Efficient Differential Evolution using Speciation for Multimodal Function Optimization

Xiaodong Li
School of Computer Science and IT
RMIT University
Melbourne, Australia
xiaodong@cs.rmit.edu.au

ABSTRACT

In this paper differential evolution is extended by using the notion of speciation for solving multimodal optimization problems. The proposed species-based DE (SDE) is able to locate multiple global optima simultaneously through adaptive formation of multiple species (or subpopulations) in an DE population at each iteration step. Each species functions as an DE by itself. Successive local improvements through species formation can eventually transform into global improvements in identifying multiple global optima. In this study the performance of SDE is compared with another recently proposed DE variant CrowdingDE. The computational complexity of SDE, the effect of population size and species radius on SDE are investigated. SDE is found to be more computationally efficient than CrowdingDE over a number of benchmark multimodal test functions.

Categories and Subject Descriptors

G.1 [**Numerical Analysis**]: Optimization; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms

Keywords

Evolutionary Computation, Differential Evolution, Multimodal Function Optimization

1. INTRODUCTION

Multimodal function optimization has been the subject of intense study for many years within the Evolutionary Computation research community. To achieve multimodal optimization, an evolutionary algorithm is required to locate all the global optima (or multiple optima including global and some second best optima) in the search space instead of just a single global optimum. Some classical techniques that can enhance an EA with this ability include crowding [7, 10], fitness sharing [4], derating method [1], restricted tournament selection [5], parallelization [2] and speciation [12, 8].

Differential evolution is a relatively new optimization technique compared with other more established EAs such as Genetic Algorithms, Evolutionary Strategy, and Genetic Programming. Similar to EAs, DE is population-based, but unlike EAs, DE modifies individuals via the use of the differences of randomly sampled pairs of individual vectors from the population. Since the distribution of the differences of these randomly sampled individual pairs also reflects the topographic feature of the fitness landscape of an objective function, DE is self-adaptive to the fitness landscape in its search for the global optimum. DE has proven to be a fast and effective global optimizer [13]. However the basic DE algorithm first proposed by Storn and Price [14] was primarily designed to search for a single global optimum. It is not surprising this basic DE is not suitable for multimodal optimization where multiple global optima must all be located [8, 15]. Some recent works have been made to extend DE to handle multimodal optimization problems. For example, MMDE (Multiresolution multipopulation DE) by Zaharie [16] and MultiDE by Hendershot [6] which both adopted an 'island model' approach similar to that of a traditional coarse-grained parallel GA. An interesting work done by Thomsen proposed to extend DE with a crowding scheme (CrowdingDE) to allow it to tackle multimodal optimization [15]. Thomsen showed that CrowdingDE (with the crowding factor set equal to the population size) outperformed a DE variant with a fitness sharing scheme. In this paper, a new DE algorithm based on the notion of speciation is proposed to handle multimodal optimization problems. This species-based DE (SDE) is shown to be more computationally efficient than CrowdingDE, particularly when there is a large number of global optima present hence requiring a large population size.

The remainder of the paper is structured as follows. Section 2 first describes the basics on Differential Evolution, the concept of crowding and the recently proposed CrowdingDE. Section 3 presents the proposed species-based DE, including the definition of speciation, how to determine species seeds and an analysis on the time complexity of the algorithm for determining species seeds. Detail of the SDE algorithm is also presented in Section 3. Section 4 proposes the performance measurement used in this study. Section 5 provides

GECCO'05, June 25–29, 2005, Washington, DC, USA.

numerical results over some test functions. Finally Section 6 gives the conclusions.

2. BACKGROUND

2.1 Differential Evolution

The basic differential evolution algorithm was described by Storn and Price in [14, 13]. Let's consider a maximization problem: for a function $f : X \rightarrow Y$, we need to find $\vec{x^*} \in X$ such that $\forall \vec{x} \in X, f(\vec{x^*}) \geq f(\vec{x})$. We let $\vec{x}^{(i,t)} = (x_1^{(i,t)}, x_2^{(i,t)}, \ldots, x_d^{(i,t)})$ represent the i-th variable vector (of d-dimensional) at the t-th iteration. Let's denote $P^t = \{x^{(1,t)}, x^{(2,t)}, \ldots, x^{(n,t)}\}$ the current population of size n ($n \geq 4$), and the offspring population will be $P^{(t+1)} = \{x^{(1,t+1)}, x^{(2,t+1)}, \ldots, x^{(n,t+1)}\}$. In the DE initial population, each vector $\vec{x}$ is generated by sampling along each dimension of the variable vector a random value uniformly between the lower and upper bounds of the variable range. An offspring $\vec{x}^{(i,t+1)} = (x_1^{(i,t+1)}, x_2^{(i,t+1)}, \ldots, x_d^{(i,t+1)})$ is then generated after initialization, according to the following procedure shown in Fig. 1.

