan et al. [26] introduced an effective surrogate fitness model in order to improve the performance of standard BOA in terms of reducing the number of fitness evaluations. In relation to this, we propose a new evaluation relaxation strategy (ERS) based on the entropy measurement for BOA in the next section.

4. BOA with Conditional Entropy Measurement-Based Evaluation Relaxation Strategy.

The idea behind this approach is to recognize whether a new offspring belongs to the better half or the worse half of the population, using the concept of entropy measurement. In other words, it needs to judge whether a newly generated individual should be estimated by a surrogate model or be evaluated by the actual fitness function. To this end, each offspring is put into the better half and the worse half of the population, and then compute their corresponding entropy values, respectively. It is clear that a new offspring is associated with the set having the smaller new entropy value. If the offspring is similar to candidate solutions of the better set, we can estimate its fitness value by the surrogate model. If it belongs to the worse set, it should be evaluated by the actual fitness function. Algorithm 2 outlines the overall procedures of our algorithm, termed *en-BOA*.

Algorithm 2 BOA with conditional entropy measurement-based ERS(en-BOA)

Set t:= 0.
Generate the first population $\mathcal{P}(0)$ at random.
Evaluate $\mathcal{P}(0)$.
while termination criterion is not met **do**
    Divide population $\mathcal{P}(t)$ into two sub-populations: selected set $\mathcal{P}_s$ and unselected set $\mathcal{P}_u$.
    Learn a Bayesian network $\mathcal{B}(t)$ to model the selected set $\mathcal{P}_s$.
    Compute entropies $\mathcal{H}_s(t)$ and $\mathcal{H}_u(t)$ of $\mathcal{P}_s$ and $\mathcal{P}_u$, using Equation (3).
    Generate offspring $\mathcal{O}(t)$ by sampling from $\mathcal{B}(t)$.
    if $\mathcal{H}_s(t) \leq \delta * \mathcal{H}_s(0)$ **then**
        Consider each new offspring ϑ of $\mathcal{O}(t)$.
        Put ϑ into $\mathcal{P}_s$ and $\mathcal{P}_u$ to create $\mathcal{P}'_s$ and $\mathcal{P}'_u$.
        Compute $\mathcal{H}'_s(t)$ and $\mathcal{H}'_u(t)$ of $\mathcal{P}'_s$ and $\mathcal{P}'_u$, using Equation (3).
        if $\mathcal{H}'_s(t) \leq \mathcal{H}'_u(t)$ **then**
            Estimate ϑ by using Equation (2).
        else
            Evaluate ϑ.
        end if
    else
        Evaluate $\mathcal{O}(t)$.
    end if
    Replace some solutions of $\mathcal{P}(t)$ with $\mathcal{O}(t)$ to create $\mathcal{P}(t+1)$.
    t:= $t+1$.
end while

It starts with evaluating all individuals (of the initial population) using the actual fitness function. We divide the population into two sets: the parents set $\mathcal{P}_s$ and the unselected set $\mathcal{P}_u$. Let $\mathcal{H}_s(0)$ and $\mathcal{H}_s(t)$ be entropy values of $\mathcal{P}_s$ at the initial generation and t^{th} generation, respectively. The parameter δ, taking a value in $[0, 1]$, is considered as the starting point to initiate the proposed ERS. Thus, the standard BOA is run unless $\mathcal{H}_s(t) \leq$

$\delta * \mathcal{H}_s(0)$; after that, our en-BOA starts to operate. Notice that if our ERS is applied earlier, the estimation model may not be accurate enough to give a good approximation. On the contrary, if we apply our ERS nearly at the end of the optimization progress, the minimal number of evaluations cannot be achieved. As for different problems, we need to assign different appropriate values to the starting point δ. More details on this issue is investigated in the next section.

Evaluating the fitness of a candidate solution is often a very expensive task; it is economical if we are able to replace the fitness function with a cheaper approximate function. In this work, the fitness inheritance in Equation (2) is used to estimate the fitness values of individuals. Whether a new offspring ϑ is estimated or evaluated depends on the entropy values $\mathcal{H}'_s(t)$ and $\mathcal{H}'_u(t)$, which are obtained after putting ϑ into the parents set $\mathcal{P}_s$ and the unselected set $\mathcal{P}_u$. The set having a smaller entropy value would be considered more stable; thus, the offspring should belong to that set. Here, all the information needed for computing the conditional entropy using Equation (3) is already obtained in the Bayesian network construction phase. If $\mathcal{H}'_s(t)$ is smaller than $\mathcal{H}'_u(t)$, the individual (ϑ) is estimated; otherwise, it is evaluated. These procedures iterate until the population converges.

5. **Experiments and Results.**

5.1. **Test problems.**

To validate the effectiveness of our method, we perform experiments on some widely-known test problems: *OneMax*, *Separable Deceptive* and *Nonseparable Deceptive*.

In the OneMax problem, the fitness value is defined as the sum of all bit values:

$$f_{\text{onemax}}(X_0, X_1, \ldots, X_{n-1}) = \sum_{i=0}^{n-1} X_i, \tag{4}$$

where $(X_0, X_1, \ldots, X_{n-1})$ denotes the string of n bits. OneMax is a simple problem whose optimal solution is a string of all 1s. Since the fitness contribution of each bit is independent, most algorithms can work well on this problem; thus, any linkage learning is not required for solving the problem.

