Adaptive Discretization on Multidimensional Continuous Search Spaces

Jiun-Jiue Liou
Ying-ping Chen

NCLab Report No. NCL-TR-2008002
February 2008

Natural Computing Laboratory (NCLab)
Department of Computer Science
National Chiao Tung University
329 Engineering Building C
1001 Ta Hsueh Road
HsinChu City 300, TAIWAN
`http://nclab.tw/`

Adaptive Discretization on Multidimensional Continuous Search Spaces

Jiun-Jiue Liou and Ying-ping Chen
Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
`vcore.tw@gmail.com, ypchen@nclab.tw`

February 02, 2008

Abstract

This paper extends an adaptive discretization method, Split-on-Demand (SoD), to be capable of handling multidimensional continuous search spaces. The proposed extension is called *multidimensional Split-on-Demand* (mSoD), which considers multiple dimensions of the search space as a whole instead of independently discretizing each dimension as SoD does. In this study, we integrate mSoD and SoD with the extended compact genetic algorithm (ECGA) to numerically examine the effectiveness and performance of mSoD and SoD on the problems with and without linkage among dimensions of the search space. The experimental results indicate that mSoD outperforms SoD on both of the test problems and that mSoD can offer better scalability, stability, and accuracy. The behavior of mSoD is discussed, followed by the potential future work.

1 Introduction

Genetic algorithms (GAs) [1] are methodologies inspired by natural evolution and widely applied to tackle enormous real-world problems as a flexible optimization framework. In genetic algorithms, promising individuals are selected from the current population to produce new solutions by utilizing the recombination and mutation operators, mimicking the biological genetic operations. According to the way genetic algorithms operate and the GA design decomposition theory [2], the key components to the GA success include identifying, reproducing, and exchanging the solution fragments. Because the recombination operator mixes promising sub-solutions and creates new solutions, genetic algorithms work well on the problems which can be implicitly or explicitly decomposed into sub-problems.

In order to further enhance the effects of sub-solution identifying and exchanging provided by the crossover and mutation operators in GAs, estimation of distribution algorithms (EDAs) [3] are proposed and developed by utilizing probabilistic models to capture and reflect the problem structure. EDAs are nowadays one of the promising branches in evolutionary computation that can offer high performance and robustness for solving optimization problems. In EDAs, thanks to the adopted models, decision variables are oftentimes encoded with some discrete coding scheme, such as binary coding. However, it is reportedly difficult to find high accuracy solutions in solving continuous problems for such a setting. To overcome this difficulty, several attempts have been made, including continuous PBIL with Gaussian distribution [4], real-coded variant of PBIL with interval updating [5], BEA for continuous function optimization [6], and the real-coded BOA [7].

Instead of directly modifying the algorithm, as an alternative and more general approach, Chen et al. [8] proposed an adaptive discretization method, called Split-on-Demand (SoD), to interface the discrete type optimization algorithms, including EDAs or GAs, with continuous variables by discretizing continuous domains. Integrated with the extended compact genetic algorithm (ECGA) [9], SoD was successfully employed to solve some real-world problems [10, 11]. Although SoD enables the discrete algorithms to tackle continuous problems, it processes each dimension of the search space independently disregarding any relationship or available linkage information among dimensions.

To develop better discretization techniques, in this study, our goal is to enhance the capability and performance of SoD and to make SoD able to take the advantage of available linkage information among dimensions, which may be obtained by a variety of linkage learning techniques [12]. In particular, we propose an extension of SoD, called *multidimensional Split-on-Demand* (mSoD), considering a group of dimensions and discretizing the multidimensional region. We integrate mSoD and SoD with ECGA to observe the performance difference.

In the next section, we will briefly review SoD and ECGA to provide a background of this study. In section 3, we will describe in detail how the proposed *multidimensional Split-on-Demand* (mSoD) operates and the integration of mSoD and ECGA. The numerical experiments and the experimental results for observing the performance difference of mSoD and SoD are given in section 4, followed by the discussion in section 5. Finally, section 6 concludes this work.

2 Background: SoD and ECGA

To provide a background of this study, in this section, we briefly review the discretization method to extend, Split-on-Demand (SoD), and the backend optimization engine, the extended compact genetic algorithm (ECGA), respectively.

2.1 Split-on-Demand

Split-on-Demand is an adaptive discretization method for encoding continuous decision variables with discrete codes by discretizing the continuous domain. The fundamental idea of SoD is simply to split the continuous interval where we wish to know in more detail and to build a more accurate probabilistic model. SoD splits the interval in which there are more than a certain number of individuals or search points. In order to determine which interval to split, a split rate γ, where $0 < \gamma < 1$, is used. Let the population size be N. If an interval contains more than $N \times \gamma$ individuals, the interval should be split at a random.

The pseudo code for SoD is shown in Figure 1. For a given interval, described by [*lower_bound*, *upper_bound*], to split, a split point m is generated at random. The interval is split into two intervals: $[\ell, m]$ and $[m, u]$. The split process runs recursively until the number of individuals in the target interval is no more than the threshold. Along the evolutionary process, the split rate, γ, is decreased with a split rate decreasing factor ϵ. The decreasing factor is utilized to manipulate the split rate such that the backend optimization algorithm will look into certain parts of the search space in more and more detail.

