Enabling the Extended Compact Genetic Algorithm for Real-Parameter Optimization by using Adaptive Discretization

Ying-ping Chen
Chao-Hong Chen

NCLab Report No. NCL-TR-2007006
March 2007

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/`

Enabling the Extended Compact Genetic Algorithm for Real-Parameter Optimization by using Adaptive Discretization

Ying-ping Chen and Chao-Hong Chen
Department of Computer Science
National Chiao Tung University
HsinChu City 300, Taiwan
{ypchen, chchen}@nclab.tw

March 24, 2007

Abstract

This paper proposes an adaptive discretization method, called *Split-on-Demand* (SoD), to enable probabilistic model building genetic algorithms (PMBGAs) to solve optimization problems in the continuous domain. The procedure, effect, and usage of SoD are described in detail. As an example of using SoD with PMBGAs, the integration of SoD and the extended compact genetic algorithm (ECGA), named *real-coded* ECGA (rECGA), is proposed and numerically examined in the study. The numerical experiments include a set of benchmark functions and a real-world application, the economic dispatch problem. The results on the benchmark functions indicate that SoD is a better discretization method than two well-known methods: the fixed-height histogram (FHH) and the fixed-width histogram (FWH). Moreover, the experimental results on the economic dispatch problems demonstrate that rECGA works quite well, and the solutions better than the best known results presented in the literature are achieved.

1 Introduction

Genetic algorithms (GAs) [1, 2] are methodologies inspired by Darwinian evolution and designed according to the biological genetic operations. As a flexible optimization tool, genetic algorithms are nowadays widely applied to tackle a number of real-world optimization problems. In principle, genetic algorithms select good, promising individuals from the current population and generate new candidates of solutions by employing recombination and mutation.

According to the theory of design decomposition [3], the key components to the GA success include identifying, reproducing, and exchanging the structure of the solutions. Recombination, one of the main GA operator, mixes the promising sub-solutions, called *building blocks* (BBs), and creates new solutions. Genetic algorithms therefore work very well for the problems which can be somehow decomposed into sub-problems. However, the problem-independent recombination operator with fixed chromosome representations often breaks building blocks and results in ineffective mixing. It is the reason when traditional genetic algorithms meet complex solution structures which consist of a group of related genes, they oftentimes fail to effectively identify and efficiently exchange the building blocks to create good final solutions [4].

In order to appropriately mix genes, the evolutionary algorithms based on utilizing probabilistic models were proposed and developed [5, 6]. In such schemes, the offspring population is generated according to the estimated probabilistic model of the parent population instead of using regular recombination and mutation operators. The probabilistic model is expected to reflect the problem structure, and better performance can be achieved via exploring and exploiting the relationship between genes. These evolutionary algorithms are called probabilistic model building genetic algorithms (PMBGAs) or estimation of distribution algorithms (EDAs) [5, 6].

In PMBGAs, decision variables are often coded with binary coding schemes. It is reportedly difficult to find high accuracy solutions in solving continuous problems for PMBGAs. Moreover, many real-world engineering problems are real-parameter optimization problems, such as structural optimization problems and the design of transonic wings of aircrafts. In the literature, several attempts to apply PMBGAs to problems in the continuous domain have been made, including continuous PBIL with Gaussian distribution [7], real-coded variant of PBIL with interval updating [8], BEA for continuous function optimization [9], and the real-coded BOA [10].

However, these approaches require the knowledge of and are clearly specialized for the modified algorithms. In order to provide a good, general interface between problems of continuous variables and algorithms for discrete variables, in this paper, we propose a framework that enables the PMBGAs designed for handling bit-strings to tackle real-valued optimization problems. Particularly, we develop a new, adaptive discretization encoding scheme that can be easily integrated into PMBGAs or other algorithms for discrete variables, and we use the extended compact genetic algorithm (ECGA) [11] as an illustrative example in the present work.

In next section, we will first briefly introduce ECGA. In section 3, we will describe in detail how the proposed *Split-on-Demand* (SoD) encodes the solutions of real values into discrete numbers. In section 4, we use SoD to enable ECGA to handle real-valued decision variables, and the numerical experiments on benchmark functions are presented in section 5, followed by handling the economic dispatch problem in section 6. Finally, section 7 concludes this work.

2 Extended Compact Genetic Algorithm

The extended compact genetic algorithm (ECGA), which was proposed by Harik [11] based on the idea that probability distributions can be used to model the population in genetic algorithms and the choice of a good probability distribution can be viewed as equivalent to learning linkage between decision variables. The probabilistic models adopted in ECGA are a class of models known as the marginal product models (MPMs). ECGA utilizes MPMs to model partitions of decision variables. The measurement of distribution quality is quantified according to the minimum description length (MDL) principle [12], which can be considered as a realization of *Occam's razor*. The essential concept of MDL is that all things being equal, simpler distributions are preferred to more complex ones. The MDL criterion penalizes both inaccuracy as well as complexity and therefore, leads to providing high quality probability distributions.

ECGA can be algorithmically outlined as

1. Initialize a population of size N at random.
2. Apply tournament selection of size S.
3. Model the population by using a greedy MPM search.
4. Stop if the MPM model has converged.
5. Generate a new population with the MPM model.
6. Return to step 2.

The complexity measurement of the MPM model 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 for subset I. According to the MDL principle, the goal for the MPM search is to find an MPM model with the minimal combined complexity as

$$\text{Combined Complexity} = \text{Model Complexity} + \text{Compressed Population Complexity} \ .$$

Instead of applying traditional crossover and mutation, ECGA creates the new population according to the MPM model obtained in step 3. By doing such an operation, offspring individuals are created without destroying building blocks represented in the form of linkage groups of decision variables. In the original framework, ECGA can handle only binary variables. In order to make ECGA capable of tackling the real-parameter optimization problems, certain technique is required to play as an interface between the method for optimization and the problem to solve. In the present work, we adopt an adaptive discretization method, called split-on-demand, described in the next section.

3 Split-on-Demand

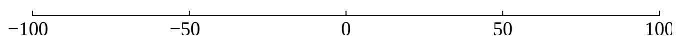
In this section, we present an adaptive discretization method, called *Split-on-Demand* (SoD), which encodes real-number decision variables into discrete numerical codes. The main idea of SoD is to encode with more integer codes those regions that we are more interested in and demand to know more about. For this purpose, SoD splits the real-number intervals in which there are equal to or more than a specified number of individuals. Thus, more accurate probabilistic models regarding these regions may be built, while less computational resource may be spent on modeling regions of fewer individuals. A *split rate* γ, where $0 < \gamma < 1$, is employed to determine whether or not a real-number interval should be split. If the population size is N, an interval containing more than $N \times \gamma$ individuals should be split. By adjusting γ, we can control the precision of the probabilistic model (of discrete variables) that we want to build to depict the population as well as avoid having unnecessarily long bit-strings for discretization. Thanks to the behavior of splitting the intervals, we call the proposed encoding scheme *Split-on-Demand*.