Prior to describing the deceptive problems, we need to introduce a trap function defined as follows:

$$f_{\text{trap\_k}}(u) = \begin{cases} k & \text{if } u = k, \\ \gamma \cdot (k - 1 - u) & \text{otherwise,} \end{cases} \tag{5}$$

where u is the number of 1s in the input of k bits, and $\gamma \in \left(0, \frac{k}{k-1}\right)$ is the *noise-to-signal* ratio. Note that the problem becomes harder as γ goes to $\frac{k}{k-1}$.

Separable deceptive problem is formed by disjointly concatenating several trap functions of order k; thus, the values of all the trap functions are added together to obtain the overall fitness value. The problem is formulated as

$$f_{\text{dec\_k}}(X_0, \cdots, X_{n-1}) = \sum_{i=0}^{\frac{n}{k}-1} f_{\text{trap\_k}}(X_{ki}, X_{ki+1}, \cdots, X_{ki+k-1}). \tag{6}$$

It has one global optimum at a string of all 1s. Any algorithm can hardly solve this problem without performing a proper decomposition of order k.

Nonseparable deceptive problem also consists of order-k trap functions, but each trap function is laid overlapped by m bits with its direct left and right adjacent functions

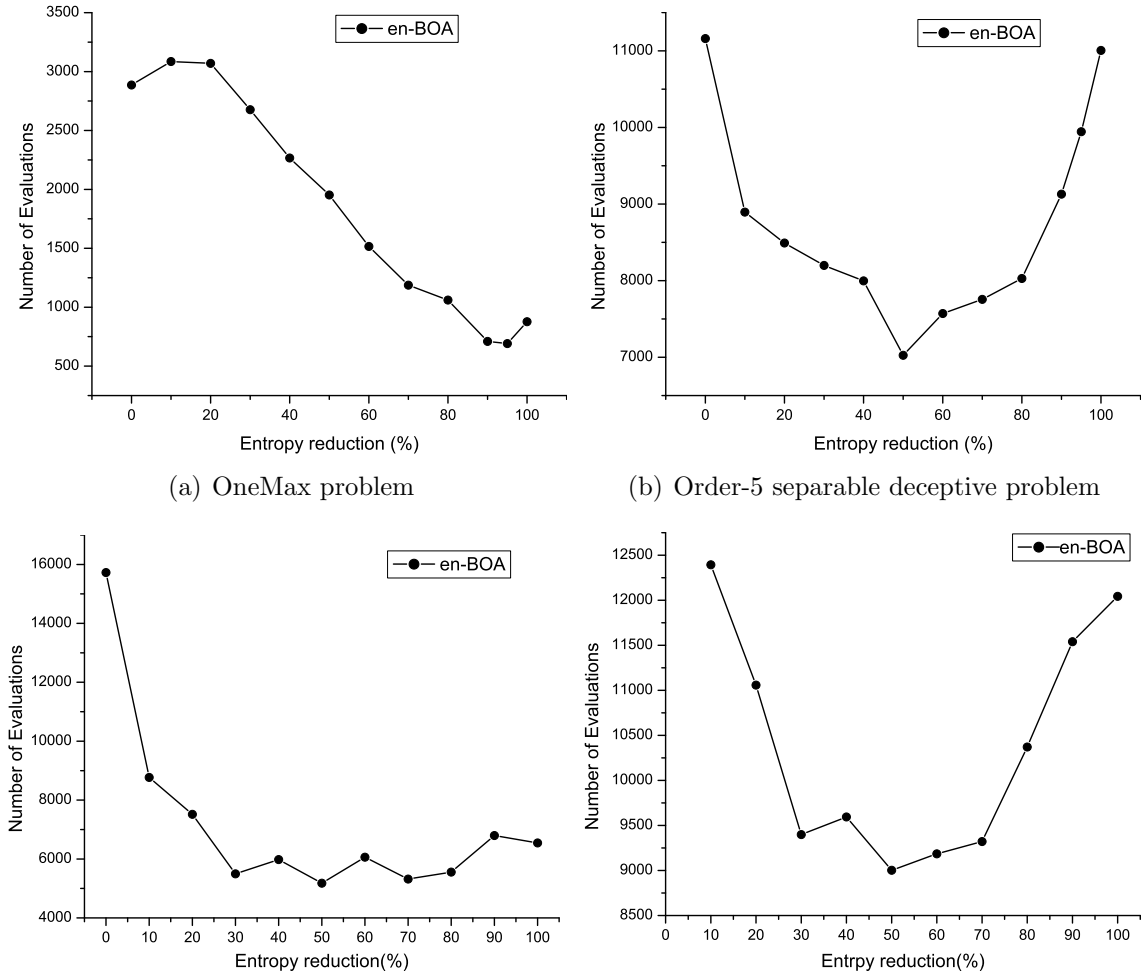
$(m < k)$. It can be formulated as follows with $d = k - m$:

$$f_{\text{overlap\_m\_k}}(X_0, \cdots, X_{n-1}) = \sum_{i=0}^{\frac{n-k}{d}} f_{\text{trap\_k}}(X_{di}, X_{di+1}, \cdots, X_{di+k-1}). \tag{7}$$

The optimum is a string of all 1s. This problem becomes even harder than the previous ones due to its complicated structure.

5.2. **Experiments for the starting points of ERS.** The juncture of applying our proposed ERS depends on each problem. In relation to this, we test en-BOA with different starting points of ERS on all the benchmark problems. Specifically, the experiments are conducted in terms of entropy reduction.

