Evolutionary robotics
Prof. Dario Floreano

Grand challenge project tips

Tips:

- Try to not run too many evolutions/visualizations of the robots (better if only one). This can lead to a crash of your web browser and therefore the end of your evolution.
- Sometime the RoboGen app seem to have a problem saving the files of the evolution. You can always try to refresh the page before you start an evolution.

Submission Deadline Reminder:

- The submission platform for the Grand Challenge will be opened on 24th May 2022 and close on 31st May 2022.
- The submission of your best robot will be analysed by the platform and double-checked by the assistants.
- Student can create and simulate the robot in different arenas using the arena generation tool available at https://github.com/Arth-ur/robogen-arena-generator
- Remember, the score of the submission platform and the performance of the robot are **not** graded.
- The grade from the laboratory component of the course will only depend on the student presentation. A template of the presentation will be provided on moodle.
- The grade will be based on the scientific approach, clarity and completeness of the presentation and work.

Q: Are there optimal parameters for an EA?

A: There is no set of parameters that guarantees robust performances on all possible problems. The performance of an EA heavily depends on the structure of the fitness function and its properties (uni/multi-modality, separability, number of dimensions). Therefore, to get the best possible performance, the EA's parameters (crossover rate, mutation rate, tournament size, pop size) must be tuned on each specific problem at hand.

Q: What are the advantages and disadvantages of

- Low/high mutation rates?
- Low/high selection pressure?

A: Again, the effects of mutation rate and selection pressure depend on the fitness landscape at hand. However, some general intuitive rules can be drawn:

- A low mutation rate and low mutation magnitude (i.e., std. dev. of the Gaussian mutations) have the advantage of not being too "disruptive", i.e. most information from the previous generation is preserved which allows for a relatively slow gradient descent towards the optimum. This might be useful when existing solutions are already close to

the optimum. On the other hand, high mutation rates and magnitudes makes the evolutionary process more exploratory as the new individuals are relatively far from their parents in the search space. This might be useful when the existing solutions are far from the optimum and/or in highly multimodal fitness landscape where you want to explore the search space as much as possible in the attempt to find the global optimum. However, excessively high mutation rates and magnitudes might compromise the inheritance of genes from the parents to the offspring, simply because they introduce too much noise in the search, almost turning the EA into random search.

- Low selection pressure (e.g. a small tournament size) is especially useful when the problem is multimodal and you want to maintain a certain diversity of individuals in the population. In fact, low selection pressure means that even low-fitness individuals still have some chance of reproducing. On the other hand, high selection pressures (e.g. a large tournament size) give "weak" individuals a much smaller chance of reproducing compared to a few high-fitness individuals, resulting in less diversity. This can be advantageous when the landscape is unimodal and it is possible to quickly converge to the global optimum by favouring exploitation over exploration, with no need to maintain diversity.

Q: Why are hidden units sometimes needed for a neural network to solve a given task? What is the defining characteristic of problems that networks without hidden units are unable to solve?

A: Hidden units are needed when the function to learn is not linearly separable. A typical case is the XOR where it is not possible to draw a straight line that separates ones from zeros. Networks without hidden layers are not able to solve this kind of problems because they can only learn linearly separable patterns. Introducing a hidden layer allows a transformation of the input space so that problems that are not linearly separable become separable.

Q: Why are recurrent connections needed to solve certain problems? What is the defining characteristic of problems that networks without recurrent connections are unable to solve? Are there problems that require recurrent connections and multiple hidden units?

A: Recurrent connections are needed, in general, whenever there is a need for one or more bits of memory. This is the case of temporal patterns where the function at any given timestep depends on the current input, as well as one or more previous inputs. Networks without recurrence are not able to solve this kind of problem because they are not able to store their previous (internal) state, i.e. they don't have "memory" of the past. In addition to recurrent connections, hidden units may be needed in cases where the temporal pattern is not linearly separable.