As described, SoD splits a dimension of real numbers into several intervals and gives each of them an integer code. We can then translate a vector of real numbers to a vector of integers, which can be represented by bits or binary codes more directly. As an example, given a real-parameter optimization problem of two dimensions, one possible code table constructed by SoD is shown in Figure 1. According to the code table, the solution $[-72.3, 24.8]$ is encoded as $[0, 1]$, and the solution $[13.8, -5.3]$ as $[2, 0]$. Figure 2 shows the solution space split by the code table given in Figure 1 as an illustration. Figure 2(a) is the split configuration on dimension 1, Figure 2(b) is the split configuration on dimension 2, and Figure 2(c) is the combined split configuration on [dimension 1, dimension 2], which is the whole solution space. The code table splits the solution space into 12 regions.

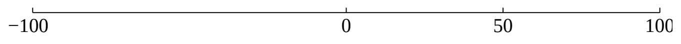
After describing the usage of the SoD code table, we now discuss the way to construct it. The principle of the proposed encoding scheme is to split the real number interval in which there are a lot of search points. Because the tournament selection operator is applied to choose the

Dimension 1		Dimension 2	
Interval	Code	Interval	Code
-100 $\sim$ -50	0	-100 $\sim$ 0	0
-50 $\sim$ 0	1	0 $\sim$ 50	1
0 $\sim$ 50	2	50 $\sim$ 100	2
50 $\sim$ 100	3		

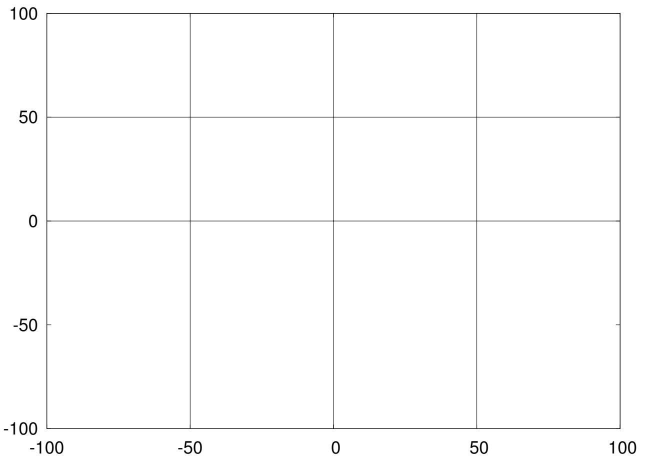
Figure 1: An example code table constructed by Split-on-Demand for a real-parameter optimization problem of two dimensions.



(a) Split configuration on dimension 1.



(b) Split configuration on dimension 2.



(c) Combined split configuration on both dimensions.

Figure 2: An illustration of the solution space being split according to the code table given in Figure 1.

promising individuals at each generation, if there are a host of individuals in certain region after selection, we consider that region important and believe the probability to find good solutions in that region is higher. Therefore, we split the promising region to gain higher resolution as well as achieve better accuracy to assist the PMBGA to build high quality probabilistic models.

In order to determine which real number interval to split, as previously mentioned, we employ a *split rate* γ, where $0 < \gamma < 1$. Assume that the population size is N, if an interval contains more than $N \times \gamma$ individuals, the interval should be split. By adjusting the split rate, we can control the accuracy of the probabilistic model which we want to build. If more accurate probabilistic models are necessary, smaller split rates should be used such that the value range of the decision variable is split to more intervals. Furthermore, for the same reason, the split rate can also be used to control the overall code length. The higher the split rate, the shorter the code length, and vice versa.

The procedure of Split-on-Demand can be describe as follows, and the pseudo code of SoD is shown in Figure 3. Subroutine `Split-on-Demand` first calls subroutine `Split` on the interval $[lower\_bound, upper\_bound]$, where the $lower\_bound$ and $upper\_bound$ are the bounds of this dimension. `Split` generates a random number m in the interval in question and counts the individuals in the two intervals: $[lower\_bound, m]$ and $[m, upper\_bound]$. If an interval contains

```
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
        Add a code for the range $[\ell, m]$
    end if
    if $N_u \geq N \times \gamma$ then
        Split($m$, $u$)
    else
        Add a code for the range $[m, u]$
    end if
end procedure
```

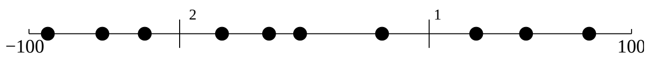
Figure 3: Pseudo code for SoD.

more than $N \times \gamma$ individuals, `Split` will be recursively called to split that interval until no interval should be further split.

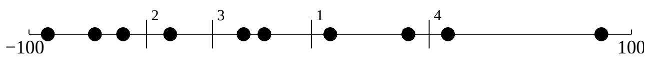
When all split operations are done, we decrease the split rate by a factor ϵ, where $0 < \epsilon < 1$. The reason to decrease the split rate is to have higher split rates at the early search stage to keep the diversity and to implement a coarse-grained, global search. As the search process goes by, we obtain more and more information about the solution space and know where to put more search points to find good solutions. Hence, at the late stage, a lower split rate is needed to build accurate probabilistic models for conducting a fine-grained, local search. The factor ϵ can be set to control the speed of convergence. An appropriate ϵ can help the search algorithm to avoid wasting time on useless regions as well as being trapped at local optima and therefore is key to an efficient search process.

We now give a typical example of how SoD runs on populations for demonstration. Assume that the population size is 10, and the initial split rate $\gamma = 0.5$. Figure 4 depicts how the individuals distributed at different generations. Initially, Figure 4(a) shows that the first position to split, marked by 1, is randomly generated. We then discover that the number of individuals in the left interval is larger than $10 \times \gamma = 5$. Under this condition, SoD calls `Split` to perform a random split on the left interval and gets the second split position, marked by 2. After the second split, the numbers of individuals in the two intervals, the left interval and the right interval to the second split position, are both less than $10 \times \gamma = 5$. As a consequence, SoD stops the split operation and decreases the split rate.

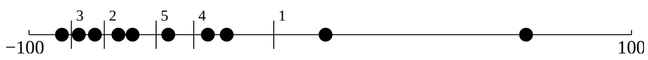
Figure 4(b) shows the population distribution and the split positions at generation 10. The split rate γ is now 0.4. Similar to the procedure described in the previous paragraph, SoD performs a random split to cut the whole interval into two intervals. It can be observed that both the left and the right intervals contain more than $10 \times \gamma = 4$ individuals, and as a result, SoD calls `Split` on both the left and the right intervals. For the left interval, SoD randomly splits it into two intervals and finds out that its right interval still contains more than 4 individuals. SoD recursively calls `Split` to split that interval. By conducting the recursive split operation



(a) Population distribution and 2 split positions at generation 1. $\gamma = 0.5$. $10 \times \gamma = 5$.



(b) Population distribution and 4 split positions at generation 10. $\gamma = 0.4$. $10 \times \gamma = 4$.



(c) Population distribution and 5 split positions at generation 20. $\gamma = 0.3$. $10 \times \gamma = 3$.

Figure 4: Population distribution and the split positions at different generations.

