Homework 3 - Solutions

Cellular Automata, Particle Swarm Optimization

(Max useful score: 100 - Available points: 130 + 65)

15-382: Collective Intelligence (Spring 2018)

OUT: February 25, 2018, at 13:30pm

DUE: March 11, 2018 at 9:30am - Available late days: 1

Instructions

Homework Policy

Homework is due on Autolab by the posted deadline. As a general rule, you have a total of 6 late days. For this homework you cannot use more than 1 late day. No credit will be given for homework submitted after the late day. After your 6 late days have been used you will receive 20% off for each additional day late.

☛ This homework includes two set of questions on Cellular Automata, in Section 1 and Section 2. *You only need to answer to one set of questions*, at your choice. Of course, if you also address the other set of questions you'll (possibly) earn additional points. However, in order to do this, you must first address some of the questions on PSO in Section 3. In practice, the aim is that you deal with both topics.

If you find a solution in any source other than the material provided, you must mention the source.

Submission

Create a zipped archive including: a PDF file with the answers to the provided questions (they can be handwritten, but in this case you must have / use a "readable" handwriting), files that have been used for dealing with the questions that require programming, a README file that specifies how to use / run the programming files. The zipped archive should be submitted to Homework 3 on Autolab.

Contents

1 Traffic studies using Cellular Automata (65 points)

You work for the traffic department and have to make an analysis of the traffic flows along one major busy road that goes through the capital city. The purpose of the analysis is to understand how fluid or jammed / irregular traffic flows are, and then, possibly, devise some corrective measures. For the analysis you decide to use a microscopic traffic model based on a cellular automata according to the Nagel-Schreckenberg + DVR model and Rickert-Nagel-Schreckenberg model (as described in the lecture slides and in the original research papers provided on the course website).

The road is two-lanes one-way, has in and out ramps, and traffic lights. here are the parameters describing the scenario:

- Length $L = 2.7$ Km
- A car is 7.5m long
- $v_{max} = 5$/cells per time-step
- Periodic boundary conditions
- Inflow ramp at 0.75 Km, probability of a car entering is p_i, subject to the space in the road
- Outflow ramp at 2.4 Km, probability of a car exiting is p_o subject to the car having $v > 0$
- Three traffic lights, s_1, s_2, s_3, at the following positions (in Km): 0.75, 1.5, 2.1. Each traffic signal has a periodic timing for switching between green and red light status, that we indicate, respectively, with τ_1, τ_2, τ_3, (e.g., s_1 changes status from green/red to red/green every τ_1 time steps).

Let's set $\tau_1 = \tau_2 = \tau_3 = 5$. The average density ρ (number of cars per cell) of vehicles traveling in the road is constant, with $p_i = 0.3$. The randomization parameter p that models human (mis)behavior in the Nagel-Schreckenberg model is set to 0.5. The slow-to-start randomization parameter is set to $p_0 = 0.75$.

1.1 Design and implement the agent-based CA model (30 points)

Implement a CA that models the scenario as described. Set all missing parameters according to some "reasonable" guess. Run simulations that correspond to 3h of real time.

Implement a simple visualization of the time evolution of the CA cells in the road lanes, as it is normal practice to visualize a CA (i.e., a new CA cell line per time-step)

1.2 Graphical visualization (10 points)

In order to appreciate (and better understand, for you and for me) what's going on, implement a graphical visualization of the traffic along the road, that should be kept fixed (i.e., show how the flows proceed over time along the road).

1.3 Analysis of the observed behaviors (25 points)

Make an analysis of what happens, reporting the plots of:

- Flux vs. the density $\rho \in [0.05, 1]$;
- Average speed vs. density;
- Variance of the speed vs. density;
- Average length of traffic jams vs. density;
- Spatial distribution of traffic jams for the critical density at which the flux starts decreasing.
- Start with an empty road $\rho = 0$, set all signals to constant green, close the outflow ramp. Set the $p = 0.015$, $p_0 = 0.75$, and $p_i = 0.4$. As time passes, the density will increases steadily because of the inflow. Run the simulation until $\rho = 1$. Show flux vs. density.

