Computational Intelligence (CI-MAI) EAs exercise 3

Lluís Belanche

Dec 4, 2023

Optimizing a difficult function with Genetic Algorithms

The application of formal tools to the minimization/maximization of multivariate binary functions is a long-standing problem in mathematics. Combinatorial optimization problems can be viewed as searching for the best element of some set of discrete items. Perhaps the most universally applicable approaches are branch-and-bound (an exact algorithm which can be stopped at any point in time to serve as heuristic), branch-and-cut (uses linear optimisation to generate bounds), dynamic programming (a recursive solution construction with limited search window) and tabu search (a greedy-type swapping algorithm)[1].

As in the continuous case, it is not possible to compile the numerous approaches existing in the literature. Again, arguably the best approach is to consult good specialized books on the matter, like those written by Schrijver[2] or the now classic by Papadimitriou and Steiglitz *Combinatorial Optimization: Algorithms and Complexity*.

Some examples of combinatorial optimization problems that are covered by this framework are shortest paths and shortest-path trees, flows and circulations, spanning trees, matching, and matroid problems. Famous examples include the travelling salesman problem (TSP), the minimum spanning tree problem (MST), and the knapsack problem. Specifically for the TSP, there are good web resources like this[3].

These are not the only problems that may have a natural binary representation. Many other problems in search, scheduling, planning, etc, have been tackled by Genetic Algorithms, to an astonishing number of applications (in the thousands!). Even discrete or integer optimization is possible (by using Gray codes or by using alphabets of cardinality larger than two).

You are required to select a problem. Ideally, you may bring your own problem, but there are many (mostly artificial) problems designed as a benchmark suite; a couple of starting papers are:

- https://www.sciencedirect.com/science/article/abs/pii/0303264796016218
- https://dl.acm.org/doi/abs/10.1145/3205651.3208251

The exercise consists in creating or choosing a problem like the ones described above and attack it using GAs. The requirement for this practical work is that the search space must be a hypercube of dimension n (the number of bits and also the chromosome length, of your choice).

You are then required to choose appropriate:

- representation for the candidate solutions
- selection mechanism
- crossover operator(s)
- mutation operator(s)
- stopping criteria

The use of synthetic data allows to be in control of:

[1] This last method could be considered an EA, under a wide sense of the definition.
[2] Freely available at http://homepages.cwi.nl/~lex/files/dict.pdf
[3] https://www.math.uwaterloo.ca/tsp/optimal/index.html

- the dimension of the problem n
- the way you represent the individuals
- the amount of available data for the fitness function
- the problem hardness (controlled by a single hyperparameter) [4]

All other conditions, settings, hyperparameters, etc, are left to your decision. You can also choose any other (evolutionary or non evolutionary) method that you want to test in comparison to the GAs. The problem hardness should not be extremely difficult or trivial, otherwise the study is pointless.

What to report in the comparison? An obvious choice is to set a predefined performance and then report the (average):

- execution time needed to achieve a given performance
- number of generations needed to achieve a given performance
- number of fitness function calls needed to achieve a given performance
- Fraction of times the algorithm reaches a given performance

  ... as a function of population size, n or problem hardness. Alternatively, one can let the algorithm run for a predefined number of generations and study all the other quantities. The study is quite flexible and left to your criterion.

If you use R, a good tool is to use *Rmarkdown* to produce the document, integrating LaTeX code and R code. If you do, have a look at

https://bookdown.org/yihui/rmarkdown/

There are other similar tools for other languages.

Important information:

- **Write a brief pdf document (7 sheets maximum, including everything) that describes only the relevant information (problem setup, previous work, work done by you, discussion and conclusions)**
- **Delivery date: no later than December 28, 2023, via the Racó**
- **To be done in groups of 2 students**
- **Please include all names in the final document and upload only one document per group**
- **Add a plain text file named "README.txt" with complete instructions about how to obtain your results**
- **If you use ChatGPT (or another similar tool) in the document, indicate it every time you do. We want to evaluate your work, not someone else's!**
- **There is no obligation to use any particular programming language; but I ask you to limit the choice to R, Julia, MATLAB and python. Choose any library that you find adequate as a complement, provided you cite them**

[4] It is desirable that you know the theoretically optimal solution.