A Hierarchical Hybrid Evolutionary Computation for Continuous Function Optimization

SAID MOHAMED SAID
SENLIN GUAN
MORIKAZU NAKAMURA
University of the Ryukyus

In this paper, we propose a hybrid master/slave approach to optimization problems on the basis of estimation of distribution algorithms (EDAs) and genetic algorithms (GAs). The master process estimates the probability distribution of the search space on the basis of the non-dependency model at each iteration and sends probability vectors to the slaves. The slaves use the vectors to generate a new initial population for their GA operations. We employ the simplest probability models and we compensate for the reduced accuracy problems by applying GAs to the solutions sampled using the simplest model. Moreover, our method can be incorporated with strategy research, and it easily can be parallelized. Lastly, we conduct experiments to verify the effectiveness of our method.

Keywords: Evolutionary computation, Genetic algorithms, estimation of distribution algorithms, parallel processing

1. INTRODUCTION

Optimization problems have been classically investigated, mainly in the fields of computer science and the operations research. Efficient algorithms have been developed for numerous real-life applications. However, modern optimization problems are quite large and complicated. Thus, new algorithms are required to solve such problems efficiently. Population-based methods have been widely adopted to solve such optimization problems, and in many cases, they yield promising results. Genetic algorithms (GAs) are well known techniques for solving optimization problems; they enable us to easily design programs and efficiently obtain high quality solutions when we set proper parameters [Holland 1972]. They have already shown effectiveness in variety of problem domains (see [Davis 1991] for examples). They became robust, easy to use and applicable in diverse areas, such as machine learning, combinatorial problems, VLSI design and numerical optimization [Salomon 1996].

Recently, several efficient optimization techniques such as ant colony optimization (ACO) [Dorigo 1996]; [J. Zhang et al. 2006] and particle swarm optimization (PSO) [Eberhart and Shi 2001]; [Hu 2004] have been developed on the basis of natural phenomena. Estimation of distribution algorithms (EDAs) were introduced as novel evolutionary computation techniques [Mühlenbein and Paaß 1996], and several researchers have investigated this interesting methodology [Larrañaga and Lozano 2002].

An EDA estimates the probability distribution of the search space in each iteration on the basis of a certain probabilistic model. The simplest model lacks dependency, that is, we assume that there is no dependency between any pair of variables. Useful algorithms such as the univariate marginal distribution algorithm (UMDA) [Mühlenbein 1998], population-based incremental learning (PBIL) [Baluja 1994], and the compact genetic algorithm (cGA) [Harik et al. 1998] have been developed for combinatorial optimization. Furthermore, algorithms such as the UMDA of the continuous domains (UMDAc) and the PBIL for continuous search spaces (PBILc) [Sebag and Ducoulombier 1998] have been proposed for continuous function optimization. A

Author's address: S. Mohamed Said, S. Guan, and M. Nakamura, Faculty of Information Engineering, University of the Ryukyus, 1 Senbaru, Nishihara, Okinawa 903-0213 JAPAN.

more complicated model for both combinatorial and continuous function optimization involves bivariate dependence, where we assume dependency between two variables, and the most complicated model involves multiple dependency ([Perikan et al. 1999]; [Bosman and Thierens 2000]; [Gallagher et al. 1999]). Although more complicated model seems to be more precise, it requires intensive computation.

[Ocenasek 2002] and [Mendiburu-Alberro 2006] proposed parallel processing schemes in order to reduce the computation time. However, further reduction in the computation time is required for large-scale complicated problems. In this paper we propose a hybrid master/slave approach on the basis of the simplest EDA and GAs. This paper is an extended version of approach we proposed in [Said and Nakamura 2010]. The master process estimates the probability distribution of the search space on the basis of the mixture model with non-dependency at each iteration, and it sends probability vectors to the slaves. The slaves use the vectors to generate a new initial population. Note that in EDAs, the vectors are used for sampling, whereas in our approach, they are used for generating the initial population. We employ a simple model of the EDA, and we compensate for the reduced accuracy by applying GAs to the solutions sampled using the simple model. Subsequently, all the slaves send the obtained solutions to the master.

Our method has two advantages. First, slave computation can be performed in parallel as an island model of evolutionary computation [Alba 2005]. Recent technological innovations in computer systems have yielded several useful platforms for parallel processing, such as PC-clusters and many-cores processors. We can easily develop parallel programs for our method. Second, our method can be easily incorporated with some strategies for controlling the diversity and concentration of searching.

We evaluate the performance of our hybrid approach by solving fifteen well-known function optimization problems. Experimental results verify the effectiveness of our method, as compared to GAs, PBIL and PSO. In addition, the experiments indicate that a strategic approach affects the searching performance.

The remainder of this paper is organized as follows. In Section 2, we briefly explains GAs, and EDAs, and we state some mathematical notations. In Section 3, we present our hybrid method, and in Section 4 we describe experiments conducted to evaluate the proposed method. Finally, in Section 5, we conclude the paper with some remarks.

2. PRELIMINARIES

2.1 Evolutionary Computation

GAs are computational models that simulate the process of genetic selection and natural elimination in biological evolution. Pioneering work in this field was conducted by Holland [Holland 1972]. As highly efficient search strategies for global optimization, GAs demonstrate favorable performance in solving combinatorial optimization problems. Figure 1 shows a pseudo code for GAs.

EDAs ([Mühlenbein and Paaß 1996]; [Larrañaga and Lozano 2002]) have a theoretical foundation in probability theory, and they do not use any recombination operators. First, EDAs generate an initial population P of solutions. Then, the following steps are iterated until the termination conditions are met. Good solutions in P are selected, and from the selected solutions, a probability distribution of the good solutions is estimated for representing the potential space. The probability distribution is used for producing new solutions by sampling based on the distribution. Figure 2 shows a pseudo code for EDAs.