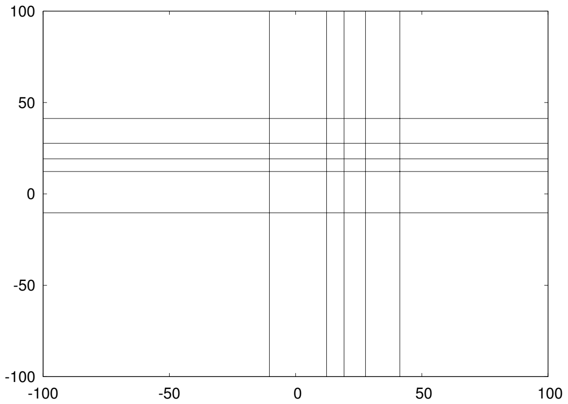
until no more interval has to be split, 4 splits make the value range 5 intervals. Moreover, in Figure 4(c), the population is at generation 20, and the split rate is 0.3. SoD runs on the population, and the interval is split into 6 regions by 5 splits.

One might wonder that the proposed encoding scheme seems similar to the marginal fixed-height histogram (FHH) introduced in [13]. In fact, there are two significant differences between SoD and FHH. The first difference is the size of the code table. In FHH, the height of the histogram is fixed, and for any population, the number of bins employed in the algorithm is fixed. However, in SoD, even with the same split rate, for different populations, SoD may generate code tables of different sizes. That is, the code table size in SoD may vary. For the other difference, the MPM model built according to the individuals encoded by SoD is not of the identical height. Such a flexibility might make the MPM model more accurate than that built according to the individuals encoded by FHH.

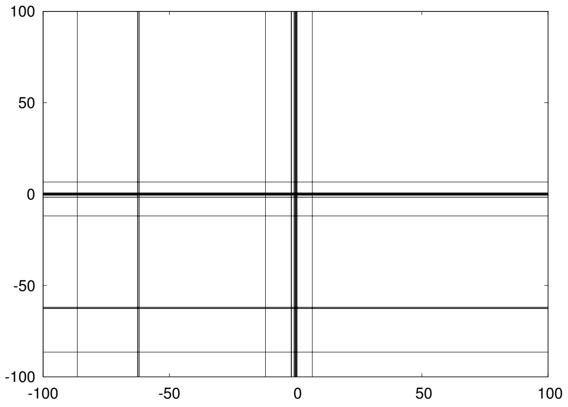
For handling the adaptive discretization during an optimization process, Figure 5 shows an example of how SoD cooperating with ECGA splits the solution space at different generations when minimizing a two-dimensional objective function $F_1 = \sum x_i^2$, where the bound of every dimension is $[-100, 100]$, and the global optimum is $(0, 0)$. Figure 5(a) depicts the split configuration on the solution space at generation 1. The split configuration seems random because the whole population is highly diverse at generation 1. Later on, at generation 50, the population begins to converge, and Figure 5(b) shows that SoD splits the solution space around $(0, 0)$ into many regions and leaves other parts of the solution space unencoded. Finally, in Figure 5(c), it can be observed that SoD focuses on the solution space close to $(0, 0)$ at generation 100. With the population converging to $(0, 0)$, ECGA is able to explore the promising solution space more thoroughly and to find the solutions of higher precision.

Another example is the two-dimensional objective function $F_2 = \sum 10 - |x_i|$. Figure 6 depicts how SoD splits the solution space, of which the bound of each dimension is $[-10, 10]$, when minimizing F_2. There are four global minima located at $(-10, -10)$, $(-10, 10)$, $(10, -10)$, and $(10, 10)$, respectively. Figure 6(a) is the split configuration on the solution space at generation 1. Because the population is initially random, the split configuration seems random. In Figure 6(b), we can observe that at generation 10, because the population begins to converge to the global minima, the split points are close to the four corners where the global minima of F_2 are located. Finally, Figure 6(c) shows that almost all split points are around the region close to $(10, 10)$ because the population converge to one of the four global optima at generation 20.

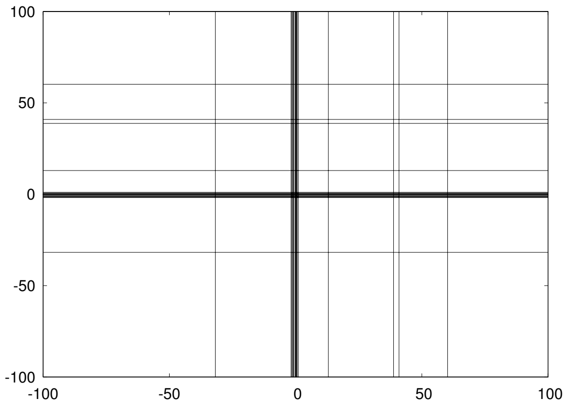
These two examples demonstrate that the split configuration established by SoD appropriately responds to the status of the population. The split configuration can encode the individuals as precise as necessary for the cooperating PMBGA to build probabilistic models. Hence, SoD is an effective encoding scheme to make PMBGAs able to tackle the real-parameter optimization



(a) Generation 1.

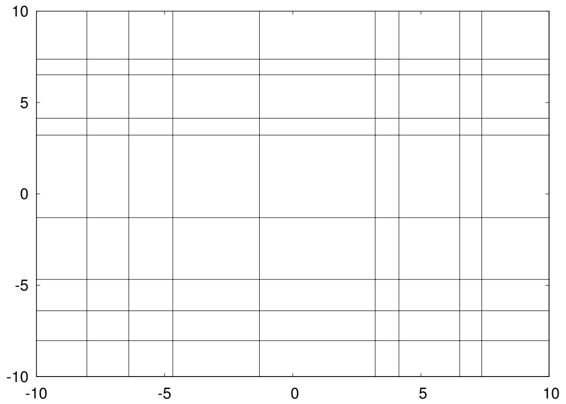


(b) Generation 50.

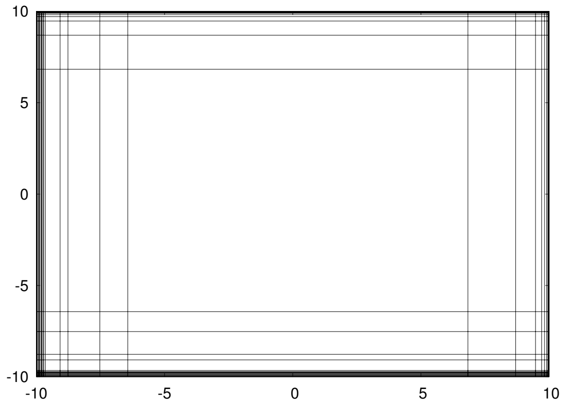


(c) Generation 100.

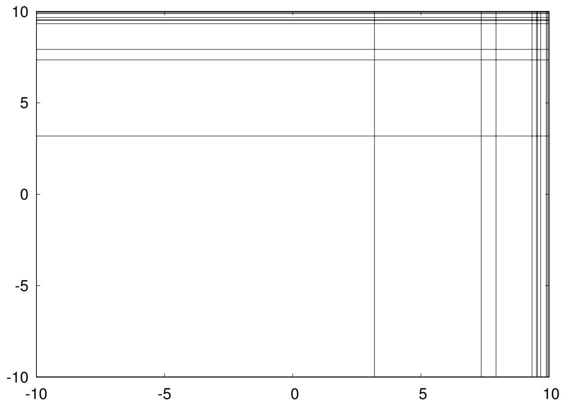
Figure 5: Split configurations at different generations for $F_1 = \sum x_i^2$.



