The Boundary Archive Genetic Algorithm
Hokkaido University  ○XU ZHUORAN, IKUO SUZUKI, MASAHITO YAMAMOTO, MASASHI FURUKAWA

Abstract
Radius-based evolutionary algorithms for Multimodal Optimization are criticized for the difficulty of the proper choice of radius which is crucial in assuring accurate results. TSC2 uses Detect-Multimodal Method to detect different peaks on landscape without assigning a radius. However, it needs mass number of additional evaluations which makes it expensive on time cost. This paper proposed a new algorithm to estimate the boundaries of peaks and archives them to reuse. It costs much fewer evaluations comparing to TSC2.

Keywords: Genetic Algorithm, Niching method, Estimation optima, Solution Archive

1. Introduction
Standard Evolutionary Algorithm has the tendency to converge to a single optimum(peak) with small variations of it. Niching Method manages to maintain several subpopulations within the whole population to overcome this problem. Most Niching Methods introduce a radius parameter which is hard to set. TSC2[1] gets rid of this radius parameter by using detect-multimodal method. Although robust, TSC2 requires mass additional fitness evaluations, which makes it expensive on time cost. This paper proposed a new algorithm which has similar performance but much efficient than TSC2.

2. Archive
This paper introduces an explicit Archive to store detected optima and their boundaries. The boundary of an optimum is represented as a hyper-sphere. The content of each item of the Archive is a tetrad.

<genotype, fitness, lowerbound, upperbound>
where the lowerbound is used as the estimation of the radius of the hyper-sphere. The lowerbound and upperbound are initialized to zero and infinity respectively.

3. Boundary Estimation
For each archived optimum, we aim to find a hyper-sphere which has as few as possible valleys in it. We estimate its radius through detect-multimodal method. The method is shown in algorithm 1, x and y are two optima in Archive, m is a point between them.

Algorithm 1 EstimateBoundary (x,y,m)

\[
\text{dis} = \text{distance between } x \text{ and } y;
\]
if \( m.\text{fitness} < \text{min}(x.\text{fitness}, y.\text{fitness}) \)
then
\[
x.\text{upperbound} = \text{min}(x.\text{upperbound}, \text{dis});
\]
\[
x.\text{lowerbound} = \text{dis};
\]
\[
y.\text{upperbound} = \text{min}(y.\text{upperbound}, \text{dis});
\]
\[
y.\text{lowerbound} = \text{dis};
\]
else
\[
\text{if } x.\text{fitness} > y.\text{fitness}
\]
\[
x.\text{lowerbound} = \text{max}(x.\text{lowerbound}, \text{dis});
\]
else
\[
y.\text{lowerbound} = \text{max}(y.\text{lowerbound}, \text{dis});
\]
end if
end if

4. Boundary Estimation Test
The algorithm generates a number of test points between randomly selected optima in Archive at each generation. They are used as m element of inputs in EstimateBoundary function. The test points are always generated between an optimum in Archive and its neighbors. The number of neighbors of an optimum that can be tested with it is limited by a parameter p. If an optimum y is ranked as the i-th neighbor of x, and i is bigger than p, then the test between x and y is forbidden. Algorithm 2 shows this process, S is a set used to store the generated test points.

Algorithm 2 CreateTest(S,p)

randomly select one optimum x from the Archive;
for \( i = 1:p \)
\[
y = \text{the } i\text{-th neighbor of } x \text{ in Archive};
\]
if y has not been tested with x
\[
\text{marked } y \text{ has been tested with } x;
\]
\[
m = x + (y - x) \times 0.5;
\]
\[
S = S \cup \{<x,y,m>\};
\]
break;
end if
end for

5. Overlap Detection
The function for verifying whether x is overlapped by y is shown in algorithm 3. A parameter p is introduced to denote the overlap area allowed to increase the stability of the Archive.

Algorithm 3 Overlap(x,y,p)

\[
\text{dis} = \text{distance between } x \text{ and } y;
\]
\[
\text{rate} = \text{max}(\text{0},(2y.\text{lowerbound-dis})/2x.\text{lowerbound});
\]
if \( \text{dis} \leq x.\text{lowerbound} \text{ or } \text{dis} \leq y.\text{lowerbound} \)
\[
\text{if } y.\text{fitness} \leq x.\text{Fitness} \text{ or } \text{rate} < p
\]
\[
\text{return } \text{FALSE};
\]
else
\[
\text{return } \text{TRUE};
\]
end if
end if
return FALSE;
6. The Boundary Archive Genetic Algorithm

The Boundary Archive Genetic Algorithm (BAGA) is presented in the algorithm 4. At each generation, first, we estimate the boundary of optima in Archive using EstimateBoundary. The inputs are recorded in a set \( S \). Then, the overlapped optima in Archive are deleted according to a parameter \( p_{\text{overlap}} \). After that, the Archive is expanded by \( N_{\text{overlap}} \) individuals selected from the population. Those individuals should not be overlapped by any optima in Archive.

\( N_{\text{seq}} \) test points are generated by CreateTest using a parameter \( p_{\text{overlap}} \) and store in the triple \( S \). \( N_{\text{overlap}} \) optima in Archive are conserved back to the population to increase the diversity. The entire population is divided into subpopulations according to Archive. \( P' \) contains the individuals selected from the population by a standard weighted mating selection operator. Because the test points of the estimation occupy the population, the size of \( P' \) is \( N - N_{\text{seq}} \), where \( N \) is the size of population. The recombination and mutation are executed on \( P' \).

The current population is the union of \( P' \) and \( m \) elements in \( S \). Hence, the evaluations needed by the estimation are integrated into the evaluation of current population. Furthermore, they contribute to the building of next generation.

**Algorithm 4 BAGA Algorithm**

\[
S = \emptyset, \text{ Archive} = \emptyset \;
\]

Initialize \( P \);

Evaluate \( P \);

while not termination condition

EstimateBoundary \((S.x,S.y,S.m); \)

delete the overlapped optima in Archive;

insert \( N_{\text{overlap}} \) unoverlapped individuals into Archive;

\( S = \emptyset \)

create \( N_{\text{seq}} \) test using CreateTest \((S, p_{\text{overlap}})\)

conserved \( N_{\text{overlap}} \) optima back to the population

divide subpopulations on \( P \);

select \( P' = N - N_{\text{seq}} \) individuals;

Recombination and Mutation on \( P' \);

\( P = P' \cup S.m \);

Evaluate \( P \);

end while

7. Experiments

Four functions, shown in table 1, are chosen from[1] to compare BAGA and TSC2. They have 10, 6, 18, 5 optima respectively.

Table 1 shows the comparison of BAGA and TSC2. Peak ratio and peak accuracy are percentage of optima found by algorithms and the difference on fitness comparing to the theoretical value. Time cost is the number of evaluations needed to reach 80% accuracy on peak ratio. The BAGA catches more than 90% optima on all test functions. It is also significantly superior to TSC2 on F2. However, TSC2 performs better on the peak accuracy, except F2. The time cost of BAGA is significant lower than TSC2.

Figure 1-4 shows the contour line graphs and the contents of Archive of BAGA on F1-F4. Red points represent the stored optima in Archive, and the circles around them represent the estimated boundary. The estimation is accuracy, for the major parts of optima on landscape are covered by circles and no circles cover two optima.

8. Conclusions

The BAGA is an improvement of TSC2. It inherits the advantage of TSC2 on tracking multiple peaks and overcomes the drawback of mass time costs. We will further investigate its performance on high-dimension space.

**Reference**