Let $\mathcal{H}_s(0)$ and $\mathcal{H}_s(t)$ be the entropy values of the parents set (i.e., selected set) at the initial iteration and at the t^{th} iteration, respectively. The starting point for our en-BOA is defined as $\rho = 1 - \delta$, where the parameter ρ indicates the percentage of reduction in the initial entropy value before applying our ERS to BOA. If ρ is set at 0%, our ERS starts at the beginning of the optimization process. When parameter ρ is 100%, en-BOA becomes the standard BOA, which means no ERS is applied.



(a) OneMax problem

(b) Order-5 separable deceptive problem

(c) Order-3 separable deceptive problem

(d) Order-3 nonseparable deceptive problem with 1-bit overlapping

FIGURE 3. Performance of en-BOA with different starting points

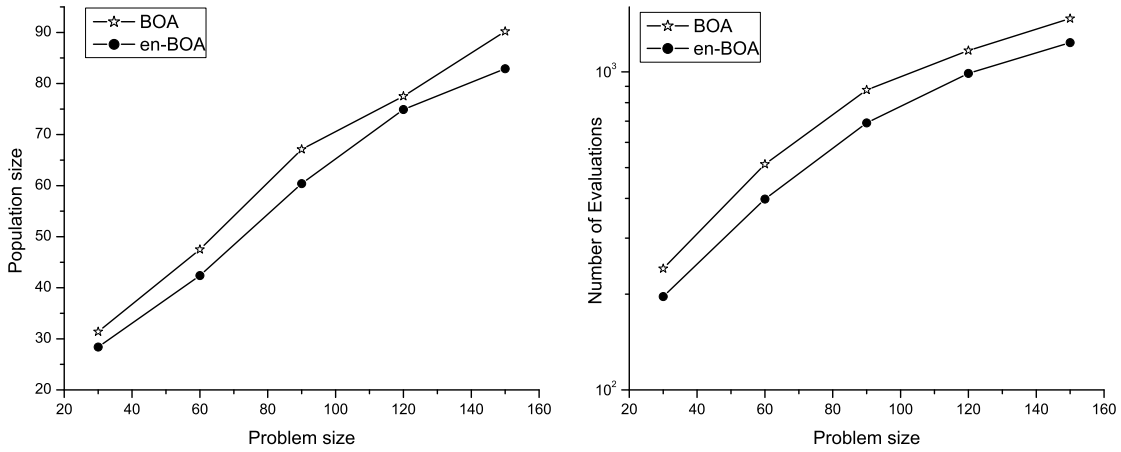
Empirical results are demonstrated in Figure 3, exhibiting that the en-BOA achieves the minimal number of evaluations when ρ is 95% for OneMax problem, and 50% for the deceptive problems. At such junctures, the structure of Bayesian network has become robust enough for operating ERS and computing the surrogate model.

5.3. **Bisection method for experiments.** We use a bisection method as the test framework to find the minimal population size for each test case. Bisection will be run many times to obtain sufficient statistical results. Every bisection begins by running an algorithm (BOA or en-BOA) with a small population size as the initial lower bound. After each run, if the algorithm cannot find the optimal solution, the current population size will be the new lower bound, and we start again with a double population size. If the optimum is reached, the current population size will be the new upper bound, and then the algorithm starts again with the population size as the middle point of lower bound and upper bound. Bisection terminates when the algorithm converges to the optimum under the condition that the distance between the lower bound and upper bound is small enough so that any change in the current population size would not yield significant differences. Our bisection method is described in Algorithm 3. Truncation selection is used to select the better half of the population as the parents set (i.e., threshold = 50%). New offspring replace the worse half of the previous generation to construct the new population. The optimization process terminates when 99% of all individuals in the population are the same (i.e., the population converges to a specific point) or the maximal generation is reached.

Algorithm 3 Bisection method for supplying the minimum population size

```
1: lower ← initial lower bound
2: upper ← initial upper bound
3: current ← initial population size
4: while true do
5:     success ← current population size can find the optimum?
6:     if success = false then
7:         lower ← current
8:         current ← current * 2
9:     else
10:        upper ← current
11:        if |upper − lower| ≥ current/10 then
12:            current ← (upper+lower)/2
13:        else
14:            break
15:        end if
16:    end if
17: end while
18: return current
```

5.4. **Empirical results and discussion.** The proposed en-BOA is compared with the traditional BOA on benchmark problems of different sizes: 30, 60, 90, 120, 150 bits for OneMax and Separable deceptive problems, and 31, 61, 91, 121, 151 bits for Nonseparable deceptive problems. The population size and the number of fitness evaluations required for discovering the optimum are used as the performance measures. All results are averaged over 50 independent runs on our bisection test framework.



(a) Population size comparison (b) Fitness evaluations comparison

FIGURE 4. Population size and number of evaluations required in solving the OneMax problems

TABLE 1. Statistical comparison of the population size for the OneMax problem

Size	**30**	**60**	**90**	**120**	**150**
BOA	31.4	47.5	67.1	77.5	90.2
σ	7.28	7.74	15.5	14.7	15.9
en-BOA	28.4	42.4	60.4	74.9	82.9
σ	7.38	7.54	10.18	9.36	14.2
Statistical t-test: (BOA and en-BOA)					
p-value	0.2	0.0156	0.0583	0.4302	0.0561