(a) Generation 1.



(b) Generation 10.



(c) Generation 20.

Figure 6: Split configurations at different generations for $F_2 = \sum 10 - |x_i|$.

problem. In next section, ECGA, as an example of PMBGAs, will be employed to show the feasibility of integrating SoD and PMBGAs.

4 Real-Coded ECGA

In the previous sections, we proposed the adaptive discretization method, *Split-on-Demand* (SoD), described the behavior of SoD, and demonstrated the effect of SoD. In this section, we will show the way to plug SoD into ECGA, as a showcase for the integration of SoD and PMBGAs. The outcome is a new algorithm, called the *real-coded* ECGA (rECGA), for solving real-parameter optimization problems with the search powered by ECGA. rECGA can be put as:

1. Initialize a population of size N at random.
2. Apply tournament selection of size S.
3. Use SoD to encode each dimension of the variables.
4. Model the population composed of the encoded individuals by using a greedy MPM search.
5. Stop if the MPM model has converged.
6. Generate a new population using the MPM model.
7. (Optional local search) for every L generations, run the Nelder-Mead [14] method on the best 10% individuals.
8. Return to step 2.

5 Comparisons with FHH & FWH

After proposing SoD and rECGA, firstly, we would like to know how SoD performs compared to other well-known discretization methods, such as the fixed-height histogram (FHH) and the fixed-width histogram (FWH) [13]. In this section, we use the identical search engine, ECGA, as the platform and plug SoD, FHH, and FWH into ECGA to examine how well they can perform under the same condition. When using FHH and FWH, the optimization procedure, similar to rECGA, can be described as:

1. Initialize a population of size N at random.
2. Apply tournament selection of size S.
3. Use FHH or FWH to encode each dimension.
4. Model the population composed of the encoded individuals by using a greedy MPM search.
5. Stop if the MPM model has converged.
6. Generate a new population using the MPM model.
7. (Optional local search) for every L generations, run the Nelder-Mead [14] method on the best 10% individuals.
8. Return to step 2.

The following eight test functions were used to serve as a testbed for observing the relative performance of the three discretization methods:

1. Sphere function

$$F_1(x) = \sum_{i=1}^{D} x_i^2$$

2. Rosenbrock's function

$$F_2(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

3. Ackley's function

$$F_3(x) = -20 \exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$$

4. Griewanks's function

$$F_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$$

5. Weierstrass function

$$F_5(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{kmax} (a^k \cos(2\pi b^k (x_i + 0.5)))\right) - D\sum_{k=0}^{kmax} (a^k \cos(2\pi b^k \cdot 0.5)) \ ,$$
$$\text{where } a = 0.5, b = 3, kmax = 20$$

6. Rastrigin's function

$$F_6(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

7. Noncontinuous Rastrigin's function

$$F_7(x) = \sum_{i=1}^{D} (y_i^2 - 10\cos(2\pi y_i) + 10) \ ,$$
$$\text{where } y_i = \begin{cases} x_i, & \text{if } |x_i| < \frac{1}{2} \\ \frac{\text{round}(2x_i)}{2}, & \text{if } |x_i| >= \frac{1}{2} \end{cases}$$

8. Schwefel's function

$$F_8(x) = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(|x_i|^{\frac{1}{2}})$$

The global optimum positions, x^*, the global optimum objective values, $f(x^*)$, and the search intervals, $[x_{min}, x_{max}]$, of each test function are listed in Table 1. The number of dimension, D, in all the experiments is 10.

The parameters for rECGA (i.e., ECGA+SoD) we used in this series of experiments are given in the following: population size = 250, crossover probability = 1.0, tournament size = 8, L = 5, the maximum function evaluations = 30,000, and 50 independent runs for each test function. Because the code length is a constant for FHH and FWH, while it varies for SoD, in order to achieve fair comparisons, we conducted two sets of experiments to examine the effect of different code lengths on the three discretization methods. In each experiment set, we executed

F	x^*	$f(x^*)$	Search Intervals
F_1	$[0, 0, ..., 0]$	0	[-100,100]
F_2	$[0, 0, ..., 0]$	0	[-2.048,2.048]
F_3	$[0, 0, ..., 0]$	0	[-32.768,32.768]
F_4	$[0, 0, ..., 0]$	0	[-600,600]
F_5	$[0, 0, ..., 0]$	0	[-0.5,0.5]
F_6	$[0, 0, ..., 0]$	0	[-5.12,5.12]
F_7	$[0, 0, ..., 0]$	0	[-5.12,5.12]
F_8	$[0, 0, ..., 0]$	0	[-500,500]

Table 1: Global optima and search intervals of the test functions

ECGA with FHH or FWH for one code length slightly shorter as well as for one slightly longer than the mean SoD code length.

In the first set of experiments, for SoD, $\gamma = 0.7$, and $\epsilon = 0.99$. Under this condition, the mean SoD code length for each dimension is 13.23. As aforementioned, we compare the results of rECGA to that of ECGA+FHH and ECGA+FWH with 10 and 15 bins. The mean objective values, variances, and T-test results of each function are shown in Table 2. In the other set of experiments, for SoD, $\gamma = 0.45$, and $\epsilon = 0.988$. In this case, the mean SoD code length for each dimension becomes 22.81. Accordingly, we compare the results of rECGA to that of ECGA+FHH and ECGA+FWH with 20 and 25 bins. The results are given in Table 3.

According to the experimental results presented in Tables 2 and 3, we can first observe that SoD outperforms FHH or FWH on the eight test functions by comparing the obtained mean objective values. We can also see that SoD in general provides smaller variances of the solutions than FHH and FWH can. The t-values listed in the tables further indicate that the experimental results are statistically significant. Except for FHH-25 on F_7 (marked by † in Table 3), all t-values are greater than the critical value, 1.68, for one tail significance $\alpha = 0.05$ and degree of freedom around 50. As a consequence, we can conclude that SoD can better discretize the continuous search intervals than FHH and FWH can, at least when cooperating with ECGA.

6 Real-World Applications

After verifying the discretization capability of SoD, we are interested in putting the proposed framework, rECGA, into action. In this section, we employ rECGA to handle the economic dispatch (ED) problem, which is an essential topic in the power system because several important facets of power systems are involved. Thanks to the importance and significance of the ED problem, researchers have been making numerous attempts to find better solutions. Among the promising sets of evolutionary optimization methods for tackling the ED problem are genetic algorithms [15, 16, 17, 18, 19], evolutionary programming [20, 21, 22, 23], and particle swarm optimization [24, 25, 26, 27]. In order to obtain even better solutions, we will apply rECGA on the ED problem in this section. First, we will briefly introduce the ED problem, and then the high quality solutions offered by rECGA are presented.