Randomly select parents
$r_1, r_2, r_3 \in \{1, 2, \ldots, n | r_1 \neq r_2 \neq r_3 \neq i\}$;
$j_{rand} = int(U[0, 1] \cdot d) + 1$;
for *j=1* **to** *d* **do**
  if *U[0,1] < CR* $\vee$ *j=*j_{rand} **then**
  $x_j^{(i,t+1)} = x_j^{(r3,t)} + F \cdot (x_j^{(r1,t)} - x_j^{(r2,t)})$;
  else $x_j^{(i,t+1)} = x_j^{(i,t)}$
end

Figure 1: The procedure for generating an offspring in DE.

The above procedure is applied to all individuals of the current population $P^{(t)}$ to generate n new individuals in $P^{(t+1)}$ for the next iteration. The population size n must be greater than 3. CR and F are user-specified control parameters, ranging from [0,1) and (0, 1+) respectively. DE uses a simple replacement scheme in which a parent $\vec{x}^{(i,t)}$ is only replaced by its offspring $\vec{x}^{(i,t+1)}$ if $\vec{x}^{(i,t+1)}$ is fitter than $\vec{x}^{(i,t)}$. For more detailed information, refer to [13]. There are a number of DE variants in the literature. In this paper, we use *DE/rand/1/exp* which is the procedure shown in Fig. 1. We set F to 0.5 and CR to 0.9 following the user guidelines in[13], for all the experiments carried out in this research.

2.2 Crowding

A crowding based EA is generally a steady state GA (SSGA) [7], which differs from a simple genetic algorithm (SGA) in its way of how to replace individuals from the population (this is where its selection pressure originates from). Crowding can be briefly described as follows - from a GA population, we randomly pick two parent individuals to mate (and/or mutate) to produce two offspring. In order to insert an offspring immediately back to the population, we randomly select C individuals from the population for potential replacement. The number C, referred to as the *crowding factor*, commonly set to 2 or 3. We then check to see which individual from these C individuals is most similar to this offspring. The similarity is measured by the Euclidean distance (for the real-coded GA) or Hamming distance (for the binary GA) of the genotype of the two individuals. Finally we replace the most similar individual with the offspring. The same procedure is repeated for the second individual as well. This process repeats for all individuals in the population before proceeding to the next generation. Crowding has shown to be effective in maintaining better population diversity, therefore to some extent alleviate the problem of premature convergence. One known problem associated with the crowding method is that when C is set to a small number, the offspring does not always replace the most similar individual with respect to the population [10], which is what so called replacement error. This problem can be overcome by setting C equal to the size of the population. Obviously the number of comparisons will be expensive if the population size is large.

2.3 Crowding DE

Thomsen in [15] extended DE to handle multimodal function optimization with De Jong's crowding method [7]. Basically CrowdingDE is a steady state DE algorithm that makes use of crowding information from the population. In CrowdingDE, when an offspring $\vec{x}^{(i,t+1)}$ is generated, its fitness is only compared with that of the most similar individual from the current population (in this case *crowding factor* C is set equal to the population size n), rather than its parent $\vec{x}^{(i,t)}$ as in the standard DE algorithm. The similarity measure is calculated based on the Euclidean distance between two individuals in genotype space. The offspring $\vec{x}^{(i,t+1)}$ will only replace the most similar individual if it is fitter. The selection pressure driving the population to seek out multiple optima comes from this replacement scheme that encourages the population to remain diverse in the search space. CrowdingDE uses the following steps:

1. Use standard DE to produce an offspring $\vec{x}^{(i,t+1)}$ (see Fig. 1).
2. Calculate the Euclidean distance values of the offspring $\vec{x}^{(i,t+1)}$ to the other individuals in the DE population.
3. Sort all individuals according to their Euclidean distances to $\vec{x}^{(i,t+1)}$.
4. Compare the fitness of $\vec{x}^{(i,t+1)}$ and the fitness of the individual that has the smallest Euclidean distance to $\vec{x}^{(i,t+1)}$. $\vec{x}^{(i,t+1)}$ will only replace this individual, if $\vec{x}^{(i,t+1)}$ is fitter than this individual.
5. Go back to step 1) to generate a new offspring, if the number of offspring is smaller than the population size; Otherwise proceed to next iteration, or stop if some termination criteria are met.

Thomsen showed that CrowdingDE outperforms a fitness sharing DE variant over a set of multimodal test functions [15]. The simplicity of CrowdingDE makes it easy to implement, however it suffers from a higher computational cost since every offspring has to be compared with every other individual in the population for similarity measurement. This procedure has a complexity of $O(N^2)$. This problem gets significantly worse when a larger population size is used.