2.2 Extended Compact Genetic Algorithm

The extended compact genetic algorithm (ECGA) [9] is one of the estimation of distribution algorithms. The key idea of ECGA is that finding good probability models is equivalent to linkage learning. The probabilistic models adopted by ECGA are a class of models known as the marginal product models (MPMs). ECGA uses MPMs to model the partitions of decision

```
procedure SPLIT-ON-DEMAND
    Split(lower_bound, upper_bound)
    $\gamma \leftarrow \gamma \times \epsilon$
end procedure
```

```
procedure SPLIT($\ell$, $u$)
    $m \leftarrow$ random$[\ell, u]$
    $N_\ell \leftarrow$ number of individuals in $[\ell, m]$
    $N_u \leftarrow$ number of individuals in $[m, u]$
    if $N_\ell \geq N \times \gamma$ then
        Split($\ell$, $m$)
    else if $N_\ell > 0$ then
        Add a code for the range $[\ell, m]$
    else
        Ignore the range $[\ell, m]$
    end if
    if $N_u \geq N \times \gamma$ then
        Split($m$, $u$)
    else if $N_u > 0$ then
        Add a code for the range $[m, u]$
    else
        Ignore the range $[m, u]$
    end if
end procedure
```

Figure 1: Pseudo code for SoD.

variables. The measure of good distributions is quantified according to the minimum description length (MDL) principle [13]. The MDL principle penalizes both the inaccuracy and the complexity of models to achieve the balance.

The complexity measurement of MPM is the sum of Model Complexity, formulated as Equation (1), and Compressed Population Complexity, formulated as Equation (2).

$$\text{Model Complexity} = \log N \sum_I 2^{S[I]} \ , \tag{1}$$

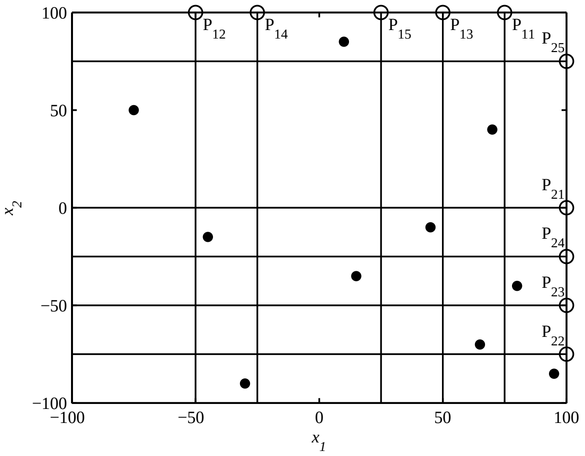
where N is the population size, and $S[I]$ is the length of the Ith subset of genes.

$$\text{Compressed Population Complexity} = N \sum E(M_I) \ , \tag{2}$$

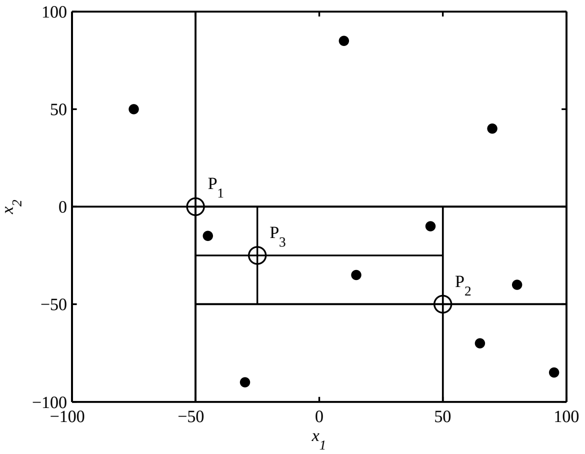
where $E(M_I)$ is the entropy of the marginal distribution of subset I. The goal for the greedy MPM search is to find an MPM model with the minimal combined complexity:

$$\begin{aligned} \text{Combined Complexity} = &\text{ Model Complexity} + \\ &\text{ Compressed Population Complexity} \ . \end{aligned}$$

Instead of using traditional crossover and mutation operators, ECGA generates the offspring population from the constructed MPM. New individuals are generated without breaking building blocks represented in the form of gene groups. In ECGA, the original framework handles only binary decision variables. Hung and Chen [14] extended the variable domain of ECGA to integers, and therefore, ECGA can be integrated with SoD as real-coded ECGA (rECGA) for tacking continuous optimization problems.



(a) Split configuration generated by SoD.



(b) Split configuration generated by mSoD.

Figure 2: Possible split configurations generated by SoD and mSoD, respectively, for the identical population of size 10. In Figure 2(a), P_{ij} denote the jth split point on the x_i axis. In Figure 2(b), P_j denote the jth split point on the x_1-x_2 plane.

3 Multidimensional SoD

In this section, we present the extension of SoD, called *multidimensional Split-on-Demand* (mSoD), which is able to split multidimensional continuous search spaces. We will also integrate mSoD and ECGA, called mrECGA, for the comparison study on the performance difference between mSoD and the original SoD.

