Information Sciences 188 (2012) 100–120

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

# Entropy-based efficiency enhancement techniques for evolutionary algorithms

Hoang Ngoc Luong, Hai Thi Thanh Nguyen, Chang Wook Ahn *

*School of Information and Communication Engineering, Sungkyunkwan University, Republic of Korea*

ARTICLE INFO

*Article history:*
Received 4 February 2011
Received in revised form 4 November 2011
Accepted 9 November 2011
Available online 18 November 2011

*Keywords:*
Evolutionary algorithms
Efficiency enhancement
Evaluation relaxation
Entropy measurement
Bayesian optimization algorithm
Network coding

ABSTRACT

This paper introduces the notion of an entropy measurement for populations of candidate solutions in evolutionary algorithms, developing both conditional and joint entropy-based algorithms. We describe the inherent characteristics of the entropy measurement and how these affect the search process. Following these discussions, we develop a recognition mechanism through which promising candidate solutions can be identified without the need of invoking costly evaluation functions. This on-demand evaluation strategy (ODES) is able to perform decision making tasks regardless of whether the actual fitness evaluation is necessary or not, making it an ideal efficiency enhancement technique for accelerating the computational process of evolutionary algorithms.

Two different evolutionary algorithms, a traditional genetic algorithm and a multivariate estimation of distribution algorithm, are employed as example targets for the application of our on-demand evaluation strategy. Ultimately, experimental results confirm that our method is able to broadly improve the performance of various population-based global searchers.



## 1. Introduction

Evolutionary algorithms (EAs) [10,32] are considered by both researchers and practitioners as effective optimizers. Their applications range from academic research [26] and industrial engineering [27,39] to financial analysis [40] and even art production (e.g., music composition, drawings) [19]. These nature-inspired algorithms have grown from being simply *ad hoc* solutions for optimization tasks to being the cornerstones for new philosophies of problem solving. Originating from more traditional approaches to problem solving, such as evolution strategy (ES) [4] and evolutionary programming (EP) [8], especially genetic algorithms (GAs) [17] and genetic programming (GP) [22], EAs have developed into diverse branches. For example, particle swarm optimization (PSO) [20], based on animal flocking behavior, and differential evolution (DE) [31,47], based on vector differences, are well-tailored for numerical optimization. Likewise, by mimicking the ant foraging mechanism, ant colony optimization (ACO) [7] carves a niche for itself in many combinatorial optimization problems. Another state-of-the-art line of EAs, estimation of distribution algorithms (EDAs) [24] utilizes machine learning techniques and statistical methods to learn the underlying structures of problem landscapes. Then these algorithms bias the sampling process (in creating new individuals) toward the promising regions encoded in probabilistic models; thus, the optimal solution can be more readily discovered. In addition, EDAs can combine with other types of EAs to obtain new optimization algorithms, such as PSO-EDA [2] or DE-EDA [48,49]. In sum, evolutionary computation (EC) is a versatile tool that is vastly enhancing the field in terms of both quantity and quality.

* Corresponding author.
*E-mail address:* cwan@skku.edu (C.W. Ahn).

0020-0255/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2011.11.010

While EAs are effective at finding acceptable solutions, they need to be coupled with efficiency enhancement techniques in order to be competitive with other conventional optimizers. Before converging to some optimum in the search space, EAs must evaluate an adequate number of candidate solutions whereby the low-fitness individuals are discarded and the high-fitness ones are selected to generate new offspring. Artificial test problems for designing algorithms usually take negligible amounts of time to be evaluated. In contrast, the cost of a fitness evaluation for an industrial solution may be considerably expensive in terms of time and computing resources. Therefore, an EA solving real-world problems with polynomial complexity can still be impractical due to costly fitness (function) evaluations.

Conveniently, a large corpus of efficiency enhancement techniques (EETs) exists for EAs. In fact, Sastry [41] mentions a classification of EETs, known as the Illinois Decomposition, which categorizes various techniques into four groups: parallelization, hybridization, time continuation/utilization, and evaluation relaxation. Parallelization [6] manages multiple processors at the same time to distribute computational cost (e.g., processing time or power) according to some topologies (e.g., master–slave, fine-grained, coarse-grained, or hierarchical). Hybridization [12,43] combines effective global-search EAs with some efficient local-search methods by taking advantage of domain-specific knowledge to speed up the optimization process. Time continuation [11,46] considers the optimal usage of computational resources to obtain acceptable solutions; for example, practitioners can choose between an EA that has a small population for many generations and an EA that has a large population for a few generations. Evaluation relaxation methods [13,42,44] try to partially replace accurate, but expensive, fitness functions with estimation/approximation models that are more error-prone, but also more economical. Notably, Jin [18] conducted a comprehensive survey of fitness approximations in EAs; however, a more detailed discussion of the aforementioned methods is beyond the scope of this paper.

In this paper, we analyze the inherent randomness of the EA population. More specifically, we discuss the variation tendency of the entropy value of a population, calculated with respect to some model. The entropy observation, combined with the characteristics of the building block structure of the problem at hand, can be used to develop several strategies that enhance the performance of the evolutionary optimization process. This paper demonstrates that there are different types of entropy variations, depending on the search landscape. In some cases, where the entropy value of the population has been monotonically reduced, it can be used to indicate the optimization progress (e.g., initialization, stabilization, or convergence). For other cases, in which the randomness of a population has fluctuated, the entropy value can still be used to signal the existence of an individual with promising or unpromising building blocks. Based on these entropy computations, we can design an on-demand evaluation strategy (ODES) to recognize which individuals should be evaluated by the fitness function and which individuals can be selected without requiring exact evaluations. Note that some of our preliminary research on this topic has been published elsewhere [29,30], but in this paper, we generalize the method, and conduct extensive experiments in terms of both theoretical and real-world problems. In addition, we emphasize three advantages of our method: first, the entropy value of evolutionary populations is an inherent characteristic of an EA itself; thus, ODES can be used without requiring any external approximation function. Second, because the entropy measurement reflects the current status of the optimization process, our method can adapt to changes in the population. Third, since ODES has a wide applicability to different types of EAs, from traditional GAs to advanced EDAs, such as Bayesian optimization algorithm (BOA [36]) or extended compact genetic algorithm (ECGA [14]), this research is useful to many fields.

Note that the proposed technique is categorized as an evaluation relaxation method. Thus, it can be combined with other EETs to accelerate the algorithm's performance. For instance, ODES can be incorporated with a local searcher to effectively estimate the fitness values of individuals in the neighborhood of a candidate solution. More details on this issue; however, are beyond the scope of this work.

The rest of this paper is organized as follows: Section 2 introduces the entropy measurements of populations in EAs, including joint entropy and conditional entropy values in different contexts. Section 3 proposes a conditional entropy-based efficiency enhancement technique and discusses how it can be incorporated into EAs. Section 4 exhibits how a joint entropy-based evaluation relaxation method helps GAs solve real-world problems (e.g., the network coding problem) faster and more efficiently. Extensive experiments with statistical results are provided in both Sections 3 and 4 to provide evidence for the applicability of our approach. Finally, Section 5 concludes the paper and outlines future work.

## 2. Entropy measurements of populations

Ocenasek [33] first proposed an entropy measurement of the evolutionary population as a convergence criterion for the Bayesian optimization algorithm (BOA) [36]. As a multivariate EDA making use of Bayesian networks [34], BOA captures the regular patterns (i.e., the building blocks) existing among promising candidate solutions of the current population. The Bayesian network describes the conditional (in)dependencies (i.e., the interactions) between variables of the optimization function, and then encodes their probability distributions. Instead of employing traditional variation operators, such as crossover or mutation operators, the process of sampling the Bayesian network generates new individuals that possess similar regularities encoded in the model. Thus, the evolutionary population converges toward the promising regions of the search space. The general process of BOA is outlined briefly in Algorithm 1.

**Algorithm 1:** Bayesian optimization algorithm (BOA [36])

1: $g \leftarrow 0$/$*g$ `is the generation counter`$*$/
2: **Initialization:** Generate uniformly and evaluate the initial population $\mathcal{P}(g)$.
3: **while** termination criteria not met **do**
4: **Selection:** Select a set of promising candidates from $\mathcal{P}(g)$ as the parents set $\mathcal{S}(g)$.
5: **Model Building:** Learn a Bayesian network $\mathcal{B}(g)$ from $\mathcal{S}(g)$.
6: **Model Sampling:** Generate a set of new offspring $\mathcal{O}(g)$ by sampling $\mathcal{B}(g)$.
7: **Evaluation:** Evaluate entire $\mathcal{O}(g)$.
8: **Replacement:** Replace some solutions of $\mathcal{P}(g)$ by $\mathcal{O}(g)$ to create $\mathcal{P}(g+1)$.
9: $g \leftarrow g + 1$
10: **end while**

A Bayesian network [34] encodes the following joint probability distribution

$$p(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} p(X_i|\Pi_i), \tag{1}$$

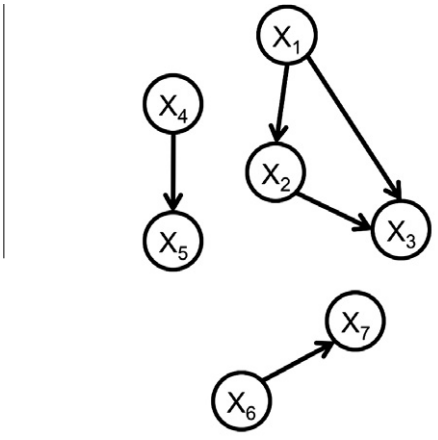
where each node $X_i$ is the $i$th random variable, $\Pi_i$ is the set of parent nodes of $X_i$ in the network (i.e., from each node of the set $\Pi_i$, there exists an edge directing into $X_i$), and $p(X_i|\Pi_i)$ is the conditional probability of $X_i$ given its parent $\Pi_i$. Each variable $X_i$ has a corresponding conditional probability table (CPT) that stores its conditional probabilities concerning all possible values of $\Pi_i$. A more detailed introduction to BOA can be found in Appendix A.

The entropy measurement proposed by Ocenasek [33] can be computed by summing the conditional entropy of each variable given its parents, with respect to the current Bayesian network and population as follows:

$$\begin{aligned} H(\mathbf{X}) &= \sum_{i=1}^{n} H(X_i|\Pi_i) = \sum_{i=1}^{n} \sum_{\pi_i \in \mathcal{P}_i} p(\pi_i) H(X_i|\Pi_i = \pi_i) = -\sum_{i=1}^{n} \sum_{\pi_i \in \mathcal{P}_i} p(\pi_i) \sum_{x_i \in \mathcal{X}_i} p(x_i|\pi_i) \log_2 p(x_i|\pi_i) \\ &= -\sum_{i=1}^{n} \sum_{\pi_i \in \mathcal{P}_i} \sum_{x_i \in \mathcal{X}_i} p(x_i, \pi_i) \log_2 p(x_i|\pi_i) = -\sum_{i=1}^{n} \sum_{\pi_i \in \mathcal{P}_i} \sum_{x_i \in \mathcal{X}_i} \left(\frac{m(x_i, \pi_i)}{N}\right) \log_2 \left(\frac{m(x_i, \pi_i)}{m(\pi_i)}\right) \end{aligned} \tag{2}$$

where $\mathcal{P}_i$ is the set of all instances of $\Pi_i$, $\mathcal{X}_i$ denotes the set of all possible values of $X_i$, $N$ is the population size, $m(x_i, \pi_i)$ is the number of individuals having $(X_i, \Pi_i)$ set to $(x_i, \pi_i)$, and $m(\pi_i)$ is the number of individuals having $\Pi_i$ equal to $\pi_i$. Fig. 1 presents an illustrative example of a Bayesian network and its entropy computation.