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

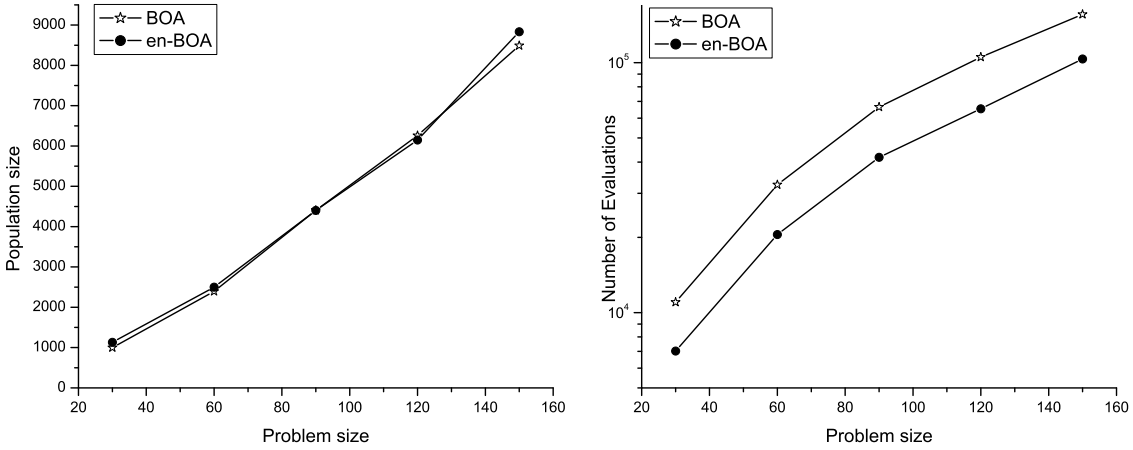
TABLE 2. Statistical comparison of the number of evaluations for the One-Max problem

Size	**30**	**60**	**90**	**120**	**150**
BOA	240.7	512.4	877	1166.4	1470.5
σ	55.6	75.2	191.7	204.7	235.2
en-BOA	196.5	398.2	690.9	988.7	1235.7
σ	70.0	84.3	121.9	99.9	193.4
Statistical t-test: (BOA and en-BOA)					
p-value	0.0348	1.86E-5†	7.69E-5†	2.38E-4†	8.37E-5†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

Figure 4, Table 1 and Table 2 show the performance comparison results between the existing BOA and our en-BOA when solving OneMax problems. While the population sizes required by en-BOA are similar to those of BOA, the efficiency of BOA is improved when combined with our entropy measurement-based ERS. On average, the en-BOA saves 18.5% of the number of fitness evaluations of BOA.

Figure 5, Table 3 and Table 4 compare the performances of BOA and en-BOA on Order-5 separable deceptive problems with $\gamma = 1.0$. While our algorithm does not require any larger population size, the number of fitness evaluations of en-BOA is considerably



(a) Population size comparison (b) Fitness evaluations comparison

FIGURE 5. Population size and number of evaluations required in solving the Order-5 separable deceptive problems with $\gamma = 1.0$

TABLE 3. Statistical comparison of the population size for the Order-5 separable deceptive problems with $\gamma = 1.0$

Size	**30**	**60**	**90**	**120**	**150**
BOA	999.1	2392.8	4404.6	6257.5	8489.6
σ	163.4	279	461.1	1020.2	1016.6
en-BOA	1131.1	2498.6	4402	6146.9	8832.5
σ	243.5	412.1	622.6	725.5	2007.3
Statistical t-test: (BOA and en-BOA)					
p-value	0.01999	0.2258	0.9852	0.6263	0.3735

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

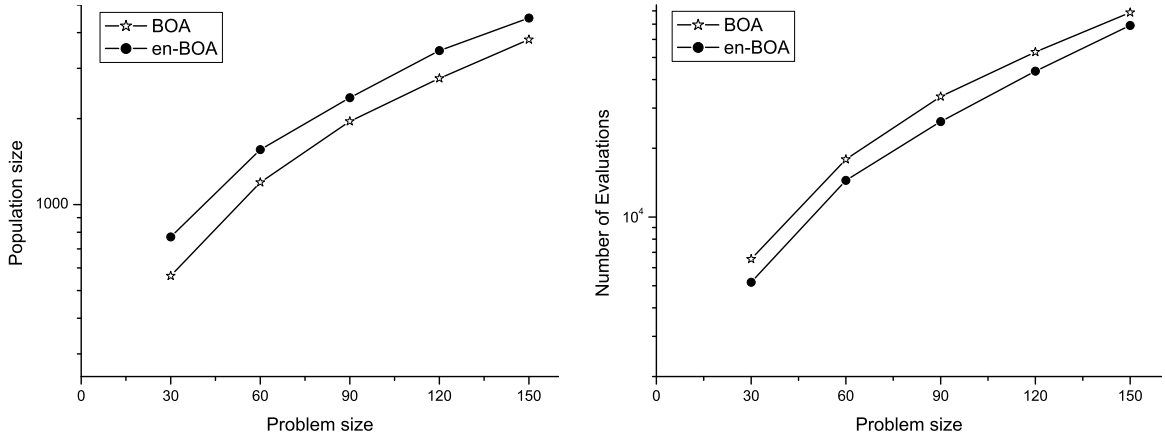
TABLE 4. Statistical comparison of the number of evaluations for the Order-5 separable deceptive problems with $\gamma = 1.0$

Size	**30**	**60**	**90**	**120**	**150**
BOA	11005.4	32496.4	66453.1	105245	155950.4
σ	1519.4	3771.1	5118.9	13198.5	15729.4
en-BOA	7025.3	20525.5	41828.9	65356.5	103402.1
σ	1551.2	2932.5	4663.6	6286.7	19786.8
Statistical t-test: (BOA and en-BOA)					
p-value	8.09E-12†	1.31E-15†	3.94E-19†	9.12E-15†	6.00E-13†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

smaller than that of BOA. Our proposed algorithm reduces 36% of the number of fitness evaluations when compared with the standard BOA.