Thoroughly discuss the results that you observe for each one of the above plots. Point out any interesting behaviors you might observe, such as phase transitions, metastability, ...

Make possible "suggestions" that the traffic department could use for improving the traffic situation.

2 Cellular Automata as an agent model for studying interacting populations (65 points)

Using the formalism of dynamical systems, we have studied a few models of competing populations / species. Among these models, the Lotka-Volterra model that considers a basic prey-predator scenario is a remarkable one.[1] The Lotka-Volterra model is a form of a population model based on the *well-mixing* (or homogeneous mixing) hypothesis: the rate of encounter between members of two different populations is proportional to the product of the population sizes. The well-mixing hypothesis implies that doubling the size of either population results in twice as many encounters, and this implies that members of both populations are *homogeneously* distributed in space and do not mix in any smaller subgroups. In practice, the well-mixing assumption it allows the use of ODEs instead of PDEs (or of agent-based models), that would otherwise require the introduction (and modeling) of the spatial structure (including, for instance, mobility).

The well-mixing hypothesis is a form of *mean field* assumption, since an *average* effect is assumed when considering multiple individuals interacting to each other. In other words, in the case of two populations N_1, and N_2, instead of considering each possible location for an individual and one-to-one interactions, a mean (force / interaction) field is considered, reducing in this way the $N_1 \times N_2$ multi-agent problem to a two-agent one. Each population is reduced to one single entity that has everywhere the same mean value (in relation to the properties of interaction). This approximation works well for large populations, since they tend to be uniformly distributed, and local fluctuations become less important. However, it is clear that when one of the two populations decreases because of the competition, the mean field approximation can become significantly wrong because the assumption of uniformity (in space) is hardly met. Also when space has a well defined structure (that in turn affects interaction), the mean field assumption might not work well.

In order to possibly overcome the limits of the mean field approximation, let's define a Cellular Automata model, that can bring a few advantages: a spatial structure (defined in terms of a lattice model), (more complex) agent-based interactions, probabilistic components.

2.1 CA model for prey-predator systems (25 points)

Define a Cellular Automata model for a prey-predator system. The CA must model a relatively large rectangular territory (with periodic boundaries) where an initial population of P preys and K predators are present (scattered around). The following aspects of the evolution of each population and their mutual interaction must be accounted for in the state transition function for CA's cells.

- The same mechanisms that are modeled by the Lotka-Volterra equations should found a counterpart in the CA model: exponential growth rate (birth) rate of the prey population in the absence of predators; exponential decrease (mortality by starving) in the predator population in the absence of prey; mortality of preys as the result of encounters with predators; increase of the predator population as a result of encounters with preys.
- *Mobility* of preys and predators. Mobility must happen respecting adjacency relationships between cells (i.e., no "jumps"). Mobility must be driven by a purpose (e.g., escaping from preys, escaping from crowded areas, going to where grass is present).

It can be observed that the elements above potentially provide a more detailed (realistic) model of a prey-predator scenario compared to the Lotka-Volterra equations.

You must precisely specify ALL the elements that define a Cellular Automata model: states, neighborhoods, transition functions, lattice, updating mechanisms.

You can (should) use probabilistic rules. You can also consider to subdivide each updating step in sub-steps (e.g., first interaction-reproduction rules, then mobility rules).

[1] The model was developed independently by Alfred Lotka and by Vito Volterra around 1925. It has been the basis of a large number of studies and extensions of the basic model.

Once you have built the model and implemented the CA, you are asked to perform a simulation study of the behaviors resulting from the model, as it is specified below. The simulation study must be supported by some 2D *visualization* showing the status of a cell (e.g., check for instance the visualization in the wolf-sheep model in NetLogo https://ccl.northwestern.edu/netlogo/).

2.2 Simulation study of the CA model (20 points)

Consider a 100×100 grid lattice. If your rules are "well" designed, by starting with different initial populations (of prey and predators), you should be able to observe an exponential growth of preys in absence of predators, and an exponential decay of predators in absence of preys. Moreover, you should observe (more or less) the cycling behaviors that arise in the phase space of the Lotka-Volterra equations, with the cyclic evolution of the preys always lagging behind that of the predators.