Using (2), we can compute the entropy of particular portions of the population with respect to the current network. Fig. 2 shows that the entropy of the better half of the population (i.e., the parents set) decreases after every iteration. When the BOA converges, this entropy value reaches its minimum value of 0. This result is naturally logical as the BOA continually narrows its sampling region toward a certain promising region of the search space: The closer the algorithm moves to a specific convergence point, the more predictable it becomes. Thus, the inherent randomness of the probability distribution in the population gradually decreases.




$$H(\boldsymbol{X}) = H(X_1) + H(X_2|X_1) + H(X_3|X_1, X_2) + H(X_4) + H(X_5|X_4) + H(X_6) + H(X_7|X_6)$$

**Fig. 1.** A Bayesian network and its corresponding entropy computation.

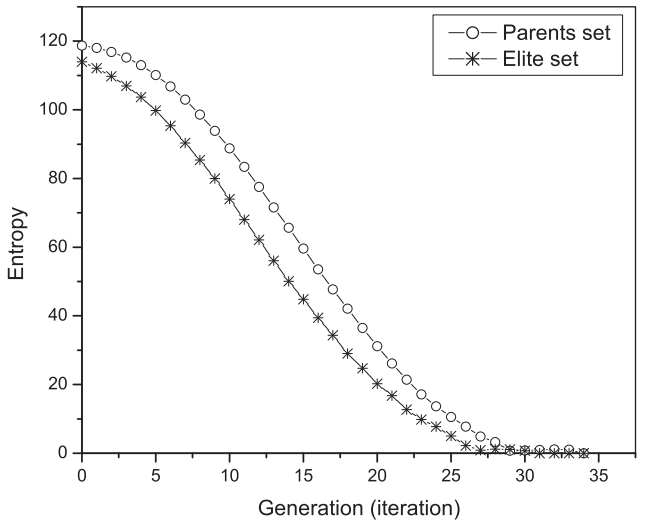



**Fig. 2.** Entropy reduction in a BOA solving a 120-bit trap-5 problem.

We define an elite set as a set of the most promising individuals selected from the population. In Fig. 2, the top $\tau = 5\%$ of the population (in terms of fitness values) were selected as the elite set. This set also exhibited a tendency to decrease its entropy measurement after each iteration. Obviously, this regional entropy was always smaller than the overall entropy of the entire population, and it reached the minimum value of 0 sooner. From such an observation, we conjecture that a promising candidate solution is an individual whose appearance in the elite set causes a reduction in the entropy measurement of that set.

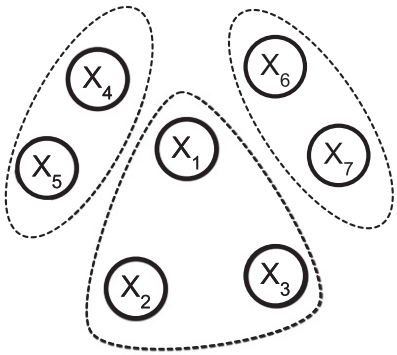
Besides BOA, other instances of EAs exist that simply model the interactions between variables as undirected linkage groups. For example, an extended compact genetic algorithm (ECGA) [14] divides variables into disjoint partitions that each corresponds to one independent component of the problem decomposition. Variables belonging to the same linkage group are considered to have interactions with each other. Apart from the joint probability distributions encoded in the model, ECGA does not describe any conditional (in)dependencies between variables. Similarly, in real-world optimization problems, the only problem-specific information available at hand for many tasks is the interaction between its variables – not their conditional (in)dependencies. For these cases, we devise the following joint entropy measurement of evolutionary populations:

$$\begin{aligned} H(\mathbf{X}) &= \sum_{i=1}^{m} H_i\left(X_1^i, X_2^i, \ldots, X_{n_i}^i\right) = -\sum_{i=1}^{m} \sum_{x_1^i, x_2^i, \ldots, x_{n_i}^i} p\left(x_1^i, x_2^i, \ldots, x_{n_i}^i\right) \log_2 p\left(x_1^i, x_2^i, \ldots, x_{n_i}^i\right) \\ &= -\sum_{i=1}^{m} \sum_{x_1^i, \ldots, x_{n_i}^i} \left(\frac{m\left(x_1^i, \ldots, x_{n_i}^i\right)}{N}\right) \log_2 \left(\frac{m(x_1^i, \ldots, x_{n_i}^i)}{N}\right) \end{aligned} \tag{3}$$

where $m$ is the number of linkage groups, $n_i$ is the number of interacting variables in the $i$th linkage groups, $X_j^i$ is the $j$th variable in the $i$th linkage group, $N$ is the population size, and $m\left(x_1^i, x_2^i, \ldots, x_{n_i}^i\right)$ is the number of individuals having $\left(X_1^i, X_2^i, \ldots, X_{n_i}^i\right)$ set to $\left(x_1^i, x_2^i, \ldots, x_{n_i}^i\right)$. Fig. 3 shows an example of the topology of linkage groups, and how its corresponding entropy can be calculated.

Depending on the types of EAs being used or prior knowledge about the linkage groups, different entropy computation models can be developed. In this context, this section presented two equations, termed *conditional* and *joint* entropy, which can be straightforwardly computed from the probability distribution models of EDAs, such as BOA and ECGA. Note that problem-specific information helps to define linkage groups in traditional GAs. Then by computing the probability distribution of the linkage groups, we can calculate the joint entropy value.

The building block is a fundamental notion in evolutionary computation theory. An evolutionary process should efficiently assemble "good" building blocks (a group of genes) to create chromosomes (i.e., genotypes) representing promising individuals (i.e., phenotypes), which ultimately produces the optimal solution. The exact quality of candidate solutions can only be specified by invoking costly fitness functions to evaluate the phenotypes. A heuristic can be used to estimate the quality of an individual by considering its genotype. The elite set, defined as a collection of individuals having the highest fitness values, can be seen as a reference for determining which building blocks are "good". If an individual has *many* good building blocks that are similar to the ones contained in the elite set, it can be deemed a promising solution. Meanwhile, if an individual has *few* good building blocks, its quality is unknown. In this paper, we propose a method to quantify the terms




$$H(\boldsymbol{X}) = H(X_1, X_2, X_3) + H(X_4, X_5) + H(X_6, X_7)$$

**Fig. 3.** Building blocks and their corresponding joint entropy computation.

*many* and *few* by computing the entropy values before and after the addition of an individual into the elite set. An individual having building blocks similar to the elite set should decrease the entropy value of the elite set, and vice versa. Therefore, when an individual has been previously determined to be a promising solution, we can select it for the next generation without evaluating the fitness function. The following sections will demonstrate how our method can be applied to different types of EAs.

## 3. A conditional entropy-based technique

### 3.1. Proposed algorithm

Algorithm 2 presents an example of the application of our entropy-based evaluation relaxation technique to the Bayesian optimization algorithm (eBOA). Based on the conditional (in)dependencies encoded in the Bayesian network, the conditional entropy value of the population can be computed. In general, the eBOA is nearly the same as the original BOA [36], except that it includes an additional elite-set selection step as well as an on-demand evaluation phase. Besides selecting the parents set of promising individuals to build the dependencies model, the eBOA requires another selection operation of elite candidate solutions which comprise a small groups of individuals having the highest fitness values. In the evaluation phase, instead of evaluating all the newly generated offspring with an actual fitness function, the eBOA recognizes which individuals should be estimated by a surrogate model.

**Algorithm 2:** Entropy-based evaluation relaxation technique for BOA

1: $g \leftarrow 0$ /*$g$ is the generation counter*/
2: $t \leftarrow 0$ /*$t$ is the ODES counter*/
3: **Initialization:** Generate uniformly and evaluate the initial population $\mathcal{P}(g)$.
4: **while** termination criteria not met **do**
5: **Selection:**
6: • Select a set of promising candidates from $\mathcal{P}(g)$ as the parents set $\mathcal{S}(g)$.
7: • Select a set of solutions having highest fitness values from $\tau\%$ of $\mathcal{P}(g)$ as the elite set $\mathcal{E}(g)$.
8: **Model Building:** Learn a Bayesian network $\mathcal{B}(g)$ from $\mathcal{S}(g)$.
9: **Model Sampling:** Generate a set of new offspring $\mathcal{O}(g)$ by sampling $\mathcal{B}(g)$.
10: **On-demand Evaluation**: Consider each individual of $\mathcal{O}(g)$, and evaluate it when necessary using the information from $\mathcal{B}(g)$ and $\mathcal{E}(g)$.
11: **Replacement:** Replace some solutions of $\mathcal{P}(g)$ by $\mathcal{O}(g)$ to create $\mathcal{P}(g+1)$.
12: $g \leftarrow g + 1$
13: **end while**

Evaluating the fitness of a candidate solution is an expensive operation, so it should only be done when necessary. As stated in Section 2, an individual is a promising candidate solution if its appearance results in a reduction in the entropy value of the elite set. If an individual causes such an entropy reduction, it has the same characteristics (in terms of building blocks) as the other candidate solutions in the elite set. Thus, it can be selected without being evaluated by the actual fitness function. Otherwise, the individual does not belong to the elite set, and it should be evaluated to obtain its correct fitness value. We have named this mechanism the *on-demand evaluation strategy* (ODES). The details of ODES are described in Algorithm 3.

**Algorithm 3:** On-demand evaluation strategy (ODES)

1: **Entropy Computation**: Based on the Bayesian network $\mathcal{B}(g)$, compute the entropy $\mathcal{H}(g)$ of $\mathcal{E}(g)$.
/*$\eta$ is the starting point of ODES*/
/*$\kappa$ is the interval of entire offspring evaluations*/
2: **if** $\mathcal{H}(g) \leqslant \eta$ **and** $t < \kappa$ **then**
3: **for** each individual $\mathcal{X}$ in $\mathcal{O}(g)$ **do**
4: • $\mathcal{E}_{\mathcal{X}}(g) \leftarrow \mathcal{E}(g) + \mathcal{X}$.
5: • Compute $\mathcal{H}_{\mathcal{X}}(g)$ for $\mathcal{E}_{\mathcal{X}}(g)$.
6: **if** $\mathcal{H}_{\mathcal{X}}(g) \leqslant \mathcal{H}(g)$ **then**
7: Estimate $f(\mathcal{X})$ as $f_{\text{estimation}}(\mathcal{X}) = f(\mathcal{Y})$.
$\{\mathcal{Y} \in \mathcal{E}(g), \forall \mathcal{Z} \in \mathcal{E}(g), f(\mathcal{Y}) \leqslant f(\mathcal{Z})\}$
8: **else**
9: Evaluate $f(\mathcal{X})$.
10: **end if**
11: **end for**
12: $t \leftarrow t + 1$.
13: **else**
14: **Evaluation:** Evaluate entire $\mathcal{O}(g)$.
15: $t \leftarrow 0$.
16: **end if**

If an individual is deemed to belong to the elite set, its fitness value should be approximated rather than evaluated. Conveniently, a number of different approximation models exist that effectively estimate the quality of candidate solutions (see [18]). In particular, the equation of fitness inheritance in BOA [37] has been shown to produce good results. However, these models have not been employed in this paper because we chose to focus on the true effectiveness of ODES – i.e., any improvement in the efficiency of BOA should stem primarily from the contribution of ODES. Thus, instead of using a complicated surrogate model, we chose a simple fitness assignment method: If an individual is recognized as a promising solution belonging to the elite set, it will be assigned the fitness value of the worst individual in the elite set. The justification for this method is that it minimizes the severity of incorrect estimation errors. For the same reason, the elite set should only be selected from candidates previously evaluated by the actual fitness function.

Note that large elite sets generally contain some estimated individuals, unless enough evaluated individuals exist. However, this inclusion does not negatively affect the performance for discrete problems (i.e., problems with finite possible states). On the other hand, for continuous problems (i.e., problems with infinite states), such inclusion allows the estimation errors to accumulate in the elite set, but this issue is outside the scope of this paper and should be addressed in other work.