Apart from its simplicity, an appealing attribute of CrowdingDE is that it does not require any additional user-specified parameters such as species radius r_s, which must be prespecified for the proposed species-based DE. See the following section.

3. SPECIES-BASED DE

3.1 Identifying species

Using speciation has shown to be an effective technique for multimodal optimization [8, 9]. A niching method based on speciation can be used to classify an EA population into groups according to their similarity measured by Euclidean distance. The smaller the Euclidean distance between two individuals, the more similar they are:

$$dist(\vec{x}^{(i)}, \vec{x}^{(j)}) = \sqrt{\sum_{k=1}^{d} (x_k^{(i)} - x_k^{(j)})^2}, \qquad (1)$$

where $\vec{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)})$ and $\vec{x}^{(j)} = (x_1^{(j)}, x_2^{(j)}, \ldots, x_d^{(j)})$ are d-dimensional vectors of real numbers representing two individuals i and j from the EA population.

The definition of a speciation also depends on another parameter r_s, which denotes the radius measured in Euclidean distance from the center of a species to its boundary. The center of a species, the so-called species seed, is always the fittest individual in the species. All individuals that fall within the r_s distance from the species seed are classified as the same species.

input : L_{sorted} - a list of all individuals sorted in decreasing fitness values
output: S - a list of all dominating individuals identified as species seeds

```
begin
    S = ∅;
    while not reaching the end of L_sorted do
        Get best unprocessed p ∈ L_sorted;
        found ← FALSE;
        for all s ∈ S do
            if d(s,p) ≤ r_s then
                found ← TRUE;
                break;
            end
        end
        if not found then
            let S ← S ∪ {p}
        end
    end
end
```

Figure 2: The algorithm for determining species seeds.

3.2 Determining species seeds

The algorithm for determining species seeds, introduced by Petrowski in [12] and also Li et al. in [8], is adopted here. By applying this algorithm at each iteration step, different species can be identified using species seeds. These species can be then treated as subpopulations running DE independently themselves. Fig.2 summarizes the steps for determining the species seeds.

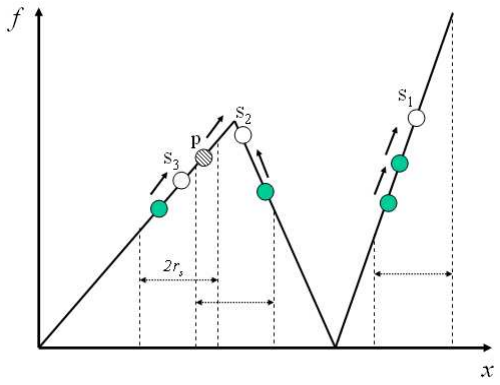


Figure 3: An example of how to determine the species seeds from the population at each iteration step. s_1, s_2 and s_3 are chosen as the species seeds. Note that p belongs to the species led by s_2.

The algorithm (as given in Fig. 2) for determining the species seeds is performed at each iteration step. The algorithm takes as an input, L_{sorted}, a list containing all individuals sorted in decreasing order of fitness. The species seed set S is initially set to $\emptyset$. All individuals are checked in turn (from best to the least-fit) against the species seeds found so far. If an individual does not fall within the radius r_s of all the seeds of S, then this individual will become a new seed and be added to S. Fig. 3 provides an example to illustrate the working of this algorithm. In this case, applying the algorithm will identify $s1$, $s2$ and $s3$ as the species seeds. The 3 species formed around the 3 species seeds which are the fitter as well as different individuals from the population. To locate multiple global optima, it seems sensible to run an DE within each species group so that each species will be able to improve locally, resulting in multiple species groups improving simultaneously across the entire DE population. More importantly, over future iterations, new formations of species will help transform such local improvements into global improvements leading to identifying multiple global optima.

Since species seeds in S are sorted in the order of decreasing fitness, when deciding which individual belongs to which species, the algorithm presented in Fig. 2 naturally allows fitter seeds get allocated with individuals from the population before the less fit seeds in S. This is an important attribute as it helps the optimization algorithm to increase its likelihood for finding the global optima before the local ones.