Make plots with the above results and discuss the impact of the different parameters that you have used. That is, in order to achieve the goal of producing results comparable to Lotka-Volterra equations, you might need to adjust parameters, neighborhoods, transitions, etc. Therefore, you are asked to comment on the different choices that you have made and why do they work or didn't work as expected.

2.3 Extended prey-predator model including preys' food (10 points)

In order to properly complete the food cycle in the ecosystem, a third of type of agent must be included in the system: the *food* F for preys (e.g., grass). F is expected to have an exponential growth rate in the absence of preys (consumers). In the absence of food, the birth rate of preys gets dramatically decreased (this models "crowded" scenarios: if a number of preys are in an area that do not provide enough food they will starve and die).

Consider the full model, including F. How does this change the observed behaviors? Are the observations reasonable from a real-world point of view?

2.4 Adding a shelter for the preys (10 points)

Introduce an additional spatial component to the model: a *shelter* S, of 5×5 cells. This is an area to where predators cannot enter. However, the shelter doesn't have food, such that being too long in the shelter makes the preys starving and then eventually die.

Again, perform a simulation study and discuss the observed behaviors.

3 Continuous optimization with PSO (65 points)

The optimization of complex multi-modal functions is a difficult task. When the number of local optima is very large, the use of gradient descent procedures cannot really help to find the global optimum. PSO performs multi-agent black-box optimization, and might be a suitable choice in these cases.

Given the intrinsic difficulty of the task, at a number of major international conferences it is common practice to set up black-box and/or white-box optimization competitions. A benchmark set of optimization problems is defined and the goal is to find the best (optimal) solutions given a specified maximum number of function evaluations (or of cpu time). For instance, you can peek at the black-box competition currently open at the GECCO conference, a major venue for nature-inspired algorithms, https://bbcomp.ini.rub.de/.

Below, some of the functions that are regularly used in these competitions are considered for you to optimize. For instance, you can check how these functions (and many other benchmark functions) look like at: http://al-roomi.org/benchmarks/unconstrained/n-dimensions.

☛ For all the tasks discussed below, you'll be scored on the soundness of your design choices, on the quality of your data reporting, and on the quality of the minima that you'll be able to find (poor minima will imply a low score, this is a *competition!*).

3.1 Basic PSO (20 points)

You are asked to implement an instance of the basic PSO (as shown in the lectures, with the constriction factor) for finding the minimum of the following set of d-dimensional scalar functions:

- Grienwank:
$$f_1(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,$$
where $d = 30$, $-600 \leq x_i \leq 600, \forall i = 1, \ldots, d$. The known optimum is found at $\boldsymbol{x}^* = \mathbf{0}$, with $f(\boldsymbol{x}^*) = 0$.

- Schwefel:
$$f_2(\boldsymbol{x}) = -\sum_{i=1}^{d}\left(x_i \sin\left(\sqrt{|x_i|}\right)\right),$$
where $d = 30$, $-500 \leq x_i \leq 500 \forall i = 1, \ldots, d$. The known optimum is found at $x_i^* = 420.968746$, $i = 1, \ldots, d$, with $f(\boldsymbol{x}^*) = -418.9828 \cdot d$

- Rastrigin:
$$f_3(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2 - 10\cos(2\pi x_i) + 10,$$
with $d = 10$, $-5.12 \leq x_i \leq 5.12 \forall i = 1, \ldots, d$. The known optimum is found at $\boldsymbol{x}^* = \mathbf{0}$, with $f(\boldsymbol{x}^*) = 0$.

- Rosenbrock:
$$f_4(\boldsymbol{x}) = \sum_{i=1}^{d-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right],$$
with $d = 30$, $-30 \leq x_i \leq 30 \forall i = 1, \ldots, d$. The known optimum is found at $x_i^* = 1\ \forall i = 1, \ldots, d$, with $f(\boldsymbol{x}^*) = 0$.

You will have to make a number of design choices for your PSO algorithm (number or particles, neighborhoods, local vs. global best, etc.). The educational task is precisely to explore the possible alternatives by reasoning about their impact and mutual interaction (i.e, you can't really use a brute force approach trying out all possible alternatives, you must make reasoned choices and assumptions). You MUST describe and discuss your design choices in the report.