The condition $\mathcal{H}(g) \leqslant \eta$ denotes that we wait until the entropy of the elite set decreases to some appropriate value before applying ODES. In the experiment with BOA in Section 3.4, test results will show that $\eta = \frac{\mathcal{H}(0)}{2}$ is the most suitable starting point for ODES (i.e., when the entropy of the elite set decreases by half of its initial value). Our choice for this particular juncture is based on the research conducted by Hauschild et al. [15] on the probabilistic models constructing dynamics of hierarchical BOA (hBOA). In particular, at the beginning of the optimization process, the Bayesian network contains only a few of its necessary dependencies along with a number of spurious linkages; the candidate solutions are distributed over a large search space. Therefore, if we were to start ODES during this period, the algorithm would not have an adequate opportunity to recognize the good individuals due to the environment of high entropy values. At the same time, if we were to wait until the algorithm is nearly done converging, we would miss the chance of achieving the maximal reduction in the number of fitness evaluations. On the other hand, in the middle of the run, the Bayesian network would already discover and capture nearly all the necessary dependencies [15], and it would discard some unnecessary linkages. Thus, in the case of BOA, the iteration (i.e., generation) $g$, in which $\mathcal{H}(g) \leqslant \frac{\mathcal{H}(0)}{2}$, is a rational choice.

Moreover, if ODES is continuously employed, estimation errors would accumulate and prevent BOA from converging. Conveniently, we can simply eliminate such accumulated errors by sporadically performing actual evaluations of the entire offspring population. In other words, we can perform this full evaluation only at every certain number of generations rather than conducting it at each iteration as in the original BOA. More complicated dynamics can be designed to adaptively specify suitable iterations in order to apply the sporadic full evaluations. In this paper, however, for the sake of simplicity, we set a fixed parameter such that after every $\kappa$ generations, the actual function would be employed to evaluate the fitness values of the entire offspring population (see the condition $t < \kappa$ in Algorithm 3).

### 3.2. Test problems

For this paper, we employ several versions of concatenated trap functions as the benchmarks for the algorithms. These functions are the classic problems for testing linkage-learning EAs because they are easy to extend for more complicated structures.

OneMax [35], the simplest test set, can be easily evaluated as the sum of all bit values in a binary string (i.e., the number of ones).

$$f_{\text{onemax}}(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} X_i, \tag{4}$$

where $(X_1, X_2, \ldots, X_n)$ is an input binary string of $n$ bits. Thus, the optimum of an $n$-bit OneMax problem is an $n$-bit string of all ones. This test case is so simple that BOA is already efficient at obtaining the optimal solution with a small number of fitness evaluations. However, we wish to demonstrate that the additional complexity that ODES contributes does not have negative effects on the performance of BOA when solving easy problems.

A concatenated 5-bit trap problem [35] consists of several non-overlapping deceptive functions of order 5. Each trap-5 function can be defined as follows:

$$f_{\text{trap5}}(u) = \begin{cases} 5 & \text{if } u = 5 \\ 4 - u & \text{if } u < 5 \end{cases} \tag{5}$$

where $u$ is the number of ones in a single trap-5 function. The overall fitness value of the trap-5 problem is the sum of all the separate trap functions; hence, an $n$-bit trap-5 problem has one global optimum with a string of all ones and $(2^{n/5} - 1)$ local optima. Note that this trap-5 problem cannot be solved if the optimizer fails to decompose the problem into order-5 subproblems since statistics on any orders lower than 5 would be misleading. Recall that the purpose of this test case is to show that ODES can accelerate BOA in solving nontrivial problems of bounded difficulty.

An overlapped trap problem [25] is also the concatenation of several trap functions, but each pair of adjacent functions can share some bits together. For example, in a 2-bit overlapped trap-5 problem, each trap function is overlapped by two bits via its direct left and right neighbor functions. As before, the optimum of this function is a string of all ones. Due to its complicated structure and decomposition, the overlapped trap problem requires a large number of fitness evaluations by BOA in order to obtain the optimal solution. We wish to demonstrate that ODES can speed up BOA by considerably reducing the number of costly evaluations in solving complicated problems.

### 3.3. Experimental configuration

In order to convincingly show that our approach does indeed improve the performance on the above test problems, we need to conduct our experiments thoroughly and collect sufficient statistical data. Thus, in this experiment, we employ the bisection method, as described in Pelikan [35], which is a popular test framework for EAs. The bisection test framework is outlined in Algorithm 4.

**Algorithm 4:** Bisection method for determining the minimum population size [35]

```
N ← initial population size
success ← all n runs with pop. size N successful?
if success then
  while (success = true) do
    N ← N/2
    success← all n runs with pop. size N successful?
  end while
  low ← N
  high ← 2 N
else
  while (success = false) do
    N ← 2N
    success ← all n runs with pop. size N successful?
  end while
  low ← N/2
  high ← N
end if
while ((high − low)/low ⩾ 0.10) do
  N ← (high + low)/2
  success ← all n runs with pop. size N successful?
  if (success = false) then
    low ← N
  else
    high ← N
  end if
end while
return high
```

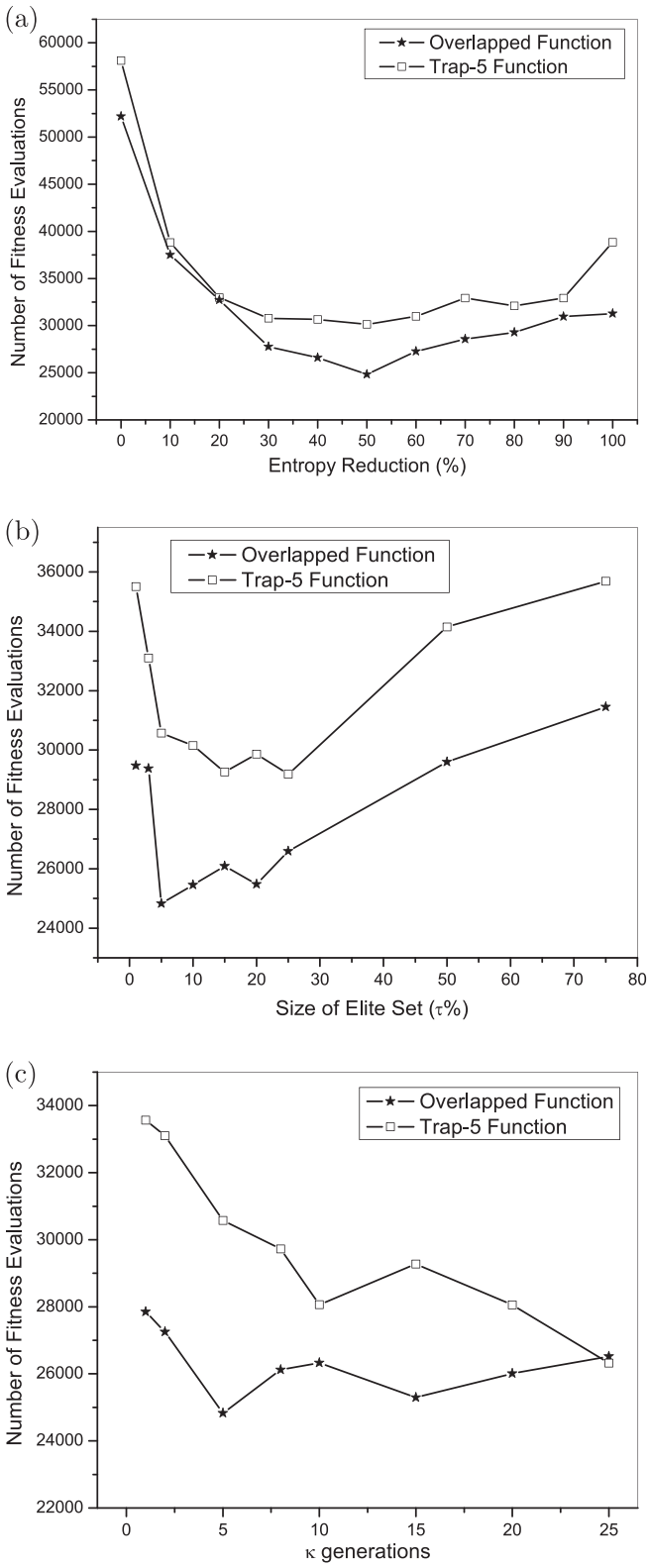



**Fig. 4.** Effects of ODES parameters on eBOA performance. (a) Fitness evaluations *vs.* entropy reduction, (b) Fitness evaluations *vs.* size of elite set, (c) Fitness evaluations *vs.* interval of sporadic evaluation.

In this experiment, we compare the number of fitness evaluations and the population sizes that both BOA and eBOA require to obtain the optimal solution. For each test case of different problem sizes, ten independent runs of the bisection method are performed. Each bisection attempts to find the minimal population size that enables the optimizer to obtain the optimal solution in all 10 out of 10 independent runs. Consequently, the statistical results of BOA and eBOA are averaged and compared over 100 ($10 \times 10$) runs. The initial population size for each bisection is set to 500.

For both algorithms, a truncation selection (with a threshold of 50%) is employed to select the better half of the current population to become the parents set. The other half of the current population is then replaced by the newly generated offspring after being evaluated or having their fitness values estimated. The top $\tau = 5\%$ individuals (in terms of fitness values) are selected as the elite set. We set $\eta = \frac{\mathcal{H}(0)}{2}$ and $\kappa = 5$ as the starting point of ODES and as the interval of the entire offspring evaluations, respectively. In addition, we set the convergence criterion to be when the proportion of a particular value on each position reaches 99% or when the maximal number of generations is exceeded.

### 3.4. Results and discussion

The proposed technique (i.e., ODES) introduced three important parameters: the starting point of ODES, $\eta$; the size of elite set, $\tau$; and the interval of the entire offspring evaluations, $\kappa$. Since these parameters influence the algorithm's performance significantly, we examined their effects first. We performed experiments for eBOA (i.e., BOA employing ODES) solving a 60-bit concatenated trap-5 problem and a 32-bit overlapped trap-5 problem (using multiple 2-bit overlapped trap-5 functions).

Fig. 4(a)–(c) provides the rationale for our choices of the parameter settings for all the experiments in this study. For instance, Fig. 4(a) presents the experimental results for eBOA with different $\eta$ settings. In terms of entropy reduction, 0% means the start of ODES from the beginning of the optimization process, 100% indicates the original BOA (i.e., ODES was not used), and 50% comes under our choice. These experimental results showed that starting ODES after the entropy of the elite set had decreased down to half of its original value achieved the maximal reduction in the fitness evaluations, so we set $\eta = \frac{\mathcal{H}(0)}{2}$ as the starting point of ODES in eBOA.

Fig. 4(b) shows the performances of eBOA for various $\tau$ settings. For smaller values of $\tau$, such as 1% or 3%, the elite set became so small that it could not include enough promising individuals (i.e., good building blocks). Consequently, this tiny elite set was not able to recognize new promising individuals. On the other hand, for larger values of $\tau$ (more than 50%), the elite set contained too many unfavorable individuals (i.e., bad building blocks). Thus, the algorithm was not able to make precise decisions on whether a new offspring belonged to the elite set or not. Note that the performance was not sensitive to $\tau$ setting unless a very small/large value was used. As a result, we found $\tau = 5\%$ to be suitable for the elite set size in eBOA.

Fig. 4(c) depicts the experimental results for eBOA with varying $\kappa$ values. For smaller values of $\kappa(<5)$, the fitness function was actually evaluated so frequently that the maximum reduction in fitness evaluations could not be achieved. For larger values of $\kappa(\geqslant 15)$, there was no significant difference among the different $\kappa$ values. Note that the $\kappa$ setting was not critical to the performance unless a very small value was used. Generally, it is hard to employ an accurate estimation model that avoids accumulating the estimation errors. If the entire offspring are not often actually evaluated, the algorithm will stagnate – i.e., it cannot proceed to better solutions – meaning that a very large $\kappa$ value is not also desirable. Thus, we found $\kappa = 5$ to be acceptable for the interval of sporadic evaluations in eBOA.