3.2.1 Complexity

The complexity of the above procedure (Fig. 2) can be estimated based on the number of calculations of Euclidean distances between two individuals that are required. Assuming there are N individuals sorted and stored on L_{sorted}, the **while** loop steps through L_{sorted} to see if each individual is within the radius r_s of the seeds on S. If S currently contains i number of seeds, then at best the **for** loop is ex-

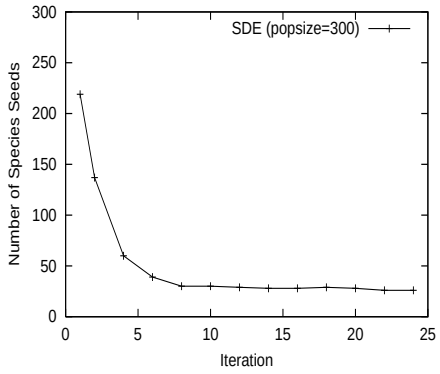


Figure 4: The number of species seeds identified at each iteration over a simulation run.

ecuted only once when the individual considered is within r_s of the first seed compared; and at worst the **for** loop is executed i times when the individual falls outside of r_s of all the seeds on S. Therefore the number of Euclidean distance calculations required for the above procedure $T(N)$ is: $N \leq T(N) \leq \frac{N(N-1)}{2}$, which shows that the worst time complexity of the procedure is $O(N^2)$ when there are N species seeds. However it is important to note that a much tighter upper bound, $\bar{N}_s \cdot N$, can be derived where $\bar{N}_s$ is the upper bound of the number of species that will be found at each iteration. $\bar{N}_s$ can be estimated according to the size of the search space and r_s. Typically $\bar{N}_s \ll N$. Fig. 4 shows that the number of species seeds decreases very quickly in a typical run of the proposed SDE (see also Fig. 6). In such a case, the procedure takes roughly $\bar{N}_s \cdot N$ Euclidean distance calculations at each iteration. Since $\bar{N}_s$ is determined by r_s but not the population size N, This procedure in fact has a linear time complexity of $O(N)$. In contrast, time complexity of the procedure for similarity measurement in CrowdingDE is $O(N^2)$, because each individual must be compared with every other individual in the population at each iteration. Especially when N becomes larger, the similarity measurement procedure gets significantly more expensive.

3.2.2 The SDE algorithm

This paper describes a species-based DE (SDE) which makes use of the algorithm for determining species in conjunction with a basic DE, can be used effectively to solve multimodal optimization problems. Following the steps in determining species seeds in Fig. 2, SDE is able to identify multiple species at each iteration step from the entire population. Each identified species by itself is an DE running the algorithm described in Fig. 1. The procedure can be summarized as follows:

1. Generate an initial population with randomly generated individuals.
2. Evaluate all individuals in the population.
3. Sort all individuals in descending order of their fitness values (i.e., from the best-fit to least-fit ones).
4. Determine the species seeds for the current population (see Fig. 2).
5. For each species as identified via its species seed, run a basic DE as described in Fig. 1.
   a. If a species has less than m individuals, then randomly generate new individuals within the radius of the species seed until there are m individuals in the species (see section 3.2.2.1 below).
   b. If a child's fitness is the same as that of its species seed, replace the 'redundant' child with a randomly generated new individual (see section 3.2.2.2).
6. Keep only the N fitter individuals from the combined population.
7. Go back to step 2), unless the termination criteria are met.

3.2.2.1 Creating local random individuals.

Note that in step 5) it is possible that an identified species has less than m individuals. Since to run a DE, we must have at least 3 or more individuals in a species, m is usually set to a number greater or equal to 3. If the number of individuals is below the m threshold, a new individual is generated randomly within the radius of the species seed and added to that species. This process is repeated until the size of the species is greater or equal to m. As a result of this, it is also possible that the size of the final combined population of all species (including these newly generated individuals) is greater than the original size of the population N. This problem can be resolved by the procedure for determining species seeds (see Fig. 2). Basically all individuals of the combined population are first sorted in decreasing fitness order, and then only the first N fitter individuals (including seeds and other individuals) are kept. The remaining individuals are simply removed so that the DE population size is always kept constant at N at each iteration step. Our experience tells us that m should be set to a number much higher than 3, for example m is set to 10 in this paper. This is because DE works well when it has a better sampling of differences among individuals in its population. 10 seems to be a reasonable number for a minimum species size.

3.2.2.2 Eliminating redundant individuals in a species.

Since multiple species may converge at different speeds, some species may have converged while others are not. The efficiency of SDE can be improved by replacing redundant individuals in the converged species with randomly generated new individuals. Redundant individuals in a species are those individuals having the same fitness as that of its species seed. This procedure should follow the conventional DE procedure for producing a child (see Fig. 1), and can be carried out by checking if the fitness of each DE child individual is the same as the fitness of its species seed; if they are the same, then the child is replaced by a randomly generated new individual; and if they are not (it means the species is not yet converged), then the child is kept as usual. This procedure should improve efficiency because it allows SDE to more effectively sample differences among a population of more diverse individuals, including these randomly generated new individuals instead of the redundant individuals in a species.