3.1 Extension to Multidimensions

The major difference between mSoD and SoD is that mSoD splits multidimensional regions while SoD splits one dimensional intervals. Collective decision variables or dimensions are considered altogether during the splitting process. For example, consider a population of size 10 spread on a two dimensional search space, where the bound of each dimension is [-100, 100]. To handle this case, SoD needs to split the x_1 and x_2 axes independently, and a possible split configuration generated by SoD is shown in Figure 2(a). On the other hand, mSoD considers the two dimensions as a whole. When a split point is generated, the target region is split in

```
procedure Multidimensional-Split-on-Demand
    mSplit($bound_1$, $bound_2$, $\ldots$, $bound_n$)
    $\gamma \leftarrow \gamma \times \epsilon$
end procedure
```

```
procedure mSplit($b_1$, $b_2$, $\ldots$, $b_n$)
    ▷ $b_i$ is the bound of the $i$th dimension and
    ▷ consists of the lower bound and the upper bound
    Generate a random split point $P$ in the target region
    Split the target region according to $P$
    for each newly created region $R_k$ do
        $N_k \leftarrow$ Number of individuals in $R_k$
        if $N_k \geq N \times \gamma$ then
            mSplit($b_1^k$, $b_2^k$, $\ldots$, $b_n^k$)
            ▷ $b_i^k$ is the bound of the $i$th dimension of $R_k$
        else if $N_k > 0$ then
            Add a code for region $R_k$
        else
            Ignore region $R_k$
        end if
    end for
end procedure
```

Figure 3: Pseudo code for multidimensional SoD.

every dimension according to the position of the split point. Thus, for the identical population, a possible split configuration generated by mSoD is shown in Figure 2(b).

Firstly, mSoD randomly generates a split point in the whole search space. Suppose the first split point P_1 is at (-50, 0). It splits the whole region into 4 sub-regions. Similarly, if the second split point P_2 is randomly generated at (50, -50), P_2 further divides the sub-region on the southeast corner of the whole search space into four new regions. Assume that the next split point is P_3, the same split operation is repeated, and the whole split process is finished because no region contains more than two individuals. When the split process terminates, each non-empty region is assigned a code. Thus, every individual in the population is encoded. Similar to SoD, the split rate, γ, and the split rate decreasing factor, ϵ, are also used to control how the region should be split. The detailed procedure of mSoD is shown as the pseudo code in Figure 3.

For more practical examples, we use mSoD to discretize the continuous domains during the ECGA optimization process. Two simple examples are provided for the visualization of the split configurations generated by mSoD at different evolutionary stages. The objective function used in the example is the sphere function of two dimensions, $\sum_{i=1}^{2} x_i^2$, where the range of each dimension is [-200, 200], and the global optimum is located at (0, 0). For a typical run, Figure 4 depicts the split configurations at generation 1, 10, and 50, respectively. The split configuration seems coarse at generation 1 because the population is highly diverse in the beginning. In later stages, mSoD conducts more split operations near (0, 0) because the population begins to converge to the global optimum.

The objective function used in the other example is the sphere function of three dimensions, $\sum_{i=1}^{3} x_i^2$, where the range of each dimension is also [-200, 200]. The global optimum is located at (0, 0, 0). The split configurations at generation 1, 10, and 50 are shown in Figure 5.

Thanks to the design of mSoD, a multidimensional region can be viewed as a whole. The search points distributed within the region can be grouped across dimensions instead of being assigned unrelated codes for each dimension independently. This design can offer an important advantage to the backend optimization engine that the linkage information among dimensions, if available, can be utilized to help the optimization process. For real-world problems, even if the domain knowledge regarding the variables or the search space exists, it is usually difficult to incorporate such information in most evolutionary algorithms. By employing mSoD, we can take advantage of known relations of decision variables to encode the related variables collectively. For instance, given a two dimensional optimization problem, if we know the two decision variables are related, we can handle them together for discretization instead of processing them individually. Furthermore, mSoD also makes possible to integrate the linkage learning techniques [12] into common evolutionary algorithms because the obtained linkage information can now be utilized.

3.2 mSoD Working with ECGA

Because mSoD handles only the task of discretizing the continuous domain, in order to examine the performance of mSoD, a backend optimization engine is needed for the optimization task. In this study, ECGA is adopted to cooperate with mSoD for tackling continuous optimization problems. The integrated optimization framework of mSoD and ECGA is called the *multidimensional real-coded ECGA* (mrECGA). The procedure of mrECGA can described as:

1. Generate a random population of size N.
2. Apply tournament selection of size S.
3. Use mSoD to encode the individuals.
4. Model the population with a greedy MPM search.
5. Stop if the obtained MPM model has converged.
6. Generate a new population with the MPM model.
7. Return to step 2.

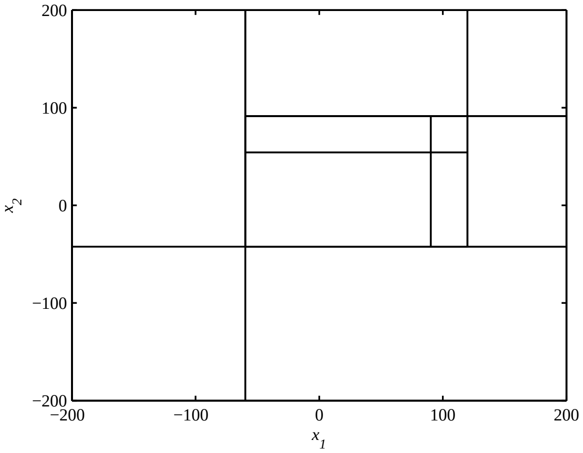
4 Experiments

In this section, we numerically examine the performance of mSoD and compare the performance difference between mSoD and SoD. We will introduce the problems adopted for benchmarking, the parameter settings for conducting the experiments, and finally the experimental results. The discussion on the experimental results and the observation in the experiments will be given in the next section.