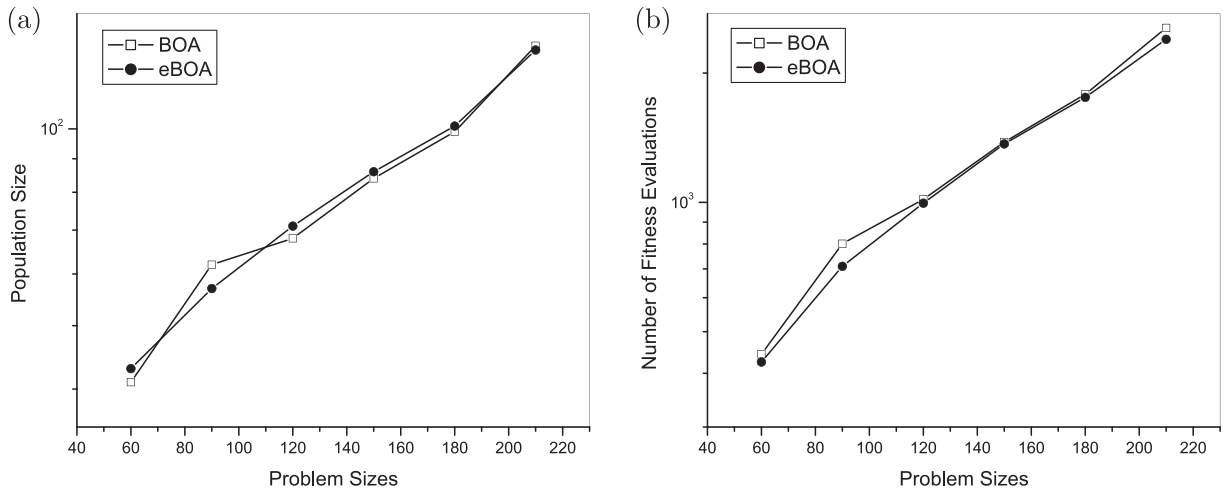



**Fig. 5.** Population size and number of evaluations required for solving the OneMax problem. (a) Population size comparison, (b) Fitness evaluations comparison.

Fig. 5(a) and (b) show the experimental results comparing the original BOA with our eBOA when applied to solve OneMax problems. The statistical tests (see Tables 1 and 2) demonstrated that applying ODES on BOA had little effect on the performance of the optimization algorithm. By experimenting on several test cases with problem sizes ranging from 60 to 210, the difference between the two optimizers was found to be insignificant in terms of both population sizes and the numbers of fitness evaluations. OneMax is such a simple function that it can be easily solved by univariate EDAs because there are no interactions between the variables. Therefore, any dependencies discovered by a multivariate EDA (e.g., ECGA, BOA) in solving OneMax are certainly false linkages; these would make the optimization process become slower at finding the optimal solution. Fortunately, while employing ODES means adding another layer of complexity, it does not have any negative effect on the performance of BOA as shown in the experimental statistics. The superiority of eBOA is

**Table 1**
Statistical comparison of the number of evaluations of algorithms for the OneMax problem.

| Size | 60 | 90 | 120 | 150 | 180 | 210 |
|---|---|---|---|---|---|---|
| BOA | 443 | 801 | 1017 | 1382 | 1785 | 2548 |
| ($\sigma$) | (80.7) | (125.0) | (141.2) | (197.1) | (332.0) | (327.4) |
| eBOA | 425 | 710 | 996 | 1368 | 1756 | 2397 |
| ($\sigma$) | (82.9) | (139.8) | (132.5) | (142.2) | (256.9) | (278.2) |
| $p$-value | Statistical $t$-test: (BOA – eBOA) | | | | | |
| | 0.11017 | 9.59E−8[†] | 0.20164 | 0.55974 | 0.4681 | 2.44E−4[†] |

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

**Table 2**
Statistical comparison of the population sizes of algorithms for the OneMax problem.

| Size | 60 | 90 | 120 | 150 | 180 | 210 |
|---|---|---|---|---|---|---|
| BOA | 41 | 62 | 68 | 84 | 99 | 134 |
| ($\sigma$) | (7.4) | (11.5) | (9.5) | (13.3) | (18.8) | (18.0) |
| eBOA | 43 | 57 | 71 | 86 | 101 | 132 |
| ($\sigma$) | (7.4) | (11.1) | (10) | (9.0) | (15.0) | (16.6) |
| $p$-value | Statistical $t$-test: (BOA – eBOA) | | | | | |
| | 0.18955 | 4.65E−5[†] | 0.02914 | 0.19346 | 0.36393 | 0.36919 |

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

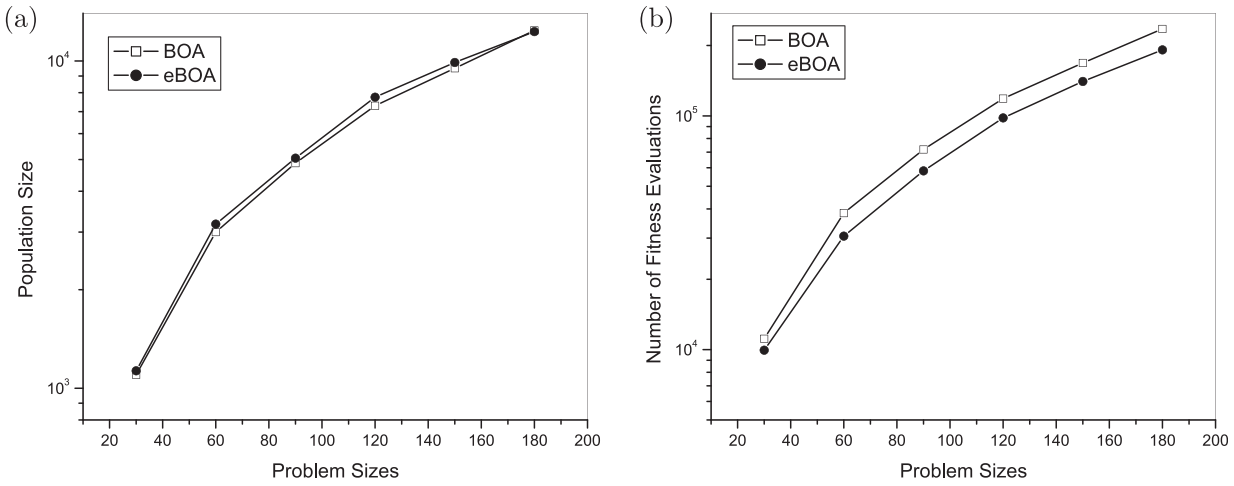



**Fig. 6.** Population size and number of evaluations required to solve the trap-5 problems. (a) Population size comparison, (b) Fitness evaluations comparison.

demonstrated more clearly when solving more complicated problems, such as concatenated deceptive functions or overlapped trap functions.

Fig. 6(a) and (b) illustrate the performance of the conventional BOA and our entropy-based approach when solving concatenated 5-bit trap problems. The statistical tests showed that, in some cases, there were differences in the population-sizing requirements of the two algorithms, in which our eBOA needed a slightly larger supply of candidate solutions in each generation (Table 4). However, the eBOA outperformed the conventional approach by significantly reducing the number of fitness evaluations (Table 3); thus, the efficiency of optimization process was considerably improved. While the population-sizing requirement is important, the number of actual function evaluations is the deciding factor in determining the performance of an optimizer. Furthermore, in all test cases with problem sizes ranging from 30 to 180, ODES accelerated

**Table 3**
Statistical comparison of the number of evaluations of algorithms for the trap-5 problem.

| Size | 30 | 60 | 90 | 120 | 150 | 180 |
|---|---|---|---|---|---|---|
| BOA | 11,132 | 38,382 | 71,813 | 118,567 | 168,415 | 235,455 |
| ($\sigma$) | (1287.0) | (3363.0) | (3935.4) | (9220.2) | (12933.3) | (15596.2) |
| eBOA | 9938 | 30,575 | 58,148 | 98,086 | 140,448 | 191,572 |
| ($\sigma$) | (1566.8) | (2532.1) | (5960.7) | (6596.5) | (11581.2) | (9822.5) |
| *p*-value | Statistical *t*-test: (BOA – eBOA) | | | | | |
| | 3.62E−20† | 2.05E−27† | 1.24E−30† | 7.88E−37† | 1.66E−37† | 2.49E−26† |

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

**Table 4**
Statistical comparison of the population sizes of algorithms for the trap-5 problem.

| Size | 30 | 60 | 90 | 120 | 150 | 180 |
|---|---|---|---|---|---|---|
| BOA | 1099 | 3000 | 4875 | 7300 | 9500 | 12,400 |
| ($\sigma$) | (129.2) | (297.2) | (257.4) | (643.5) | (870.3) | (921.1) |
| eBOA | 1131 | 3175 | 5050 | 7750 | 9900 | 12,300 |
| ($\sigma$) | (90.7) | (338.0) | (672.3) | (605.1) | (921.1) | (643.5) |
| *p*-value | Statistical *t*-test: (BOA – eBOA) | | | | | |
| | 0.06427 | 1.15E−4† | 0.00347† | 1.67E−6† | 0.00416† | 0.47093 |

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

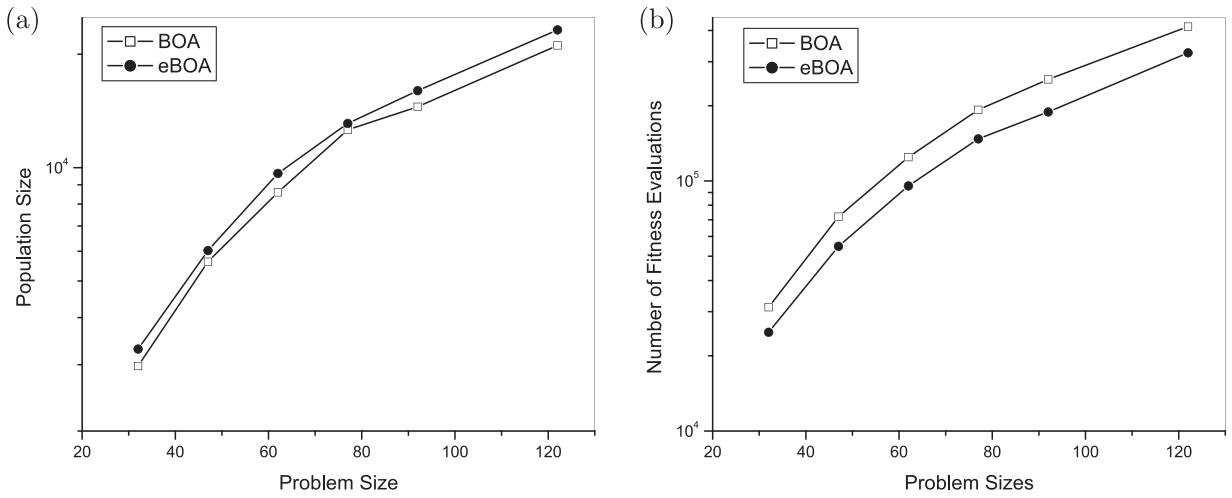



**Fig. 7.** Population size and number of evaluations required to solve the 2-bit overlapped trap-5 problem. (a) Population size comparison, (b) Fitness evaluations comparison.