At this juncture, a set of problems with higher noise-to-signal ratio and more complicated structure is considered. We wish to demonstrate that en-BOA can also speed up BOA in such difficult conditions. We first examine the two algorithms on Order-3 separable deceptive problems with $\gamma = 1.35$. The performance of BOA and en-BOA is compared in Figure 6. On average, en-BOA enlarges about 20% of population sizes of original BOA but saves about 19.4% of the number of fitness evaluations. Additionally, Table 5 and



(a) Population size comparison (b) Fitness evaluations comparison

FIGURE 6. Population size and number of evaluations required to solve the Order-3 separable deceptive problems with $\gamma = 1.35$

TABLE 5. Statistical comparison of the population size for the Order-3 separable deceptive problems with $\gamma = 1.35$

Size	**30**	**60**	**90**	**120**	**150**
BOA	562.7	1196.5	1956.7	2768.9	3781.6
σ	57.7	146.3	274.7	383	515.7
en-BOA	770.6	1557.6	2368.4	3461.9	4500.933
σ	140.2	339.2	716.02	813.5	1572.7
Statistical t-test: (BOA and en-BOA)					
p-value	4.09E-10†	1.54E-6†	4.7E-3†	8.65E-5†	0.2E-2†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

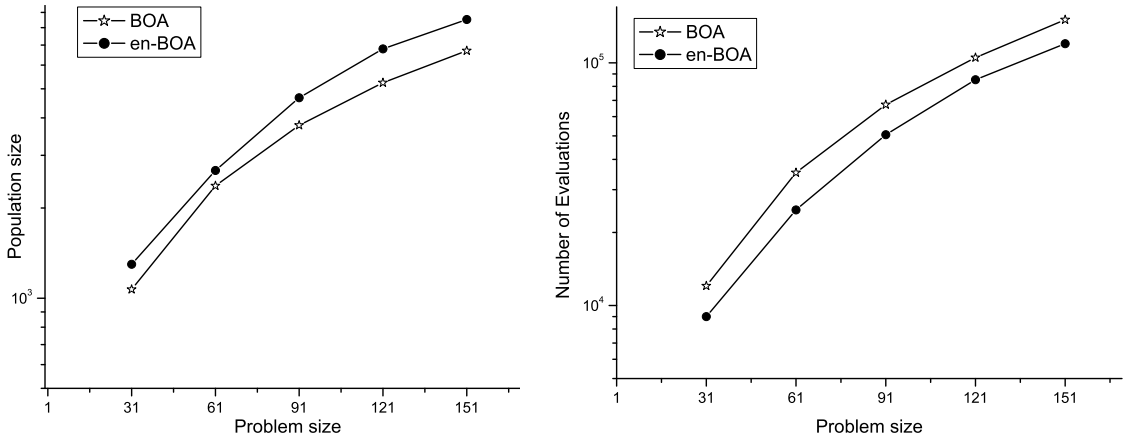
TABLE 6. Statistical comparison of the number of evaluations for the Order-3 separable deceptive problems with $\gamma = 1.35$

Size	**30**	**60**	**90**	**120**	**150**
BOA	6544.9	17884	33708.1	52852.7	78737.8
σ	708.5	1767.3	3816.7	6342	9321.8
en-BOA	5176.8	14469.4	26185.8	43560.2	69080
σ	1130.8	3172	6997.7	8945.7	18267.3
Statistical t-test: (BOA and en-BOA)					
p-value	1.0E-5†	3.24E-6†	3.03E-6†	2.02E-5†	3.19E-7†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

Table 6 prove that two algorithms, BOA and en-BOA, are significantly different at both the population size and the number of fitness evaluations.

We then test two algorithms on Order-3 nonseparable deceptive problems with 1-bit overlapping and $\gamma = 1.35$. Here, the problem structure is even more complicated than the previous case because the subproblems cannot be decomposed separately due to their overlapping decision variables. Figure 7 shows that our method requires population sizes that are 18% larger than the population sizes of the standard BOA. Nevertheless, our en-BOA reduces about 24% of the number of fitness evaluations on average. BOA and



(a) Population size comparison (b) Fitness evaluations comparison

FIGURE 7. Population size and number of evaluations for solving the Order-3 nonseparable deceptive problems with 1-bit overlapping and $\gamma = 1.35$

TABLE 7. Statistical comparison of the population size for the Order-3 nonseparable deceptive problems with 1-bit overlapping and $\gamma = 1.35$

Size	**31**	**61**	**91**	**121**	**151**
BOA	1069.7	2373.6	3777	5242.2	6700.7
σ	190.5	301.2	487.1	592.84	712.3
en-BOA	1298	2669.3	4667.6	6797.2	8522.1
σ	253.87	579.7	1125.2	6741.9	2229.5
Statistical t-test: (BOA and en-BOA)					
p-value	3.85E-6†	1.69E-3†	1.02E-6†	5.43E-8†	3.69E-7†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

