COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

21/22 PROJECT GUIDELINE

Contact:

Berfin Sakallioglu, NOVA IMS

bsakallioglu@novaims.unl.pt

1. PROJECT DESCRIPTION

You are asked to apply your Genetic Algorithm (GA) knowledge to solve an optimization problem. You can choose one of the problems suggested in this document or propose your own project. If you choose to continue with your own project, the idea must be approved.

All the communication will be done via email to: bsakallioglu@novaims.unl.pt except the "Group Choice", "Project Topic" and "Project Delivery" activities which are already on Moodle.

After the "Group Choice"; for the "Project Topic" and "Project Delivery" steps, only one student from each group will be performing these steps since your groups are saved in the system. These are two separate steps, so there is no need for the same student to do them.

It is expected that you already formed your groups on Moodle but if you did not until now, please complete it until the 16th of April. You can take the last week of the trimester and Easter Break as an opportunity to make research about the topics and choose a topic for your group.

Until the 30th of April, you are expected to complete the "Project Topic" activity on Moodle by choosing your group and topic. As stated before, if you will be working on your original idea, you should select the "other" option, send me an email before the 30th of April, and receive a confirmation from me.

This is a group project, as such you are encouraged to do the project in a group. The maximum number of members for a group is 4 - there are no exceptions for this upper bound. Groups of 3-4 people are encouraged, but not mandatory. If you are having trouble finding a group, please email me.

Due to the global situation, you can do the project alone if you can reasonably justify that you are unable to do the project in a group and receive a prior authorization from me. You can send me an email about it.

For any names not included in the "Group Choice" (even if you are doing the project alone) by the delivery deadline, we will assume that those students have opted to not partake in the project and will receive an automatic grade of 0.

NOTE: You can pass the module without undertaking the project, as it only counts for 30% of your grade. However, this is not encouraged as you will forfeit 30% of your grade.

2. PROJECTS

The projects suggested below are all expected to be solved using the Charles Genetic Algorithm Library developed throughout the semester. If you decide to undertake a more complex custom problem, you are free to use external GA libraries - although the library developed in class should be solid enough for any GA use-case.

Everyone is expected to demonstrate a good understanding of optimization problems and Genetic Algorithms. These are demonstrated by representing your problem well pragmatically, both as the representation itself as well as the definition of your fitness function. These are some basic steps expected from each project:

- Implement several selection methods
- Implement several mutation operators
- Implement several crossover operators
- Implement code abstractly, making it work for both minimization and maximization (when applicable) *

*Even though your problem will likely only be either minimization or maximization, you should ensure both are implemented. Remember, most of your work will be on implementations in the Charles library - which should allow both maximization and minimization to work.

Illustrate and compare the performance of different approaches to your problem in your report. The report should focus exclusively on your implementation and the results you obtained, hence, have no literature review or "Introduction to CIFO/ Genetic Algorithms" part. It should only focus on your project.

Try to illustrate decisions taken both in written form as well as using plots and illustrations. Your decisions should be based on statistical validation (i.e. run each selected configuration 100 times and compare the performance of your different configurations statistically).

2.1. TSP STUDY

Extend upon the code implemented during the semester. Implement and study which types of operations (selection, crossover, mutation etc.) work best for the TSP problem, justify how you determine which operators works best.

The University of Heidelberg offers a collection of TSP data along-side the best score for each set: http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html. You should use multiple TSP datasets to benchmark your performance (at least 3).

2.2. SUDOKU SOLVER

Use GAs to evolve solutions to a Sudoku problem. Think about how to represent such a problem in your code, and how to design a fitness function and appropriate genetic operators. You should be able to implement Sudoku relatively easily and without using external libraries. How your Sudoku implementation looks visually is wholly irrelevant, this will not be considered for your grading process. You just want to be able to find good solutions - ideally solving Sudoku problems completely.

2.3. LEARNING TO PLAY SNAKE

Teach a model to play Snake. For this project you will likely want to build a very basic Single-Layer Neural Network, which takes as input feed from the game (i.e., what the snake can see). The output of the network will be the various movement options the snake can take (UP, LEFT, RIGHT). Use GAs to optimize a set of weights for your Single-Layer NN.

For this project, you will want to work with a snake implementation in pure Python - to make sure your application works on all OSs (Windows, Max, Linux).

See https://www.pygame.org/ and https://www.pygame.org/tags/all for already implemented games. You are not expected to implement the game on your own - but you can obviously do so if you wish -. Still, keep in mind that the visual part of the project is wholly irrelevant like the other cases, it will not be considered during your grading process.

2.4. MORE IDEAS TO SEARCH FOR

Keep in mind that Travelling Salesperson Problem has some different problem definitions. You can look for these special cases for TSP.

One other similar (to TSP) case that you might want to consider can be the Vehicle Routing Problem (VRP).

In addition, another similar case which can be framed differently is a case related to electronics, if you are interested. When soldering chips on boards, the distances of each element should be optimized based on several conditions like the speed of the electricity.

The Snake game is just an example. You can create Neural Networks that plays other games as well. Using the pygame links given above, you can try to discover and choose other games that you can work on. (The conditions about the visuality will be valid for any case. We are not interested in how good the game is looking for the grading process.).

If you are interested in computer vision, you might want to consider working on Convolutional Neural Networks and GAs.

These examples can be extended, and original ideas are always encouraged.