**Table 5**
Statistical comparison of the number of evaluations of algorithms for the 2-bit overlapped trap-5 problem.

| Size | 32 | 47 | 62 | 77 | 92 | 122 |
|---|---|---|---|---|---|---|
| BOA | 31,278 | 71,990 | 124,612 | 192,590 | 254,805 | 414,385 |
| ($\sigma$) | (4746.9) | (9901.3) | (15060.3) | (18089.4) | (29143.6) | (45600.1) |
| eBOA | 24,826 | 54,762 | 95,602 | 147,320 | 188,799 | 325,545 |
| ($\sigma$) | (2971.5) | (6120.3) | (10372.5) | (16306.3) | (16462.3) | (41159.7) |
| *p*-value | Statistical *t*-test: (BOA – eBOA) | | | | | |
| | 1.70E−9[†] | 2.06E−37[†] | 1.60E−40[†] | 1.00E−33[†] | 9.77E−29[†] | 6.24E−36[†] |

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

**Table 6**
Statistical comparison of the population sizes of algorithms for the 2-bit overlapped trap-5 problem.

| Size | 32 | 47 | 62 | 77 | 92 | 122 |
|---|---|---|---|---|---|---|
| BOA | 2975 | 5625 | 8600 | 12,600 | 14,500 | 21,100 |
| ($\sigma$) | (306.7) | (517.9) | (627.6) | (1287.0) | (674.1) | (1982.2) |
| eBOA | 3300 | 6025 | 9650 | 13,100 | 16,000 | 23,200 |
| ($\sigma$) | (293.0) | (508.1) | (505.0) | (948.1) | (898.9) | (1608.0) |
| *p*-value | Statistical *t*-test: (BOA – eBOA) | | | | | |
| | 1.24E−18[†] | 9.63E−6[†] | 2.06E−17[†] | 0.00288[†] | 1.30E−17[†] | 2.90E−19[†] |

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

the BOA without compromising the scalability of the algorithm. Therefore, we can infer from the figures that if the problem size is extended, ODES would still maintain its positive effect on BOA.

Fig. 7(a) and (b) show the effects of applying ODES to BOA in order to solve 2-bit overlapped trap-5 deceptive problems. The results were similar to the observations in the test cases of the concatenated 5-bit trap problems. While applying ODES required a larger population size to efficiently recognize which individuals should be evaluated (see Table 6), the eBOA compensated by requiring a significantly smaller number of fitness evaluations to achieve a reliable convergence compared to the original algorithm (see Table 5). Also, these results verified that ODES did not have any negative effects on the scalability of BOA. It should be underlined that the problem of overlapped trap functions is considerably more difficult than the corresponding concatenated versions in terms of both building Bayesian networks and in the number of fitness evaluations. For instance, a traditional GA would require an exponential amount of fitness evaluations to find an acceptable solution to this problem. Although BOA can achieve subquadratic scalability [38] for a real-world optimization problem with the same level of difficulty as overlapped trap functions, the subquadratic increases in terms of fitness evaluations are still considerable. Therefore, the reduction in fitness evaluations brought about by applying ODES denotes a substantial improvement in the performance of BOA.

## 4. A joint entropy-based technique

### 4.1. Proposed algorithm

In the previous section, we proposed a case study for incorporating ODES with BOA. Bayesian networks contain such resourceful information, stored in CPTs or decision trees, that the conditional entropy can be directly computed by using (2). However, other EAs may not have a dedicated model like BOA; for example, ECGA has information about the linkage groups of interacting variables but not about their conditional (in)dependences. Consequently, in order to enhance ECGA via ODES, we need to change from using (2) and (3) (i.e., using joint entropy computation). Furthermore, we also wish to extend the applicability of ODES beyond the scope of EDAs to traditional GAs, which have been widely employed in industrial optimization problems. Admittedly, without probabilistic models, entropy values cannot be straightforwardly calculated by either (2) or (3); however, industrial applications of GAs usually come with prior knowledge about problem domains. This knowledge often provides useful information about the interactions between variables; for instance, several variables should be treated together since there are connections or correlations between them. In those cases, the variables with connections can form linkage groups, which can be used to compute the probability distributions of the building blocks, and can help obtain the joint entropy measurement of the evolutionary population as discussed in Section 2.

**Algorithm 5:** Entropy-Based Evaluation Relaxation Technique for GA

1: $g \leftarrow 0$ /*$g$ is the generation counter*/
2: $t \leftarrow 0$ /*$t$ is the ODES counter*/
3: **Initialization:** Generate uniformly and evaluate the initial population $\mathcal{P}(g)$.
4: **while** termination criteria not met **do**
5: **Selection:**
6: • Select a set of promising candidates from $\mathcal{P}(g)$ as the parents set $\mathcal{S}(g)$.
7: • Select a set of solutions having highest fitness values from $\tau$% of $\mathcal{P}(g)$ as the elite set $\mathcal{E}(g)$.
8: **Crossover:** Recombine selected individuals to create new offspring.
9: **Mutation:** Mutate several bits in the newly generated offspring.
10: **On-demand Evaluation**: Consider each individual of $\mathcal{O}(g)$, and evaluate it when necessary using the information from $\mathcal{E}(g)$ and the linkage of variables (if any).
11: **Replacement:** Replace some solutions of $\mathcal{P}(g)$ by $\mathcal{O}(g)$ to create $\mathcal{P}(g+1)$.
12: $g \leftarrow g+1$
13: **end while**

Algorithm 5 presents the framework for how traditional GAs can be combined with our entropy-based evaluation relaxation technique. The *on-demand evaluation* phase can be similarly implemented as in Algorithm 3, but the equation for joint entropy measurement is used in all steps of the entropy computation (see (3)). Algorithm 5 demonstrates that ODES can be incorporated not only into complicated EDAs but also into simple GAs without major changes to the algorithm implementation. The only requisite for computing entropy values is the problem-specific information about the correlations between the variables. This knowledge may be available when observing or investigating any nontrivial optimization problems.

### 4.2. Test problem

In this section, we wish to demonstrate the applicability of the joint entropy computation in evaluation relaxation. Instead of applying our ODES to EDAs having joint probabilistic distribution models, such as ECGA, we choose to present a case study of a traditional GA solving a real-world problem, and then apply ODES to enhance its performance. The target optimization task is to find the optimal topology for the network coding problem.

Traditional network routing employs a simple mechanism by which interior nodes can only replicate and/or forward data received from an incoming link to several outgoing links. On the other hand, network coding [1] enables network nodes to perform more complicated algebraic operations, such as addition or subtraction, on data received from multiple incoming links and then forward the combined information. Packets from several incoming links can be linearly combined according to the equations at the merging nodes. The flexibility of network coding helps to support a large amount of multicast traffic in data routing for demanding services such as video conferencing. A simple network coding topology would be to enable full coding ability at all the nodes in the network. However, network coding nodes are apparently much more expensive, in terms of construction and operating costs, than conventional network nodes. It is crucial, therefore, to determine an appropriate subset of coding nodes that are more economical without compromising on the required multicast rates. Unfortunately, finding a minimal set of coding nodes in an arbitrary network is known as an NP-hard problem [23]. Fig. 8(a) shows an example of how network $A$ employs coding at node $h$ to transmit two units of data to the sinks $t_1$ and $t_2$. If the link $h \rightarrow w$ has a capacity of 2, as in network $B$ (see Fig. 8(b)), then the multicast rate '2' is achievable without network coding. In order to achieve a multicast rate '2' from source $s$ to all the sinks $t_1$, $t_2$, and $t_3$, the network $C$ needs coding at either node $h$ or node $u$, but not both, as depicted in Fig. 8(c).

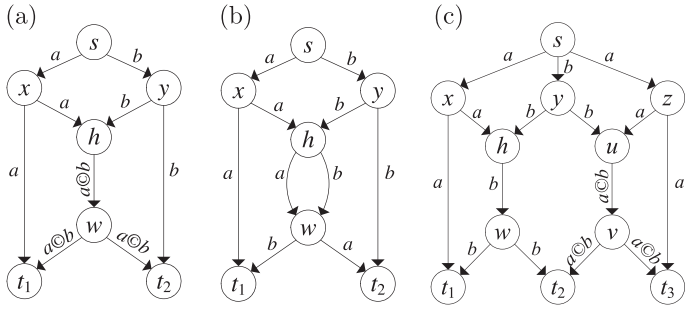



**Fig. 8.** Examples of network coding. (a) Network $A$, (b) Network $B$,(c) Network $C$.

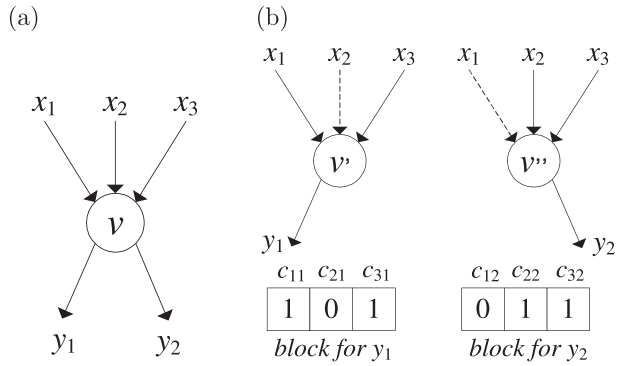



**Fig. 9.** Example: encoding at node $v$ with three incoming and two outgoing links. (a) Merging node $v$, (b) Blocks on outgoing links.

There exist a number of both deterministic and probabilistic approaches to tackling this problem [5,9,21,23,28]. A population-based stochastic search, the evolutionary algorithm solving the network coding (i.e., the network coding genetic algorithm (NCGA) [21]) was shown to be an effective solution compared to other conventional approaches. Yet, NCGA requires a large number of network evaluations to find an acceptable coding topology, and such evaluations significantly increase the time complexity when network structures are enlarged. In this paper, we combine the NCGA with our proposed on-demand evaluation strategy to improve the efficiency of the algorithm.

A more detailed problem formulation of network coding can be found in Appendix B and other references [1,21]. Hence, we will briefly describe the individual representation (i.e., encoding mechanism) of NCGA. A network is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{L}$ is the set of links. Let us consider each merging node $v \in \mathcal{V}$, where $\mathcal{L}_{in}^{(v)}$ and $\mathcal{L}_{out}^{(v)}$ are the sets of $k$ incoming links and $l$ outgoing links of node $v$, respectively (with $k = \left|\mathcal{L}_{in}^{(v)}\right| \geqslant 2$ and $l = \left|\mathcal{L}_{out}^{(v)}\right| \geqslant 1$). Then, the NCGA employs a binary individual representation, in which a chromosome $X$ is encoded as $X = \left\{c_{ij}^{(v)} \middle| \forall v \in \mathcal{V}', \forall i \in \{1,\ldots,k\},\ \forall j \in \{1,\ldots,l\}, c_{ij}^{(v)} \in \{0,1\}\right\}$. Each variable $c_{ij}^{(v)}$ (i.e., each gene) takes the value of '1' (i.e., an allele) if the data from the $i$th incoming link contributes to the coded data on the $j$th outgoing link (indicating an active link state), and '0' otherwise. Fig. 9(a) and (b) show an example of how to encode a node $v$ with three incoming links and two outgoing links into a chromosome of the NCGA.

We give an example of the genotypes for the networks in Fig. 8. Since only the outgoing links from the nodes that have multiple incoming links ($\geqslant 2$) need encoding, we can obtain the chromosomes as follows:

(a) Network A: `11`
Only the link $h \to w$ needs to be encoded. The chromosome denotes a coding event such that the data $a$ and $b$ received from two incoming links are combined at that link.

(b) Network B: `10 | 01 | 01 | 10`
There are four outgoing links that need to be encoded: two links $h \to w$ (`10|01`), one link $w \to t_1$ (`01`), and one link $w \to t_2$ (`10`). The chromosome represents no coding event; there is no combination of data from multiple incoming links.

(c) Network C: `11 | 10`
There are two outgoing links that need to be encoded: one link $h \to w$ (`11`), and one link $u \to v$ (`10`). The chromosome encodes a coding event that combines the received data $a$ and $b$ (received from the nodes $x$ and $y$) at the link $h \to w$.

Note that the other edges in the networks do not need to be encoded in the chromosomes because either they are incoming links, or they are outgoing links of the nodes with only one incoming link.

As seen from Fig. 9(b), each block of variables $c_{ij}^{(v)}$ for each $j$th outgoing link of a merging node $v$ can naturally be considered as a building block (i.e., a linkage group) in the NCGA. Because the structures of these blocks must be preserved, variation operators, such as crossover and mutation operators, are utilized in a block-wise fashion. Then, we can employ straightforwardly the aforementioned structure as a topology of linkage groups for the NCGA. Thus, the joint entropy measurement formula for evolutionary populations (see (3)) can be easily applied to NCGA without any significant changes to the algorithm.