4. PERFORMANCE MEASUREMENT

Since our main goal is to see if the SDE is more efficient than the CrowdingDE, we adopt the following two performance measurements:

1. **accuracy**: an algorithm is run for a fixed number of iteration steps, and accuracy, which measures the closeness of fittest solutions to all known global optima, is recorded in the final iteration step (see equation (2));

2. **convergence speed**: in this mode, an expected accuracy level is pre-specified and the number of evaluations required to achieve the expected accuracy is recorded.

In addition, we also measure the performance in terms of **success rate**, the percentage of runs in which all global optima are successfully located.

4.1 Accuracy

Accuracy is calculated by taking the average of the fitness differences between all known global optima to their closest species seeds:

$$accuracy = \frac{1}{||opts||} \sum_{j=1}^{||opts||} |fit(opt_j) - fit(seed_j)|, \quad (2)$$

where $||opts||$ gives the number of known global optima. A pair of opt_j and $seed_j$ represents that for each optimum opt_j, there is correspondingly a closest species seed $seed_j$ to opt_j. This $seed_j$ can be identified from S, the set of species seeds. Since a species seed is always the fittest individual in its species, equation (2) should give an accurate indication of how closely the algorithm identifies all the global optima. Note that suboptima of a test function are not taken into consideration in computing accuracy. Since our goal is to find a complete set of global optima, i.e., all the global optima of interest in a function, we allow the algorithm to run for a fixed number of iterations (e.g., 1000 iteration steps) before termination to see if all known global optima are found. When measuring convergence speed, we terminate the algorithm only after a pre-specified accuracy (as given in equation (2)) is reached for all known global optima (e.g., Table 3).

4.2 Convergence speed

To measure convergence speed at a required level of accuracy, we only need to check set S, which contains the species seeds identified so far. These species seeds are dominating individuals sufficiently different from each other, however they could be individuals with high as well as low fitness values (see Fig. 3). We can decide if a global optimum is found by checking each species seed in S to see if it is close enough to a known global optimum. An expected accuracy acceptance threshold ($0 < \epsilon \leq 1$) is defined to detect if the solution is close enough to a global optimum, and the following condition must be satisfied:

$$\forall x \in S_{opt} \exists y \in S_{seed} : min\{||x - y||\} \wedge |fit(x) - fit(y)| \leq \epsilon, \quad (3)$$

where S_{opt} is a set of all known global optima of a multimodal function, and S_{seed} is a set of identified species seeds (each should correspond closely to an optimum towards the end of a run). $min\{||x - y||\}$ returns the closest pair of a global optimum (from S_{opt}) and a species seed (from S_{seed}). Equation (3) states that for each global optimum x there must exist a species seed y such that the fitness difference between x to its closest species seed y is not greater than ϵ. This condition must be satisfied before a run can be terminated.

Table 2: Results on accuracy after 1000 iterations with 100% success rate (averaged over 50 runs).

Function	SDE (mean and std dev)	CrowdingDE (mean and std dev)
$F1$	$1.71E-09\pm$ 1.21E-08	0.00E+00 ± 0.00E+00
$F2$	$2.62E-09\pm$ 1.84E-08	4.20E-10 ± 6.79E-10
$F3$	$1.55E-06\pm$ 0.00E+00	1.55E-06 ± 0.00E+00
$F4$	$3.58E-07\pm$ 0.00E+00	3.67E-07 ± 1.32E-08

Table 3: Results on convergence speed with 100% success rate (averaged over 50 runs).

Function	SDE (mean and std dev)	CrowdingDE (mean and std dev)
$F1$	440± 268.97	2439 ± 721.77
$F2$	5286± 5166.22	20001 ± 2016.32
$F3$	723± 123.81	7272 ± 1481.01
$F4$	4360± 2799.38	18620 ± 3161.39

5. NUMERICAL RESULTS

Table 1 shows the test functions used in this paper. Firstly, to compare the accuracy and convergence speed of SDE and CrowdingDE on some relatively simple multimodal functions, $F1$, $F2$, $F3$ and $F4$ were used. These functions are widely used and they are typical 2-dimensional multimodal test functions with a limited number of global optima. After this, a more challenging test function $F5$ was used. $F5$ the 2-dimensional Shubert function is different from $F1$ to $F4$ in the fact that it has a large number of global and local optima (760 optima including 18 global optima); hence it would pose greater difficulties to a multimodal optimization method. In the second part of this section $F5$ was used with various parameter setups to test SDE's ability to handle the situation when there are a large number of optima present.

5.1 Accuracy

Table 2 shows the results of accuracy for SDE and CrowdingDE. In this experiment, the population size was set to 50, and the SDE species radius r_s was set to 0.05 for $F1$, and 0.5 for $F2$, $F3$ and $F4$. Both SDE and CrowdingDE were run for 1000 iteration steps and their accuracies were recorded. As can be seen from Table 2, SDE has obtained very similar results as that of CrowdingDE. Both SDE and CrowdingDE have achieved 100% success rate, which means both of them have reached an accuracy smaller than the required accuracy ϵ of 0.0001 within 1000 iterations.