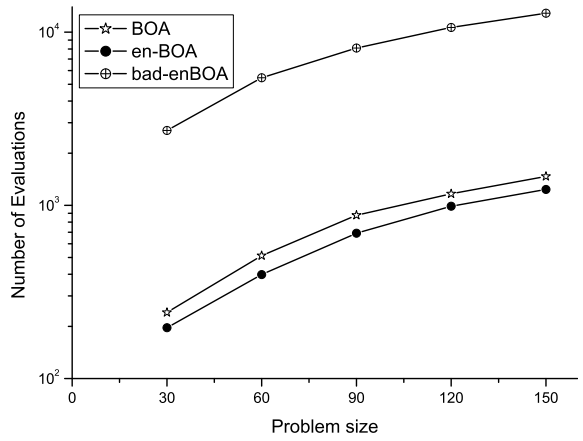
TABLE 8. Statistical comparison of the number of evaluations for the Order-3 nonseparable deceptive problems with 1-bit overlapping and $\gamma = 1.35$

Size	**31**	**61**	**91**	**121**	**151**
BOA	12043.3	35247.8	67208.5	105012.8	150419.9
σ	2238.4	4192.5	8759.0	10520.6	15290.2
en-BOA	9002	24770.6	50572.8	85213.8	119846
σ	1489.4	4693.7	9332.8	18322.2	25682.3
Statistical t-test: (BOA and en-BOA)					
p-value	8.39E-13†	1.68E-20†	1.44E-14†	8.94E-10†	6.98E-11†

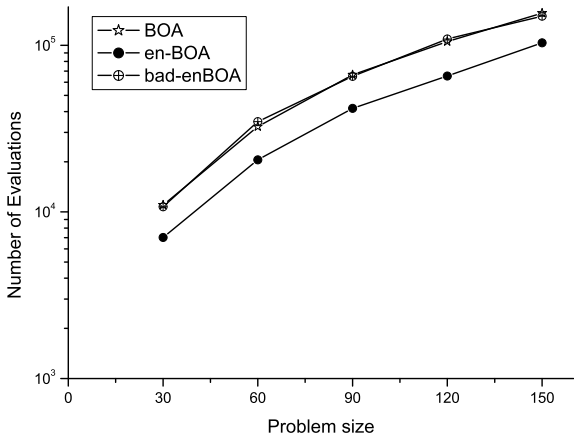
†Significance by a paired, two-tailed test with $\alpha = 0.01$.

en-BOA are also significantly different at both population sizing requirements and the numbers of fitness evaluations (see Table 7 and Table 8).

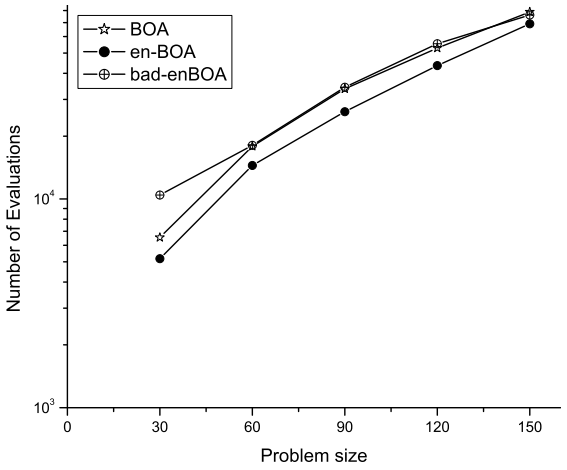
Moreover, we also perform experiments to prove that estimating on the parents set is better than estimating on the unselected set. In the proposed en-BOA algorithm, we estimate the fitness of an individual if it is judged to belong to the parents set; in other case, it is evaluated. On the contrary, we carry out the reverse experiments such that an individual which is judged to belong to the unselected set is estimated; in other case, we evaluate it. This method is called *bad-enBOA*. The results of these experiments are shown



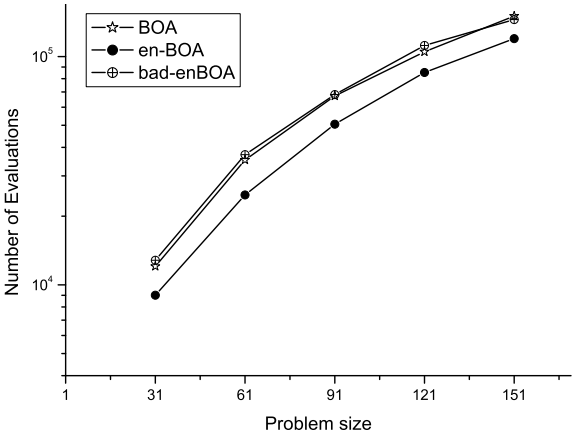
(a) OneMax problem



(b) Order-5 separable deceptive problem with $\gamma = 1.0$



(c) Order-3 separable deceptive problem with $\gamma = 1.35$



(d) Order-3 nonseparable deceptive problem with 1-bit overlapping and $\gamma = 1.35$

FIGURE 8. Number of evaluations required on the three different methods

TABLE 9. Statistical comparison for the three different methods on the OneMax problems

Size	**30**	**60**	**90**	**120**	**150**
BOA	240.7	512.4	877	1166.4	1470.5
bad-enBOA	2709.2	5459.7	8097.5	10641.6	12880.7
en-BOA	196.5	398.2	690.9	988.7	1235.7
Statistical t-test: (BOA and bad-enBOA — BOA and en-BOA)					
p-value 1	5.63E-19†	3.1E-27†	9.53E-28†	5.53E-21†	3.98E-27†
p-value 2	2.97E-19†	1.7E-27†	3.31E-28†	7.07E-22†	1.77E-27†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

in Figure 8, Table 9, Table 10, Table 11 and Table 12. We compare the three methods, BOA, en-BOA and bad-enBOA, in view of supporting that the estimation/approximation based on the unselected set is not efficient. It is seen that the bad-enBOA is significantly worse than the en-BOA in terms of the number of fitness evaluations on all the test cases.