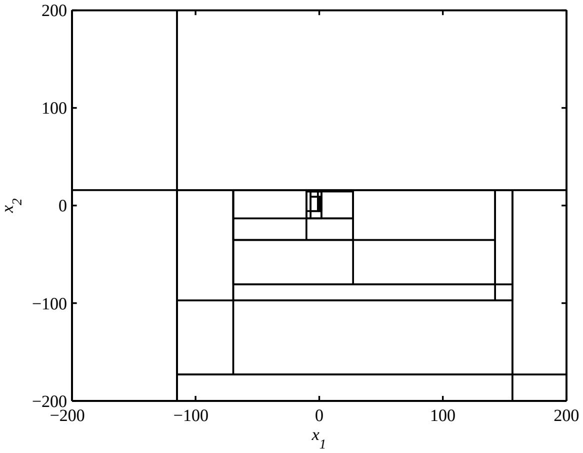
4.1 Problem Definitions

Since the goal is to compare the performance difference between mSoD and SoD, we conduct the experiments on two types of problems, the problems with linkage and the problems without linkage. Because mSoD is explicitly designed for discretizing multidimensional search spaces, it is expected to perform well on continuous problems consisting of groups of interrelated decision variables or dimensions. We call such problems as the problems with linkage in this study. In order to properly control the dimension relationship of the test problem, we define the problem with linkage of $n = m \times k$ dimensions as

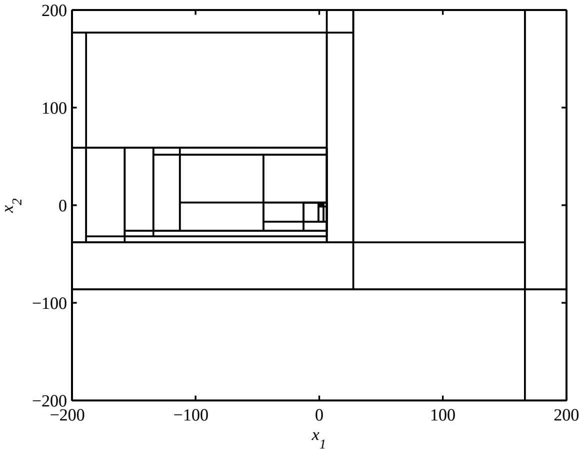
$$f_{\text{linkage}}(m, k, \vec{x}) = \sum_{i=1}^{m} g_i(k, \vec{x_i}) \ , \tag{3}$$