2.2 Mathematical Notation

We use X_i to denote a random variable. An instance of X_i is denoted by x_i. $\mathbf{X} = (X_1, X_2, ..., X_n)$ denotes an n-dimensional random variable, and $\mathbf{x} = (x_1, x_2, ..., x_n)$ is one of its instances. If variable X_i is continuous, we denote the density function of X_i by $f(X_i)$. If all the variables in $\mathbf{X}$ are continuous, we denote the joint density function by $f(\mathbf{x})$. If X_i is discrete, the mass probability for X_i is denoted by $p(X_i)$. If all the variables in $\mathbf{X}$ are discrete, $p(\mathbf{x})$ denotes the

```
1: P ← GenerateInitialPopulation();
2: Evaluate(P);
3: while termination condition is not met do
4:     P′ ← Recombine(P);
5:     P″ ← Mutate(P′);
6:     Evaluate(P″);
7:     P ← Select(P, P″);
8: end while
9: Output the best solution found;
```

Figure 1: Pseudo code for GAs

```
1: P ← GenerateInitialPopulation();
2: Evaluate(P);
3: while termination condition is not met do
4:     P_sel ← ChooseFrom(P);
5:     p(x) ← EstimateProbabilityDistribution(P_sel);
6:     P ← SampleBasedOn(p(x));
7: end while
8: Output the best solution found;
```

Figure 2:Pseudo code for EDAs

joint probability mass.

3. HYBRID APPROACH ON THE BASIS OF MASTER-SLAVE COOPERATION

In this section, we propose a novel hybrid scheme for the EDA and GAs.

3.1 Hybrid Method

The proposed method is based on master-slave formulation, in which the master manages the set of local optimal solutions obtained by the slaves, estimates the probability distribution of the potential search space, and indirectly controls the searching of the slaves.

The master achieves this *indirect control* by specifying a probability vector for each slave to generate its initial population. The probability vector is constructed by the master according to a strategy. Although traditional GAs initiate their evolution from a uniformly distributed initial population, at some points our method starts from a strategically biased initial population generated by the probability vector. Moreover, our method is iterative, and each iteration corresponds to one execution of evolutionary computation. Thus, in our method, the slaves iterate optimization on the basis of evolutionary computation, and they send the best solution, i.e., the conversed solution, of each iteration to the master. The master receives the solutions, stores them in a database DB, and generates the probabilistic vectors by using the solutions in DB for the next iteration. Figure 3 shows the outline of the master-slave model.

The strategy can affect searching performance, and it comprises tactics such as wide-range searching, outside cluster focusing, cumulative clustering and best cluster focusing. These tactics are implemented by the slaves while searching, where the slaves initiate their evolutionary computation from the biased initial population generated by the probability vectors, as described above.

The slaves perform parallel evolutionary computation of the well-known island model in isolation, i.e., no communication occurs between the slaves. All the slaves (islands) iterate their evolutionary computations independently, and on completion, they return their local optimal solutions to the master. The master stores all the local optimal solutions sent from the slaves in its database, calculates probability vectors, and then returns a vector to each slave so that it can initiate the next evolutionary computation.

Note that the role of the master is analogous to that of an EDA because the master generates

probability vectors on the basis of solutions already obtained for the next iteration. However, our method is different from EDAs in that it incorporates the strategic searching and parallel evolutionary computation instead of simple solution sampling as in the case of traditional EDAs. Although GAs are used as searching algorithms by the slaves in our method, other algorithms are also applicable.

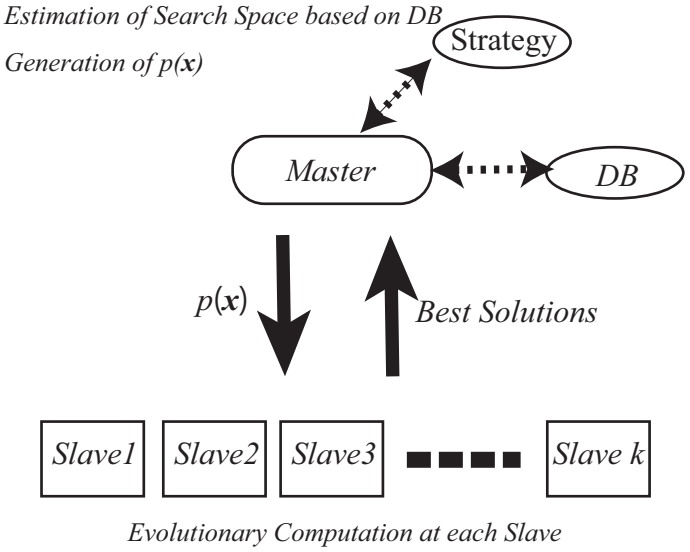

Figure 3:Hybrid Model based on Master-Slave Cooperation

Regarding the Master-Slave model, some limitations are mentioned in [Gagne and Parizeau 2003], but our present approach takes the limitations in favor of accuracy and performance(solution quality). For the Master process being the backbone of the architecture, it enables better control of probabilistic estimation for slaves initialization process, and in case of failure the slaves will assume random initialization. According to previous studies creating network of slaves (migration of chromosomes) helps to boost searching performance [Gong and Nakamura 2008], but we tried to reduce algorithm complexity as well as communication overheads by completely cutting inter-slaves communication.