### 4.3. Experimental configuration

In this section, we test the combination of NCGA with ODES on two sets of network topologies. The first set is comprised of several networks that are synthetically constructed by concatenating a number of copies of the network $B$ such that the

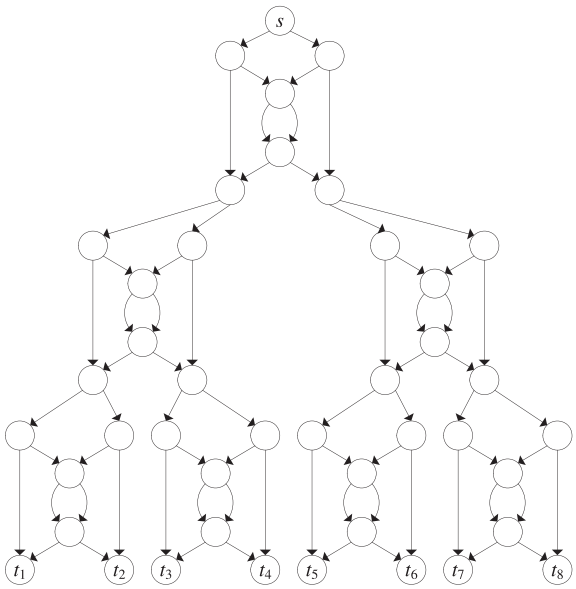



**Fig. 10.** Test network *B*-II consisting of 7 copies of network *B*.

**Table 7**
Details of the synthetic test networks.

| Test network | Number of *B* Copies | Number of blocks (problem size) | Space size (scale: $\log_{10}$) |
|---|---|---|---|
| *B*-I | 3 | 16 | 9.63 |
| *B*-II | 7 | 40 | 24.08 |
| *B*-III | 15 | 88 | 52.98 |
| *B*-IV | 31 | 184 | 110.78 |

sinks of the upper networks are the sources of the succeeding lower networks. Visually, a synthetic network created in this way would take the form of a full binary tree; Fig. 10 shows an example of how we built a synthetic network with seven copies of network *B*. We perform the experiment on four networks: *B*-I, *B*-II, *B*-III, and *B*-IV which comprise 3, 7, 15, and 31 copies of *B*, respectively. For all four cases, the multicast rate '2' can be achievable without the need of coding at any of the network nodes – i.e., the optimal number of coding links is '0' [3]. Details on the search space of this test set are described in Table 7 [3]. The second test set comprises several topologies obtained from ISP 1755 (Ebone with multicast rates of

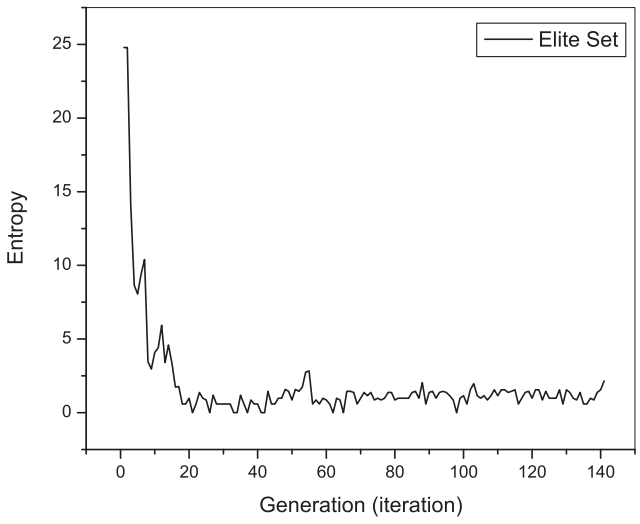



**Fig. 11.** Entropy reduction in an NCGA solving the *B*-I network.

'2', '3', and '4') and ISP 3967 (Exodus with multicast rates of '2' and '3') of the Rocketfuel Project [45]. In these test sets, the required multicast rates can also be achievable without network coding.

Next, we employ a modified version of the bisection method to measure the performance of both the NCGA and the NCGA combined with ODES (referred to here as eNCGA). Each algorithm has to go through 100 runs of the bisection method on all the test cases to collect statistical data about the minimal numbers of fitness evaluations required to reach the global optimum of '0' as well as the corresponding population sizes. (Note: we only perform 30 runs of the bisection method with the case of the *B*-IV network due to its extreme evaluation time.) Tournament selection with replacement is employed to randomly choose 5% of the current population size, *N*, each time, and the fittest individual is then selected to populate the mating pool until its size reaches *N*. Block-wise uniform crossover and mutation probabilities are set to 0.8 and 0.012, respectively (as in the original NCGA paper [21]). Generational replacement renews all the population with newly generated offspring while elitism ensures the existence of the best individual in the next generation. In particular, for the cases of Ebone and Exodus topologies, we also examine the selection with a fixed tournament size of 100 as in the original NCGA report [21] because this setting has yielded better results for both algorithms on Ebone and Exodus. All the results are averaged over 100 runs of bisections (30 runs for the case of *B*-IV).

We investigate the entropy tendency of the evolutionary population in the original NCGA to determine the suitable starting point of ODES. Differenting from BOA, where the entropy value gradually decreases until reaching '0' value, the entropy of the elite set in NCGA drops dramatically just after a few generations when compared with its initial value (see Fig. 11).

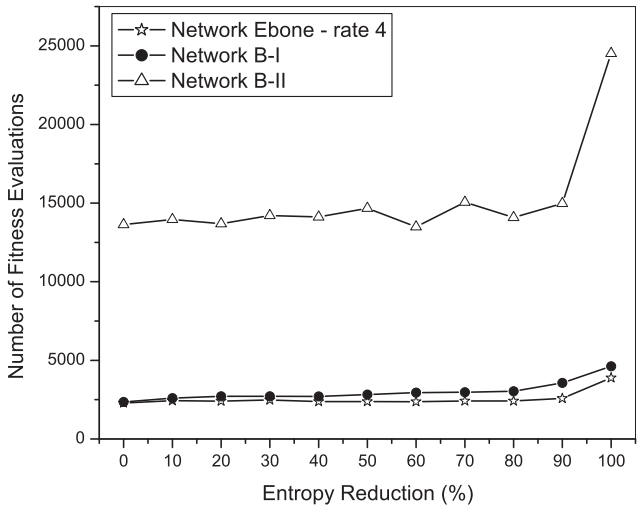



**Fig. 12.** eNCGA with different starting points of ODES.

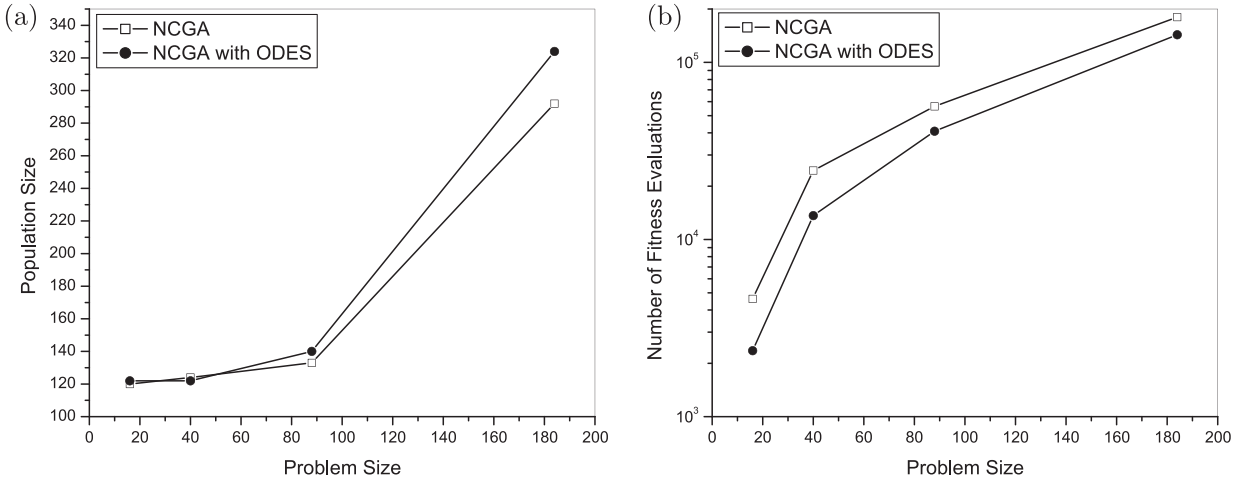



**Fig. 13.** Population size and number of evaluations required by NCGA and NCGA combined with ODES for solving synthetic networks. (a) Population size comparison, (b) Fitness evaluations comparison.

After that, the entropy measurement fluctuates around some small values. Therefore, we do not need the ODES starting point parameter $\eta$ in this experiment; we can start ODES right after the first generation.

### 4.4. Results and Discussion

Fig. 12 displays the performance of eNCGA with respect to different starting points of ODES. Note that finding the optimal coding topologies is a constrained optimization problem, in which the feasible search space is extremely narrow and complicated. In addition, all the infeasible solutions receive an infinite fitness value as a penalty. These reasons explain why the elite set in NCGA had a dramatic decrease after a few generations. Therefore, whether we started ODES at the beginning or waited until a 90% entropy reduction had been achieved did not significantly affect the performance. In all the following experiments, we applied ODES after the first generation.

Fig. 13(a) and (b) demonstrate the effects of ODES when applied to NCGA in order to find the optimal coding topologies for the set of synthetic networks *B*-I, *B*-II, *B*-III, and *B*-IV. The results showed that the eNCGA required larger population sizes than the original NCGA, but the gaps were shown to be insignificant (see Table 8(b)). Therefore, while there were no considerable differences between the population sizing requirements, ODES accelerated the search process of NCGA finding the optimal solution in terms of fewer costly network evaluations. More specifically, ODES helped NCGA recognize which coding

**Table 8**
Statistical data of performance of NCGA and eNCGA on synthetic networks.

| Size | 16 | 40 | 88 | 184 |
|---|---|---|---|---|
| *(a) Number of fitness evaluations statistical data* | | | | |
| NCGA | 4625 | 24,523 | 56,435 | 180,050 |
| ($\sigma$) | (5182.6) | (13281.1) | (17609.7) | (39944.8) |
| eNCGA | 2356 | 13,636 | 40,848 | 143,234 |
| ($\sigma$) | (1337.2) | (6679.7) | (12456.1) | (33703.4) |
| *p*-value | Statistical *t*-test: (NCGA – NCGA with ODES) | | | |
| | 4.45E−5[†] | 2.25E−10[†] | 1.01E−9[†] | 2.74E−4[†] |
| *(b) Population sizes statistical data* | | | | |
| NCGA | 120 | 124 | 133 | 292.6 |
| ($\sigma$) | (15.0) | (18.2) | (31.6) | (65.0) |
| eNCGA | 122 | 122 | 140 | 324 |
| ($\sigma$) | (16.2) | (16.5) | (40.3) | (147.7) |
| *p*-value | Statistical *t*-test: (NCGA – NCGA with ODES) | | | |
| | 0.29996 | 0.5191 | 0.19743 | 0.27552 |

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

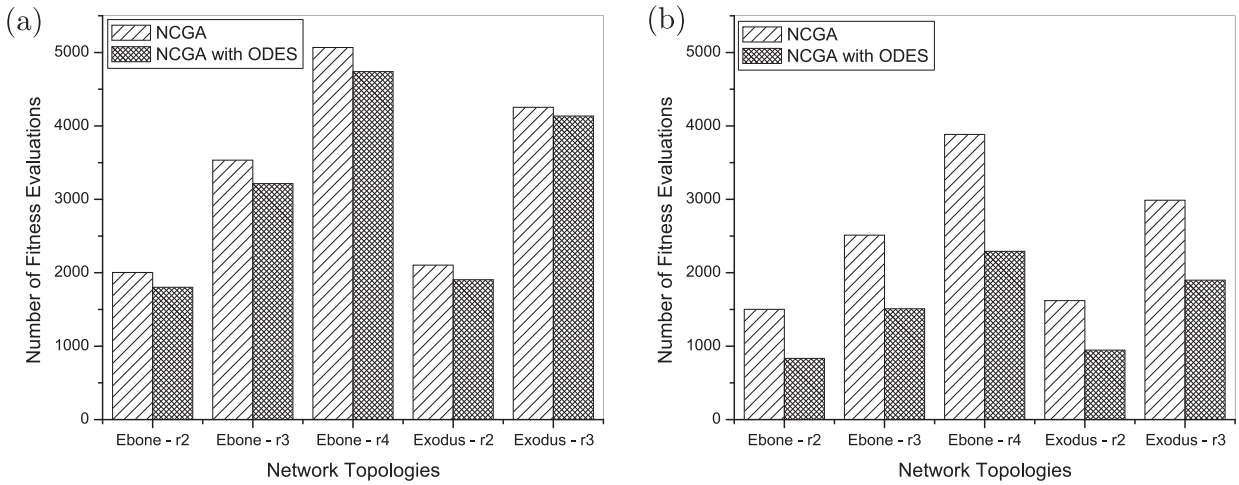



**Fig. 14.** Performance of NCGA and NCGA combined with ODES for solving real network topologies. Ebone and Exodus are the names of two ISPs. The notations of r2, r3, and r4 denote multicast rates of '2', '3', and '4', respectively. (a) Adaptive tournament size = 5% population size, (b) Fixed tournament size = 100.