Table 1: Test functions.

Function	Range	Comments
Deb's 1st function [3]: $F1(x) = sin^6(5\pi x)$	$0 \leq x \leq 1$	5 equally spaced global optima
Himmelblau [1]: $F2(x) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	$-6 \leq x, y \leq 6$	4 global optima
Six-Hump Camel Back [11]: $F3(x) = -4[(4-2.1x^2+\frac{x^4}{3})\cdot x^2+xy+(-4+4y^2)\cdot y^2]$	$-1.9 \leq x \leq 1.9$; $-1.1 \leq y \leq 1.1$	2 global optima and 4 local optima
Brainin RCOS [11]: $F4(x,y) = (y - \frac{5.1}{4\pi^2} \cdot x^2 + \frac{5}{\pi} \cdot x - 6)^2 + 10 \cdot (1 - \frac{1}{8\pi}) \cdot cos(x) + 10$	$-5 \leq x \leq -10$; $0 \leq y \leq 15$	3 global optima
Shubert [8]: $F5(x,y) = \sum_{i=1}^{5} icos[(i+1)x+i] \cdot \sum_{i=1}^{5} icos[(i+1)y+i]$	$-10 \leq x, y \leq 10$	760 optima including 18 global optima

5.2 Convergence speed

In this experiment, the same parameter setup was used as for accuracy in the previous section. Table 3 compares the convergence speed of SDE and Crowding DE over $F1$ - $F4$, in terms of the number of evaluations. Both SDE and CrowdingDE were run until it reached the required accuracy ϵ of 0.0001 for all global optima over 50 runs (that is 100% success rate). For this experimental setup, SDE used far fewer evaluations to reach convergence than CrowdingDE did. With consideration of time complexity on the number of comparisons required for CrowdingDE and SDE, the next section will show that CrowdingDE is even more costly when a larger population size is used.

5.3 Effect of varying population size

The effect of using varying population sizes was tested on $F5$ the 2-dimensional Shubert function. In this experiment, SDE and CrowdingDE were both run until a required accuracy ϵ of 0.1 (to all known 18 global optima) or the 5000th iteration step was reached.

It would be difficult to compare SDE's efficiency with CrowdingDE based purely on the number of evaluations, because CrowdingDE has much higher time complexity in identifying the similar individuals than SDE in determining species seeds at each iteration step (see section 3.2.1). To make a fair comparison, both CrowdingDE and SDE were run 50 times on a Pentium 4 machine and the time for each run was measured in milliseconds. The average over the 50 runs, the highest, and lowest amount of time were recorded. It can be seen from Fig. 5, the amount of time taken by CrowdingDE quickly went up, when the population size was increased from 50 to 100. This would get much worse if CrowdingDE uses a population size greater than 100, since the number of similarity measurements in CrowdingDE has the complexity of $O(N^2)$. In contrast, SDE used much less time in finding all the global optima even when the population size was increased from 150 to 400. This is largely attributed to SDE's more efficient way of identifying species, which has a linear time complexity of $O(N)$.

Another observation from Fig. 5 is that overall CrowdingDE has a different 'optimal' range of population sizes than SDE. CrowdingDE works well on smaller population sizes, while SDE favours relatively larger population sizes. Similar results were also obtained for F1 - F4. As can be seen from the plotted data points in Fig. 5, CrowdingDE only managed to find all 18 global optima when a population size of 80 or 100 was used, but not with a population of 50 or 60. SDE managed to find all 18 global optima with

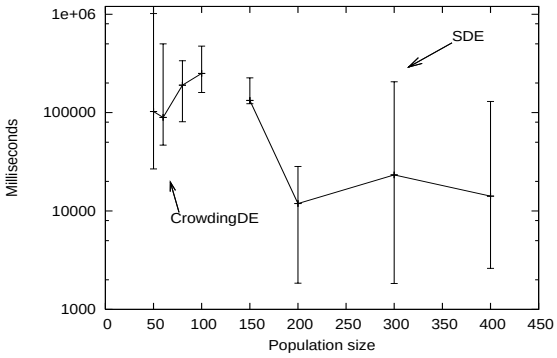


Figure 5: The effect of varying population size on time measured in milliseconds (on F5 the 2-dimensional Shubert function). The average over 50 runs, the highest and lowest values were recorded.