Pseudo codes for the master and the slave are shown in Figures 4 and 5, respectively.

```
1: while termination condition is not met do
2:    Recieve(p(x)) from Master;
3:    P ← GenerateInitialPopulation(p(x));
4:    Evaluate(P);
5:    for i = 0... MAXGENERATION do
6:       P' ← Recombine(P);
7:       P'' ← Mutate(P');
8:       Evaluate(P'');
9:       P ← Select(P, P'');
10:   end for
11:   Send the best solution to Master;
12: end while
```

Figure 4: Pseudo code for Slaves

```
1: Initialize(DB);
2: for i = 0...k − 1 do
3:    T ← WS;
4:    p_i(x) ← GenerateProbabilityVector(T, DB);
5:    Send p_i(x) to Slave i;
6: end for
7: while termination condition is not met do
8:    ReceiveSolutionFromSlave(DB, i);
9:    EstimationOfDistribution(DB);
10:   T ← Strategy();
11:   p_i(x) ← GenerateProbabilityVector(T, DB);
12:   Send p_i(x) to Slave i;
13: end while
```

Figure 5: Pseudo code for Master

3.2 Estimation based on Mixture Model without Dependency

Several models for the estimation of distribution have been reported in the literature on EDAs [Larrañaga and Lozano 2002]. For simple approaches to efficient computation, univariate models are used. Such models assume an n-dimensional joint probability distribution function as a product of n univariate and independent probability distributions. For more precise estimation, multiple dependency models are used. These models exhibit good performance for target problems with a complicated landscape. However, their computational costs are very high.

In our approach, the EDA plays the role of the master for efficient control of searching of the slaves. More accurate estimation leads to more efficient control. We can avoid the high computation costs for the role of the master because the evolutionary computation performed by the slaves can compensate for the inaccuracy of the estimation.

For the master's EDA, we focus on the mixture model of the Gaussian distribution. The master estimates the distribution of good solutions in the search space as a set of clusters and assumes that the good solutions in each cluster are normally distributed (Gaussian distribution). This model is very simple and efficient; it is more adaptable for a complicated search space than simple univariate models. Another advantage is its suitability for parallel searching. The master can assign the search space cluster-wise. That is, clusters with high potential are assigned to the slaves.

We consider the joint Gaussian probabilistic density function

$$f_l(\mathbf{X}) = \prod_{i=1}^{n} f_l(x_i) \tag{1}$$

as the probabilistic model, where $f_l(x_i)$ is denoted by the sum of component distributions at the l-th iteration,

$$f_l(x_i) = \sum_{j=1}^{K_l} \pi_{l,j} f_l(x_i|j), \tag{2}$$

where K_l is the number of mixture components and $\pi_{l,j}$ is the coefficient of mixing of the j-th component. Because we do not assume any dependency among variables, $f_l(x_i|j) \equiv \mathcal{N}(x_i; \mu^l_{i,j}, \sigma^{2,l}_{i,j})$, that is, the univariant Gaussian density function corresponding to the j-th component.

The EDA, i.e., the role of the master process, starts with a clustering algorithm that classifies solutions sent from the slaves into K_l clusters. After clustering, the master calculates the mean

vector for each group. Let us denote the mean vector by

$$\mu_j^l = (\mu_{1,j}^l, \mu_{2,j}^l, ..., \mu_{n,j}^l), \tag{3}$$

$$\mu_{i,j}^l = \frac{1}{m_j^l} \sum_{k=1}^{m_j^l} x_{i,j}^{l,k}, \tag{4}$$

where m_j^l is the total number of solutions in the j-th cluster, and $x_{i,j}^{l,k}$ is the k-th component of the i-th solution in the j-th cluster at the l-th iteration.

For discrete optimization problems, we can have similar definitions. The joint probability mass is represented by a product of independent univariate distributions

$$p_I(\mathbf{X}) = \prod_{i=1}^{n} p_l(x_i), \tag{5}$$

where $p_l(x_i)$ is denoted by the sum of component distributions at the l-th iteration

$$p_I(x_i) = \sum_{j=1}^{K_l} \pi_{l,j} p_l(x_i|j), \tag{6}$$

where K_l is the number of mixture components, $\pi_{l.j}$ is the coefficient of mixing of the j-th component, and $p_l(x_i|j) \equiv \frac{1}{m_j^l} \sum_{k=1}^{m_j^l} \delta_j(X_i = x_{i,j}^{l,k})$ is the univariant marginal frequency of the j-th component.

3.3 Strategic Search Control

As described above, the control is achieved through the probability vector for indicating the biased area of the initial population. A slave generates the initial population by sampling based on the Gaussian distribution with variance 1 and the probability vector as the mean vector.

Search control is achieved via the following population initialization tactics.

—**Wide-Ranging Searching (WRS)** This tactic involves behavior similar to that of the traditional GA. Slaves generate their initial populations uniformly and randomly. The objective is to collect good solutions from a wider area of the search space. This tactic is used during the early stages of searching. For this purpose, the master sends the probability vectors generated randomly with the message "UNIFORM" to the slaves.

—**Outside Clusters Searching (OCS)** For robust searching, we need to carefully search the unsearched area. For this purpose, this tactic starts by clustering the local optimal solutions obtained in the previous iterations. Then, areas that are not in the same cluster are assigned by the master to the slaves. Let $\mathbf{p}_i = (p_{i,1}, p_{i,2}, ..., p_{i,n})$ be the mean vector of the i-th cluster. The master generates their complementary vectors (for 0-1 integer optimization, $\bar{\mathbf{p}}_i = (1 - p_{i,1}, 1 - p_{i,2}, ..., 1 - p_{i,n})$) and sends them to slave i.

—**Cumulative Clustering (CC)** For more detailed searching, we try to explore the clusters. That is, we gradually increase the number of good solutions in the clusters. At this stage, the mean vector of each cluster, $\mathbf{p}_i = (p_{i,1}, p_{i,2}, ..., p_{i,n})$, is sent to slave i to initialize the population for the next iteration.

—**Best Cluster Focusing (BCF)** Having explored the wider area of the search space, we need to concentrate only on the area with better feedback. To do so, the master determines the cluster that yields good results, as compared to other clusters and assigns its vector $\mathbf{p}_i = (p_{i,1}, p_{i,2}, ..., p_{i,n})$ to all the slaves for them to focus their searching.

