Limitation on the operator amount for making process less time-consuming

Kiarash Jamshidi

July 1, 2021

1 Introduction

In this report, we introduce two new variables, which are Fog-Density-Threshold-For-Generating-Seed-Min, Fog-Density-Threshold-For-Generating-Seed-Max. In the beginning section, there is an explanation of why these two variables were required. Then, we calculate the right amount for these two variables and make the new configuration specifications in the following part. In section five, we add two variables to the program and then experiment with the new configuration to investigate our implementation.

First experiment has the configuration below:

```
MUTATION_TYPE = Config.MUT_FOG
FOG_DENSITY_threshold_max = 1
FOG_DENSITY_threshold_min = 0
MUTATION_FOG_PRECISE = 0.0
MUTATION_EXTENT = 6
generator_name = Config.GEN_DIVERSITY
POOLSIZE = 20
POPSIZE = 12
NUM_GENERATIONS = 10
ARCHIVE_THRESHOLD = 35.0
```

There is one type of threshold(FOG-DENSITY-threshold-max = 1 FOG-DENSITY-threshold-min=0) for operator Fog. This type of threshold is for the mutation and also generating the seeds. Generating the seeds also gets the same threshold as the mutation process. The MUTATION-TYPE is the type of operation that the system has to run. In this experiment, we choose the fog operation, which the name in this program is MUT-FOG. Also, in this example, the approach is DIVERSITY, so generator-name = Config.GEN-DIVERSITY. It means testing all seeds firstly to choose only the seed that behaves correctly(it doesn't go out from the lane and arrive at the final point).

The Poolsize is the number of seeds which is 20. Pop size is the number of individuals in every generation. For this test, Popsize is chosen 12.

The first section is the duration of the whole process per generation. The timing in this program is the solid specification. We have to do our best to decrease this amount.

2 Duration of process

In this part, we want to see how much time every generation will take.

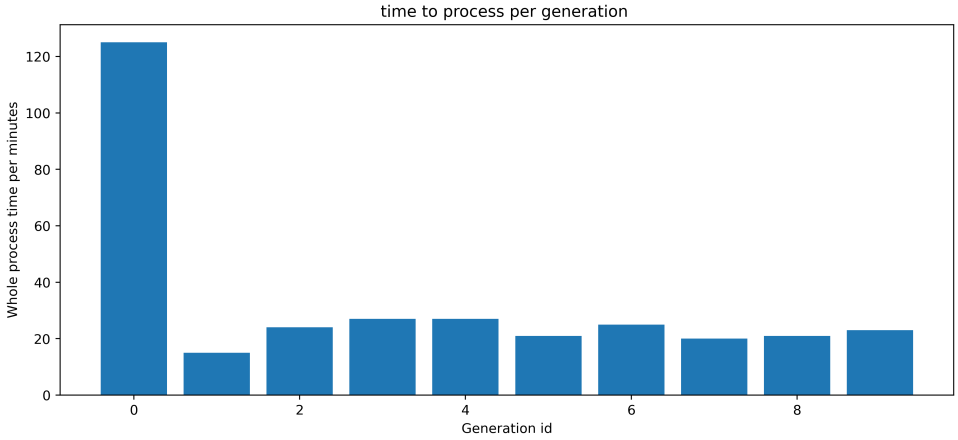

Figure 1: Duration in minutes per generation with DIVERSITY-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 , Smart archive

3 Details of the Duration

The first generation in Figure 1 was different from the other generation. If you look at the first generation, you will see the big difference between the first and others. For more information, we divided the generation into three-partparts:1- generating seed and population 2 - evaluation of the population, 3 - calculation of spareness and selecting the archive. Figure 2 is showing the result.

