Paper Title

Zhao Zhixiang 11711621
dept. CSE of SUStech
11711621@mail.sustech.edu.cn

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

I. Introduction

The Evolutionary Algorithms (EAs) are inspired by nature. They design this kind of algorithm based on the procedure of biological evolution described in Darwin's masterpiece *The Origin of Species*. Then Joseph Grinnell came up with the concept of the ecological niche in 1917. He found the functional role that it plays within an ecosystem drives divergent evolution and speciation. Since then, the concept of the niche is not only used in biology but also widely used in Social Science and Economics, which is related to the social division of labor and market theory. Also, EAs designed with a niching mechanism can locate multiple globally optimal or suboptimal solutions for the problem of optimization. In this work, a simple EA with crowding and fitness sharing is designed for optimizing multimodal functions given by Benchmark for CEC'2013 special session and competition. Beyond optimization, in a way, EAs are simulations for natural and civilized evolution. They can be used as mirrors to reflect the world. There exists a deceptive fitness landscape when I deployed fitness sharing to EA, which drives the divergent evolution but also affects locating peaks.

II. Background

A. Munkres Algorithm [1]

1) Assignment Problem: Let C be an $n \times n$ matrix representing the costs of each of n workers to perform any of n jobs. The assignment problem is to assign jobs to workers so as to minimize the total cost. Since each worker can perform only one job and each job can be assigned to only one worker the assignments constitute an independent set of the matrix C.

2) Algorithm Steps: The following 6-step algorithm is a modified form of the original Munkres' Assignment Algorithm (sometimes referred to as the Hungarian Algorithm).

- **Step 0**: Create an nxm matrix called the cost matrix in which each element represents the cost of assigning one of n workers to one of m jobs. Rotate the matrix so that there are at least as many columns as rows and let k=min(n,m).
- **Step 1**: For each row of the matrix, find the smallest element and subtract it from every element in its row. Go to Step 2.
- **Step 2**: Find a zero (Z) in the resulting matrix. If there is no starred zero in its row or column, star Z. Repeat for each element in the matrix. Go to Step 3.
- **Step 3**: Cover each column containing a starred zero. If K columns are covered, the starred zeros describe a complete set of unique assignments. In this case, Go to DONE, otherwise, Go to Step 4.
- **Step 4**: Find a noncovered zero and prime it. If there is no starred zero in the row containing this primed zero, Go to Step 5. Otherwise, cover this row and uncover the column containing the starred zero. Continue in this manner until there are no uncovered zeros left. Save the smallest uncovered value and Go to Step 6.
- **Step 5**: Construct a series of alternating primed and starred zeros as follows. Let Z_0 represent the uncovered primed zero found in Step 4. Let Z_1 denote the starred zero in the column of Z_0 (if any). Let Z_2 denote the primed zero in the row of Z_1 (there will always be one). Continue until the series terminates at a primed zero that has no starred zero in its column. Unstar each starred zero of the series, star each primed zero of the series, erase all primes and uncover every line in the matrix. Return to Step 3.
- **Step 6**: Add the value found in Step 4 to every element of each covered row, and subtract it from every element of each uncovered column. Return to Step 4 without altering any stars, primes, or covered lines.
- **DONE:** Assignment pairs are indicated by the positions of the starred zeros in the cost matrix. If C(i,j) is a starred zero, then the element associated with row i is assigned to the element associated with column j.

3) Munkres Crowding: Since crowding techniques insert new individuals into the population by replacing similar individuals. So I deploy the Munkres algorithm to match offsprings to the close parents such that the total distance between parents and offsprings is minimum. Comparing the fitness of the parent with the fitness of the offspring that picked up through the Munkres algorithm, it decides whether to replace the parent or not. Thus we achieve elitism and crowding strategy of replacement.

III. Proposed Algorithm

The simple Evelutionary Algorithm (SEA) framework with components as followings:

1) Mechanism for Divergent Evolution: Fitness Sharing;
2) Crossover Operator: Symetric Simple Arithmetic Recombination;

3) Mutation Operator: Nonuniform Mutation with Cauchy Distribution;
4) Selection Scheme: Exponential Ranking Selection;
5) Replacement: Munkres Crowding;

A. Fitness Sharing

Fitness Sharing is a dynamic penalty for each agent in population. The sharing radius σ_{share} defines the niche size and the distance between two agents i, j defines the similarity between them.

Then both assumptions induce the sharing function:

$$sh(d_{i,j}) = \begin{cases} 1 - \left(\frac{d_{i,j}}{\sigma_{share}}\right)^{\alpha} & \text{, if} \quad d_{i,j} < \sigma_{share} \\ 0 & \text{, otherwise} \end{cases} \tag{1}$$

with α determines The extent of the penalty.

And the shared fitness of i_th agent can be

$$f_{share}(i) = \frac{(f_{raw}(i))^{\beta}}{\Sigma_{j=1}^{\mu} sh(d_{i,j})} \tag{2}$$

with μ is the population size and β is a scaling factor.