By combining the tactics described above, various strategies can be constructed. Table I lists some strategies. Strategy 0 includes only WRS. The hybrid approach with Strategy 0 behaves like the isolated island model of GAs, that is, an independent GA processes search solutions. Strategy 1 comprises two tactics: WRS and BCF. That is, the diversity of searching can be implemented by WRS, and the concentration, by BCF. Strategy 2 comprises four tactics. The first phase, WRC, implements the diversity of searching, and the second phase tries to explore areas where less solutions were found in the first phase. The third phase implements the concentration for multiple clusters. We focus on the best cluster in the final phase. For experimental analysis, we also employed Strategies 3, 4, and 5. These strategies were employed to compare the influence of different combinations of tactics while exploring the search space. In this paper, we consider only a static strategy, even though we are investigating a dynamic one. In a dynamic strategy, the hybrid method can adaptively employ tactics.

Table I: Examples of Strategies

Strategy 0		Strategy 1		Strategy 2	
Phase	Tactic	Phase	Tactic	Phase	Tactic
1	WRS	1	WRS	1	WRS
				2	OCS
		2	BCF	3	CC
				4	BCF

4. EXPERIMENTAL EVALUATION

In order to evaluate the effectiveness of our strategic approach, we implemented our method using the C programming language with the POSIX Threads (Pthreads) library on a Mac Pro (Intel Dual Quad Core, i.e., 8 cores in total; 3.0 GHz; 12 GB RAM). The algorithm can also be implemented using OpenMP as a shared memory parallelism. However in distributed memory environment of multicomputer system, message passing library (MPI) might as well be used for parallelization. All the libraries are available in C/C++ and Fortran programming languages. In this paper, we have solved some well-known benchmark continuous function optimization problems.

4.1 Target Problems

In this experiment, we evaluated our approach by solving fifteen continuous function optimization problems. Table II lists the functions, some of them are treated by [J. Zhang et al. 2006]. F_1 to F_5 are unimodal functions, whereas F_6 to F_{15} are multimodal functions. In the table, D denotes the dimension and the values within brackets indicate search space; its value varies among functions. We set D as 30 for $F_1 - F_4$ and $F_6 - F_8$, 10 for F_5 and F_9, 2 for F_{12}, F_{13} and F_{15}, 5 for F_{10}, F_{11}, and 40 for F_{14}. These values of D and search space were used to suit the pre-determined standard PSO parameters for fair comparison.

4.2 Effects of Strategic Approaches

Table III lists five strategies used in the experiment: three strategies, Strategies 0, 1, and 2, from Table I, and two additional strategies, Strategies 4 and 5. The tactic (TC) is one of WRS, OCS, CC, and BCF.

Strategy 0 comprises no tactics, that is, the slaves generate the initial population uniformly and randomly at every iteration. Strategy 1 uses only two tactics, WRS in the first phase and BCF in the second phase, whereas Strategy 2 employs four tactics in the four phases, WRS,

Table II: Test Functions

Test Function	Range
$F_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$
$F_2(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j^2)^2$	$[-100, 100]^D$
$F_3(x) = \max_i(\|x_i\|, 1 \leq i \leq D)$	$[-100, 100]^D$
$F_4(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	$[-30, 30]^D$
$F_5(x) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$
$F_6(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-100, 100]^D$
$F_7(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \Pi_{i=1}^{D} \cos \frac{x_i}{\sqrt{i}} + 1$	$[-600, 600]^D$
$F_8(x) = -20\exp(-0.2\sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2})$ $- \exp(\frac{1}{D} \sum_{i=1}^{D} \cos 2\pi x_i) + 20 + \exp(1)$	$[-32, 32]^D$
$F_9(x) = -\sum_{i=1}^{D} \sin x_i [\sin((i+1)\frac{x_i^2}{\pi})]^{20}$	$[0, \pi]^D$
$F_{10} = \sum_{k=1}^{D} [\sum_{i=1}^{D} (i^k + \beta)((x_i/i)^k - 1)]^2$	$[-5, 5]^D$
$F_{11} = [\frac{1}{6.931)} - \frac{x_0 x_1}{x_2 x_3}]^2$	$[12, 60]^D$
$F_{12} = p(x_2)(1 + p(x_1)) + \|x_1 + 50p(x_2)(1 - 2p(x_1))\|$ $+\|x_2 + 50(1 - 2p(x_2))\|$ $p(x) = 1$, if $x \leq 0, p(x) = 0$ if $x < 0$.	$[-100, 100]^D$
$F_{13} = 0.5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$	$[-100, 100]^D$
$F_{14} = \sum_{i=1}^{D} [(x_i - 1)^2] + 1 + \sum_{i=2}^{D} [x_i(x_{i-1})]$	$[-1600, 1600]^D$
$F_{15} = (1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 13x_1^2$ $-14x_2 + 6x_1x_2 + 3x_2^2)) * (30 + 2x_1 - 3x_2)^2$ $(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2))$	$[-100, 100]^D$

OCS, CC, and BCF. We add Strategy 3, 4 , and 5 for comparison. Input parameters used in all strategies are the same as those of Hybrid approach explained in subsection 4.3 below.

For clustering in the tactics, we employ the well-known k-means algorithm. For simplicity we set k, the number of slaves, as eight.

Table III: Strategies in the Experiment

Strategy 0		Strategy 1		Strategy 2		Strategy 3		Strategy 4		Strategy 5	
PS	TC	PS	TC	PS	TC	PS	TC	PS	TC	PS	TC
1	WRS	1	WRS	1	WRS	1	WRS	1	WRS	1	BCF
				2	OCS						
		2	BCF	3	CC	2	OCS	2	OCS		
				4	BCF			3	CC		

Tables IV and V list the values of the final solution. MEAN denotes the average value, MIN, the minimum value among thirty executions, and STDEV, the standard deviation. Note that zero is the global minima for all functions, except for F_9 and F_{15}, which have optimum solutions -10 and 3, respectively. From the tables, we observed that Strategy 2 was the best strategy in terms of both MEAN and MIN values for most functions, even though the other strategies were slightly better for a few functions. We also observed that Strategies 4 and 5 have good solutions in most cases owing to the fact that they incorporate the final tactics of Strategy 2. In the case of Strategy 5, the solution is always close to the best values, which implies that BCF is the tactic that yields the best solutions, mostly in Strategy 2. Strategy 0, the isolated island model, exhibited the worst performance for most of the functions. The computation time for one execution was just a few seconds. Among the six strategies, Strategy 2 found the final solution at an earlier generation on average.