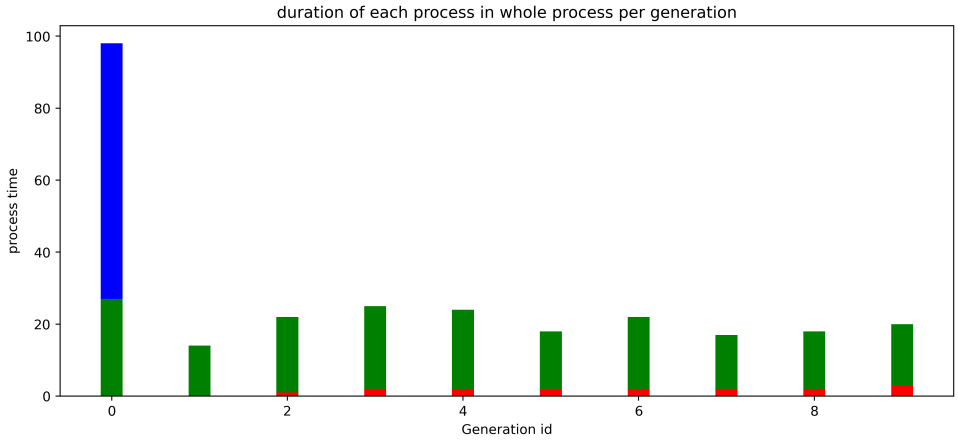

Figure 2: details of duration per generation

As we can see in the figure, there is a time-consuming process in generating the seed, and the population in the first generation is different from other generations. It happens because, in the Diversity generator way, the seed will always test to be a successful version(the seed with the random road and random fog density don't go out of the lane and reach the last point of the road). Testing all the seeds takes a lot of time and, if we find a way to decrease the amount of the first generation, we save a lot of time.

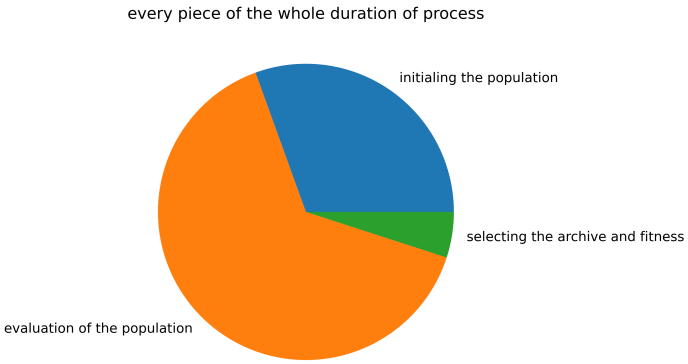

Figure 3: all the duration details together

As you can see, generating the population and seeds has a considerable proportion of the whole process time.

4 Fog density amount during the process

In this section, we have a significant purpose. We want to see the various fog amounts in the individuals. Every individual has a pair of failing and successful members. We take all the individuals in the archive and show THE Fog Densities of them in the Figure 4, the blue is for the successful fog amount, and orange is for the failure fog amount.

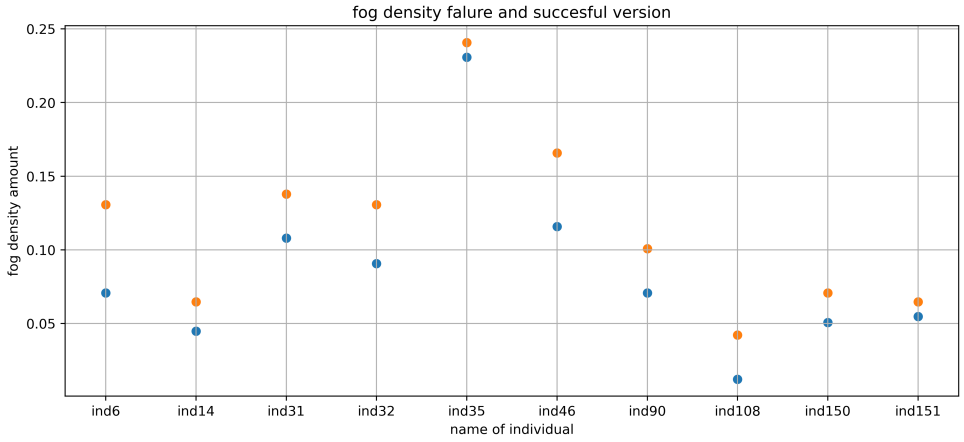

Figure 4: Fog amounts (successblue and failureorange) in every Individual of archive with DIVERSITY-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 , Smart archive

