Università
della
Svizzera
italiana

Particle Methods **2022**

Student: Filippo Casari

Report for Assignment 1

Introduction

My first approach for this assignment was using python. Although the logic behing the code seemed to work, by using $H > 4$ as parameter, I got some troubles about the performances of the program. Consequently, I switched to Matlab to exploit multithreading and GPU power.
The program asks to users to decide whether to save the result image, to apply random search, and to choose the number H as well.

Implementation

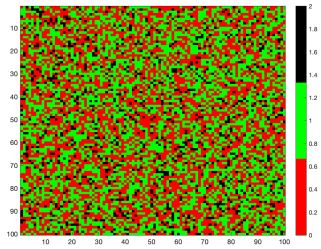
I used 2 strategy for implementing the algorithm for moving the agents:

- **Random search**: The unhappy agents move to a random position in the grid. The direction is chosen randomly.
- **NN**: Nearest Neiighbour. The unhappy agents move to the nearest empty position in the grid. The direction goes from the left to the right, and from the top to the bottom.

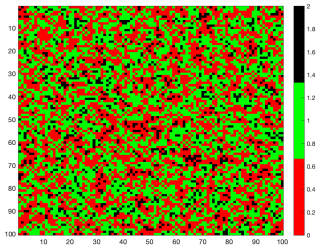
Since it could happen for large number of H that the algorithm is going to converge very slowly or in the case of NN never converges, I implemented a so-called **early stopping** condition. It consists in checking if the number of unhappy agents is the same for 1000 consecutive iterations. If this happens, the algorithm stops.
Speaking about the best strategy, it turns out that the Random one is the best one. Indeed, sometimes the NN does not let the algorithm converge to 100 percent of happiness.
In contrast, the Random movement agent strategy for large H takes very long time to converge, and this leads to an early stopping criterion.

Results

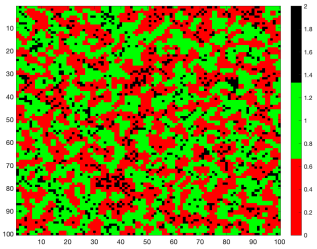
When the algorithm reaches 100 percent of happiness, the result image shows a really nice patterns which look like a sort of segregation between green and red points.



