Biologically Inspired Computation: Coursework 2
Implementing and Comparing Optimisation Algorithms

Due: Midnight, Monday 26th November 2018

This is individual assessed coursework. You are allowed to discuss this assignment with other students, but you should not copy their work, and you should not share your own work with other students. We will be carrying out automated plagiarism checks on both code and text submissions.

Special note for re-using existing code. If you are re-using code that you have not yourself written, then this must clearly be indicated, making clear which parts were not written by you and clearly stating where it was taken from. If your code is found elsewhere by the person marking your work, and you have not mentioned this, you may find yourself having to go before a disciplinary committee and face grave consequences.

Late submission and extensions. Late submissions will be marked according to the university's late submissions policy, i.e. a 30% deduction if submitted within 5 working days of the deadline, and a mark of 0% after that. The deadline for this work is not negotiable. If you are unable to complete the assignment by the deadline due to circumstances beyond your control (e.g. illness or family bereavement), you should complete and submit a mitigating circumstances application: https://www.hw.ac.uk/students/studies/examinations/mitigating-circumstances.htm

Learning Outcome: The learning outcome of this assessment is to increase your understanding of two of the most popular biologically-inspired optimisation algorithms, the genetic algorithm (GA) and particle swarm optimisation (PSO), and gain some insight into how they compare when evaluated on benchmark problems. It is worth 25% of your overall course mark.

Variants: This coursework has two variants. You should **choose one of these** to do:

Variant 1: This is intended for students who have previous experience of programming, and involves implementing both a GA and PSO and then comparing the performance of these two implementations on a small number of benchmark problems.

Variant 2: This is intended for students who do not have previous experience of programming, and involves doing a literature review and then comparing the performance of a GA and PSO in a more rigorous way using an existing software tool.

GA and PSO: Both variants require knowledge of GA and PSO. These algorithms were covered in the lectures. However, more information is readily available elsewhere. You might start by looking at the recommended course book "Essentials of Metaheuristics", which also includes pseudocode: https://cs.gmu.edu/~sean/book/metaheuristics/

Benchmark functions: Both variants involve evaluating GA and PSO on mathematical benchmark functions, where the goal is to find a global optimum, i.e. the point within the function's domain (i.e. its inputs) that minimises (or sometimes maximises) the output of the function. For an *n*-dimensional function, *f*, the goal is to optimise a list of its *n* real-valued numerical inputs, *x,* so that *f(x)* returns its lowest possible value. Information about benchmark functions is readily available on the internet. These are some places to start looking:
https://en.wikipedia.org/wiki/Test_functions_for_optimization , https://bit.ly/2EpTGPI

Comparing the performance of optimisation algorithms

Both variants involve comparing the performance of GA and PSO. When comparing algorithms, you have to be careful to do this in a fair way; otherwise your conclusions may not be valid.

Most importantly, the algorithms should be able to carry out the same number of solution evaluations when they are run. For GA and PSO, the number of solution evaluations is usually the population size multiplied by the number of iterations (or generations) the algorithm is run for.

GA and PSO both have hyperparameters that affect their performance, e.g. mutation rate, number of informants, and the various weights (acceleration coefficients) in the PSO velocity update equation. Before comparing two algorithms, you should expend equal effort trying to find values of their hyperparameters that maximise their performance.

Finally, GA and PSO are both stochastic algorithms, meaning that for the same problem and hyperparameter values, you will likely get different results each time you run the algorithm. Therefore, the distribution of results across a series of runs is more informative than the result of a single run. For GA and PSO, it is normal to give the mean result across at least 10 repeated runs.

Variant 1: Implement and compare GA and PSO

What you are asked to do:

1. Implement these two algorithms using a language of your choice
2. Compare your implementations using a small number of benchmark problems
3. Write a report and submit both this and your code to Vision

1. Implement these two algorithms using a language of your choice

- Your GA and PSO implementations should both represent solutions using fixed-length vectors of real numbers (i.e. they should not use a binary encoding).
- Your GA implementation should use two-point crossover and tournament selection, and a suitable mutation operator.
- For your PSO implementation, each particle should have a group of informants (these can be randomly allocated at the start of a run), and the velocities of particles should be updated using the equation described in the lectures.
- You can implement the algorithms using any standard procedural or object-oriented language, e.g. Java, Python, Matlab, C, C++. Both algorithms should be written using the same language and the same programming style. You should write the code yourself. If you copy any snippets of code from a book, the internet or elsewhere, this should be clearly indicated.