6.1 The Problem: Economic Dispatch

With the development of modern power systems, the economic dispatch (ED) problem has been receiving an increasing attention. The ED problem is essential for the real-time control of power system operations. It consists of allocating the total generation required among the available thermal generating units, assuming that a thermal unit commitment is previously determined.

		F_1	F_2	F_3	F_4
SoD	mean	**2.84e-05**	**8.46e+00**	**1.23e-03**	**1.09e-02**
	var.	2.12e-09	5.79e+01	2.96e-07	2.44e-04
FHH	mean	6.65e+01	3.05e+01	4.32e+00	1.59e+00
10	var.	1.30e+03	1.70e+02	5.41e-01	8.12e-02
	t-value	1.30e+01	1.03e+01	4.15e+01	3.92e+01
FHH	mean	6.44e+00	1.51e+01	1.84e+00	8.71e-01
15	var.	2.52e+01	1.35e+02	3.53e-01	4.88e-02
	t-value	9.07e+00	3.40e+00	2.19e+01	2.74e+01
FWH	mean	3.56e+02	1.67e+01	8.29e+00	4.51e+00
10	var.	1.18e+04	4.64e+00	4.99e-01	6.72e-01
	t-value	2.32e+01	7.39e+00	8.30e+01	3.88e+01
FWH	mean	5.81e+01	1.06e+01	4.53e+00	1.56e+00
15	var.	6.20e+02	9.26e+00	2.47e-01	6.27e-02
	t-value	1.65e+01	1.88e+00	6.45e+01	4.37e+01

		F_5	F_6	F_7	F_8
SoD	mean	**1.03e-01**	**1.24e+00**	**4.04e+00**	**1.55e-04**
	var.	1.13e-03	1.22e+00	2.11e+00	1.03e-09
FHH	mean	1.12e+00	4.82e+00	5.88e+00	5.97e+01
10	var.	8.39e-02	4.82e+00	2.65e+00	3.59e+03
	t-value	2.48e+01	1.03e+01	5.98e+00	7.05e+00
FHH	mean	3.67e-01	3.29e+00	4.38e+00	1.12e+01
15	var.	2.90e-02	2.46e+00	1.82e+00	3.14e+02
	t-value	1.07e+01	7.58e+00	1.22e+00	4.46e+00
FWH	mean	4.86e+00	2.78e+01	2.24e+01	4.09e+02
10	var.	3.46e-01	2.91e+01	9.37e+01	1.32e+04
	t-value	5.71e+01	3.42e+01	1.33e+01	2.51e+01
FWH	mean	2.61e+00	1.76e+01	1.56e+01	4.10e+02
15	var.	5.33e-02	1.02e+01	1.35e+01	1.12e+04
	t-value	7.59e+01	3.42e+01	2.07e+01	2.73e+01

Table 2: The mean SoD code length is 13.23

The problem aims to minimize the fuel cost subject to the physical and operational constraints. As a result, the ED problem is to find the optimal combination of generations that minimizes the total cost while satisfying the specified constraints. To model the ED problem, a simplified cost function [28] of each generator, represented as a quadratic function, can be put as:

$$\begin{aligned} C &= \sum_{j \in J} F_j(P_j) \ , \\ F_j(P_j) &= a_j P_j^2 + b_j P_j + c_j \ , \end{aligned}$$

where

- C: the total generation cost;
- J: the set for all generators;
- P_j: the electrical output of generator j;
- F_j: the cost function for generator j;

		F_1	F_2	F_3	F_4
SoD	mean	**6.86e-08**	**6.99e+00**	**2.47e-04**	**1.24e-02**
	var.	4.86e-15	2.68e+00	1.27e-08	1.34e-04
FHH	mean	2.31e+00	1.13e+01	1.16e+00	5.93e-01
20	var.	5.03e+00	1.30e+02	5.24e-01	6.67e-02
	t-value	7.27e+00	2.65e+00	1.13e+01	1.59e+01
FHH	mean	1.33e+00	8.30e+00	1.23e+00	5.55e-01
25	var.	1.46e+00	5.00e+00	4.25e-01	6.95e-02
	t-value	7.81e+00	3.36e+00	1.34e+01	1.45e+01
FWH	mean	1.02e+02	1.11e+01	5.46e+00	1.94e+00
20	var.	4.69e+02	1.57e+01	2.19e-01	2.82e-02
	t-value	3.32e+01	6.74e+00	8.25e+01	8.07e+01
FWH	mean	5.42e+01	1.11e+01	4.35e+00	1.46e+00
25	var.	1.14e+03	5.14e+01	6.16e-01	1.11e-01
	t-value	1.14e+01	3.92e+00	3.92e+01	3.08e+01

		F_5	F_6	F_7	F_8
SoD	mean	**9.35e-02**	**1.32e+00**	**2.49e+00**	**1.27e-04**
	var.	1.18e-03	1.23e+00	2.00e+00	1.22e-14
FHH	mean	2.32e-01	2.19e+00	3.71e+00	3.94e+00
20	var.	2.00e-02	1.27e+00	1.34e+00	1.53e+01
	t-value	6.74e+00	3.89e+00	4.70e+00	7.13e+00
FHH	mean	1.99e-01	1.90e+00	2.71e+00	3.52e+00
25	var.	1.08e-02	1.14e+00	1.46e+00	1.41e+01
	t-value	6.83e+00	2.66e+00	†8.21e-01	6.64e+00
FWH	mean	3.34e+00	3.72e+01	1.32e+01	9.45e+01
20	var.	1.12e-01	6.27e+01	3.94e+01	1.18e+03
	t-value	6.84e+01	3.18e+01	1.18e+01	1.94e+01
FWH	mean	1.89e+00	7.97e+00	6.50e+00	1.83e+02
25	var.	4.78e-02	2.75e+00	3.98e+00	2.15e+03
	t-value	5.75e+01	2.35e+01	1.16e+01	2.78e+01

Table 3: The mean SoD code length is 22.81

- a_j, b_j, c_j: the cost coefficients for generator j.

In the real world, the total generation should be equal to the total system demand plus the transmission network loss. However, in this study, the network loss is not considered for simplicity as in many studies. Thus the constraints of the problem include two parts. The first part is the equality constraint. The total system power demand must be equal to the summation of the output of each generator:

$$D = \sum_{j \in J} P_j \ , \tag{3}$$

where D is the total system demand.

Moreover, the generation output of each unit should be within its minimum and maximum limits. Such a condition introduces the inequality constraint for each generation unit, such as for generator j:

$$P_{jmin} \leq P_j \leq P_{jmax} \ ,$$

where P_{jmin} and P_{jmax} are the minimum and maximum output of generator j, and P_j is the desired output.

In reality, the objective function of economic dispatch problem is more complicated due to the valve-point effects, the change of fuels, and other potential practical factors. Therefore, nonsmooth cost functions should be taken into consideration instead of the simplest form as Equation (3). The inclusion of the valve-point effects makes the modeling of the incremental fuel cost function of the generation units more practical. Such a modification increases the nonlinearity as well as the number of local optima in the search space. Hence, the employed search algorithm may be trapped in the local optimal more easily. The incremental fuel cost function of the generator with the valve-point effects can be put as [15]:

$$F_j(P_j) = a_j P_j^2 + b_j P_j + c_j + |e_j \sin(f_j \times (P_{jmin} - P_j))| \ , \tag{4}$$

where e_j and f_j are the coefficients for generator j to reflect the valve-point loading effects.