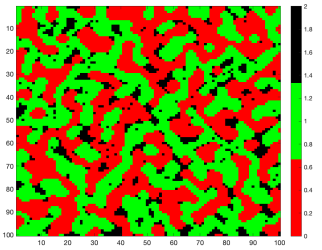
(a) Image 1



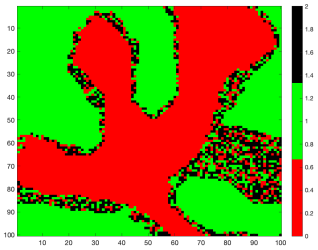
(b) Image 2



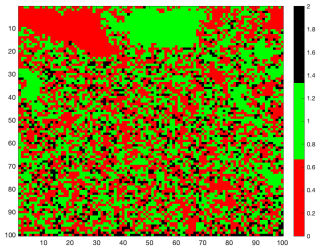
(c) Image 3



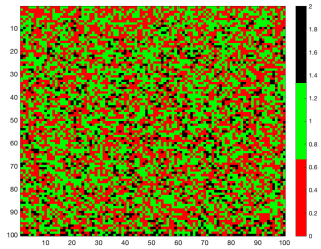
(d) Image 4



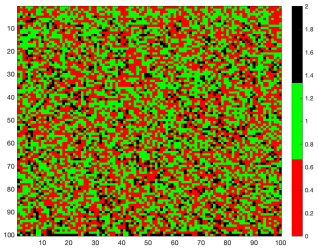
(e) Image 5



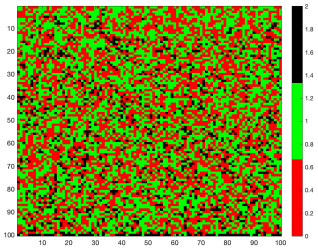
(f) Image 6



(g) Image 7

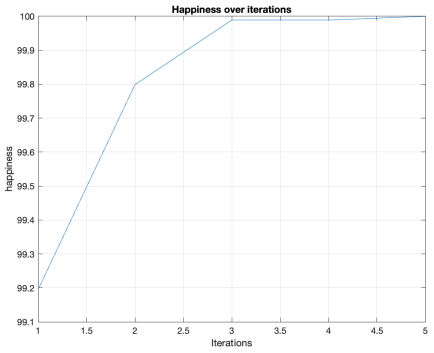


(h) Image 8

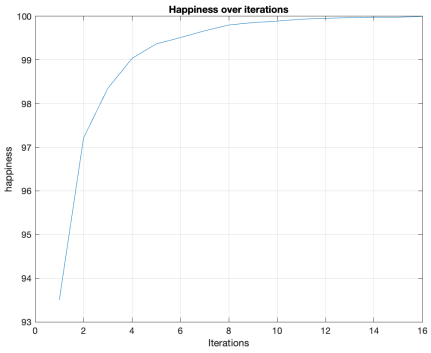


(i) Image 9

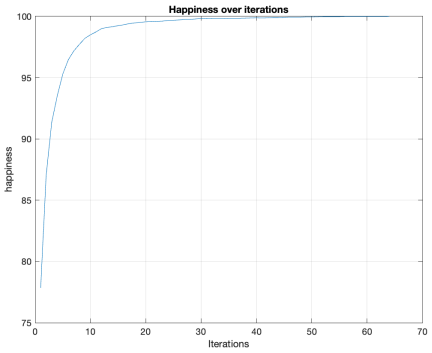
I reported above the grids at the end of the algorithm for different values of H and Random=True. I also saved the grids for Random=False (NN) in "results" directory.
It turns out that with $H > 5$ (from image 6) the segregation is not so clear or there isn't at all. Below I reported also the level (percentage) of happiness for each case. If the level si not 100 percent is because maybe the algorithm converges to a local "minimum" or because of the early stopping criterion was met. This could happen if the happiness did not improve within a time window of 1000 iterations.



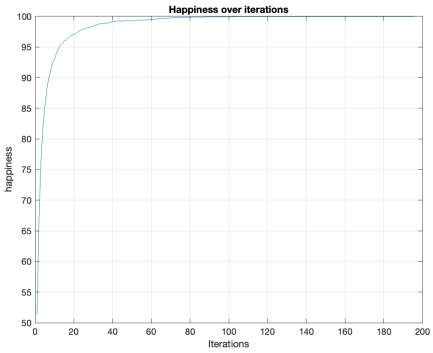
(a) Image 1



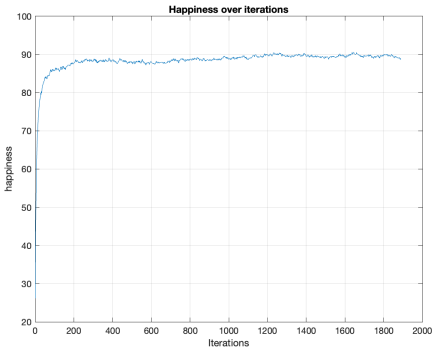
(b) Image 2



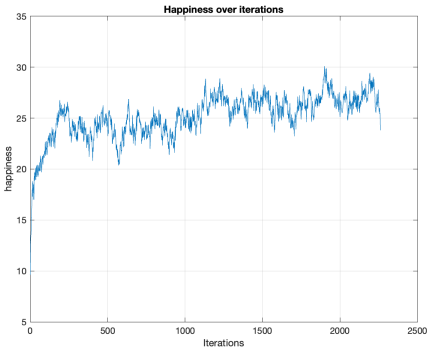
(c) Image 3



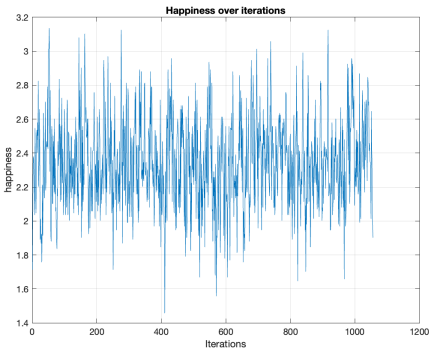
(d) Image 4



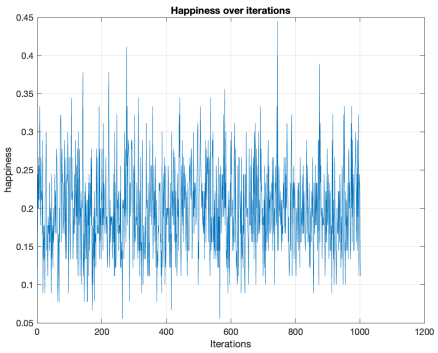
(e) Image 5



(f) Image 6



(g) Image 7



(h) Image 8

As one can see the heppiness with $H < 6$ it converges whereas with H higher the level is going up and down without reaching a good level of happiness.