(a) Generation 1

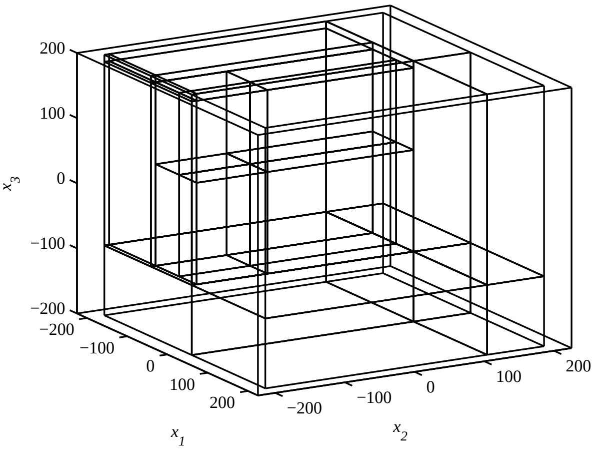


(b) Generation 10

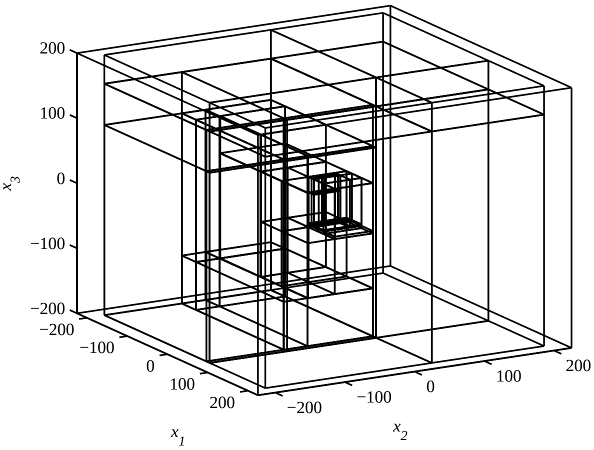


(c) Generation 50

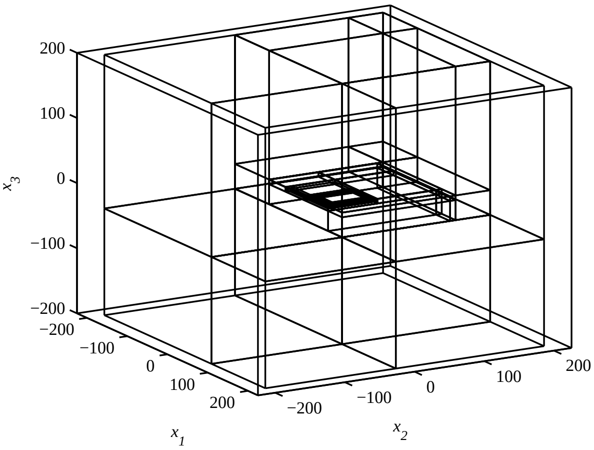
Figure 4: Split configurations generated by mSoD at different generations during the ECGA evolutionary process for the sphere function of two dimensions, $\Sigma_{i=1}^{2} x_i^2$.



(a) Generation 1



(b) Generation 10



(c) Generation 50

Figure 5: Split configurations generated by mSoD at different generations during the ECGA evolutionary process for the sphere function of three dimensions, $\Sigma_{i=1}^{3} x_i^2$.

where m is the number of subproblems, k is the number of dimensions of each subproblem, $\vec{x}$ is an n dimensional vector,

$$\vec{x} = [x_1, x_2, \ldots, x_n] \ ,$$

and $\vec{x_i}$ is the ith k dimensional vector within $\vec{x}$,

$$\vec{x_i} = [x_{(i-1)k+1}, x_{(i-1)k+2}, \ldots, x_{(i-1)k+k}] \ .$$

$g(\cdot,\cdot)$ is the elementary subproblem designed to make all dimensions interrelated and defined as

$$g(k, \vec{y}) = (y_1 - c)^2 + \sum_{j=1}^{k-1} (y_{j+1} - y_j)^2 \ , \tag{4}$$

where k is the number of dimensions, c is a constant for controlling the location of the global optimum, and $\vec{y}$ is a k dimensional vector,

$$\vec{y} = [y_1, y_2, \ldots, y_k] \ .$$

The function $g(k, \vec{y})$ is a specifically designed function to be used as the elementary subproblem. The global optimum is 0 and can be obtained when each y_i is equal to y_{i-1} for $i > 1$ and y_1 is equal to the assigned constant c. Because each y_i cannot be optimized independently, all the variables of $g(k, \vec{y})$, $y_1, y_2, \ldots, y_k$, have to be considered together to obtain the global optimum.

The function $f_{\text{linkage}}(m, k, \vec{x})$ is composed of m subproblems $g(\cdot,\cdot)$, and the constant c_i is assigned different values for each subproblem $g_i(\cdot,\cdot)$. The function $f_{\text{linkage}}(m, k, \vec{x})$ is the test problem with linkage in the experiments.

Since the decision variables in each subproblem $g_i(\cdot,\cdot)$ are strongly interrelated, we apply mSoD on each $g_i(\cdot,\cdot)$ accordingly. In order to analyze and observe the performance of mSoD and SoD on different numbers of related variables, subproblems, and total dimensions, we conduct the experiments of both mSoD and SoD with various k and m.

For the test problem without linkage, in this study, we adopt the sphere function, which can be described as

$$f_{\text{sphere}}(n, \vec{x}) = \sum_{i=1}^{n} x_i^2 \ , \tag{5}$$

where n is the number of dimensions, and $\vec{x}$ is an n dimensional vector. The global optimum is 0 when all x_i's are equal to 0. Since all decision variables are independent, each of them can be optimized independently, and no relation exists among them. We apply both mSoD and SoD on all the dimensions.

4.2 Experimental Settings

The error value is defined as $|f(x) - f(x^*)|$, where x^* is the location of the global optimum, and $f(x^*)$ is the global optimum. The lower bounds and upper bounds for all decision variables are -200 and 200, respectively. The parameter settings in the experiments for both mrECGA (mSoD+ECGA) and rECGA (SoD+ECGA) are

- maximum generation: 2000;
- population size: 200;
- tournament size: 8;
- crossover probability: 0.975;

- split rate (γ): 0.5;
- split rate decreasing factor (ϵ): 0.998.

All the experimental results are averaged over 50 runs. If the algorithm reaches a given accuracy level $\zeta = 10^{-6}$ in a run, we consider the run as a *successful* run. The successful rate (SR) is calculated as the ratio of the number of success runs divided by the number of total runs.

4.3 Experimental Results

Tables 1 and 2 report the statistics of mSoD and SoD on the test problems with linkage of various k, the number of dimensions of the subproblem, and m, the number of subproblems. The statistics include the averaged error values, the standard deviations of the error values, and the successful rates over the 50 independent runs. Table 3 shows the experimental results on the test problems without linkage for both mSoD and SoD.

For easy observations, we also plot the experimental results listed in Tables 1, 2, and 3 in Figures 6 and 7. The left axis represents the mean error value in the log scale (base 10). The standard deviations are shown in the form of errorbars with an exception that if the standard deviation is greater than the mean, the lower part of the errorbar is not shown on the plot because of the log scale. The right axis represents the successful rate for each individual experiment. These results will be discussed in the next section.