In this study, we focus on solving the ED problem with the valve-point loading effects, which is modeled as Equation (4). We applied the proposed method as an optimization tool. The equality and inequality constraints are handled through repair. Detail descriptions are given in the following section.

6.2 Our Solution: rECGA for Economic Dispatch

The integration of ECGA and SoD, rECGA, can generally handle global optimization problems. However, certain extra efforts have to be made to employ rECGA to tackle the ED problem. Among the most important topics for solving the ED problem may be the equality and inequality constraints. These constraints divide the problem search space into complicated areas. This condition renders the local search mechanism inefficient and ineffective. As a result, we did not use the optional local search operator when handling the ED problem.

Furthermore, in order to deal with the constraints, based on the repair concept, we developed a constraint handling technique specifically for the ED problem. Repairing solutions stands for transforming infeasible solutions into feasible one through certain procedure. For the equality constraint (Equation (3)) in the ED problem, we repair infeasible solutions in the following way: First, we create a sequence from 1 to the number of generator in a random order. Each number in the sequence represents one designated generator. The sequence indicates the order in which we adjust the output of the generation for making the solution feasible. For example, if the generated sequence is 4, 2, 1, 5, 3, we will firstly process generator 4, then generator 2, and so on. To process a generator, we check the equality constraint, i.e., the sum of the output has to be equal to the total power demand. If the equality constraint is not satisfied, the output of the generator under processing is modified according to:

$$P_i' = \min(\text{UBound}(P_i), \max((D - \sum_{j=1, j\neq i}^{n} P_j), \text{LBound}(P_i))) \ , \tag{5}$$

where D is the system power demand, LBound(P_i) and UBound(P_i) are the lower bound and upper bound of P_i, i.e., the inequality constraint of P_i.

The proposed algorithm, rECGA, incorporating the constraint handling technique is capable of tackling the ED problem effectively. With the adoption of the proposed repair mechanism, rECGA for solving the ED problem can be outlined as:

1. Initialize a population of size N at random according to the constraints posed to the generator output.

2. Apply tournament selection of size S.

3. Use SoD to encode each dimension of the variables.

Generator	P_{min}(MW)	P_{max}(MW)	a	b	c	e	f
1	100	600	0.001562	7.92	561	300	0.0315
2	100	400	0.00482	7.97	78	150	0.063
3	50	200	0.00194	7.85	310	200	0.042

Table 4: Parameters for test case I (3-unit system) with the valve-point loading effect. a, b, c, e, and f are the cost coefficients in the fuel cost function: $F_j(P_j) = a_jP_j^2 + b_jP_j + c_j + |e_j \sin(f_j \times (P_{jmin} - P_j))|$.

4. Model the population composed of the encoded individuals by using a greedy MPM search.
5. Stop if the MPM model has converged.
6. Generate a new population with the MPM model.
7. Repair the infeasible individuals in the population.
8. Return to step 2.

In order to observe the effectiveness and to verify the performance of rECGA for ED, two ED problem instances, one consisting of 3 generators and the other consisting of 40 generators, are served as a testbed in the study. The experimental results on the two ED problems are presented in the next section.

6.3 Verification: Numerical Experiments

In this paper, we focus on solving the ED problem with nonsmooth cost functions considering the valve-point effects for verifying the utility of the proposed framework, rECGA. The nonsmooth cost functions were described as Equation (4). In order to examine the performance, rECGA for ED was applied to two ED problems which were adopted as test problems in the literature [15, 21] for the comparison purpose. One consists of 3 generation units, and the other consists of 40 generation units. The input data for the 3-generator system are given by Walters and Sheble [15], and those for the 40-generator system are given by Shinha et al. [21]. The detailed problem parameters for the two test problems, including the lower bound and upper bound for the output of each generator as well as the coefficients for computing the cost functions, are given in Tables 4 and 5. The total system demand for the 3-unit system is 850MW, and that for the 40-unit system is 10500MW. It has been proven that for the 3-unit system, the global optimal solution is 8234.07 [29]. As for the 40-unit system, the global optimal solution has not been determined. To the best of our limited knowledge, the known best solution reported in the literature is 122252.265 [26].

The parameter settings in rECGA for ED are that population size = 400, crossover probability = 0.975, tournament size = 8, $\gamma = 0.5$, $\epsilon = 0.999$, and the maximum function evaluations is 200000. 100 independent trails were conducted for each problem to collect statistically significant results. The obtained results for the 3-unit system are shown in Table 6 and are compared to those obtained by IEP [30], EP [22], and MPSO [26]. The results for the small ED problem indicate that rECGA was able to find the global optimal solution presented by Lin et al. [29].

In the case of the 40-unit system, the results are compared with those obtained by using other methods given in [21] such as classical EP (CEP), fast EP (FEP), modified FEP (MEFP), and improved FEP (IFEP) as well as those obtained by using MPSO in [26]. The minimum costs, i.e., the best solutions, achieved by each method are presented in Table 7. We can see that the best solution offered by rECGA is 121462.3591, which is better than the known best solution, 122252.265, presented in [26]. For the purpose of access and verification, the generation

Generator	P_{min}(MW)	P_{max}(MW)	a	b	c	e	f
1	36	114	0.0069	6.73	94.705	100	0.084
2	36	114	0.0069	6.73	94.705	100	0.084
3	60	120	0.2028	7.07	309.54	100	0.084
4	80	190	0.00942	8.18	369.03	150	0.063
5	47	97	0.0114	5.35	148.89	120	0.077
6	68	140	0.01142	8.05	222.33	100	0.084
7	110	300	0.00357	8.03	287.71	200	0.042
8	135	300	0.00492	6.99	391.98	200	0.042
9	135	300	0.00573	6.6	455.76	200	0.042
10	130	300	0.00605	12.9	722.82	200	0.042
11	94	375	0.00515	12.9	635.2	200	0.042
12	94	375	0.00569	12.8	654.69	200	0.042
13	125	500	0.00421	12.5	913.4	300	0.035
14	125	500	0.00752	8.84	1760.4	300	0.035
15	125	500	0.00708	9.15	1728.3	300	0.035
16	125	500	0.00708	9.15	1728.3	300	0.035
17	220	500	0.00313	7.97	647.85	300	0.035
18	220	500	0.00313	7.95	649.69	300	0.035
19	242	550	0.00313	7.97	647.83	300	0.035
20	242	550	0.00313	7.97	647.81	300	0.035
21	254	550	0.00298	6.63	785.96	300	0.035
22	254	550	0.00298	6.63	785.96	300	0.035
23	254	550	0.00284	6.66	794.53	300	0.035
24	254	550	0.00284	6.66	794.53	300	0.035
25	254	550	0.00277	7.1	801.32	300	0.035
26	254	550	0.00277	7.1	801.32	300	0.035
27	10	150	0.52124	3.33	1055.1	120	0.077
28	10	150	0.52124	3.33	1055.1	120	0.077
29	10	150	0.52124	3.33	1055.1	120	0.077
30	47	97	0.0114	5.35	148.89	120	0.077
31	60	190	0.0016	6.43	222.92	150	0.063
32	60	190	0.0016	6.43	222.92	150	0.063
33	60	190	0.0016	6.43	222.92	150	0.063
34	90	200	0.0001	8.95	107.87	200	0.042
35	90	200	0.0001	8.62	116.58	200	0.042
36	90	200	0.0001	8.62	116.58	200	0.042
37	25	110	0.0161	5.88	307.45	80	0.098
38	25	110	0.0161	5.88	307.45	80	0.098
39	25	110	0.0161	5.88	307.45	80	0.098
40	242	550	0.00313	7.97	647.83	300	0.035