The whole process population was made with fog density in the range of (0,1). In Figure 4, the range of fog densities shorter than the whole range, most of the numbers are in a particular range.

So we decide to move forward with this attitude. We want the enormous archive for the new experience. Changing the generator approach for the seed and changing the attitude for keeping the archive was the solution. We use the RANDOM Generator to have a giant archive. It captures all individuals which have the pair of success and failure. Figure 5 is set with the same configuration of the last experiment, only the generator name change From Diversity to RANDOM and the way of keeping an archive which is changing from the Smart archive to the Greedy archive. And it keeps all the individuals which have the succesful and failure pair.

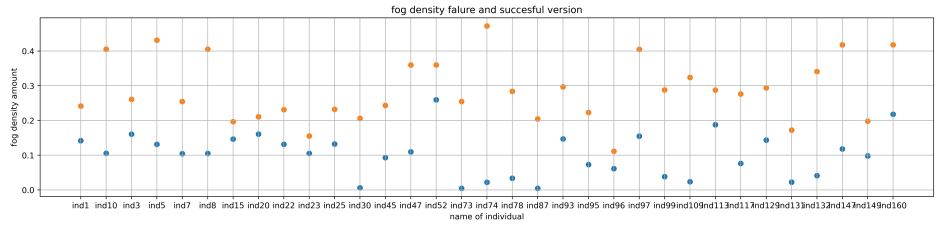

Figure 5: Fog amounts (successblue and failureorange) in every Individual of archive with RANDOM-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 , Greedy archive

As you can see in Figure 5, all of the fog amount (the failure and success) is in the particular number range. With this experiment, it will easy to find out the range of the successful version. If we understand the range of fog density of successful members, we can generate some limitations in generating the seed and making the time-consuming shorter. With a shorter range, it will cause more probability of success. In the first range of fog density in the seed, the program will make fewer attempts to find the successful member. That means reducing time in the first generation(which has a lot of time producing the seed). In the following Figure, we eliminate the failure members and put the successful members in the plot.

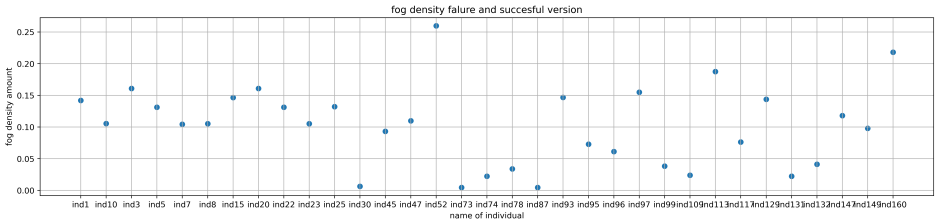

Figure 6: Fog amounts (successblue in every Individual of archive with RANDOM-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 , Greedy archive

The range of the fog density amount is between (0.021, 0.26). It means something. So if I want to set one threshold for this amount according to our assessment, we can put it between (0,0.35) because there is no successful fog density amount even close to 0.3. we can say we can generate the seeds(which have to be successful) in the range of (0, 0.35). We executed the program four with DIVERSITY-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 and Smart archive approach, But this time with 50 generations to see our assumption is valid or not.