5 Discussion

From an overall point of view, mSoD outperforms SoD on both of the test problems with and without linkage when cooperating with ECGA. For all the experiments, the successful rates of mSoD are equal to or greater than that of SoD. For most of the experiments, mSoD can provide better solutions than SoD can. Therefore, mSoD offers better discretizing capability when working with ECGA. In the following sections, we will discuss the experimental results for the different types of the test problems.

5.1 Test Problems with Linkage

For the problems with linkage, we can observe in Tables 1 and 2 as well as in Figure 6 that mrECGA successfully solved the problems with k=2, 3, and 4 in all runs. mrECGA also solved the problems with k=5 and m=1, 2, and 3 but failed to solve the problems with m=5 and 6. rECGA failed to solve the problems on most runs as k increased. It can be argued that it was a series of unfair experiments because mSoD utilized the linkage information while SoD did not. However, the purpose of these experiments are twofold: (1) verifying that mSoD can outperform SoD on the problems with linkage as expected; (2) demonstrating that mSoD can utilize the linkage information if available while SoD cannot. The results indicate that mSoD achieves the design goal.

5.2 Test Problems without Linkage

For the problems without linkage, Table 3 and Figure 7 indicate that mrECGA solved all the problems in all runs, and the success rates of mrECGA are equal to or greater than that of rECGA. When the problem size increased, the performance of rECGA largely reduced. Furthermore, it was unexpected that although the variables in the sphere function are all independent and can be optimized individually, mrECGA still provided better solutions than rECGA did. In addition to the outcome that mSoD outperformed SoD on both types of the problems, the reason why mSoD can offer better performance on the sphere function needs further investigations.

$k = 2$	m	1	3	5	7	9	11	13	15
	$n = m \times k$	2	6	10	14	18	22	26	30
	Mean	7.06e-22	1.72e-17	1.51e-15	2.92e-14	2.08e-13	9.55e-13	1.63e-12	6.20e-12
	Std. Dev.	5.61e-22	9.59e-18	5.42e-16	1.14e-14	6.06e-14	2.94e-13	1.18e-12	3.58e-12
	SR	100%	100%	100%	100%	100%	100%	100%	100%
$k = 3$	m	1	3	5	7	9	11		
	$n = m \times k$	3	9	15	21	27	33		
	Mean	1.21e-19	5.05e-15	5.46e-13	8.98e-12	4.85e-11	1.88e-10		
	Std. Dev.	8.83e-20	2.75e-15	2.20e-13	3.38e-12	1.54e-11	5.34e-11		
	SR	100%	100%	100%	100%	100%	100%		
$k = 4$	m	1	2	3	4	5	6	7	8
	$n = m \times k$	4	8	12	16	20	24	28	32
	Mean	2.10e-18	6.40e-15	4.62e-13	5.81e-12	3.67e-11	1.74e-10	7.70e-10	2.96e-09
	Std. Dev.	1.33e-18	3.96e-15	2.65e-13	2.63e-12	1.48e-11	5.95e-11	2.78e-10	1.07e-09
	SR	100%	100%	100%	100%	100%	100%	100%	100%
$k = 5$	m	1	2	3	4	5	6		
	$n = m \times k$	5	10	15	20	25	30		
	Mean	1.39e-16	5.71e-13	4.88e-11	1.82e-08	2.99e-06	1.36e-04		
	Std. Dev.	2.19e-16	3.10e-13	3.24e-11	2.99e-08	5.84e-06	1.84e-04		
	SR	100%	100%	100%	100%	40%	0%		

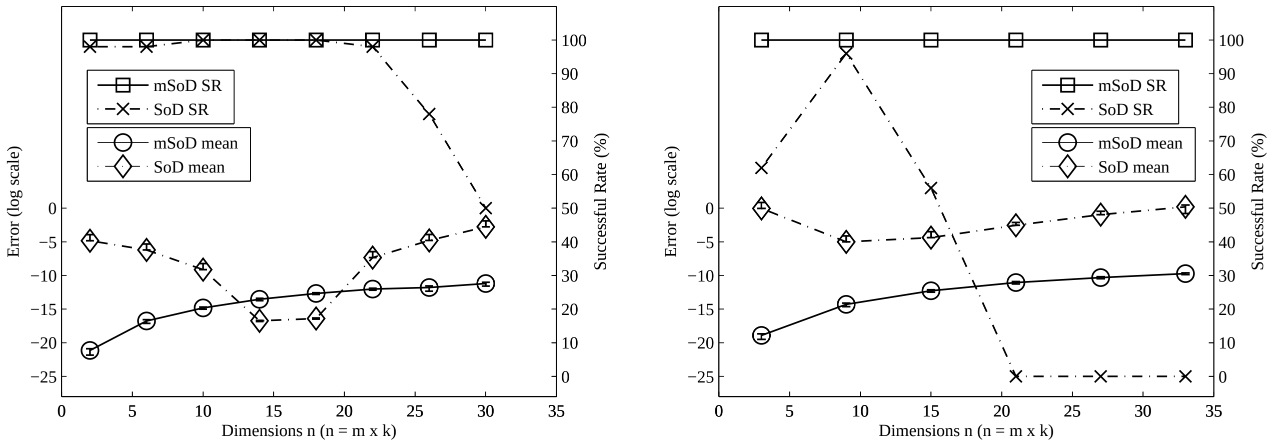
Table 1: mSoD on the problems with linkage of various k and m.