3. PROJECT DELIVERABLES

- Code Implementation (git)
- Report (pdf)

The code of your project should be delivered through a git repository. You may use any git service you like (GitHub, GitLab, etc.) or even host the git yourself - as long as it can be cloned (https://git-scm.com/docs/git-clone). All submitted projects will be cloned at exactly 00:00, the minute following delivery deadline, so no commits or changes made to the original repo after the deadline will be considered for evaluation.

NOTE: If your repository is private and it is not possible to clone it, you will receive 0 points.

Upload your report to Moodle before the deadline with the link to the git repository on the first page. Do not forget to include your report in git as well. Do not forget to check if the link on the report is really working. Otherwise, I will not be able to grade your project and you will receive 0 points.

The report should be delivered as a PDF. Your report should not exceed 5 pages of text (i.e., if you use a cover page, with just the title and team members, you can have a total of 6 pages: Cover Page + 5 pages of text). In this module we trust you (or perhaps evolution?) enough to allow you to choose font and font size yourself*. Just keep it easy to read and relatively professional.

You will want to show what implementations you have done, and how the various implementations differ in terms of performance. Justify why and how you took decisions during your project.

Your report should mention the division of labor in your group.

*Reports delivered in font Comic Sans MS will receive 0 points.

A strong project will have implemented three selection approaches (i.e., FPS, Ranking, Tournament), three mutation operators, and three crossover operators. The operators must be coherent with the type of problem being addressed. These implementations will then be benchmarked and statistically compared. The code will be well formatted and commented. The report will be clear and easy to understand with useful illustrations. The following questions will be answered in the report:

Why did you choose the representation you did?

How did you design the fitness function? Did you try using different fitness functions, to see the impact on your GA?

Which configurations worked best together? How many did you try, and how did you determine the "best" one?

Do different operators affect the convergence of your GA?

Have you implemented elitism? Does the inclusion or exclusion of elitism impact your GA?

Are your implementations abstract and work with both minimization and maximization?

Do you get good results for the project you chose? What could be improved?

3.1. PROJECT DEADLINE

The project deadline is the 31st of May at 23:59, that is in 63 days from the publication of this handout. Given the possibility of passing this course without undertaking the project, there is no ability to submit projects late, as any student who has not delivered theirs by the deadline will be considered as having opted to pursue solely the exam. Thereby, forfeiting 30% of their grade.

3.2. RECOMMENDATIONS

You are strongly recommended to use git versioning control as you develop the project, to avoid a sudden loss of all code on the day of delivery of the project. If you are unfamiliar with git, see: https://git-scm.com/book/en/v2 and https://youtu.be/2sjqTHE0zok.

Keep code simple and commented. Do not copy code from Stack Overflow without knowing what it does! Obviously, take inspiration from Stack Overflow whenever you are stuck. But try to understand how the code works and adapt solutions to your code. If you have parts of code that stand out from the rest of your code, and that particular part of the code is not commented - you will be convened for an in-person project defense.

If you have questions or require clarifications, use the practical lectures, or contact me.

3.3. EVALUATION CRITERIA

• 40% Code Functionality: Assessment of the functionality of the code, the correct implementation of the GA techniques and interaction of the various components. While you will work on your projects in group, you should ensure code is well-structured and works well in conjunction.

• 15% Code Structure: Assessment of the code quality and structure, the code should be clear and well-commented to be inspectable and understandable by anyone.

• 35% Report Content: Assessment of the contents of the report. The report should illustrate your project and not contain vast literature reviews (this is a report not an essay). A good report will justify choices undertaken in the project as well as demonstrate findings and results.

• 10% Report Structure: Assessment of the report structure, the report should be well organized, well written and grammatically correct. Proof-read your report well. An allowance for syntactical English errors is given.

*In situations where there are significant inconsistencies between the code and the report, or plagiarism doubts arise, your group will be called to defend the project in person.

*Keep in mind that we have no tolerance for the plagiarism. In NOVA IMS, plagiarism is punished with the failure of both Epoch 1 and Epoch 2 of the exam (failure of the whole academic year).

4. FAQ

Any questions answered below, will not receive an answer if asked via email.

My *insert excuse here*, can I get an extension on the project deadline?

No.

Can we do the project in 5 people?

See answer above.

Can we do the project in 2, 3 or 4 people?

Yes.

Can I do the project alone?

Only if you emailed me and received a confirmation from me. See the last paragraphs of the Project Description above.

Hi Berfin, it's the day before the deadline and I forgot to tell you I was going to do the project on my own. Is that fine?

No.

My *insert excuse here*, I lost all my code. What should I do?

Use a versioning control system from the start. See git.

Does every member of the group have to choose the topic from Moodle/ upload the deliverables on Moodle?

No. Just one person from the group will choose the topic/ upload the deliverables. It is not necessary for the same person to do these two steps.

Will we upload the code on Moodle?

No. You will only upload your report which will have the functioning link (it is crucial, please check it very well) for your git repository on the first page.

Can I submit my project multiple times before the deadline?

Yes, I will not start grading or reviewing projects before the deadline. So, until then you can resubmit as many times as you want.

Can I get full grades by doing the TSP implementation?

Yes, every project has the potential to get full points. However, the simpler the project the more implementations you are expected to do. See Project Deliverables above.

Can I change the name of the library?

Yes, if you do not want to use the name Charles (inspired by Charles Darwin) you can rename it to whatever you want.