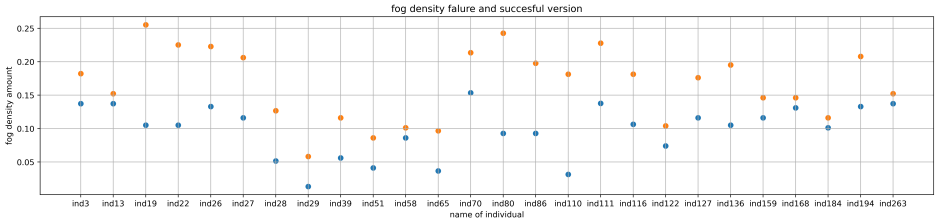

Figure 7: Fog amounts (success(blue) and failure(orange)) in every Individual of archive with DIVERSITY-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 , Smart archive

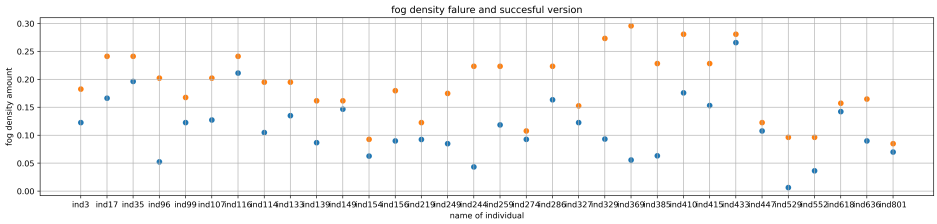

Figure 8: Fog amounts (success(blue) and failure(orange)) in every Individual of archive with DIVERSITY-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 , Smart archive

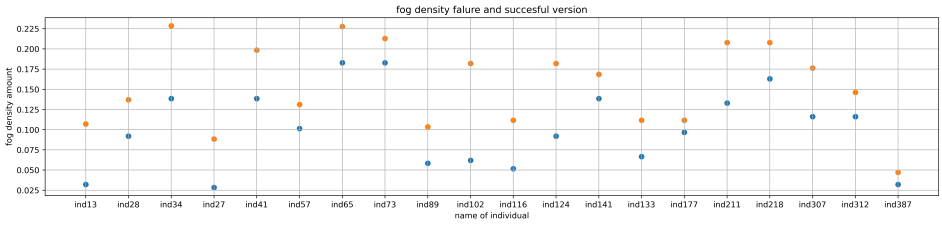

Figure 9: Fog amounts (success(blue) and failure(orange)) in every Individual of archive with DIVERSITY-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 , Smart archive

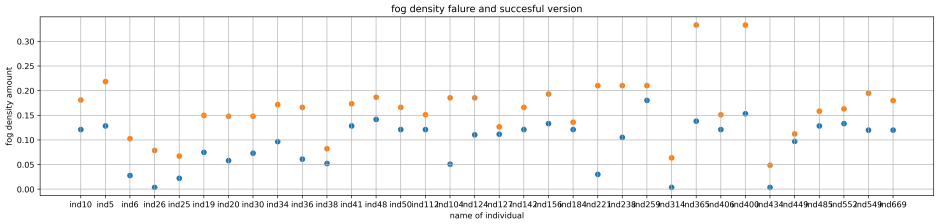

Figure 10: Fog amounts (success(blue) and failure(orange)) in every Individual of archive with DIVERSITY-generator, MUTATION-FOG-PRECISE=0.01, and MUTATION-EXTENT=6 , Smart archive

In all the experiments, the successful()the point with blue color) fog densities are between range (0, 0.3), so we understand from figures that if we set the fog density more than 0.3, the outcome will fail. With this information, we define two more variables for configuration. FOG-DENSITY-threshold-for-generating-seed-max and FOG-DENSITY-threshold-for-generating-seed-min. So in generating step, the program generates

the seed with fog density in the range between these two new variables. We set the range between (0, 0.35). We are sure that fog density will be no successful with higher than this amount because the highest value for the successful member in our experiments was 0.28 in the Figure 9.