$k = 2$	m	1	3	5	7	9	11	13	15
	$n = m \times k$	2	6	10	14	18	22	26	30
	Mean	1.48e-05	6.12e-07	7.37e-10	1.78e-17	4.05e-17	4.60e-08	1.64e-05	1.70e-03
	Std. Dev.	1.04e-04	4.28e-06	5.16e-09	3.37e-18	7.39e-18	2.97e-07	1.04e-04	1.12e-02
	SR	98%	98%	100%	100%	100%	98%	78%	50%
$k = 3$	m	1	3	5	7	9	11		
	$n = m \times k$	3	9	15	21	27	33		
	Mean	9.27e-01	9.74e-06	4.29e-05	2.86e-03	1.08e-01	1.56e+00		
	Std. Dev.	6.04e+00	6.78e-05	2.84e-04	4.91e-03	2.45e-01	1.40e+00		
	SR	62%	96%	56%	0%	0%	0%		
$k = 4$	m	1	2	3	4	5	6	7	8
	$n = m \times k$	4	8	12	16	20	24	28	32
	Mean	1.89e+00	1.91e-04	3.94e-03	1.48e-01	1.41e+00	6.32e+00	1.78e+01	5.70e+01
	Std. Dev.	9.07e+00	7.68e-04	5.00e-03	1.86e-01	1.52e+00	5.21e+00	1.20e+01	5.14e+01
	SR	10%	12%	0%	0%	0%	0%	0%	0%
$k = 5$	m	1	2	3	4	5	6		
	$n = m \times k$	5	10	15	20	25	30		
	Mean	1.37e+00	2.35e-01	4.01e+00	2.92e+01	1.62e+02	3.14e+02		
	Std. Dev.	3.53e+00	2.84e-01	3.60e+00	2.79e+01	1.20e+02	1.98e+02		
	SR	2%	0%	0%	0%	0%	0%		

Table 2: SoD on the problems with linkage of various k and m.

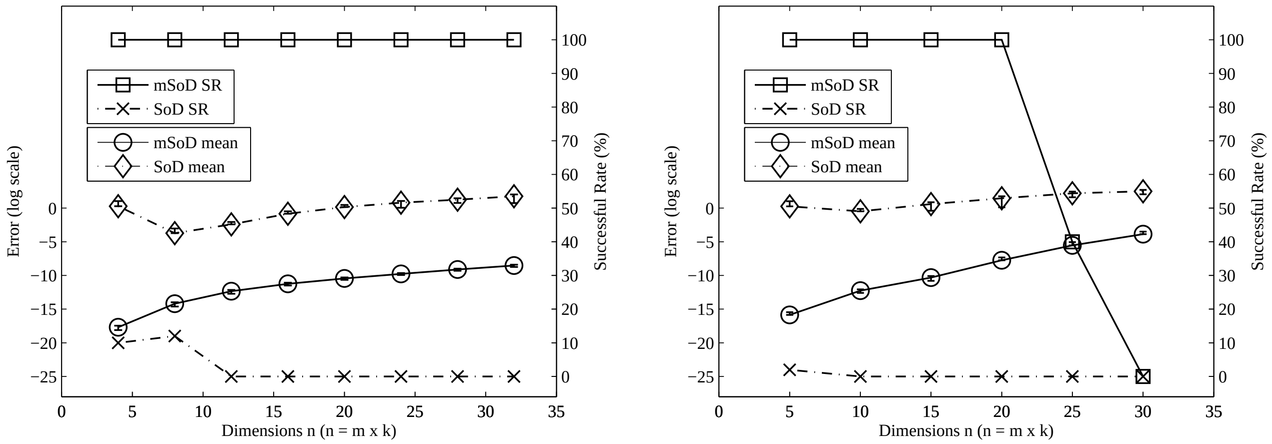
	n	10	20	30	40	50	60
mSoD	Mean	1.29e-14	1.33e-11	6.75e-10	7.88e-9	4.85e-8	2.55e-7
	Std. Dev.	6.85e-15	4.58e-12	2.12e-10	2.03e-9	1.12e-8	5.89e-8
	SR	100%	100%	100%	100%	100%	100%
	n	10	20	30	40	50	60
SoD	Mean	4.62e-18	4.56e-17	1.22e-9	2.80e-4	1.20e-1	2.58e-1
	Std. Dev.	1.10e-18	6.17e-18	6.23e-9	1.96e-3	8.30e-1	1.49e+0
	SR	100%	100%	100%	92%	50%	14%

Table 3: mSoD and SoD on the problems without linkage of different problem sizes.



(a) $k = 2$ with different m.

(b) $k = 3$ with different m.



(c) $k = 4$ with different m.

(d) $k = 5$ with different m.

Figure 6: Performance of mSoD and SoD on the problems with linkage.