2. Compare your implementations using a small number of benchmark problems

- You should compare the performance of the two algorithms on a small number (e.g. 5) of benchmark functions. You should pick functions that have a variety of fitness landscapes.
- It's fine if you want to implement these functions yourself. However, you are encouraged to use existing implementations, and will not lose marks for doing so.
- A good choice would be to use functions from the CEC 2005 benchmark set, which is described in https://bit.ly/2EpTGPI. Java, Matlab and C implementations are available at https://bit.ly/2RRN7YS. There are also a couple of Python implementations available, e.g. https://github.com/dmolina/cec2005real and https://pydigger.com/pypi/optproblems
- Make sure it is a fair comparison (see information at the beginning of Page 2).

3. Write a report and submit both this and your code to Vision

Your report (up to a maximum of 3 pages in length) should:

- Briefly describe your implementations of GA and PSO, noting any interesting aspects.
- Briefly describe the benchmark functions you used to evaluate your implementations, and the motivation (if any) for choosing these particular functions.
- Report how well your implementations performed on these benchmarks. For instance, you might use tables that show the average results achieved for each problem by each algorithm over a series of (at least 5) repeated runs. You should also discuss what your comparative results tell you (if anything) about the strengths and weaknesses of GA and PSO.
- Discuss any observations you have about how changes to the various hyperparameters affected the performance of your implementations of GA and PSO. You do not need to give detailed experimental results for each hyperparameter, though examples would be useful.
- Relate your findings to the wider optimisation literature. For instance, try to find some papers where other people have compared the performance of GA and PSO and state whether their observations agree with yours.

You should submit both your report (as a **pdf** file) and your code (as a **zip** file) to Vision using the links provided. Grading will use the assessment criteria given in the table below.

Criteria	Weight	A (70-100%)	B (60-69%)	C (50-59%)	D (40-49%)	E/F (<40%)
Implementation of algorithms *(i.e. code for GA and PSO and its documentation, binding or implementation of benchmarks)*	50%	Algorithms are implemented as described in the coursework specification. Code is easy to read, well structured and appropriately commented.	Some minor issues in terms of how the algorithms are implemented, structured or documented.	Some significant issues in terms of how the algorithms are implemented, structured or documented.	Some major issues: for example, only one algorithm is implemented, or the algorithms are implemented using different styles and/or languages.	Some critical errors: for example, the code does not compile and/or run, or inappropriate algorithms have been implemented.
Comparative study *(i.e. choice and motivation of benchmarks, experiments performed, presentation of results)*	25%	Benchmarks and hyperparameter settings are well motivated and described. Suitable results have been collected and are clearly presented and meaningful.	Some minor issues in terms of the motivation or description of benchmarks and hyperparameters, the experiments performed, or the presentation of results.	Some significant issues in terms of the motivation or description of benchmarks and hyperparameters, the experiments performed, or the presentation of results.	Some major issues: the comparison is unfair, the experiments do not support the conclusions, or the study is not adequately described.	Some critical issues: the comparison is nonsensical or missing, the experiments are inappropriate, or the description of the study is uninformative.
Wider discussion *(i.e. intro, interpretation of results, conclusions, comparison against results reported in the literature)*	25%	Clear, insightful discussion that shows a good understanding of GAs, PSO, and the role and nature of benchmarks. Good use of the wider literature.	Generally clear and insightful, but shows some misunderstanding of GAs, PSO, or the role and nature of benchmarks. Adequate use of the wider literature.	The discussion is limited in terms of the depth or volume of understanding it demonstrates. Little use of the wider literature.	Some major issues in terms of depth or volume of understanding. No use of the wider literature.	No real demonstration that the subject matter has been understood, or very limited in its scope.

Variant 2: A tool-based comparison of GA and PSO

What you are asked to do:

1. Familiarise yourself with EvA2
2. Do some research into mathematical benchmark functions
3. Carry out a rigorous comparison of GA and PSO using EvA2
4. Write a report and submit it to Vision

1. Familiarise yourself with EvA2

For this variant, you will be using an existing software toolkit to compare GA and PSO. It is strongly recommended that you use EvA2, a GUI-based toolkit that includes implementations of a wide range of optimisation algorithms and numerical benchmark functions. If you would prefer to use another toolkit, please discuss this with your course lecturer.

- EvA2 is available here: http://www.ra.cs.uni-tuebingen.de/software/EvA2/download.html
- Documentation is here: http://www.ra.cs.uni-tuebingen.de/software/EvA2/shortdoc.html
- In order to run EvA2, your computer will need to have a Java runtime environment installed.

2. Do some research into mathematical benchmark functions

Mathematical benchmark functions are a common means of comparing optimisation algorithms. You should carry out a literature review of work in this area, with the aim of discussing:

- the motivation behind using them to compare optimisation algorithms
- the kinds of functions that are used, and their properties
- the aspects of optimisation that they are assessing
- any interesting findings of previous comparative studies; for example, is there any agreement over which algorithms perform well?