Table IV: Solution Quality ($F_6 - F_{10}$)

F		MEAN 1	STDEV	MIN
F_6	Strategy 0	1.894761E+02	15.839868	1.394067E+02
	Strategy 1	4.612843E-01	1.323513	1.335374E-05
	Strategy 2	2.999541E-01	0.816529	1.760697E-07
	Strategy 3	1.024819E+02	22.46609	3.663270E+01
	Strategy 4	9.979855E+00	6.243674	1.035758E+00
	Strategy 5	4.896217E-01	1.222234	1.687662E-04
F_7	Strategy 0	3.037367E+01	4.533935	2.003931E+01
	Strategy 1	2.300298E-01	0.576456	1.149834E-03
	Strategy 2	2.320089E-01	0.419868	9.176119E-04
	Strategy 3	2.767637E+01	12.56069	5.684698E+00
	Strategy 4	1.179731E+00	0.07433739	7.433739E-02
	Strategy 5	1.620464E-01	0.3743754	1.989410E-03
F_8	Strategy 0	1.256719E+01	0.5572259	1.172345E+01
	Strategy 1	1.256019E-02	0.009977694	1.332268E-15
	Strategy 2	1.332268E-15	0.000000	1.332268E-15
	Strategy 3	1.332268E-15	6.017600E-31	1.332268E-15
	Strategy 4	2.251204E-01	0.6912618	1.332268E-15
	Strategy 5	8.448955E-03	0.007588	1.332268E-15
F_9	Strategy 0	-4.579745E+00	0.274385	-5.154176E+00
	Strategy 1	-4.406482E+00	0.345473	-5.637859E+00
	Strategy 2	-4.216858E+00	0.448879	-5.366555E+00
	Strategy 3	-4.132271E+00	0.332019	-5.266340E+00
	Strategy 4	-4.298698E+00	0.523935	-5.764212E+00
	Strategy 5	-2.358188E+00	1.341437	-4.113095E+00
F_{10}	Strategy 0	5.285506E+02	307.989852	1.040280E+02
	Strategy 1	1.106116E+03	1196.894877	2.883050E+02
	Strategy 2	8.094983E+02	387.685231	2.415661E+02
	Strategy 3	1.607404E+03	916.306200	2.358568E+02
	Strategy 4	2.183783E+03	992145500	3.748262E+02
	Strategy 5	6.616938E+04	93167.270000	4.319299E+02

Figure 6 compares the ability of three strategies to improve (minimize) solutions evaluated using the Schaffer function (F_{13}) over a number of evolutionary generations; F_{13} is a multimodal function with a global optimum at 0. Strategy 0 is the worst performer among the three. With the application of some more tactics that introduce hybridization and partition effects, further improvements are observed with Strategy 1 and 2. Owing to advancements in the statistical exploration of the search space, Strategy 2 is observed to be far better than the other strategies.

Table V: Solution Quality ($F_{11} - F_{15}$)

F		MEAN 1	STDEV	MIN
F_{11}	Strategy 0	6.298387e-16	1.911565e-15	8.721350e-26
	Strategy 1	7.719056E-13	0.000000	2.545271E-26
	Strategy 2	4.671965e-16	2.109362e-15	1.745478e-28
	Strategy 3	4.969175E-10	1.287356E-09	1.566654E-18
	Strategy 4	1.505645E-08	5.459028E-08	2.227331E-14
	Strategy 5	1.574747E-08	8.592779E-08	1.961375E-24
F_{12}	Strategy 0	4.579862E-02	0.062411	2.655849E-04
	Strategy 1	1.049913E-01	0.192152	1.429740E-05
	Strategy 2	5.948715E-02	0.137409	2.032152E-12
	Strategy 3	1.269969E+00	0.9704151	3.315504E-02
	Strategy 4	9.740593E-01	0.7914018	1.713301E-03
	Strategy 5	1.855347E+00	1.860671	1.416840E-04
F_{13}	Strategy 0	7.804370E-03	0.003141	4.798975E-04
	Strategy 1	1.211043E-11	4.172366E-11	1.054712E-15
	Strategy 2	5.532964E-13	0.000000	4.996004E-16
	Strategy 3	7.103085E-09	2.124199E-08	4.872991E-12
	Strategy 4	4.150992E-12	7.877735E-12	3.330669E-16
	Strategy 5	2.973029E-13	8.774498E-13	5.551115E-16
F_{14}	Strategy 0	4.505036E+05	7.146276E+04	3.128791E+05
	Strategy 1	3.848346E+01	2.732852	3.445721E+01
	Strategy 2	3.819127E+01	4.019584	2.504938E+01
	Strategy 3	4.100000E+01	0.000000	4.100000E+01
	Strategy 4	5.972326E+01	102.5515	4.100000E+01
	Strategy 5	3.666976E+01	4.998478	2.268158E+01
F_{15}	Strategy 0	4.707042E+02	228.463800	5.490632E+00
	Strategy 1	6.534466E+01	104.670407	3.000000E+00
	Strategy 2	1.485090E+01	22.387410	3.000002E+00
	Strategy 3	3.401902E+02	239.601500	3.369548E+00
	Strategy 4	7.753362E+01	78.76845	3.072684E+00
	Strategy 5	1.399849E+02	221.498700	3.000013E+00

