Emergent computing for Optimization – SET11508 2019/2020 – Alessio Williams Gava 40212064

Introduction

The initial attempts at improvement were focused on the modification of the default parameters, understanding their effect on the race time and on the algorithm learning-speed.

Fitness function has been explored, trying to create a function that would encompass multiple variables, trying to reach for a good optimum.

A goal was set to refine the fitness function at its best: find a very good solution, in a mid-long time.

Different variants of the main components of the Evolutionary Algorithm have been explored to verify their effect on the population and help in choosing the most appropriate to the type of problem comparing their results obtained. Lastly, different types of evolutionary algorithms have been tested.

Method

The chromosome representation has been utilized the default one which is composed by the pacing and transition strategies because it was found that they summarize very well a chromosome ability in the race. For an experiment, the energies left were added to the chromosome to investigate their effect on the race competition, but not normally utilized.

The fitness function rewards the groups that took less time in completing the race. For groups that didn't finish, a high constant is added and value variable is calculated based on the distance left to the finishing line, rewarding the closest. The fitness for finishing groups is simply their time race taken, because it is wanted to improve their time. It has been investigated a possible fitness function that would take into account time taken and energy expended, trying to maximize the total energy expended. Unfortunately, this path was discovered to be unsuccessful because the algorithm would output mostly non-finishing groups because they usually use all the energies of a cyclist. Additionally, it was found challenging the correct comparison between energy and times since it would need to normalize the values and it is hard to perform it correctly without one variable dominating over the other. The goal of the project is minimizing the time taken, not minimize the energy left. Being a black box problem, it is not known how exactly the energy affects the finishing time, so it is incorrect assume that energy is the only cause, ergo it has been discarded from the fitness function.

The algorithms implemented are: simple Evolutionary Algorithm, Hill Climber, Simulated Annealing, Island model. After implementing these algorithms it was understood that the most suitable are the ones that can easily work with a rough search space. The hill climber is not very suitable and the simulated annealing works slightly better. The EA and the Island are the most suitable because there were implemented functions to diversify population and escape optimums.

In most of the algorithms implemented are present similar building blocks functions as: initialization, selection, crossover, mutation, replacement. Mostly for EA and the Island model, different approaches of these functions have been implemented because some methods adopted are more suitable for certain type of problems and it was wanted to record which were the best combination of building blocks to solve the racing problem. More specifically, have been attempted:

- selection: tournament, fitness proportionate, ranking, species.
- crossover: uniform, arithmetic, npoints
- mutation: gaussian, random

- replacement: random, tournament, oldest, worst

Diversity mechanism were adopted as island model, population replacement and infusion.

Hill Climber is a simple implementation which focuses on the exploitation of results, there is not much exploration in the algorithm. On the other hand, the Simulated Annealing is more explorative in its default version, but for the coursework it has been implemented with a rehaetable temperature which will get reheated multiple times to a lower maximum point every time, until it will not get reheated anymore.

The initial Island model implemented is composed by a number of simple EA which after every set of iterations will exchange some of their members with their neighbor island. This is done one island after the other, in order to make it possible that an individual from island 1 has the chance to be moved further than only 1 island. Afterwards, each island assumed a specialized task using different parameters and building block functions.

Experiments

In this section, there will be presented the experiments undertaken in chronological order to show the findings and their relative choices to improve the results obtained.

As first step an average best finish time was calculated using the initial clean project. Evolutionary Algorithms being stochastics, are needed to be run multiple times to take advantage of their randomness and draw their performance. The usual experimentation approach followed was to run for 1000 iterations for 10 times.

The performance measure utilized are mean to measure the centrality, median, standard deviation to measure the spread, and success runs which is based on whether a group finished or didn't finish the race. When comparing methods, the most regarded measure is the mean because the aim of the experimentation is to find a very good solution. The results presented do not comprehend results of non-finishing groups because it is not possible to utilize their finishing time (infinite). Only the finishing groups results are taken into account;

The initial project has been run following the methodology above described and it scored as follows

Mean	Median	Std Deviation	#Success Runs
235.7778	235.7735	0.1077	10

Table1. Initial clean project result – pop = 25 mutationP = 0.5 #mutatedGenesMax = 6

The initialization of the population plays a key role on the time taken of the algorithm before it finds a good result. The two main variables to be initialized in the cyclist population are the *transition* and *pacing* strategy. To be noted is the range of valid pacing variables: they span between 200 and 1200 Joules. The initial project was set with a default pacing strategy with all the values either being 300 or 350. It has been discovered that this technique is reasonable to achieve in a very quick time a good result, in fact out of 20 groups (population size), in average no group would be initialized in a manner that they would not finish the race. The average initial standard deviation of the groups' result is usually quite low, meaning that the initialized population is not very spread, having many similar instances, risking to easily get stuck in a local optima from the beginning. In fact, the results show that most of the time there is not big change from the initial population.