Make sure you correctly cite any papers you use, and make it clear if you use material from a source (even if you have paraphrased it). Try to cite at least 5 relevant papers.

3. Carry out a rigorous comparison of GA and PSO using EvA2

Using EvA2, and considering the findings of your literature review, you should compare the performance of GA and PSO across a range of different single-objective benchmark functions. The aim of this work is to gain an understanding of what kind of problems the two algorithms are good at solving, so you should aim to use a diverse range of (around 10) problems in your investigation.

- EvA2 includes standard implementations of GA and PSO. You can choose these from the algorithm selection dialogue box (which appears when you click the "…" button next to "Optimizer" in the "Optimisation parameters" tab). The GA is called GeneticAlgorithm and PSO is called ParticleSwarmOptimisation. Press the start button to begin a run.
- EvA2 includes a large selection of benchmark functions, including the standard CEC 2005 benchmark set, described in https://bit.ly/2EpTGPI. You can specify the problem by clicking on the "…" button next to "Problem" in the "Optimisation parameters" tab.
- As noted on EvA2's download page, you need to download both the base package and the problems package in order to access the full range of benchmarks, and start the program using the command "java -cp EvA2.jar:EvA2Problems.jar eva2.gui.Main". If you are unsure how to do this, talk to your course lecturer.
- To ensure your comparison is fair (see information at the beginning of Page 2), you should spend some time finding values for the various hyperparameters that work well, for both GA and PSO. Hyperparameter settings can be changed in the algorithm selection dialogue box

(which appears when you click the "…" button next to "Optimizer" in the "Optimisation parameters" tab).

- EvA2 can automatically generate results across a number of repeated runs using the same parameter settings. To use this functionality, set a number above 1 in the "Multi runs" property in the Statistics tab before pressing the start button.

4. Write a report and submit it to Vision

Your report (up to a maximum of 5 pages in length) should:

- Summarise the literature review you carried out into the use of mathematical benchmark functions to assess the performance of optimisation algorithms.
- Briefly describe the benchmark functions you used to evaluate your implementations, and describe your motivation (if any) for choosing these particular functions.
- Report how well your implementations performed on these benchmarks. For instance, you might use tables that show the average results achieved for each problem by each algorithm over a series of (at least 10) repeated runs. You should also discuss what your comparative results tell you about the strengths and weaknesses of GA and PSO.
- Discuss your observations, and give examples (in numerical terms of their effects), of how changes to the various hyperparameters affected the performance of GA and PSO.
- Relate your findings to the wider optimisation literature. For instance, try to find some papers where other people have compared the performance of GA and PSO and state whether their observations agree with yours.

You should submit your report (as a **pdf** file) using the link provided. Grading will use the assessment criteria given in the table below.

Criteria	Weight	A (70-100%)	B (60-69%)	C (50-59%)	D (40-49%)	E/F (<40%)
Literature review *(i.e. a survey of the use of mathematical benchmark functions for assessing optimisation algirithms)*	35%	Clear, insightful survey of the use of mathematical benchmark functions within optimisation. A number of good quality sources have been used and referenced.	Generally clear and insightful, but has minor limitations in terms of scope, content or understanding demonstrated.	The survey is limited in terms of the depth or volume of understanding it demonstrates. Limited use of the literature.	Some major issues in terms of depth or volume of understanding. Poor or inappropriate use of the wider literature.	No real demonstration that the subject matter has been understood, or very limited in its scope.
Comparative study *(i.e. choice and motivation of benchmarks, experiments performed, presentation of results)*	40%	Benchmarks and hyperparameter settings are well motivated and described. Suitable results have been collected and are clearly presented and meaningful.	Some minor issues in terms of the motivation or description of benchmarks and hyperparameters, the experiments performed, or the presentation.	Some significant issues in terms of the motivation or description of benchmarks and hyperparameters, the experiments performed, or the presentation.	Some major issues: the comparison is unfair, the experiments do not support the conclusions, or the study is not adequately described.	Some critical issues: the comparison is nonsensical or missing, the experiments are inappropriate, or the description of the study is
Wider discussion *(i.e. intro, interpretation of results, conclusions, comparison against results reported in the literature)*	25%	Clear, insightful discussion that shows a good understanding of GAs, PSO, and the role and nature of benchmarks. Good use of the wider literature.	Generally clear and insightful, but shows some misunderstanding of GAs, PSO, or the role and nature of benchmarks. Adequate use of the literature.	The discussion is limited in terms of the depth or volume of understanding it demonstrates. Little use of the wider literature.	Some major issues in terms of depth or volume of understanding. No use of the wider literature.	No real demonstration that the subject matter has been understood, or very limited in its scope.