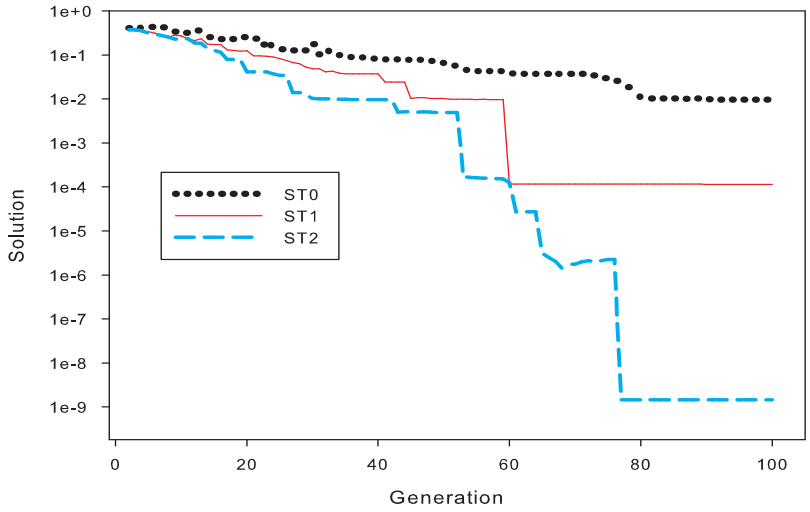

Figure 6: Strategic Solution Improvement Comparison for F_{13}

Even though the experimental evaluation is not sufficient, the effectiveness of our method can

be estimated from the results. The advantages of our approach are such that the method can be incorporated with strategy research, and it can be parallelized for speedup. Therefore, we plan to investigate strategy and scalability analysis with parallelization in greater detail.

4.3 Comparison with Other Approaches

To evaluate the efficiency of our hybrid approach, we compared it with other heuristic approaches, GA, PSO, and EDA (PBIL). For all approaches, we performed 30 independent executions (runs) with the same number of evaluations for each. However, the number of evaluations vary among functions based on predetermined maximum number of evaluations set in standard PSO. The best (minimum) solution is recorded in every execution.

In the hybrid approach, each of the eight slaves uses a total of 16 iterations equally divided among four tactics (i.e., the tactic is changed after every four iterations), and we employ Strategy 2, as shown in Table III. Four iterations are statistically enough to give satisfactory number of local optima solutions necessary for clustering and vector sampling by Master process in each tactic (phase). With four tactics, it makes total of sixteen sequentially executed iterations. Synchronous model of our approach was set for all slaves to finish their search at the same time in a given iteration (fixed number of generations as a termination condition). All eight slaves are parallelly executed, each running its own independent instance of the algorithm, and they are set to accommodate a similar number of predefined parameters such as population size, search space, and generations. The K-means clustering is performed just after WRS, OCS, and at each iteration within CC. We have used eight slaves in our experiment to balance the processing load between our computer processor cores, however the number of slaves is not limited. The total number of evaluations is given by

$$NEval = SLAVES \cdot POPSIZE \cdot ITER \cdot GEN, \tag{7}$$

where $SLAVES$ is the number of slaves, $POPSIZE$ is the population size in each slave, $ITER$ is the number of iterations, and GEN is the average number of generations.

We used Standard PSO 2007, where the number of evaluations is controlled by setting it within the specific function, controlling the swarm size, and carefully observing the average $NEval$ in every run. Owing to improvement strategies set in PSO and early terminations caused by erroneous evaluation, we needed to monitor $NEval$ by observing the final number of successfully evaluated solutions.

For GAs and EDA, the total number of evaluations was obtained as

$$NEval = POPSIZE \cdot GEN \tag{8}$$

For a fair comparison, we set the same number of total evaluations in every approach. We solved every function listed in Table II 30 times and recorded the average data in Tables VI and VII. From Tables VI and VII, we conclude that our hybrid approach yields better solutions on average for ten functions among fifteen.

Table VI: Solution Quality ($F_1 - F_8$)

F		MEAN	STDEV	MIN
F_1	GA	2.171826E+01	8.033171	1.097725E+01
	PSO	6.638152E+02	132.358560	3.830402E+02
	EDA	5.830295E-01	1.083213E-01	3.139274E-01
	HYBRID	1.479625E-05	5.601400E-05	8.073568E-13
F_2	GA	1.173199E+03	387.679700	5.531206E+02
	PSO	3.440920E+03	633.273975	2.2253382E+03
	EDA	1.142110E+05	2.548724E+04	6.105655E+04
	HYBRID	6.034880E+00	15.845680	8.054725E-03
F_3	GA	1.913765E+00	0.3616138	1.266923E+00
	PSO	9.158852E-01	0.659434	1.035516E-01
	EDA	1.784505E+00	1.328715E-01	1.296474E+00
	HYBRID	2.838989E-02	2.861382E-02	1.171564E-07
F_4	GA	1.803249E+02	48.16632	1.047767E+02
	PSO	5.793312E+01	45.130289	1.258991E+01
	EDA	5.654497E+01	6.075012E+00	4.665599E+01
	HYBRID	2.898902E+01	0.02081097	2.891641E+01
F_5	GA	1.600000E+00	1.753814	0.000000E+00
	PSO	1.926667E+01	7.465178	8.000000E+00
	EDA	1.566667E+00	5.295954E+00	0.000000E+00
	HYBRID	1.266667E+00	1.257620	0.000000E+00
F_6	GA	4.938599E+01	27.811390	1.818478E+01
	PSO	4.964892E+01	13.439184	2.28884048E+01
	EDA	1.685275E+02	1.104919E+01	1.477889E+02
	HYBRID	2.999541E-01	0.816529	1.760697E-07
F_7	GA	1.290777E+00	0.101764	1.130136E+00
	PSO	1.006164E+00	0.032570	8.914737E-01
	EDA	6.832971E-01	5.458563E-02	5.954656E-01
	HYBRID	2.320089E-01	0.419868	9.176119E-04
F_8	GA	3.543511E+00	0.345603	2.738278E+00
	PSO	1.424752E+00	0.347108	8.08992754E-01
	EDA	4.055419E-01	5.203604E-02	3.118305E-01
	HYBRID	1.332268E-15	0.000000	1.332268E-15

Figure 7 shows the solution improvement curves for the Rastrigin function (F_6) for GA, EDA, PSO, and the proposed hybrid approach during one hundred generations of searching. The curves indicate that there is considerable variation in the fitness values obtained by the hybrid approach; however, it finally yields the best solution as compared to the other approaches. The considerable variations is attributed to the application of strategies during searching. Independent GA has the lowest capability to improve the solutions.

