1. Module number	*SET10117*
2. Module title	*Emergent Computing for Optimisation*
3. Module leader	*Emma Hart*
4. Tutor with responsibility for this Assessment Student's first point of contact	*As above*
5. Assessment	*Report*
6. Weighting	*75% of overall module total:*
7. Size and/or time limits for assessment	*Report – 6 pages*
8. Deadline of submission Your attention is drawn to the penalties for late submission	Hand-in: by 9am Monday 4th December
9. Arrangements for submission	Via Moodle

10. Assessment Regulations All assessments are subject to the University Regulations**.**	
11. The requirements for the assessment	*Please see attached document*
12. Special instructions	*See attached document*
13. Return of work	*within 3 weeks of submission.*
14. Assessment criteria	*See attached document* *Normal academic conventions for acknowledging sources should be followed.*

SET10117 Coursework: The Museum Protection Problem

A national museum needs to protect its exhibits from thieves. As they can't afford security guards, they want to install a set of cameras on each floor of the museum to monitor activity. Each camera has a fixed range of view which is a circle centered at the camera position. Within its range (i.e. within the circle), the probability of detection by the camera decreases as the distance from the centre increases. Outside of the range, it can't detect anything. Cameras can't see through walls or around them! If two or more cameras cover the same area, the probability of detection is increased in that area. Any area that has a probability of being detected that is greater than 0.5 is considered 'covered'.

Given a floor plan and a fixed number of cameras, the goal is to write an optimisation algorithm that positions each camera such that the amount of floor-space covered by the cameras is maximised. Ideally the entire floor space is covered. You are provided with a number of different floor layouts – you need to find a solution for each one given.

What to do

Your goal is to develop a *stochastic search algorithm* to find the best possible solutions to the set of instances provided. For each instance, the floor-space is divided into a grid of *n* equally size 'cells' and there may be walls at various positions within the grid. A camera can be placed in any cell. The optimisation problem is cast as a *minimisation* problem, i.e. the objective is to minimise the number of cells that are ***not covered*** by any camera, so a perfect solution has fitness 0.

Although you might want to experiment with multiple algorithms and discuss these in your report, your end goal is to produce a single algorithm that should work with any layout of walls and empty space provided in a CSV file – you should not develop a separate algorithm per instance

You can implement any meta-heuristic or local search algorithms (or a hybrid method) of your choice. You can use any of the methods covered in the module, but you are also free to do some research and use another method if it falls under the category of a stochastic search algorithm*. If you want to use a method not covered in the lecture material, please check with me before going ahead.*

You can implement your algorithm(s) using the DEAP libraries using the knowledge you have learned in class, but if you prefer to write in another language then that is fine, although this will likely involve much more work. Note there are no marks associated with the style of the code or for its efficiency or for choosing one language over another.

No marks will be awarded for taking a 'brute-force' approach, i.e. throwing a large amount of computational power at the problem to evaluate a very large number of possible solutions – please read the marking rubric carefully to see what marks are allocated for

Read the following carefully:

- You will be provided with a set of instances in CSV files showing different floor layouts. You can use these instances to experiment with to design the best possible algorithm. *Your final design of algorithm should work on all three instances without having to change its parameters.*

- You will have to design a representation to use with a stochastic search algorithm (or algorithms) of your choice.

- You need to undertake a methodological set of experiments to select good representations, algorithm(s), operator settings, parameter tuning etc.

- You need to write a report in the form of an academic paper that documents what you have done, covering what you have done in terms of designing an algorithm and experiments undertaken, and provide evidence in the form of statistics and/or diagrams.

- You are **required** to compare your results to those obtained by a random-search algorithm that evaluates exactly the same number of solutions as the algorithm you have designed.

- Following submission of the report, you will be provided with a test set of 3 more instances. You should run your algorithm once on each instance and will be asked to report the results on each instance via Moodle.

The report will be evaluated according to the following criteria (the marks associated with each are also shown):

Approach (15 marks): Explain the representation(s) used to represent an individual solution, including a description of why you chose these formats. Clearly set out what your investigation covers. You should describe which type of algorithm(s) you intend to develop and set out what your experiments will cover, for example comparing operators/comparing algorithms/parameter tuning

Algorithm & Operator Design (30 marks): For the algorithm(s) you chose, describe the operators used and any customisation you might have made to the algorithm itself. You should also explain any modifications you made to the evaluation function, for example to handle solutions with incorrect number of cameras. You should clearly describe any custom operators designed and the rationale behind their design. If you use the off-the-shelf operators provided in DEAP without modification (e.g. EASimple) you should state which ones used but you don't need to describe how they work. However, you should describe why you chose them. If your algorithm does not follow a standard approach then you should give pseudo-code.

Experimental Design & Analysis (30 marks) : describe experiments conducted to test version(s) of your algorithm. For each experiment, you should state what the experiment was designed to test. Give sufficient detail that experiments could be reproduced, including parameter settings. Present results of experiments in an appropriate form using graphs and/or tables. Use statistical significance testing where appropriate. Provide a discussion that describes results/highlights interesting findings.

Solution Quality (5 marks): you should provide a table that has one row for each of the instances provided with columns indicating the following information:

1. the best fitness you found for each instance provided for experimentation
2. the coordinates of the camera positions for this solution
3. the total number of evaluations used to obtain your solution (typically equal to the (population_size x num_generations) if you use a standard EA, or otherwise can be counted as calls of the evaluation function.
4. the result of the best solution found by a random search algorithm that evaluates the *same number of solutions* as indicated in column 3

Evaluation (10 marks): provide a reflective commentary on the quality of the results obtained and the approach taken, highlighting any strengths or weaknesses and making suggestions for future work.