Table 5: Parameters for test case II (40-unit system) with the valve-point loading effect. a, b, c, e, and f are the cost coefficients in the fuel cost function: $F_j(P_j) = a_j P_j^2 + b_j P_j + c_j + |e_j \sin(f_j \times (P_{jmin} - P_j))|$.

outputs (the values of the decision variables) and the corresponding cost (the objective values) of the best solution offered by rECGA are given in Table 8.

Generator	GA	IEP (pop=20)	EP	MPSO (par=20)	rECGA
1	300	300.23	300.26	300.27	300.267
2	400	400	400	400	400
3	150	149.77	149.74	149.73	149.733
TP	850	850	850	850	850
TC	8237.6	8234.09	**8234.07**	**8234.07**	**8234.07**

Table 6: Comparison of the experimental results obtained by various methods on the nonsmooth cost function considering the valve-point loading effect. For the 3-unit system, EP, MPSO, and rECGA were able to find the global optimum [29].

	CEP	FEP	MFEP	IFEP	MPSO	rECGA
Minimum Cost	123488.3	122679.7	122647.6	122624.35	122252.265	**121462.3591**

Table 7: Comparison of the experimental results obtained by various methods on the nonsmooth cost function considering the valve-point loading effect. For the 40-unit system, rECGA was able to find the best solution.

Because of the stochastic nature of evolutionary computation methods, to avoid reporting the results of a “lucky shot”, comparison of the experimental results in a statistical manner has to be conducted. First of all, Table 9 shows the range of the results in the 100 trials obtained by CEP, FEP, MFEP, IFEP, MPSO, and rECGA, where the listed results except for those of rECGA are given in [21, 26]. As we can observe in Table 9, the distribution of the rECGA results may be considered better than those for the other evolutionary algorithms.

Furthermore, to more carefully and accurately compare the performance of rECGA and MPSO [26] on the 40-unit problem, the t-test was conducted for the statistical significance of the obtained experimental results. Since the actual results of the 100 trials for MPSO is not available, in order to get a fair performance comparison and capability assessment, we set up two conditions under which the t-test was conducted. Based on the data given in Table 9, the first condition is that the MPSO results contain forty-seven 122252.265, which is the optimum reported for MPSO [26], and fifty-three 122750, which is the mean value of 122500 and 123000. Table 10 demonstrates the t-test results for condition 1. Given the p-value: 2.26×10^{-55}, which is smaller than the commonly used statistical significant levels, such as 0.05 (5%), 0.01 (1%), or 0.001 (0.1%), we can conclude that the performance of rECGA on the 40-unit ED problem is statistically significantly better than that of MPSO on the same problem. For condition 2, the MPSO results contain forty-seven 122252.265, which is the optimum reported for MPSO [26], and fifty-three 122500, which is the best value in the range from 122500 to 123000. The t-test results under condition 2 are presented in Table 11. Due to the change of the standard deviation, the p-value becomes 9.09×10^{-91}. The small p-value prevents us from accepting the null hypothesis, which is interpreted as that the performance of rECGA and MPSO on the problem is equivalent.

According to the experimental results, we can known that the proposed algorithm, rECGA = ECGA with SoD, performs well on the two ED problems. Particularly, for the 40-unit ED problem, we improved the known best solution from 122252.265 [26] to 121462.3591. Moreover, from Tables 9, 10, and 11, we can observe that rECGA statistically significantly outperformed MPSO on the 40-unit ED problem. Therefore, rECGA is able to solve ED problems effectively.

Generator	P_{min}(MW)	P_{max}(MW)	Output	Cost
1	36	114	110.80098	925.11565
2	36	114	110.88806	926.56631
3	60	120	97.40449	1190.63739
4	80	190	179.73300	2143.55011
5	47	97	96.15215	840.66343
6	68	140	140.00000	1596.46432
7	110	300	299.99898	3216.41474
8	135	300	284.62219	2780.24662
9	135	300	284.61234	2798.46198
10	130	300	130.00001	2502.06532
11	94	375	94.00003	1893.30606
12	94	375	94.00027	1908.17291
13	125	500	214.76169	3792.11715
14	125	500	394.27878	6414.85790
15	125	500	304.52026	5171.21428
16	125	500	394.28449	6436.71537
17	220	500	489.27966	5296.71703
18	220	500	489.27855	5288.76474
19	242	550	511.27996	5540.94200
20	242	550	511.28163	5540.95823
21	254	550	523.28030	5071.30855
22	254	550	523.28419	5071.38735
23	254	550	523.28495	5057.33548
24	254	550	523.28151	5057.26621
25	254	550	523.28214	5275.14526
26	254	550	523.27977	5275.09678
27	10	150	10.00013	1140.52698
28	10	150	10.00517	1140.64280
29	10	150	10.00018	1140.52812
30	47	97	87.84287	707.21302
31	60	190	189.99927	1643.98840
32	60	190	189.99996	1643.99109
33	60	190	189.99993	1643.99098
34	90	200	199.99994	2101.01644
35	90	200	199.99993	2043.72638
36	90	200	199.99972	2043.72436
37	25	110	110.00000	1220.16612
38	25	110	109.99978	1220.16484
39	25	110	109.99871	1220.15859
40	242	550	511.28401	5541.02984
Total Generation & Total Cost			10500	121462.3591

Table 8: The generator outputs and the corresponding costs of the best solution obtained by rECGA.

	Range of Cost											
Method	127.0 - 126.5	126.5 - 126.0	126.0 - 125.5	125.5 - 125.0	125.0 - 124.5	124.5 - 124.0	124.0 - 123.5	123.5 - 123.0	123.0 - 122.5	122.5 - 122.0	122.0 - 121.5	121.5 - 121.0
CEP	10	4	-	16	22	42	4	2	-	-	-	-
FEP	6	-	4	2	10	20	26	24	6	-	-	-
MFEP	-	-	-	-	-	14	26	50	10	-	-	-
IFEP	-	-	2	-	4	4	18	50	22	-	-	-
MPSO	-	-	-	-	-	-	-	-	53	47	-	-
rECGA	-	-	-	-	-	-	-	-	-	2	97	1

Table 9: Comparison of methods on relative frequency of convergence in the ranges of cost.

	rECGA	MPSO
mean	121777.649963	122516.06455
t-value	27.8068829451749	
p-value	2.2645299161711E-55	

Table 10: The t-test for the experimental results obtained by rECGA and MPSO under condition 1, where the rECGA data set contains the actual results, and the MPSO data set contains forty-seven 122252.265 and fifty-three 122750.

	rECGA	MPSO
mean	121777.649963	122383.56455
t-value	39.4214198098397	
p-value	9.0857670116394E-91	

Table 11: The t-test for the experimental results obtained by rECGA and MPSO under condition 2, where the rECGA data set contains the actual results, and the MPSO data set contains forty-seven 122252.265 and fifty-three 122500.