Table VII: Solution Quality ($F_9 - F_{15}$)

F		MEAN 1	STDEV	MIN
F_9	GA	-2.444638E+00	1.865700	-5.327922E+00
	PSO	1.007508E+00	0.033592	4.185728E-01
	EDA	-7.807477E+00	2.794171E-01	-8.245333E+00
	HYBRID	-4.216858E+00	0.448879	-5.366555E+00
F_{10}	GA	3.614425E+02	369.5434000	5.500511E+00
	PSO	1.930132E+02	137.607458	4. 0.000000E+00
	EDA	3.232064E+02	3.441669E+02	2.830059E+00
	HYBRID	8.094983E+02	387.685231	2.415661E+02
F_{11}	GA	4.586383E-09	0.000000	1.776972E-17
	PSO	2.945888E-10	0.000000	4.8.571489E-16
	EDA	1.099034E-14	1.553807E-14	6.990018E-19
	HYBRID	7.719056E-13	0.000000	2.545271E-26
F_{12}	GA	3.627734E-01	0.5304258	3.121911E-46
	PSO	4.747350E-02	0.124603	4. 5.093622E-04
	EDA	5.363002E-01	6.577885E-01	5.173846E-11
	HYBRID	5.948715E-02	0.137409	2.032152E-12
F_{13}	GA	2.900864E-03	0.004231	0.000000E+00
	PSO	4.046250E-03	0.003521	4. 3.829711E-05
	EDA	1.027831E-02	1.035801E-03	9.715927E-03
	HYBRID	5.532964E-13	0.000000	4.996004E-16
F_{14}	GA	8.703169E+03	2477.476000	4.417168E+03
	PSO	3.228840E+03	2634.603429	4. 1.648598E+02
	EDA	8.561090E+03	1.598077E+03	6.322911E+03
	HYBRID	3.819127E+01	4.019584	2.504938E+01
F_{15}	GA	3.729539E+03	11024.459109	3.000084E+00
	PSO	3.000000E+00	0.000000	4.3000000E+00
	EDA	3.965054E+00	2.354626E+00	3.000000E+00
	HYBRID	1.485090E+01	22.387410	3.000002E+00

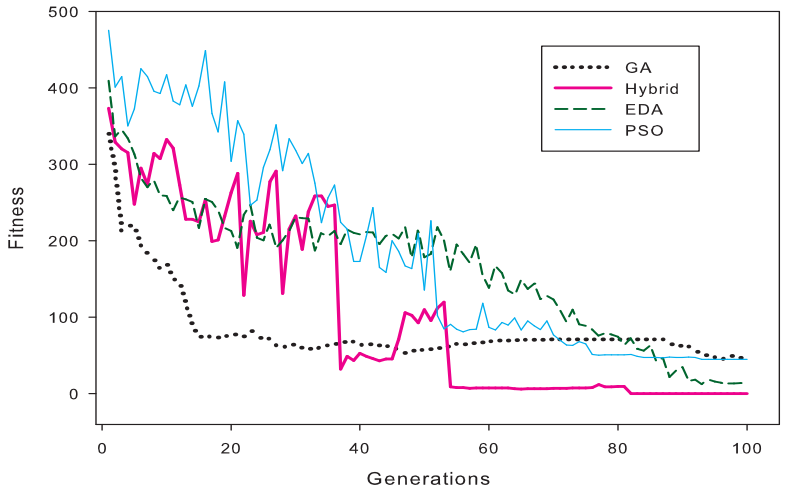

Figure 7: Solution Improvement Curves for F_6

A similar behavior is observed in Fig. 8 for the Schaffer function (F_{13}). Although the results are promising, this approach requires further investigation. For instance, the applicability of dynamic K-means clustering to optimize the clustering outcome by considering the migration

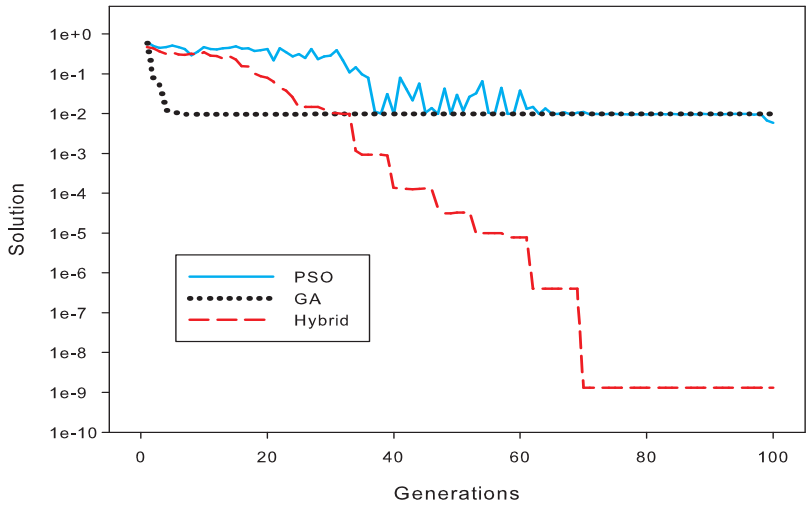

Figure 8: Solution Improvement Curves for F_{13}

effect between islands, their topologies, etc.