So we define these two variable and set then to 0 and 0.35 .So the following is new configuration:

```
MUTATION_TYPE = Config.MUT_FOG
FOG_DENSITY_threshold_max = 1
FOG_DENSITY_threshold_min = 0
MUTATION_FOG_PRECISE = 0.0
MUTATION_EXTENT = 6
generator_name = Config.GEN_DIVERSITY
POOLSIZE = 20
POPSIZE = 12
NUM_GENERATIONS = 10
ARCHIVE_THRESHOLD = 35.0
FOG_DENSITY_threshold_for_generating_seed_max = 0.35
FOG_DENSITY_threshold_for_generating_seed_min = 0
```

5 Re-attempt the simulation with new configuration

After adding these two variable to the configuration, we want to investigate the result.

We re-run the program again with the same configuration as the first experiment (only two new variables are added). In Figure 11, we show the whole duration time per generation.

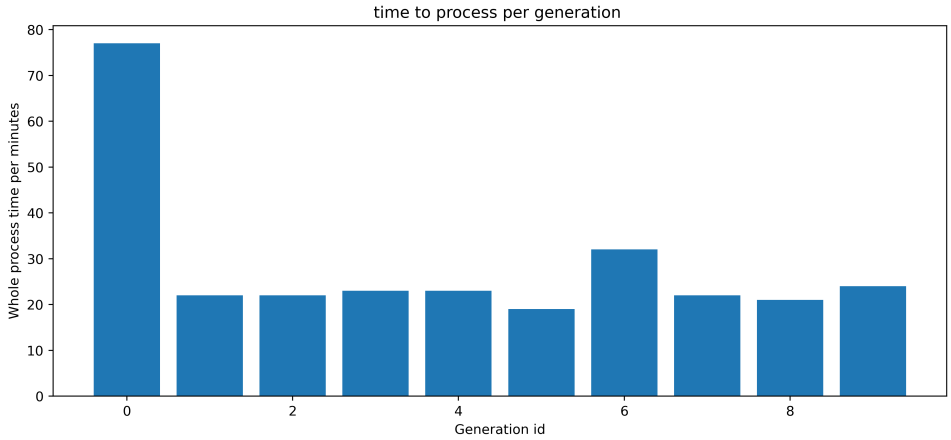

Figure 11: duration in minutes per generation

As you can see, the time-consuming in the first generation decreases a lot. So it means our implementation for reducing the time was successful. In the following Figure 13 we analyze the time of the program in more detail.

Figure 12: details of duration in minutes per generation

Generating seed and initial value in first-generation decreases. This part will not become a big part of the whole time process of the program anymore.

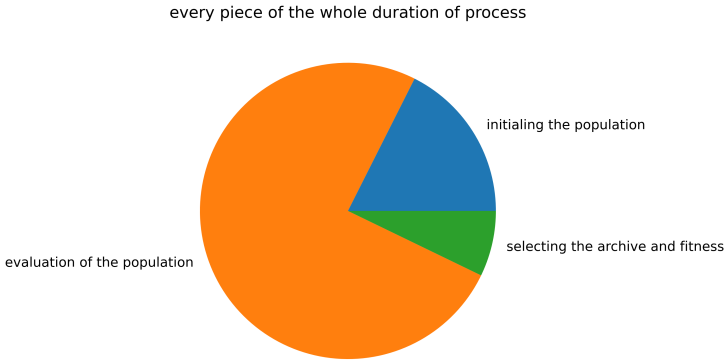

Figure 13: Proportion of each part of program in whole process time

Compared with the last pie chart, this one has the less blue part, which is related to generating seed and population.

6 Conclusion

One of the essential factors in our process is time. By defining the constraint amount for generating the seeds, we spend less time on the process. These types of limitation of variables for producing the range of reasonable amounts will help the program. It can also be implemented in other operators with diverse quantities. At first, we should find the correct range, then put it in the constraint to reduce the time.