7 Summary and Conclusions

In this study, we proposed an adaptive discretization method, called *split-on-demand* (SoD), to enable the probabilistic model building genetic algorithms (PMBGAs) designed for handling discrete variables for real-parameter optimization. SoD was described in detail with its procedure, effect, and usage. For showing the utility of SoD, the extended compact genetic algorithm (ECGA) was employed as an optimization engine, and SoD was used as a variable-type interface. By combing ECGA and SoD, the real-coded ECGA (rECGA) were applied to a set of benchmark functions and two economic dispatch (ED) problems. The results on benchmark functions indicated that SoD was better than two well-known discretization methods: the fixed-height histogram (FHH) and the fixed-width histogram (FWH). The results on the ED problems demonstrated that rECGA successfully achieved the global optimal solution of the 3-unit ED problem and was able to obtain the solutions better than the known best solution reported in the literature for the 40-unit ED problem.

The outcome of this study indicates that it is not only possible but also practical to employ an optimization method designed for handling discrete variables to tackle problems consisting of continuous variables, as long as an appropriate interface is adopted. Although many researchers in the EC field do not consider the variable-type transformation as an issue, in practice, except for some limited cases, most algorithms designed for discrete variables do not perform well on continuous problems and vice versa. By comparing the real-coded ECGA to the algorithms specifically designed for handling continuous variables, such as particle swarm optimization (MPSO) and evolutionary programming (IFEP, MFEP, FEP, CEP), this paper provides the experimental results to serve as the proof of principle for transforming the variable type while retaining the capability of the optimization algorithm.

Finally, the future work of this study includes applying rECGA to other important problems as well as developing different integrations of optimization algorithms and variable-type transforming techniques. Moreover, theoretical understandings for the quality of the transforming techniques, such as SoD, FHH, and FWH, as well as for the interaction between the engine and the interface should also be considered.

Acknowledgments

The work was partially sponsored by the National Science Council of Taiwan under grants NSC-95-2221-E-009-092, NSC-95-2627-B-009-001, and NSC-95-2815-C-009-033-E as well as by the MOE ATU Program. 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, *Genetic Algorithms in Search, Optimization and Machine Learning.* New York: Addison-Wesley, 1989.

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

[4] D. E. Goldberg, B. Korb, and K. Deb, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex Systems*, vol. 3, no. 5, pp. 493–530, 1989.

[5] 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.

[6] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[7] 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.

[8] 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.

[9] S.-Y. Shin and B.-T. Zhang, "Bayesian evolutionary algorithms for continuous function optimization," in *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, 2001, pp. 508–515.

[10] 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 (GECCO-2004)*, 2004, pp. 840–851.

[11] G. R. Harik, "Linkage learning via probabilistic modeling in the ECGA," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 99010, 1999.

[12] J. Rissanen, *Stochastic Complexity in Statistical Inquiry.* World Scientific, November 1989, ISBN: 9971508591.

[13] S. Tsutsui, M. Pelikan, and D. E. Goldberg, "Evolutionary algorithm using marginal histogram models in continuous domain," University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, IlliGAL Report No. 2001019, 2001.

[14] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–315, 1965.

[15] D. C. Walters and G. B. Sheble, "Genetic algorithm solution of economic dispatch with valve point loading," *IEEE Transaction on Power Systems*, vol. 8, no. 3, pp. 1325–1332, August 1993.

[16] G. B. Sheble and K. Brittig, "Refined genetic algorithm - economic-dispatch example," *IEEE Transactions on Power System*, vol. 10, no. 1, pp. 117–124, 1995.

[17] P. H. Chen and H. C. Chang, "Large-scale economic-dispatch by genetic algorithm," *IEEE Transactions on Power System*, vol. 10, no. 4, pp. 1919–1926, 1995.

[18] S. Baskar, P. Subbaraj, and M. V. C. Rao, "Hybrid real coded genetic algorithm solution to economic dispatch problem," *Computers and Electrical Engineering*, vol. 29, no. 3, pp. 407–419, 2003.

[19] T. Yalcinoz, H. Altun, and M. Uzam, "Economic dispatch solution using a genetic algorithm based on arithmetic crossover," in *IEEE Power Tech Conference*, 2001.

[20] T. Jayabarathia, K. Jayaprakasha, D. N. Jeyakumarb, and T. Raghunathan, "Evolutionary programming techniques for different kinds of economic dispatch problems," *Electric Power Systems Research*, vol. 73, no. 2, pp. 169–176, 2005.

[21] N. Shinha, R. Chakrabarti, and P. K. Chattopadhyay, "Evolutionary programming technique for economic load dispatch," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 83–94, 2003.

[22] H. T. Yang, P. C. Yang, and C. L. Huang, "Evolutionary programming based economic dispatch for units with non-smooth fuel cost functions," *IEEE Transactions on Power System*, vol. 11, no. 1, pp. 112–117, 1996.

[23] K. P. Wong and J. Yuryevich, "Evolutionary-programming-based algorithm for environmentally-constrained economic dispatch," *IEEE Transactions on Power System*, vol. 13, no. 2, pp. 301–306, 1998.

[24] Z. L. Gaing, "Particle swarm optimization to solving the economic dispatch considering the generator constraints," *IEEE Transactions on Power System*, vol. 18, no. 3, pp. 1187–1195, 2003.

[25] T. A. A. Victoire and A. E. Jeyakumar, "Hybrid pso-sqp for economic dispatch with valve-point effect," *Electric Power Systems Research*, vol. 71, no. 1, pp. 51–59, 2004.

[26] J. B. Park, K. S. Lee, J. R. Shin, and K. Y. Lee, “A particle swarm optimization for economic dispatch with nonsmooth cost functions,” *IEEE Transactions on Power System*, vol. 20, no. 1, pp. 34–42, 2005.

[27] T. A. A. Victoire and A. E. Jeyakumar, “Particle swarm optimization to solving the economic dispatch considering the generator constraints,” *IEEE Transactions on Power System*, vol. 19, no. 4, pp. 2121–2122, 2004.

[28] J. W. Allen and F. W. Bruce, *Power Generation, Operation, and Control.* New York: Wiley, 1984.

[29] W. M. Lin, F. S. Cheng, and M. T. Tsay, “An improved tabu search for economic dispatch with multiple minima,” *IEEE Transactions on Power System*, vol. 17, no. 1, pp. 108–112, 2002.

[30] Y.-M. Park, J. R. Won, and J. B. Park, “New approach to economic load dispatch based on improved evolutionary programming,” *Eng. Intell. Syst. Elect. Eng. Commun*, vol. 6, no. 2, pp. 103–110, June 1998.