**Table 9**
Statistical data of the performance of NCGA and eNCGA on ISP networks.

| Name | Ebone-r2 | Ebone-r3 | Ebone-r4 | Exodus-r2 | Exodus-r3 |
|---|---|---|---|---|---|
| *(a) Number of fitness evaluation statistical data* | | | | | |
| NCGA | 1501 | 2513 | 3884 | 1621 | 2987 |
| ($\sigma$) | (229.0) | (413.6) | (1465.0) | (416.3) | (415.2) |
| eNCGA | 834 | 1510 | 2292 | 946 | 1899 |
| ($\sigma$) | (143.6) | (230.1) | (466.1) | (163.2) | (212.6) |
| *p*-value | Statistical *t*-test: (NCGA – NCGA with ODES) | | | | |
| | 1.17E−45† | 8.95E−37† | 1.77E−17† | 9.46E−29† | 1.30E−40† |
| *(b) Population sizes statistical data* | | | | | |
| NCGA | 116 | 118 | 121 | 116 | 117 |
| ($\sigma$) | (13.5) | (13.9) | (20.7) | (13.2) | (14.8) |
| eNCGA | 114 | 118 | 119 | 117 | 116 |
| ($\sigma$) | (11.5) | (14.6) | (16.0) | (14.3) | (12.6) |
| *p*-value | Statistical *t*-test: (NCGA – NCGA with ODES) | | | | |
| | 0.43051 | 0.8235 | 0.49036 | 0.75163 | 0.59553 |

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

topologies share the same building blocks and which ones have different characteristics with the elite network configurations. Then, *similar* individuals (i.e., ones with the same building blocks as the elite set) that cause reductions in the entropy measurement could be selected without evaluation. On the other hand, *unsimilar* individuals (i.e., those with different building blocks than the elite set), which increase the entropy value, had to be evaluated by the actual function. More interestingly, as seen in Fig. 13(b), the eNCGA did not lose the original scalability of NCGA when more networks *B*s were concatenated. Table 8(a) verified that the efficiency of NCGA was significantly improved due to the incorporation of ODES.

Fig. 14(a) presents the experimental results from both the NCGA and the NCGA with ODES (eNCGA) on five different network topologies obtained from the Rocketfuel project [45]. Again, ODES appeared to have positive effects on the performance of NCGA in terms of reducing the number of fitness evaluations required to obtain the optimal solution (i.e., a non-coding topology). However, the overall improvement was not considerable, specifically in the cases of experiments on Ebone-r4 and Exodus-r3 topologies.

Hence, we conducted experiments for NCGA and eNCGA on this test set of ISP network topologies with a different parameter setting – one in which the tournament size of the selection operator was fixed at the value '100' (which is relatively large compared to the population-sizing requirements). As shown in Fig. 14(b), this setting produced better performances in both algorithms. In addition, this experiment verified that ODES could have significant improvements on the efficiency of NCGA via statistical tests (see Table 9(a)). For all five cases, the eNCGA required much fewer evaluations (of the coding topology) than the original NCGA did. Furthermore, while ODES substantially accelerated the search process, it did not impose a larger population-sizing requirement on the NCGA in the ISP networks (see Table 9(b)).

## 5. Conclusion

In this paper, we proposed the concept of an entropy measurement for evolutionary populations and discussed its inherent characteristics. Two formulas for computing the conditional entropy and joint entropy values were then introduced for use in two different general situations. For example, the conditional entropy measurement can be employed when the information about conditional (in)dependencies among variables are available (e.g., the Bayesian optimization algorithm [36]). The joint entropy measurement, on the other hand, can be used when there exists only knowledge of the connections between variables (e.g., building blocks in Genetic Algorithms or linkage groups in Extended Compact Genetic Algorithm [14]). In addition, we conjectured that a good candidate solution should contain building blocks similar to those existing in the elite set (i.e., a group of individuals having high fitness values). Based on the aforementioned entropy measurement, the existence of a good individual in the elite set decreased its randomness while an individual with strange building blocks distorted the elite set. Thus, *familiar* individuals could be selected without evaluation, but *unfamiliar* individuals had to be evaluated to determine their actual fitness values. To provide evidence for its effectiveness, we applied this on-demand evaluation strategy (ODES) to two evolutionary algorithms, a BOA solving artificial deceptive problems [36] and a GA tackling real-world network coding topologies [21].

Generally, the eBOA (BOA coupled with ODES) required slightly larger population sizes than the original algorithm, but it resulted in a significant reduction in the number of fitness evaluations. Thus, the performance of BOA was improved because the time necessary for costly fitness evaluations took a large proportion of time complexity required to solve real-world applications. Moreover, the scalability of BOA was preserved as the problem sizes grew, even in the extreme cases with complicated structures such as overlapped trap functions.

Similarly, when compared to the original NCGA, the eNCGA (NCGA coupled with ODES) required fewer network evaluations in order to find the optimal coding topologies. Its superior ability of recognizing promising topologies without

evaluation proved to be a valuable asset when the time consumed for calculating fitness values increased considerably as the network sizes were expanded. Furthermore, as the statistical tests illustrated, the eNCGA did not demand a larger population-sizing requirement than the NCGA. Thus, ODES can be applied to the NCGA without any significant changes to the original search algorithm.

A viable future work on this subject would be to study the incorporation of ODES with other types of evolutionary algorithms, such as ECGA [14] or hBOA [35]. EDAs can be easily coupled with our ODES due to their explicit models of interactions between variables. In addition, applying ODES to EAs in order to solve other challenging real-world problems should to be studied thoroughly. There exists a large legacy of source codes implemented in traditional genetic algorithms without having any straightforward model of the variable interactions; yet, most of time, complicated real-world problems come with domain-specific knowledge that could be useful for entropy computations. Thus, future studies in these directions would help elucidate a better understanding of the efficacy and efficiency of ODES in different contexts.

Furthermore, while ODES accelerated the original algorithms, it also introduced three important parameters: the starting point of ODES, the interval of the sporadic evaluations, and the size of the elite set. Experiments showed that these parameters must be set properly in order to achieve the optimal results. Our initial investigation suggested that the appropriate values for the parameters might be directly correlated with the tendencies of the entropy measurement during the optimization process. Ideally, a better estimation model, which should be more accurate than our simple assignment method, would be able to eliminate the need for sporadic full evaluations. Hence, future work will seek to develop an adaptive mechanism to determine the starting point of ODES in order to help users set various parameters. In addition, upcoming research will determine the ideal size of the elite set.

In conclusion, although further extensive work must be conducted, this paper has introduced and confirmed the positive effects of our ODES on reducing the number of costly fitness evaluations for different types of evolutionary algorithms. Furthermore, the entropy measurement (i.e., the randomness of evolutionary populations), with its inherent characteristics and wide applicability, promises to motivate more fascinating research into designing efficiency enhancement techniques.

## Acknowledgment

This research was supported by the Ministry of Knowledge Economy (MKE) of Korea, under ITRC NIPA-2011-(C1090-1121-0008).

## Appendix A. Bayesian optimization algorithm

The Bayesian optimization algorithm (BOA) [36] is a multivariate estimation of distribution algorithm, which can capture regularities existing in data. Discarding traditional variation operators (e.g., crossover, mutation), BOA employs Bayesian networks to model high-order interactions between variables and subsequently samples the constructed model to generate new candidate solutions having similar properties with the learned data. The original framework of BOA is outlined in Algorithm 1. In the next section, we will briefly describe the model building and the model sampling phases, which differentiate BOA from the traditional genetic algorithm.

### A.1. Model building

Constructing a Bayesian network from a given set of selected candidate solutions consists of two phases: learning the structure, and learning the parameters (i.e., the conditional probabilities). Learning the structure requires a scoring metric and a search procedure.

A scoring metric measures the fitness of the structure of a network – i.e., how well the network models the conditional (in)dependencies between variables in a given data set. The Bayesian Dirichlet (BD) metric [16] can be employed to compute the fitness of a network $B$ modeling a data set $D$ of size $N$ as follows:

$$BD(B) = p(B) \prod_{i=1}^{n} \prod_{\pi_i} \frac{(m'(\pi_i))!}{(m'(\pi_i) + m(\pi_i))!} \cdot \prod_{x_i} \frac{(m'(x_i, \pi_i) + m(x_i, \pi_i))!}{m'(x_i, \pi_i)!}$$

where $p(B)$ is the prior probability of network $B$; the product over $x_i$ runs over all instances of $X_i$ and the product over $\pi_i$ runs over all possible configurations of the parents $\Pi_i$ of $X_i$; $m(\pi_i)$ is the number of individuals in $D$ with $\Pi_i$ set to $\pi_i$; $m(x_i, \pi_i)$ is the number of individuals in $D$ with $X_i = x_i$ and $\Pi_i = \pi_i$. Prior information about $m(\pi_i)$ and $m(x_i, \pi_i)$ can be described by $m'(\pi_i)$ and $m'(x_i, \pi_i)$, respectively. The implementation of BOA in this paper employs the K2 metric [16] that sets $m'(x_i, \pi_i) = 1$, $m'(\pi_i) = \sum_{x_i} m'(x_i, \pi_i)$, and $p(B) = 1$ as no prior information is known. Details on different scoring metrics, such as the minimum description length metric, can be found in other references [35].

A search procedure traverses the search space of all possible networks to find the best network with respect to a scoring metric. However, obtaining the optimal Bayesian network is a difficult problem without any polynomial-time algorithm. Fortunately, a simple greedy algorithm is adequate for finding an acceptable network in BOA [35]. First, an empty network with no edges is assumed; the Bayesian network is then constructed stage by stage. At each generation, the algorithm considers all the possible elementary graph operations belonging to three categories: adding a new edge, removing an edge, or revers-

ing an existing edge. Due to the greedy manner, the operation that can improve the quality of the network the most, with respect to the scoring metric, is conducted. During the search process, the Bayesian network must represent an appropriate acyclic structure without exceeding the upper-bound order of interactions. This process is iterated until no further improvement in the fitness of the network can be made. The resultant network structure describes the relationships (i.e., the conditional (in)dependencies) between the variables. Then, learning the parameters for the Bayesian network can be done by simply iterating all the candidates in the selected set to compute the relative frequencies of all the possible relationships [35].