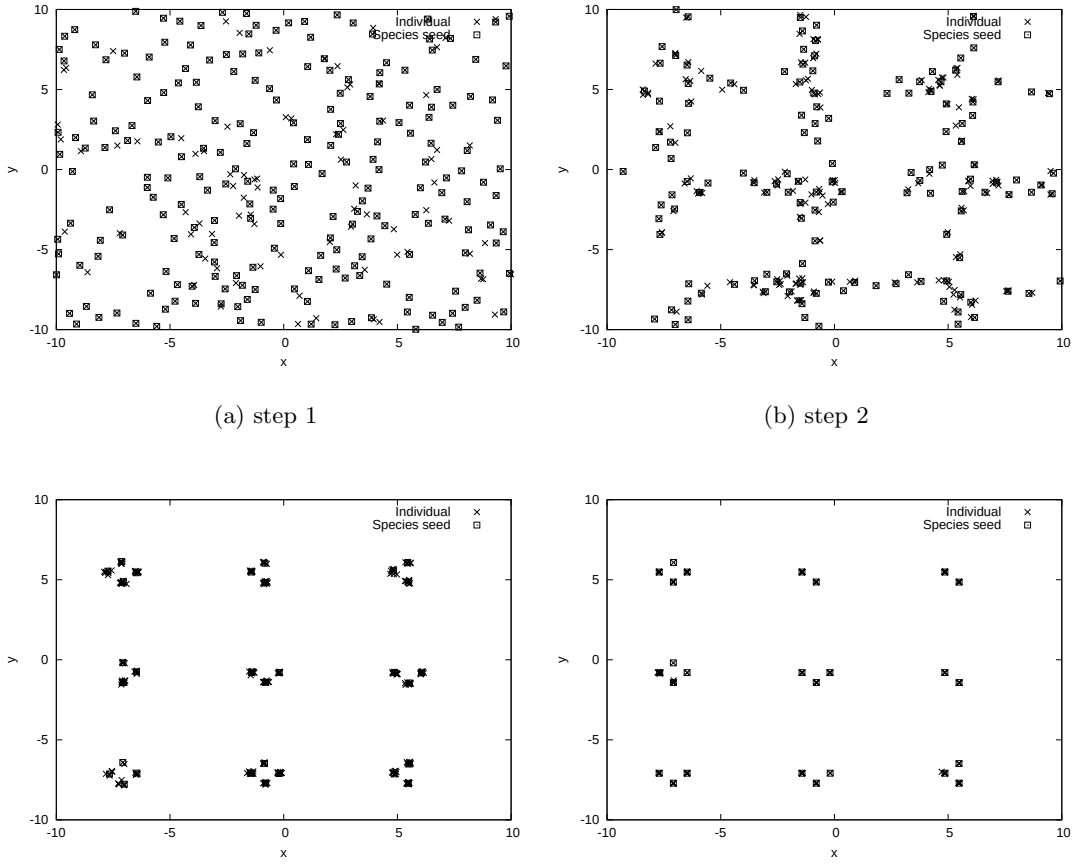
a population size of 200, 300, and 400, but not 150. SDE is much less sensitive to the increasing population sizes than CrowdingDE. It is interesting to see that in Fig. 5 SDE even used less time in average for the setup using a population size of 400 than 300. On the other hand, CrowdingDE is almost too costly to use when the population size is increased to 100 or greater. SDE does not seem to have such a limitation.

For CrowdingDE, it may be possible to reduce the crowding factor C to a smaller number than the population size, however this may cause the problem of 'replacement error' as described in section 2.2.

Fig. 6 shows a simulation run of SDE on $F5$ the 2-dimensional Shubert function. In this run, SDE found all 18 global optima in only 24 iteration steps with the expected accuracy of 0.1.

5.4 Effect of varying species radius

In this experiment, population size was set to 300, and expected accuracy ϵ was set to 0.1. SDE was run for 50 times with varying species radius values ranging from 0.1 to 14. The average number of solutions found over 50 runs with one standard deviation was recorded (see Fig. 7). SDE found most of 18 global optima when species radius r_s was set to 0.2 - 0.8 (Fig. 7 b)). This can be explained by the fact that the closest distance between any two global optima in the Shubert function is 1.6, therefore SDE using a r_s value of 0.8 or below still has the ability to distinguish any two



(a) step 1

(b) step 2

(c) step 8

(d) step 24

Figure 6: A snapshot of a simulation run of SDE on $F5$ - step 1, 2, 8 and 24.

global optima, but if r_s is increased to above 0.8, then any two closest global optima would become indistinguishable to SDE (regarded as the same species). As shown in Fig. 7 a), when r_s is set to 0.8 or above, the number of found solutions is reduced dramatically. Also noticeable is that when r_s is set to 1 - 5, SDE was still able to locate 9 different clusters among the 18 global optima (see Fig. 6 step 24 for an example), i.e., at least one global optimum from a global optima pair in a cluster was located.