Clarity/overall style (10 marks): the document should be written in a scientific style in the format of an academic paper. Be careful to label graphs/tables clearly (including axes on graphs) and use references where appropriate. The report should be a maximum of 6 pages, using Arial font, minimum size 11. **Do not** use any appendices other than the single one required with the screen shot of the checker function.

BONUS MARKS *An additional bonus of up to 5 marks will be awarded based on the quality of your solutions on the unseen test instances provided after the hand in date. The mark will be proportional to your ranking in the class in terms of the median quality of three runs of your algorithm on each of the unseen instances. If this takes your mark to >100, then the mark is capped at 100. The bonus is 0 if you do not participate*

A marking rubric is included at the end of this document

Technical Information

You are provided with a basic EA in Python in a Jupyter Notebook. You are free to convert to another language and can convert to a Python program if you prefer to work outside of a Notebook environment. The code provided has some basic functions:

- It reads in an instance file that defines the floor layout for an instance
- It provides an evaluation function that makes a call to an *external program* that returns a fitness value for a *valid solution*: **a valid solution is defined as one that defines unique locations for exactly the number of cameras specified the instance and in which there is no camera placed on a cell containing a wall**
- Three versions of the external program are provided:
    - If you run your code via Google colab, then you need to upload the file called *bit_cam_napier_colab* to Colab
    - If you run your code via Anaconda in Windows, then you need to have the file *bit_cam_napier_windows.exe* on your file system somewhere and must specify a path to this file
    - If you run your code via Anaconda on a Mac then you need to add the path the binary called bit_cam_napier_mac on your system
    - The external program must be sent a solution in the form a list of length *n* (where *n* is the number of cells in the grid for the instance being solved), where each value is 0 or 1. The number of 1s must be *exactly equal* to the number of cameras specified.
    - If you send a solution to be evaluated that does not have the correct number of cameras, the function will return a very large (poor) fitness value that is equal to the size of the grid squared by default.
    - You cannot modify any of the code in the external program
- You are also provided with additional binaries that allow you to visualise your solutions. These are named differently for each operating system as describe above. An example of their usage is given in the Jupyter notebook provided. This is just intended as an aid to help understand the problems and might inspire you to think of ways to design intelligent operators and to visualise your solutions.
- As noted in the introduction, the probability of detection of a camera decreases with range. If two cameras cover the same location then the detection probability is increased. The external program considers a location as ‘covered’ of the probability

of a cell being covered is greater than or equal to 0.5. This information is simply provided as background knowledge.

Hints and Tips

- The EA provided will return a very poor solution as many (it not all) solutions generated will have more than the specified number of cameras. You should think of ways to address this, e.g:
    - modify the *eval_solution()* function to assign a different fitness to solutions that do not have the specified number of cameras or have cameras placed on walls instead of calling the external objective function
    - consider methods of smart initialisation
    - consider defining operators that maintain 'valid' solution
    - consider repairing invalid solutions
- Although you must send a solution to the external program in the form of a binary list, your EA/search algorithm can use a different representation that might be more efficient. In this case, you just need to convert an individual from your representation to the bit string format required inside the function eval_solution() before calling the external program.

Marking Rubric

	<40	40-50	50-60	60-70	70+
Approach	Inappropriate choices; no explanation provided/explanation incorrect; no clear questions or plan defined	adequate choices of method and/or representation; some basic attempt at explanation but partially incorrect/choices that do not demonstrate understanding of ECO concepts	Good choice of algorithm/representation; good attempt at explanation but missing some depth or has some inaccuracies; choices demonstrate understanding of core material	Very good approach with explanations that demonstrate v. good understanding of material, but perhaps missing some detail or depth	Excellent approach, very clearly justified, demonstrating excellent understanding of core concepts of course, perhaps going beyond taught material
Algorithm and Operator design and customisation	Algorithm does not produce a valid solution; design shows lack of understanding of basic concepts	Uses off-the-shelf operators/algorithms with no attempt at developing any customisation to make methods suitable to solving the problem	Some attempt to provide customisation of at least one operator to enforce constraints	Very good attempt to adapt algorithm to problem taking account of constrained nature; may have customised multiple operators or shown very good insight into particular operator design	Excellent; bespoke customisations that demonstrate excellent understanding of the nature of the problem and deep insights into the algorithm; may draw on literature not covered in the course
Experimental design and analysis	Insufficient experiment; poor presentation of results, no analysis	Experimental design meets minimum requirements, but very limited in	Good design, that includes some experimentation of parameters/methods.	Very good design that covers a range of factors;	Excellent design, wide ranging or very thorough investigation;

		scope or has some flaws; presentation adequate but lacks statistical analysis	Presentation could perhaps be improved; at least some attempt at basic statistical analysis provided	results well presented, appropriate use of statistics	thoughtful presentation or results, thorough use of statistics, excellent analysis
Solution quality	No valid solution given	Solutions found but worse than random search	Solution better than random search but not optimal	Very good quality solutions, possibly close to optimal for all instances	Excellent quality of solution on all instances
Evaluation and future work	None provided or demonstrates significant misunderstandings	minimal reflection; suggestions for future work either minimal or flawed	Provides some reflection with some attempt to highlight strengths/weaknesses with at least one sensible suggestion for future work but perhaps not well linked to weaknesses	Very good and critical reflection that shows insight into domain and methods. Focused suggestions for future work linked to weaknesses	In-depth reflection, shows deep insights in both methods and the domain, with excellent suggestions for future work that address highlighted issues
Clarity/overall style	Below standard expected at this level	adequate	good	Very good	excellent