### A.2. Model sampling

Once we have established a Bayesian network modeled on a set of selected individuals, we can sample this network to obtain new candidate solutions with similar characteristics of the learned data. For convenience, the variables are sorted according to the ancestral order described in the network – the parent nodes always precede the child nodes [35]. This ordering ensures that the values of all the parents of each variable are generated first, which can then be used to look up the distribution of the variable itself in its corresponding CPT (to sample its value). After this process is complete, all the variables in a new candidate solution are specified.

## Appendix B. Network coding

A network is an acyclic directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{L}$ is the set of links. Each link $(i,j)$ – a lossless point-to-point channel from node $i$ to node $j$ – is assumed to have unit capacity. A link with a higher capacity can then be represented as multiple unit capacity links. Since only integer flows are allowed, a link has either a unit-rate flow or no flow at all. This paper considers the single multicast scenario, where a source $s \in \mathcal{V}$ transmits data at rate $r$ to a set of sinks $T = \{t_1, t_2, \ldots, t_{|T|}\} \subset \mathcal{V}$. Rate $r$ is achieved when the current coding topology allows all $|T|$ sinks to receive the sent data.

Given a multicast rate $r$, we need to find the minimal set of nodes where coding is necessary. Note that a node with a single incoming link does not need coding ability. On the other hand, nodes having multiple incoming links, termed 'merging nodes', are potential coding nodes. If the data flow on an outgoing link of a merging node is the contribution of only one incoming link, no coding should be required on that outgoing link. Hence, let $\mathcal{V}'$ and $\mathcal{L}'$ be a set of all merging nodes and a set of their outgoing links, respectively. Then, coding ability should be implemented at a node $v \in \mathcal{V}'$ if coding is required on at least one of the outgoing links of node $v$. Consequently, the number of coding links is an appropriate measure of coding cost [21,23]. In this sense, our objective is to minimize the number of coding links while ensuring the multicast rate $r$ from a single source to a set of sinks. The requirement can be formalized as follows [3]:

$$\text{Minimize} \sum_{(i,j)\in\mathcal{L}'_{out}} z_{i,j} \tag{B.1a}$$

$$\text{Subject to} \sum_{\{j|(i,j)\in\mathcal{L}\}} g_{i,j}^{(t_l)} = \sum_{\{j|(j,i)\in\mathcal{L}\}} g_{j,i}^{(t_l)}, \quad \forall i \in \mathcal{V} - \{s, t_l\}, \quad \forall t_l \in T \tag{B.1b}$$

$$\sum_{\{j|(s,j)\in\mathcal{L}\}} g_{s,j}^{(t_l)} = r, \quad \forall t_l \in T \tag{B.1c}$$

$$\sum_{\{j|(j,t_l)\in\mathcal{L}\}} g_{j,t_l}^{(t_l)} = r, \quad \forall t_l \in T \tag{B.1d}$$

$$C_{i,j} \geqslant g_{i,j}^{(t_l)} \geqslant 0, \quad \forall (i,j) \in \mathcal{L}, \ t_l \in T \tag{B.1e}$$

where $z_{i,j} \in \{0,1\}$ represents whether the outgoing link $(i,j)$ from node $i$ to node $j$ needs the coding operation or not; $g_{i,j}^{(t_l)}$ is the amount of flows to the sink node $t_l$ through the link $(i,j)$. The flow constraints ensure that in the flow decomposition of the sink $t_l$, the flow coming into any intermediate node is equal to the flow coming out of the node (see (B.1b)). They also ensure that the flows coming from the source $s$ (see (B.1c)) and into the sink $t_l$ (see (B.1d)) match the rate $r$ (see Flow Decomposition Properties [1]). A capacity constraint for each link is established by (B.1e).

The task of specifying the minimal set of coding nodes has been known as an NP-hard problem [23]. There exist several deterministic approaches to tackle this problem, which can be found in [5,9,23,28]. Additionally, details regarding the network coding genetic algorithm (NCGA) can be found in [3,21].

## References

[1] R. Ahlswede, N. Cai, S.-Y.R. Li, R.W. Yeung, Network information flow, IEEE Transactions on Information Theory 46 (4) (2000) 1204–1216.
[2] C.W. Ahn, J. An, J.-C. Yoo, Estimation of particle swarm distribution algorithms: combining the benefits of pso and edas. Information Sciences (in press).
[3] C.W. Ahn, M. Kim, Self-adaptive evolutionary network coding algorithm: a constraint handling approach, in: Third International Workshop on Advanced Computational Intelligence (IWACI), 2010, 2010, pp. 238–243.
[4] H.-G. Beyer, H.-P. Schwefel, Evolution strategies – a comprehensive introduction, Natural Computing: An International Journal 1 (1) (2002) 3–52.
[5] K. Bhattad, N. Ratnakar, R. Koetter, K. Narayanan, Minimal network coding for multicast, in: Proceedings of the International Symposium on Information Theory, 2005, ISIT 2005, 2005, pp. 1730–1734.
[6] E. Cantu-Paz, Efficient and Accurate Parallel Genetic Algorithms, Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[7] M. Dorigo, T. Stützle, Ant Colony Optimization, Bradford Company, Scituate, MA, USA, 2004.
[8] L.J. Fogel, Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming, John Wiley & Sons, Inc., New York, NY, USA, 1999.
[9] C. Fragouli, E. Soljanin, Information flow decomposition for network coding, IEEE Transactions on Information Theory 52 (3) (2006) 829–848.
[10] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[11] D.E. Goldberg, Using time efficiently: Genetic-evolutionary algorithms and the continuation problem, Proceedings of the Genetic and Evolutionary Computation Conference, vol. 1, Morgan Kaufmann, Orlando, Florida, USA, 1999, pp. 212–219.
[12] D.E. Goldberg, S. Voessner, Optimizing global–local search hybrids, in: Proceeding of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann, 1999, pp. 220–228.
[13] J.J. Grefenstette, J.M. Fitzpatrick, Genetic search with approximate function evaluation, in: Proceedings of the 1st International Conference on Genetic Algorithms, L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1985, pp. 112–120.
[14] G.R. Harik, F.G. Lobo, K. Sastry, Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ecga), in: Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, Springer-Verlag, 2006, pp. 39–61.
[15] M. Hauschild, M. Pelikan, K. Sastry, C. Lima, Analyzing probabilistic models in hierarchical boa, IEEE Transactions on Evolutionary Computation 13 (6) (2009) 1199–1217.
[16] D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: the combination of knowledge and statistical data, Machine Learning 20 (3) (1995) 197–243.
[17] J.H. Holland, Adaptation in Natural and Artificial Systems, MIT Press, Cambridge, MA, USA, 1992.
[18] Y. Jin, A comprehensive survey of fitness approximation in evolutionary computation, Soft Computing 9 (1) (2005) 3–12.
[19] C.G. Johnson, J.J.R. Cardalda, Introduction: genetic algorithms in visual art and music, Leonardo 35 (2) (2002) 175–184.
[20] J. Kennedy, R.C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.
[21] M. Kim, M. Médard, V. Aggarwal, U.-M. O'Reilly, W. Kim, C.W. Ahn, M. Effros, Evolutionary approaches to minimizing network coding resources, in: INFOCOM, 2007, pp. 1991–1999.
[22] J.R. Koza, Genetic Programming: On The Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992.
[23] M. Langberg, A. Sprintson, J. Bruck, The encoding complexity of network coding, IEEE Transactions on Information Theory 52 (6) (2006) 2386–2397.
[24] P. Larrañaga, J.A. Lozano (Eds.), Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Boston, MA, 2002.
[25] C.F. Lima, M. Pelikan, F.G. Lobo, D.E. Goldberg, Loopy substructural local search for the Bayesian optimization algorithm, in: SLS 2009, LNCS, vol. 5752, 2009, pp. 61–75.
[26] F. Liu, J. Tsai, R. Chen, S. Chen, S. Shih, Fmga: finding motifs by genetic algorithm, in: Proceedings of the Fourth IEEE Symposium on Bioinformatics and Bioengineering 2004, 2004, pp. 459–466.
[27] J. Lohn, S. Colombano, A circuit representation technique for automated circuit design, IEEE Transactions on Evolutionary Computation 3 (3) (1999) 205–219.
[28] D. Lun, N. Ratnakar, M. Medard, R. Koetter, D. Karger, T. Ho, E. Ahmed, F. Zhao, Minimum-cost multicast over coded packet networks, IEEE Transactions on Information Theory 52 (6) (2006) 2608–2623.
[29] H.N. Luong, H.T.T. Nguyen, C.W. Ahn, Entropy-based evaluation relaxation strategy for Bayesian optimization algorithm, in: Proceedings of the 23rd International Conference on Industrial Engineering and Other applications of Applied Intelligent Systems (IEA/AIE 2010), LNAI, vol. 6097, Springer-Verlag, 2010, pp. 126–135.
[30] H.N. Luong, H.T.T. Nguyen, C.W. Ahn, Entropy-based substructural local search for the Bayesian optimization algorithm, in: GECCO'10: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA, 2010, pp. 335–342.
[31] E. Mezura-Montes, M.E. Miranda-Varela, R. del Carmen Gmez-Ramn, Differential evolution in constrained numerical optimization: an empirical study, Information Sciences 180 (22) (2010) 4223–4262.
[32] M. Mitchell, An Introduction to Genetic Algorithms, MIT Press, Cambridge, MA, USA, 1996.
[33] J. Ocenasek, Entropy-based convergence measurement in discrete estimation of distribution algorithms, in: Studies in Fuzziness and Soft Computing 192: Towards a New Evolutionary Computation, Springer-Verlag, 2006, pp. 39–50.
[34] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.
[35] M. Pelikan, Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms, Springer-Verlag, 2005.
[36] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, Boa: the Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999), 1999, pp. 525–532.
[37] M. Pelikan, K. Sastry, Fitness inheritance in the Bayesian optimization algorithm, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004), 2004, pp. 48–59.
[38] M. Pelikan, K. Sastry, D.E. Goldberg, Scalability of the Bayesian optimization algorithm, International Journal of Approximate Reasoning 31 (3) (2002) 221–258.
[39] C.A. Peña-Reyes, M. Sipper, Evolutionary computation in medicine: an overview, Artificial Intelligence in Medicine 19 (1) (2000) 1–23.
[40] M. Santini, A. Tettamanzi, Genetic programming for financial time series prediction, in: Genetic Programming, Lecture Notes in Computer Science, vol. 2038, Springer, Berlin/ Heidelberg, 2001, pp. 361–370.
[41] K. Sastry, Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL, 2001.
[42] K. Sastry, D.E. Goldberg, M. Pelikan, Don't evaluate, inherit, in: Proceedings of the Genetic and Evolutionary Computation Conference (Also IlliGAL), Morgan Kaufmann, 2001, pp. 551–558.
[43] A. Sinha, Y.-p. Chen, D. Goldberg, Designing efficient genetic and evolutionary algorithm hybrids, in: W. Hart, J. Smith, N. Krasnogor (Eds.), Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing, vol. 166, Springer, Berlin, Heidelberg, 2005, pp. 259–288.
[44] R.E. Smith, B.A. Dike, S.A. Stegmann, Fitness inheritance in genetic algorithms, in: SAC'95: Proceedings of the 1995 ACM Symposium on Applied Computing, ACM, New York, NY, USA, 1995, pp. 345–350.
[45] N. Spring, R. Mahajan, D. Wetherall, T. Anderson, Measuring isp topologies with rocketfuel, IEEE/ACM Transactions on Networking 12 (1) (2004) 2–16.
[46] R.P. Srivastava, Time continutation in genetic algorithms. Master's thesis, University of Illinois at Urbana-Champaign, General Engineering Department, Urbana, IL, 2002.
[47] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359.
[48] J. Sun, Q. Zhang, E.P. Tsang, De/eda: a new evolutionary algorithm for global optimization, Information Sciences 169 (3–4) (2005) 249–262.
[49] Y. Wang, B. Li, T. Weise, Estimation of distribution and differential evolution cooperation for large scale economic load dispatch optimization of power systems, Information Sciences 180 (12) (2010) 2405–2420.