COMPUTATIONAL INTELLIGENCE FOR OPTIMIZATION

22/23 PROJECT GUIDELINES

Contact:

Berfin Sakallioglu, NOVA IMS

bsakallioglu@novaims.unl.pt

1. PROJECT DESCRIPTION

You are asked to apply your knowledge about Genetic Algorithms (GAs) 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.
- The group members and group topics will be chosen by the students filling the following sheet. The oral defense schedule will be published there as well: https://tinyurl.com/35tr39u7
- Project delivery will be done via Moodle by only one member of the group.

You can take the Easter break as an opportunity to make research about the topics and choose a topic for your group. Until the **8th of May**, you are expected to complete the sheet given in the link by choosing your group and topic. As stated before, if you will be working on your original idea, you must send an email and receive a confirmation before the 8th of May.

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. Groups of 3-4 people are encouraged, but not mandatory. If you are having trouble finding a group, please send an email. If you can reasonably justify that you are unable to do the project in a group, send an email and you can only proceed after receiving a prior authorization.

For any names not included in the sheet (even if you are doing the project alone) by the delivery deadline, it will be assumed 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 35% of your grade. However, this is not encouraged as you will forfeit 35% 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 prefer to create your own library from scratch, you are free to do so. 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 GAs. These are demonstrated by representing your problem well pragmatically, both as the representation itself as well as the definition of your fitness function. Keep in mind that the simpler the project the more implementations you are expected to do.

These are some basic steps expected from each project:

- Implement several selection methods
- Implement several mutation operators
- Implement several crossover operators

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. STIGLER'S DIET PROBLEM

Utilize GAs to evolve solutions for Stigler's Diet Problem, which is named after the Nobel laureate in economics, George Stigler. Stigler devised an economical method to satisfy fundamental nutritional requirements based on a specified set of foods. The primary aim is to minimize the overall cost while satisfying the nutritional requirements. The dataset for this problem is available here: https://developers.google.com/optimization/lp/stigler_diet, but you can increase the number of foods, nutritional requirements, and constraints, as well as introduce uncertainty to modify the problem's complexity and facilitate comparisons.

2.2. IMAGE CLASSIFICATION

Employ GAs to optimize the weights of a single-layer neural network (NN) for image classification purposes. It is sufficient to design a basic NN as the primary focus will be on the GA optimization process. To evaluate the model's performance, one may choose a simple benchmark dataset such as CIFAR-10. This dataset contains 60,000 32x32 pixel color images distributed into 10 classes with 6,000 images per class. The classes are mutually exclusive and correspond to airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. https://www.cs.toronto.edu/~kriz/cifar.html. You are free to use any other dataset like Flowers, MNIST Handwritten Digits, Fashion-MNIST etc.

2.3. FROZEN LAKE PROBLEM

Use GAs to evolve solutions for the Frozen Lake Problem. The Frozen Lake environment is a 4x4 grid that consists of four areas: Safe (S), Frozen (F), Hole (H), and Goal (G). The agent moves around the grid by taking four possible actions: Up, Down, Left, and Right, aiming to reach the goal while avoiding the holes in a minimal number of moves.

You are not expected to implement the game on your own - but you can obviously do so if you wish - OpenAI Gymnasium's Frozen Lake Environment can be used instead. https://gymnasium.farama.org/environments/toy_text/frozen_lake/#frozen-lake. 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.

The game also has a slippery ice option, which makes it more challenging for the agent as there is only a 33% chance of successfully performing the intended action. In 66% of the time, the agent will perform one of the other movements. It may be helpful to begin with the non-slippery version and subsequently test the slippery condition.

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). There are more optimization problems such as Job Shop Problem, Timetable Scheduling etc. that might be interesting.

You can create Neural Networks that play games using the pygame module. 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.) 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.

Upload your report to Moodle before the deadline with the **link to the git repository on the <u>first</u> 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. If your repository is **private** and it is **not** possible to clone it, it will not be possible 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 **must** 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 **28th of May at 23:59**, that is in 62 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 35% of their grade.

3.2. PROJECT DEFENSE

There will be a mandatory oral defense on the **31st of May**. The exact time schedule for each group will be published closer to the date via the same sheet where the group lists are written. You will not be preparing any presentation for the defense. All the group members **must** join the defense and questions will be asked to each student about their project steps. These might include questions about the code implementations, experiments, or the results.

In situations where there are significant inconsistencies between the code, the report and/or the defense and/or plagiarism doubts arise, the defense duration will be extended also with the presence of the lecturer of the theoretical part.

3.3. RECOMMENDATIONS

You are 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 and/or ChatGPT without knowing what it does! Obviously, take inspiration 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 – this will be investigated in during the project defense.

If you have questions or require clarifications, use the practical lectures, or send an email.

3.4. EVALUATION CRITERIA

• 35% 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.

• 10% Code Structure: Assessment of the code quality and structure, the code should be clear and well-commented to be inspectable and understandable by anyone.

• 25% 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.

• 5% 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.

• 25% Project Defense: Assessment of the project defense. All group members must join the defense and present knowledge about every aspect of the project.

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

4. FAQ

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

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

Yes.

Can I do the project alone or with 5 people?

Only if you have sent an email with valid reasonings and received a confirmation.

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/I was going to implement my original idea/topic. Is that fine?

No, sorry.

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

See the answer above.

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

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

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. And remember, only one member of the group will upload the report on Moodle.

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 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.