6. CONCLUSIONS

This paper proposes an extension to DE using speciation for solving multimodal optimization problems. This species-based DE (SDE) was compared with another recently proposed CrowdingDE on some widely used test functions. It has been shown in the experimental results that SDE is more computationally efficient than CrowdingDE, especially when a larger population size has to be used to deal with problems having a large number of optima.

Although CrowdingDE has the advantage of not having to specify additional user-specified parameters such as species radius r_s, it can be very computationally expensive. Our results indicate that CrowdingDE performs well only over a range of smaller population sizes. However when facing problems having a large number of global optima thereby requiring a larger population size, CrowdingDE becomes inefficient and too expensive to run. In contrast, the proposed species-based DE (SDE) is much less sensitive to larger population sizes. Our results show the efficiency of SDE becomes significantly better than CrowdingDE when both algorithms must use a large population size.

SDE also incorporates two mechanisms to further improve its efficiency. By creating local random individuals, DE running within each species can be enhanced with better convergence; by removing redundant individuals and replacing them with randomly generated individuals, the search space is better explored without any additional computational cost. These two mechanisms also help to alleviate the sensitivity of SDE to different species radius values.

SDE proves to be an efficient multimodal optimization algorithm for the problems tested in this paper. In future, SDE's ability in handling complex real-world multimodal optimization problems and problems with higher dimensionality will be studied.

7. ACKNOWLEDGMENTS

The author would like to thank Rene Thomsen for some useful discussion on CrowdingDE.

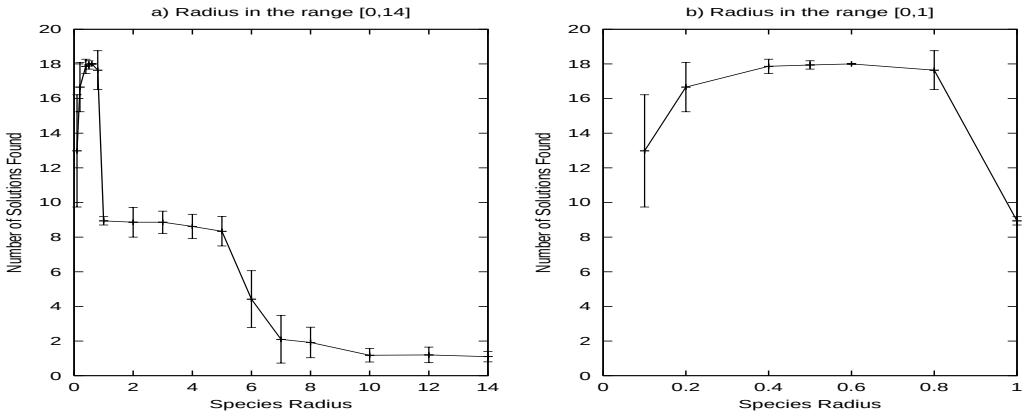


Figure 7: The average number of solutions found over 50 runs when using different species radius values, with one standard deviation error bars.

8. REFERENCES

[1] D. Beasley, D. R. Bull, and R. R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125, 1993.

[2] M. Bessaou, A. Pétrowski, and P. Siarry. Island model cooperating with speciation for multimodal optimization. In H.-P. S. et al., editor, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, Paris, France, 16-20 2000. Springer Verlag.

[3] K. Deb and D. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, 1989.

[4] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[5] G. R. Harik. Finding multimodal solutions using restricted tournament selection. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, San Francisco, CA, 1995. Morgan Kaufmann.

[6] Z. Hendershot. A differential evolution algorithm for automatically discovering multiple global optima in multidimensional, discontinues spaces. In *Proceedings of MAICS 2004, Fifteenth Midwest Artificial Intelligence and Cognitive Sciences Conference*, pages 92–97, 2004.

[7] K. A. D. Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975.

[8] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evol. Comput.*, 10(3):207–234, 2002.

[9] X. Li. Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In e. a. K. Deb, editor, *Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO'04) (LNCS 3102)*, pages 105–116, 2004.

[10] S. W. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *Parallel problem solving from nature 2*, pages 27–36, Amsterdam, 1992. North-Holland.

[11] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, New York, New York, 1996.

[12] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of the 3rd IEEE International Conference on Evolutionary Computation*, pages 798–803, 1996.

[13] K. Price. An introduction to differential evolution. *New Ideas in Optimization*, pages 79–108, 1999.

[14] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.

[15] R. Thomsen. Multimodal optimization using crowding-based differential evolution. In *Proceedings of the 2004 Congress on Evolutionary Computation*, volume 2, pages 1382–1389, 2004.

[16] D. Zaharie. Extensions of differential evolution algorithms for multimodal optimization. In *Proceedings of SYNASC'04, 6th International Symposium of Symbolic and Numeric Algorithms for Scientific Computing*, pages 523–534, 2004.