A different approach has been taken, using a random initialization into a specified range of pacing strategy values. Initially, the range used was the valid values, but it was discovered that the high values will have a negative impact on the finishing groups, creating many cyclists that would not

finish the race. Ideally, this would be an ideal initialization strategy because it highly empowers the exploration of the algorithm but the result obtained were poor, where in 25 runs of 250 iterations from 200-1000 initialized population, none were successful outputting a valid solution. After examination of the children created after each iteration, it was noticed that they do not differentiate a lot from their parents because the mutation implemented is not working on the pacing strategy. Additionally, the fitness function used is not appropriate for the type of experiments carried out. The problem resides in the initialization of a high number of groups that would not finish the race because of high energy consumption. The original fitness function used, would output a default value as result for any group that didn't finish, making it not possible to distinguish from each group. In case the initial population contains only groups that didn't finish the race, the first group would be both the best and the worst.

In the next improvement, it was decided to incorporate information about the percentage of completion of the race in order to create a more specific fitness function. More specifically, groups that finished would get their time taken as fitness function and the others would get a high value plus a variable dependent on how much percentage was missing before finishing the race. It was run for 25 times with 250 iterations and no runs outputted a valid solution. The following results are from the usual test run using same pacing initialization and mutation rate {200,1200}, initialization {200,1000}

Mean	Median	Std Deviation	#Success Runs
225.0125	225.0125	0.5575	2

Table 2. Mutation rewards closer non-finishing groups. 10 runs, 1000 iterations, init {200,1000}

A similar problem remained, in which if the initial population contains many high energy values, it would be very difficult to escape from this local low-optima. The algorithm would find an optimum of the groups that did not finish, outputting the best group with "infinite" time taken because did not cross the finish line. Three possible solutions to this problem were tested:

1) Reduce the range of pacing strategy which most probably will create an initial group in the population that manages to finish the race. They will influence the population towards the optima dominated by finishing groups. This solution is viable but is limiting the exploration of the possible groups.
2) Modify the mutation rate maximum which specify how many items can be altered in the pacing strategy. Setting a high mutation rate maximum items, will make possible to mutate a child very differently from their parent, creating an individual completely outside the local optima where the parents are set, increasing the exploration property of the algorithm.
3) Modify the mutation function to make it more efficient in creating new children that are going to be finishing groups. In case a child is created and does not finish the race, choose randomly how many genes to mutate and generate a random value that is less than the actual energy used.

With solution 1, as predicted, improvements were gained.

Initialization upper bound	1200	1000	800	600
#Successful Runs (25 total)	0	0	7 (28%)	25 (100%)

Table3. comparison of successful runs using different pacing initialization upper bound. 250 iterations, mutation {200,1200}, 4 max possible mutation genes

As it can be seen, the lower the bound is set, the more successful runs will be obtained. They are still not very successful in general, this is also because the mutation upper bound was left to 1200, but if

it is also lowered, valid results will be outputted more frequently. The 25 runs results show mostly the chance in getting an initial population with finishing groups. Instead, below it can be seen results obtained after a longer iteration run, where the average successful runs are higher because the algorithm had more chances to randomly mutate the children and find good solutions.

Initialization upper bound	1200	1000	800	600
#Successful Runs (10 total)	0	2 (20%)	5 (50%)	10 (100%)

Table3. comparison of successful runs using different pacing initialization upper bound. 1000 iterations, mutation {200,1200}

With solution 2, it was attempted to escape local optimum created in non-finishing groups using a higher number of genes to be mutated. It was applied to the experiment with initialization {200,1000}

#Possible mutation genes	2	8	15	20
#Successful Runs (25 total)	1	1	3	0

Table3. comparison of successful runs using different max possible mutation genes

The results obtained were not the expected ones, defeating the theory that it is needed many mutated genes in order to escape the optimum of non-finishing groups. This method was not pursued further.

With solution 3, it was understood that the pacing strategy is the most influential in creating valid groups, so it is tried to lower their energy usage. It had the best successful runs rate, in fact for every run a valid result was outputted.

Upper initial bound	1200	1000	800	600
Mean	221.1738	218.5698	219.2138	217.69
Median	221.996	218.3015	218.696	217.703
Standard Deviation	3.770364	4.445705	3.248659	1.004343
Success Run	10	10	10	10

Table4.

After evaluating the results obtained, it was decided that utilizing an initialized population with lower top pacing strategy bound {200,1000} was more ideal because it reflects better the goal where is wanted to get a good solution in a mid-long time even though it affects the generality of the algorithm, the gains seen from the results are quite evident. The mutation range is still kept at {200,1200} even thought it was found that it could help in finding faster solutions but doing so it would for sure exclude possible solutions because high values pacing strategy would never be assigned without mutation.

Afterwards, EA functions building block were investigated to evaluate possible alternatives.

CROSSOVER

The crossover was not applied to both the genes, only to transition, so it was modified and also applied to the pacing strategy, expanding the exploration of the algorithm because it would create children different from their first parent.