B. Crossover Operator

First there is a predefined weight α and a generating uniform-randomly selected point k. Then with a 0.5 probability:

$$I_1 = 0, ..., k-1$$

And

$$I_2 = k-1, ..., d$$

Offspring 1: $v_i' = v_{1i},\ i \in I_1(resp.\ I_2)$

. $v"_i = \alpha v_{1i} + (1-\alpha) v_{2i}\ i \in I_2(resp.\ I_1)$

Offspring 2: $v_i' = v_{2i},\ i \in I_1(resp.\ I_2)$

. $v"_i = \alpha v_{2i} + (1-\alpha) v_{1i}\ i \in I_2(resp.\ I_1)$

C. Mutation Operator

For the vector v represent the individual in $\Re^{dim}$, dim is the dimension of the solution space. Then we mutate the individual as followings:

$$v_i = v_i + \triangle \forall i \in 1, 2, 3, ..., dim \tag{3}$$

$$\triangle = \alpha \times EEL \times \text{Standard cauchy distribution}$$

The EEL is the expected edge length:

$$EEL = \sqrt[dim]{Volumn}$$

And the volumn of solution space:

$$Volumn = \prod_{i=1}^{dim} (UB_i - LB_i)$$

with LB_i (resp. UB_i) is the lower bound (resp. upper bound) of i_th dimension.

Besides, if the mutation is illegal, i.e., the mutated value is out of boundary, then there is a probability p such that roll back and do the mutation again and $1-p$ set the value to be boundary.

D. Selection Scheme and Replacement

The exponential ranking selection is not different with what we usually use. And the replacement policy is mentioned in section 2 as Munkres Crowding to maintain the preexisting diversity of a population.

E. Experimental setup

- population size: n
  Since I didn't not found a proper way to self-adaptively know the number of global optimals. So I set the population size as the number of global optimals.
- share fitness:
  share radius: $\sqrt[dim]{Volumn/n}$
  α = 2;
  α determines the shape of sharing function: the larger the alpha, the more competiton;
  β = 1 No scalling.
- crossover: weight α = uniformly random from 0.2 to 0.4 for maintain the diversity of offsprings and parents.
- mutation: mutation α = 0.005 p = 0.3 to set the result value to the boundary.

IV. Experimental Results and Discussion

A. results

The results (table I) are the average sums of PR for each scenario by running 10 times of the function benchmark:

The peek rate (PR) is,

$$\frac{num\ of\ the\ founded\ optimals}{num\ of\ the\ ground\ truth\ optimal}$$

compared with [2], which the sum of PR is 17.787, this algorithm is a failure.

TABLE I
SUM OF PR

accuracy = 0.1	6.45426
accuracy = 0.01	4.78157
accuracy = 0.001	3.74731

B. Conclusion and Discussion

Since the result is bad, I try to find out the reason. A major reason I found is that fitness sharing is a double-edged sword: On one hand, it drives the divergent evolution; on the other hand, it deceives the agents or the selector, making them mistaken for the global optimal. But actually, they are not, not even the local optimal. This kind of illusion affects their ability to locate global optimals.

Another reason is that the mutation operator has a large step compared with the small area of peak defined by accuracy,

which means the agents are hard to hit the peak with a small probability.

Is it a good idea that we simulate the competition between individuals within a niche as changing their fitness? If we go back to think about the original idea, i.e. biological niche theory, then we can see:

the Principle of Exclusion of Competition: When the ecology is balanced, the ecological niche does not coincide in principle. If there is overlap, it is bound to be unstable, and it will inevitably reduce the overlap of the ecological niche through competition between species until equilibrium is done. Competition, such as the invasive species with a similar ecological niche can lead to a reduction in the presence of indigenous species. If a niche of the indigenous species is compressed by the invaders such that the area of existence is too small, then it will lead to the extinction of a species.

Evolution, in some way, is to let related species taking up different ecological niches and reducing the chances of competition.

Therefore, the result of the competition is that the winner takes all rather than penalty of each other.

Then we may design new mechanisms by this principle which do not touch the fitness.

1) Dynamic system mechanism: Each individual has two states - survival or death, and each individual interacts with an surrounding neighbour individual within the niche by certain rules and decide the state.
2) Local selector: This selector select individuals to survive within the niche according to their fitness and distance.

As for the mutator, I think we can use the way of simulated annealing to reduce the step of mutation in late statge in evolution.

V. Future Work

A. Future Work

The above ideas are based on the niche and its size. I think we must heuristically detect the size of the niche for certain problems. For this benchmark, I think information about the niche is the valley. We treat a hill as a niche and hope every hill has one to climb the mountain. That is enough. I need to design a method to detect the valley as my next step.

And another issue is that there are a huge number of local optimals. We don't want them as our solution since it will deduct the f1 rate. A useful method could be an archive strategy. I will explore it and add it to the algorithm.

References

[1] Algorithms for Assignment and Transportation Problems, James Munkres, Journal of the Society for Industrial and Applied Mathematics Volume 5, Number 1, March, 1957

[2] Real-Valued Evolutionary Multi-Modal Optimization driven by Hill-Valley Clustering In Proceedings of the Genetic and Evolutionary Computation Conference S.C. Maree, T. Alderliesten, D. Thierens, and P.A.N. Bosman. GECCO-2018, ACM Press, New York, New York, 2018.