For each run of your PSO algorithm, you have to compute the *relative error*, defined as:

$$\varepsilon = \frac{|f^{pso} - f^{opt}|}{|f^{opt}|}.$$

The numerator is the absolute error, the magnitude of the difference between the optimal value (known in these cases) and the approximation provided by your PSO algorithm. The relative error is the absolute error divided by the magnitude of the optimal value. The *percent error* is the relative error expressed in terms of percentage.

Since PSO depends on stochastic aspects, including the initial deployment and velocities of the particles, for each function you must run your PSO algorithm (at least) $n = 5$ times starting with different initial deployment and velocities for the particles. The performance of your algorithm on each function f_i, $i = 1, \ldots, 4$ is computed as the average performance over the n runs.

For the experimental evaluation, you have a maximum of *10 minutes of cpu time per function* (imagine you are using function optimization for a real life problem that requires to take optimized decisions every 10-15 minutes based on a changing environment). You are free to use this time as you want, but of course you must assume no knowledge about where the optimum of the function is!

For each function, you must report the following data in a table:

- Average relative error;
- Variance of the relative error;
- Average time when the best solution was found.

You are warmly encouraged to write a *script* program (python was originally design precisely for this!) that makes you performing all the experiments, collect all data, and prepare the result table in automatic (you have 4 functions, each run is 10 minutes, and you have 5 runs, therefore, each "experimental campaign" will take 200 minutes, about 3h. You *must* keep your computer doing the job for you while you're doing other things!

3.2 PSO variants (25 points)

Once you have gone through the previous question, you should consider possible improvements of your PSO algorithm. A number of variants have been proposed through the years. Some of these are discussed in the book extract included with the homework. Your task is to go through the pages of the extract, read about the different variants (described in pages 317- onward), and pick-up one of them as guideline to implement your PSO variant.

1. Discuss the reasons of your choice vs. other three different possible alternatives described in the book extract, at your choice (it's not enough to say: "Because I like it", you must bring substantial arguments!)
2. Describe your implementation and strategic choices.
3. Perform an experimental campaign analogous to the one you have performed with the basic PSO and report the data in a table as before.
4. Discuss the observed differences (if any) in behavior. At this aim you will likely have to collect some extra data apart from when the minimum is achieved (e.g., best solution so far improvement trend, spatial distribution of the particles over time, diversity in the particles, ...)

3.3 *Niching* optimization (20 points)

PSO has the tendency to "converge" to a single global optimum. However, dealing with multi-modal functions, it might be interesting to discover and maintain/converge on more than one optimum. This can be thought as the case when the agent population corresponds to multiple *species*, with each species being interested in finding an ecological *niche*, which is optimized and at the same time is far enough from the ecological niche of other sub-populations / species. The optimization function precisely defines where the different niches are in the "environment".

You can check the niching competition that is currently open at GECCO'18 http://www.epitropakis.co.uk/gecco2018/

Your task consists in modifying one of the two PSO algorithms that you have developed for the previous questions in order to address the issue of niching, aiming to two niches. The goal is to have, at the end of the 10 minutes run, the particle population having discovered and more or less clustered around two good minima. The performance J_{pso} of you algorithm is measured as follows:

$$J_{pso} = \frac{\varepsilon_1 + \varepsilon_2}{2d_{12}},$$

where ε_1 and ε_2 are, respectively the relative errors for niche 1 and niche 2 minima, d_{12} is the Euclidean distance between the locations where the minima of the two niches are, normalized to the maximal distance D between two points in the function domain (for all functions f_i, $i = 1, \ldots, 4$, the domain is squared, centered in $\mathbf{0}$, such that the maximal distance D between two points is easily computed). More specifically, d_{12} is defined as:

$$d_{12} = \frac{||\boldsymbol{x}_1 - \boldsymbol{x}_2||}{D}.$$

In practice, the goal is to find two good minima that are sufficiently far apart in the function domain.

As in previous questions, you have the same time limits, must perform multiple runs, and report you result in a table. Moreover, you have to describe and discuss your design choices.