Mean	Median	Stddev	Successrun
221.0369	221.186	2.282569	10

Table5. using 2points crossover also for the pacing strategy, upper init bound 1000, mutation max genes 4

Comparing to the previous result obtained with same parameters, it can be seen that in general the algorithm had a lower performance but its spread is lower, having results time more similar. This seems to suggest that without using the crossover on the pacing, the algorithm is more subject to be stuck on local optimum, but in case a parent is a member of a very good optima, it is possible to exploit it (the same goes in case the parent is not very good optima). On the other hand, using the crossover suggests that a good solution is found, but the exploitation of the algorithm is lesser, not reaching extremely good solutions. A slight modification has been adopted in which the point of crossover is the same for both pacing and transition. This has given quite a better result, losing 2 successful runs but gaining a few seconds.

Mean	Median	Stddev	Successrun
218.6748	219.4765	3.189291	8

Table6. using 2points crossover also for the pacing strategy. Same point pacing and transition

EARLY CONVERGENCE – DIVERSITY

After analyzing the population while the algorithm is running, it was evident that there was a problem of early convergence, in fact the population would converge to a single record after 150 iterations. This would create a very static population in which crossover would be useless because it would be based on the same parent. A few solutions were attempted and the replace function has been modified from the replacing the worst population individual to a more explorative function as replace tournament. Replacing by worst would find a solution very quickly, but then it would get stuck in a local optimum, instead utilizing the tournament it was noticed that the population would retain more diversity throughout the algorithm because the lower fitness individuals have less chance to be replaced and might be able to reproduce, creating different children.

#Tournament elements	Mean	Median	Stddev	Successrun
3	222.1232	222.601	3.339348	9
5	221.066	219.9755	3.791306	10
10	220.153	220.8	2.952033	10

Table7. replace using tournament

From the results it can be seen that it was not much of an improvement from using the replace worst. In fact, examining the population, it is still converging into one dominating result with sporadic non-finishing groups. The higher the tournament size, the best result obtained, and at max size it would be like implementing a replace worst. At this point it was needed an explicit diversity mechanism to keep under control the convergence. Higher diversity was achieved utilizing infusion and population replacement. After a certain number of iterations, randomness is added using infusion which will replace the worst individuals with new random ones. The standard deviation of the population is calculated at every iterations and in case the deviation is stuck for a long time, randomness will be inputted in the population with recreating a new population with a small part random and a part created from best parents crossover.

Mean	Median	Stddev	Successrun
218.828	218.101	3.384218	10

Table8. Using replacing population after 150 same iterations and input new random every 50 iterat

Diversity was calculated using the standard deviation between each members of the population.

To enhance the exploration of very good results, it was applied a different mutation and crossover for finishing groups, while for the others the standard mutation that tried to lower the high energy values have been used. The new mutation is a gaussian mutation which outputs similar values for

each gene plus a range that is halved for every 50 iteration in which there was no change in the population. The crossover used is algorithmic, outputting a mean value between the parents used. The mutation should be helpful in pointing down the values created, trying to avoid very different values that are only creating non-finishing groups.

Gaussian Std Dev	Mean	Median	Stddev	Successrun
300	216.3948	216.7085	2.515523	10
200	217.6077	217.581	1.598517	10
100	218.0777	217.875	1.637049	10

Table9. Using mutate gaussian variable based on number of same iterations. Halved every 50 iterations with population that didn't change.

The most improvement is found using an initial larger gaussian standard deviation even though the spread of its results is higher compared to the others. It seems that a very small range (100 std deviation) is not large enough to find many improvements.

Other functions that were mentioned in the method section and have not been presented in this section, can be viewed in the provided code. They have been omitted for space reasons and because there was no improvement compared to the ones proposed.

Conclusions

The best solution was found using the last presented algorithm: 212.734 secs. 20 population, 1000 iterations, Initialized pacing strategy {200,1000}, transition strategy randomly initialized, tournament selection, 2 parents, 1 children, 2points crossover with same point for pacing and transition, random low mutation for non-finishing groups, gaussian mutation with 300 standard deviation for finishing groups, replace worst, infusion random every 50 iterations, replace population with 20% random individuals after 150 same iterations.

Improvement was achieved from the initial solution 235 secs, 216.3 was the best mean time found, gaining 19 seconds. The goal was also achieved because a good solution can be found in a mid-long time (1000 iterations in 1min13secs). Comparing the results obtained with the best results proposed from the paper "Evolving Pacing Strategies for Team Pursuit Track Cycling" it can be understood that what it was though as being a very good result, is easily beaten from the paper's authors, showing results obtained as 202 secs. Some choices adopted lowered the generality in creating a general purpose algorithm because specific functions were created for this problem and parameters were set after trial and error and not automatically from the application.

Better solutions can be found if the algorithm is let to run for a longer time because using the large initial range, there is an initial search in which the algorithm has to deal with non-finishing groups and sometimes it will not find a valid group until the 200th iteration.

Future work

Would be very beneficial to be able to take advantage from groups that didn't finish the race rather than just ignoring them. A fitness function that could take into account some of their good features is needed to be created, but it is challenging since the comparison between non-finishing and finishing is not possible using the usual fitness function based on the time taken to finish the race.

The Island algorithm seems the most promising between them all because of its structure suitable to assign a task to each island, solving the problem of having different individuals that cannot be compared (non-finishing groups and finishing).