5. CONCLUDING REMARKS

In this paper, we proposed a hybrid method of EDAs and GAs on the basis of master/slave cooperation. In this method, the master process estimates the probability distribution of the search space on the basis on the non-dependency model at each iteration and sends probability vectors to the slaves. The slaves use the vectors to generate a new initial population. We employed the simplest probability model, and we compensated for the reduced accuracy by applying GAs to the solutions sampled using the simplest model. Moreover, our method can be incorporated with strategy research, and it easily can be parallelized. We conducted experiments to verify the effectiveness of our method. However, careful evaluation with more complex functions and comparisons with other approaches are required. In particular, accuracy analysis should be carried out for more complex problems to demonstrate the effectiveness of our method. Real-world deployment of our approach is considered in the future, with a cloud having master and slaves all running on independent VM instances. As master and slave processes, might fail at some point during searching, we also need to consider fault tolerance mechanism in a real world implementation of our methodology.

ACKNOWLEDGMENTS

A part of this research was supported by Grant-inAid for Scientific Research (C) 22500031 from Japan Society for the Promotion of Science (JSPS).

REFERENCES

Alba, E., Ed. 2005. *Parallel Metaheuristics -A new class of algorithms*. Wiley-Intescience.

Baluja, S. 1994. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. Technical Report CMU-CS-94-163, Carnegie Mellon University.

Bosman, P. A. N. and Thierens, D. 2000. Continuous iterated density estimation evolutionary algorithms within the idea framework. In *Proc of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*. 197–200.

Davis, L. 1991. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York (1991).

Dorigo, M. 1996. Ant system: optimization by colony of cooperating agents. *IEEE Trans on systems man, and cybernetics - part B 26*, 29–41.

Eberhart, R. C. and Shi, Y. 2001. Particle swarm optimization: Developments, applications and resources. *IEEE 1*, 81–86.

Gagne, C. and Parizeau, M. 2003. A robust master-slave distribution architecture for evolutionary computations. In *Proc of Genetic and Evolutionary Computation Conference Late Breaking 2003*. 80–87.

Gallagher, M. R., Frean, M., and Downs, T. 1999. Real-valued evolutionary optimization using a flexible probability density estimator. In *Proc of Genetic and Evolutionary Computation Conference*. 840–846.

Gong, Y. and Nakamura, M. 2008. Migration effects of parallel genetic algorithms on line topologies of heterogeneous computing resources. *IECE Transactions on Fundamentals of Electronics, Communication and Computer Sciences E91-A(4)*, 1121–1128.

Harik, G., Lobo, F. G., and Goldberg, D. E. 1998. The compact genetic algorithm. In *Proc of the IEEE Conference on Evolutionary Computation*. 523–528.

Holland, J. H. 1972. *Adaptation in Nature and Artificial Systems*. The University of Michigan Press, Reprinted by MIT Press (1992).

Hu, X. 2004. Recent advances in particle swarm. *Evolutional Computation 1*, 90–97.

J. Zhang, W. C., Zhong, J., Tan, Z., and Li, Y. 2006. Continuous function optimization using hybrid ant colony approach with orthogonal design scheme. In *SEAL 2006, Lecture Notes on Computer Science, 4247*. 126–133.

Larrañaga, P. and Lozano, J. A. 2002. *Estimation of Distribution Algorithms*. Kluwer Academic Publishers.

Mendiburu-Alberro, A. 2006. Parallel implementation of estimation of distribution algorithms based on probabilistic graphical models. application to chemical calibration models. Ph.D. thesis, The University of the Basque Country.

Mühlenbein, H. 1998. The equation for response to selection and its use for prediction. *Evolutional Computation 5*, 303–346.

Mühlenbein, H. and Paass, G. 1996. From recombination in genes to the estimation of distribution i, binary parameters. In *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature*.

Ocenasek, J. 2002. Parallel estimation of distribution algorithms. Ph.D. thesis, Brno University of Technology.

Perikan, M., Goldberg, D., and Cantú-Paz, E. 1999. Boa: The bayesian optimization algorithm. In *Proc of the Genetic and Evolutionary Computation Conference*. 525–532.

Said, S. M. and Nakamura, M. 2010. A hybrid approach of edas and gas based on master/slave cooperation for continuous function optimization. In *Proc of NABIC 2010 Second World Congress*. 244–248.

Salomon, R. 1996. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *ELSEVIER, Biosystems 39*, 263–278.

Sebag, M. and Ducoulombier, A. 1998. Extending population-based incremental learning to continuous search spaces. In *Proc of Parallel Problem Solving from Nature ? PPSN V*. 418–427.

Said Mohamed Said was born in Zanzibar, Tanzania in 1981. He received his B.Sc. degree in Computer Science from University of Dar-Es-Salaam, Tanzania in 2008 and M.E. in Information Engineering from University of the Ryukyus, Okinawa Japan in 2011. He is an Assistant Lecturer at University of Dodoma, Tanzania. Currently he is a Ph.D. candidate at University of the Ryukyus Japan. His main research interests include Parallel and Distributed Computing, Bioinformatics, and Cloud Computing.

Senlin Guan received the B.E. from Central South University of Forestry and Technology, China in 1995, and M.E. and Ph.D. from University of the Ryukyus, Japan in 2006 and 2009. He is currently special assistant professor at University of the Ryukyus. His research interest includes Petri net modeling, optimization computation for scheduling problems, and mobile and cloud computing.

Morikazu Nakamura was born in January 28, 1966. He took B.E. and M.E. from University of the Ryukyus in 1989 and 1991, respectively and D.E. from Osaka University in 1996. He had been an Associate Professor from 1996 to 2005 and has been a Professor since 2005 at the same university. His research interests are design and analysis of parallel and distributed algorithms and Petri nets. He is a member of the Institute of Electrical and Electronics Engineers and Institute of Electronics (IEEE), Information and Communication Engineers (IEICE).