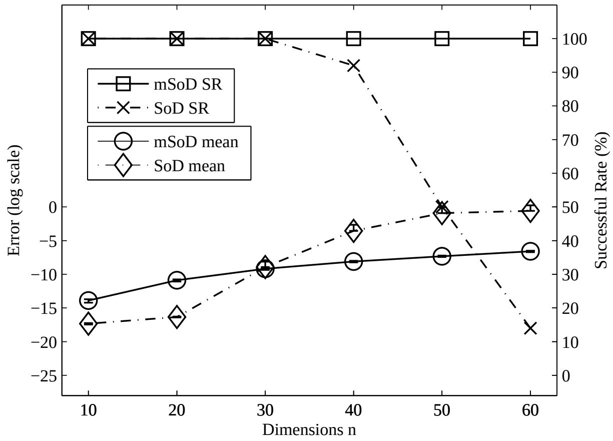


Figure 7: Performance of mSoD and SoD on the problems without linkage.

5.3 Virtual Alphabet

The experimental results may be explained by the concept of *virtual alphabets* proposed by Goldberg [15]. Since SoD maps each dimension into its own set of virtual alphabets, the blocking situation may easily occur when the landscape prevents the optimization algorithm from accessing the global optimum via all dimensions. Instead, mSoD maps the related variables into one set of virtual alphabets, the blocking situation may be automatically resolved by the mechanism of mSoD. The results on the problems with linkage evidentially support the use of the virtual alphabet concept to explain the reason why mSoD performs well. Further studies are needed for the direct evidence.

6 Summary and Conclusions

In this paper, we proposed an extension to SoD, called *multidimensional Split-on-Demand* (mSoD), considering the multidimensional space as a whole and splitting the space accordingly. To numerically examine the effectiveness, we integrated mSoD and SoD with ECGA and conducted experiments on the problems with and without linkage. The results showed that mSoD outperformed SoD on both types of the test problems, and the observations were discussed.

The ability of mSoD to adaptively discretize multidimensional search spaces opens a portal between the algorithms designed for discrete variables and the problems composed of continuous variables. The mechanism of mSoD also enables the backend optimization algorithm to incorporate the available linkage information, which may greatly promote the optimization performance. The experimental results demonstrate that advancing the adaptive discretization technique is a promising research direction. More work along this line needs to be done, and we will continue to develop better discretization methods.

Acknowledgments

The work was supported in part by the National Science Council of Taiwan under Grants NSC-96-2221-E-009-196, NSC-96-2627-B-009-001, and NSC-96-2815-C-009-047-E as well as by the ATU Plan (Aiming for the Top University and Elite Research Center Development Plan) of the National Chiao Tung University and Ministry of Education, Taiwan. The authors are grateful to the National Center for High-performance Computing for computer time and facilities.

References

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press, 1975, ISBN: 0-262-58111-6.

[2] D. E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, ser. Genetic Algorithms and Evolutionary Computation. Kluwer Academic Publishers, June 2002, vol. 7, ISBN: 1-4020-7098-5.

[3] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, ser. Genetic algorithms and evolutionary computation. Boston, MA: Kluwer Academic Publishers, October 2001, vol. 2, ISBN: 0-7923-7466-5.

[4] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces," in *Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature (PPSN V)*, 1998, pp. 418–427.

[5] I. L. Servet, L. Trave-Massuyes, and D. Stern, "Telephone network traffic overloading diagnosis and evolutionary computation techniques," in *Proceeings of the Third European Conference on Artificial Evolution (AE 97)*, 1997, pp. 137–144.

[6] S.-Y. Shin and B.-T. Zhang, "Bayesian evolutionary algorithms for continuous function optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation 2001*, 2001, pp. 508–515.

[7] C. W. Ahn, R. S. Ramakrishna, and D. E. Goldberg, "Real-coded Bayesian optimization algorithm, bringing the strength of BOA into the continuous world," in *Proceedings of Genetic and Evolutionary Computation Conference 2004*, 2004, pp. 840–851.

[8] C.-H. Chen, W.-N. Liu, and Y.-p. Chen, "Adaptive discretization for probabilistic model building genetic algorithms," in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2006*, 2006, pp. 1103–1110.

[9] G. R. Harik, "Linkage learning via probabilistic modeling in the ECGA," Illinois Genetic Algorithms Laboratory, UIUC, Urbana, IL, IlliGAL Report No. 99010, 1999.

[10] C.-H. Chen and Y.-p. Chen, "Real-coded ECGA for economic dispatch," in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2007*, 2007, pp. 1920–1927.

[11] P.-C. Hung, Y.-p. Chen, and H. W. Zan, "Characteristic determination for solid state devices with evolutionary computation: A case study," in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2007*, 2007, pp. 2029–2036.

[12] Y.-p. Chen, T.-L. Yu, K. Sastry, and D. E. Goldberg, "A survey of genetic linkage learning techniques," Illinois Genetic Algorithms Laboratory, UIUC, Urbana, IL, IlliGAL Report No. 2007014, 2007.

[13] J. Rissanen, *Stochastic Complexity in Statistical Inquiry.* World Science, 1989.

[14] P.-C. Hung and Y.-p. Chen, "iECGA: Integer extended compact genetic algorithm," in *Proceedings of ACM SIGEVO Genetic and Evolutionary Computation Conference 2006*, 2006, pp. 1415–1416.

[15] D. E. Goldberg, "Real-coded genetic algorithms, virtual alphabets, and blocking," Illinois Genetic Algorithms Laboratory, UIUC, Urbana, IL, IlliGAL Report No. 90001, 1990.