TABLE 10. Statistical comparison for the three different methods for the Order-5 separable deceptive problems with $\gamma = 1.0$

Size	**30**	**60**	**90**	**120**	**150**
BOA	11005.4	32496.4	66453.1	105245	155950.4
bad-enBOA	10742.1	34838.3	65106.7	109033	149514.2
en-BOA	7025.3	20525.5	41828.9	65356.5	103402.1
Statistical t-test: (BOA and bad-enBOA — BOA and en-BOA)					
p-value 1	0.53	0.02	0.48	0.23	0.07
p-value 2	8.36E-13†	4.11E-17†	4.26E-17†	1.39E-16 †	9.74E-10†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

TABLE 11. Statistical comparison for the three different methods on the Order-3 separable deceptive problems with $\gamma = 1.35$

Size	**30**	**60**	**90**	**120**	**150**
BOA	6544.9	17884	33708.1	52852.7	78737.8
bad-enBOA	10443.2	18081	34294	55429.9	76110
en-BOA	5176.8	14469.4	26185.8	43560.2	69080
Statistical t-test: (BOA and bad-enBOA — BOA and en-BOA)					
p-value 1	1.32E-34†	0.69	0.50	0.14	0.34
p-value 2	1.1E-31†	2.44E-6†	2.07E-7†	3.76E-7†	7.294E-4†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

TABLE 12. Statistical comparison for the three different methods on the Order-3 nonseparable deceptive problems with 1-bit overlapping and $\gamma = 1.35$

Size	**31**	**61**	**91**	**121**	**151**
BOA	12043.3	35247.8	67208.5	105012.8	150419.9
bad-enBOA	12807.5	37169.6	68178.2	111929	145226.2
en-BOA	9002	24770.6	50572.8	85213.8	119846
Statistical t-test: (BOA and bad-enBOA — BOA and en-BOA)					
p-value 1	0.34	0.07	0.3	0.02	0.06
p-value 2	3.96E-11†	7.14E-15†	2.6E-11†	2.61E-10†	5.00E-7†

†Significance by a paired, two-tailed test with $\alpha = 0.01$.

In this paper, we have tested a large class of important benchmark problems from the simple OneMax and the linear (i.e., separable) deceptive to the nonlinear (i.e., nonseparable) deceptive problems. Additionally, the efficiency of our method has been verified by the statistical comparison of performances. From the obtained results, we can claim that our approach, en-BOA, achieves a substantial reduction in the number of fitness evaluations on all the test problems. Moreover, en-BOA has not imposed any larger population size requirements on OneMax and general deceptive problems. Although en-BOA has required slightly larger population sizes when the noise-to-signal ratio of deceptive problems increases and their structure becomes more complicated, it has not affected the performance of en-BOA. We conclude that the en-BOA considerably accelerates the original BOA in discovering the optimal solution on both simple and difficult problems.

6. **Conclusion and Future Work.** In real-world optimization, it is essential to improve the efficiency of evolutionary algorithms by means of reducing the number of fitness evaluations. To this end, computationally efficient models can be constructed for fitness approximation to assist the optimizers. However, if the fitness values of all individuals are approximated, the estimation errors will be accumulated over time, and the true optimum cannot be reached. Thus, when approximate models are involved in optimization, it is important to determine which individuals should be evaluated using the actual fitness function to guarantee faster and correct convergence. This paper proposed an evaluation relaxation strategy for BOA by using the entropy measurement. The variation in entropy values caused by the appearance of a new individual in the population is used to identify whether that solution should be evaluated by the actual function or not. The fitness values of candidate solutions which need not to be evaluated are estimated by a surrogate model. Conceptually, this method is done by examining whether a newly generated solution is similar to the selected solutions of the previous generation.

Experimental results proved affirmatively that en-BOA significantly improves the performance of the standard approach. From the obtained results, we can claim that en-BOA achieves a substantial reduction in the number of fitness evaluations on all test problems. In other words, our algorithm works efficiently on both simple and difficult problems. Additionally, our en-BOA does not impose any larger population requirements on the OneMax and the deceptive problems with moderate difficulties. It enlarges slightly the population size on the deceptive problems with higher noise-to-signal ratios and more complex structures. The chosen starting points for the test problems have been obtained by experiments; generally, different problems have their own appropriate starting points.

In the future work, we will investigate the reason of the mentioned difference in the starting points of our ERS and develop a firm theory to determine a suitable starting point for a wider class of optimization problems. We will also investigate the effectiveness of our method with other estimation models and on different types of evolutionary algorithms.

Acknowledgment. This paper was supported by Faculty Research Fund, Sungkyunkwan University, 2011.

REFERENCES

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

[2] D. E. Goldberg and M. Rudnick, Genetic algorithms and the variance of fitness, *Complex Systems*, vol.5, pp.265-278, 1991.

[3] D. Martin, R. del Toro, R. Haber and J. Dorronsoro, Optimal tuning of a networked linear controller using a multi-objective genetic algorithm and its application to one complex electromechanical process, *International Journal of Innovative Computing, Information and Control*, vol.5, no.10(B), pp.3405-3414, 2009.

[4] F. Xhafa, J. Carretero and A. Abraham, Genetic algorithm based schedulers for grid computing systems, *International Journal of Innovative Computing, Information and Control*, vol.3, no.5, pp.1053-1071, 2007.

[5] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, Boston, MA, 2002.

[6] M. Pelikan and D. E. Goldberg, Hierarchical Bayesian optimization algorithm = Bayesian optimization algorithm + niching + local structures, *Proc. of the Optimization by Building and Using Probabilistic Models (OBUPM) Workshop at the GECCO'01*, pp.217-221, 2001.

[7] M. Pelikan and H. Mühlenbein, Marginal distributions in evolutionary algorithms, *Proc. of the International Conference on Genetic Algorithms Mendel*, pp.90-95, 1999.

[8] M. Pelikan and H. Mühlenbein, The bivariate marginal distribution algorithm, *Advances in Soft Computing: Engineering Design and Manufacturing*, pp.521-535, 1999.

[9] P. A. N. Bosman and D. Thierens, Linkage information processing in distribution estimation algorithms, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99)*, pp.60-67, 1999.
[10] G. Harik, F. Lobo and K. Sastry, Linkage learning via probabilistic modeling in the ECGA, *Studies in Computational Intelligence 33: Scalable Optimization via Probabilistic Modeling*, pp.39-61, 2006.
[11] M. Pelikan, D. E. Goldberg and E. Cantú-Paz, BOA: The Bayesian optimization algorithm, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99)*, pp.525-532, 1999.
[12] C. W. Ahn and R. S. Ramakrishna, On the scalability of real-coded Bayesian optimization algorithm, *IEEE Trans. on Evolutionary Computation*, vol.12, no.3, pp.307-322, 2008.
[13] C. W. Ahn, R. S. Ramakrishna and D. E. Goldberg, Real-coded Bayesian optimization algorithm: Bringing the strength of BOA into the continuous world, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'04)*, pp.840-851, 2004.
[14] T. C. Duque, D. E. Goldberg and K. Sastry, Enhancing the efficiency of the ECGA, *Proc. of the Parallel Problem Solving from Nature – PPSN X, LNCS*, vol.5199, pp.165-174, 2008.
[15] R. Santana, Estimation of distribution algorithms with kikuchi approximations, *Evolutionary Computation*, vol.13, no.1, pp.67-97, 2005.
[16] K. Sastry, D. E. Goldberg and M. Pelikan, Efficiency enhancement of probabilistic model building algorithms, *Proc. of the Optimization by Building and Using Probabilistic Models Workshop at the GECCO'04*, 2004.
[17] K. Sastry, *Evaluation-Relaxation Schemes for Genetic and Evolutionary Algorithms*, Master Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2001.
[18] E. Cantú-Paz, *Designing Efficient and Accurate Parallel Genetic Algorithms*, Ph.D. Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.
[19] A. Mendiburu, J. A. Lozano and J. Miguel-Alonso, Parallel implementation of EDAs based on probabilistic graphical models, *IEEE Trans. on Evolutionary Computation*, vol.9, no.4, pp.406-423, 2005.
[20] D. E. Goldberg and S. Voessner, Optimizing global-local search hybrids, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99)*, pp.220-228, 1999.
[21] A. Sinha, Y.-P. Chen and D. E. Goldberg, Designing efficient genetic and evolutionary algorithm hybrids, *Studies in Fuzziness and Soft Computing 166: Recent Advances in Memetic Algorithms*, pp.259-288, 2005.
[22] D. E. Goldberg, Using time efficiently: Genetic-evolutionary algorithms and the continuation problem, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99)*, pp.212-219, 1999.
[23] R. P. Srivastava, *Time Continuation in Genetic Algorithms*, Master Thesis, University of Illinois at Urbana-Champaign, Urbana, IL, 2002.
[24] J. J. Grefenstette and J. M. Fitzpatrick, Genetic search with approximate function evaluation, *Proc. of the 1st International Conference on Genetic Algorithm*, pp.112-121, 1985.
[25] K. Sastry, C. F. Lima and D. E. Goldberg, Evaluation relaxation using substructural information and linear estimation, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'06)*, pp.419-426, 2006.
[26] M. Pelikan and K. Sastry, Fitness inheritance in the Bayesian optimization algorithm, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'04)*, pp.48-59, 2004.
[27] C. F. Lima, M. Pelikan, K. Sastry, M. Butz, D. E. Goldberg and F. Lobo, Substructural neighborhoods for local search in the Bayesian optimization algorithm, *Proc. of the Parallel Problem Solving from Nature – PPSN IX, LNCS*, vol.4193, pp.232-241, 2006.
[28] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 2006.
[29] H. N. Luong, H. T. T. Nguyen and C. W. Ahn, Entropy-based evaluation relaxation strategy for Bayesian optimization algorithm, *Proc. of the 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE'10) – LNAI*, vol.6097, pp.126-135, 2010.
[30] H. N. Luong, H. T. T. Nguyen and C. W. Ahn, Entropy-based substructural local search for the Bayesian optimization algorithm, *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'10)*, pp.335-342, 2010.
[31] D. Heckerman, A tutorial on learning Bayesian networks, *TechReport: MSR-TR-95-06*, Microsoft Research, 1995.
[32] M. Pelikan, *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*, Springer-Verlag, 2005.

[33] K. Sastry, M. Pelikan and D. E. Goldberg, Efficiency enhancement of estimation of distribution algorithms, *Studies in Computational Intelligence 33: Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications*, pp.161-186, 2006.

[34] J. Ocenasek, Entropy-based convergence measurement in discrete estimation of distribution algorithms, *Studies in Fuzziness and Soft Computing 192: Towards a New Evolutionary Computation*, pp.39-50, 2006.