# Agents that eat, cooperate and live a long life

Inteligência Artificial / Introdução à Inteligência Artificial

Group work - 2017/2018

15 October 2017

## I Important information

1. This work should be done in groups of 2 to 3 students. In each submitted Python module, include a comment with the name and number of the authors.
2. A first version of the program should be submitted until ~~November 20, 2017~~ November 28, 2017. The final version should be submitted until ~~November 30, 2017~~ December 15, 2017.
3. In both submissions, the work may be submitted beyond the deadline, but will be penalized in 5% for each additional day.
4. Each group must submit a zip file `tpg.zip` containing all relevant Python code. Programs should be prepared to run in Python 3. If this is not possible, inform the teacher. The agent should be a class `StudentAgent` in a module `studentagent.py`.
5. In the final submission, include also a Powerpoint presentation (pdf format), with a maximum of four pages, summarizing the underlying design of the developed agent. If you discuss this assignment with colleagues from other groups, include a comment with the name and number of those colleagues. If you use other sources, cite them as well.
6. All submitted code must be original. While trusting that most groups will do this, plagiarism detection tools will be used. Students involved in plagiarism cases will have their work canceled.
7. The works will be evaluated taking into account: performance; quality of design and implementation; and originality.

## II Overview

This work involves concepts and techniques of three main chapters of the course, namely: programming in Python; agent types and architectures; and search techniques for automated problem solving.

Students are asked to develop a cooperative agent that lives in a simulated world. The agent needs to eat food that appears regularly in the environment and wants to live for as long as possible.

The agent should be able to play alone, as well as together with another instance of the same agent. In the second case, the two instances of the agent will cooperate in order to increase the chances of living longer.

## III World and agents

The game takes place in a world consisting of a rectangular grid of $M$x$N$ cells. Some cells are occupied by walls.

The game starts with two agents and ends when both are dead. Each agent occupies two grid cells. In each perception-action cycle, the agent can move to a free adjacent cell or stay in the current cell. Adjacent cells are those reachable in vertical (north/south) or horizontal (east/west) movement. The agent does not move in diagonal. The agent dies if it hits a wall.

The world is toroidal. This means that, when an agent crosses one edge of the grid, it will appear immediately near the opposite edge.

The world also contains food, which the agent needs to eat in order to continue living. Each piece of food is located in a particular cell. There are two types of food:

- *moving* food - It's red and moves randomly from cell to cell. Moving food is required for an agent to have physical energy.
- *static* food - It's blue and does not move. Static food is required for an agent to have mental energy.

When an agent enters a cell with food, the agent automatically eats it, i.e., the food dissappears and the agent absorbs its nutrients. New food will emerge in another cell randomly.

The agent gets a map of the world in the beginning of the game. This map identifies the size of the world and the cells with walls. Additionally, in each cycle, the agent gets information on its own location as well as the location of food pieces in its neighborhood. Therefore, the agent has local vision and cannot see all pieces of food currently existing in the world.

The body of each agent occupies two cells, with the same colors as the nutrients. The intensity of these colors is proportional to the amount of nutrients. For instance, a light blue color means that the agent is low in type $S$ nutrients.

Each agent starts with a stock of 1000 units of type $M$ nutrient and 1000 units of type $S$ nutrient. Each piece of moving food provides 100 additional units of nutrient $M$. Each piece of static food provides 100 additional units of nutrient $S$. However, the agent cannot acumulate more than 2000 units of each nutrient. (These parameters may be adjusted later.) Nutrients eaten beyond this limit are lost. If the agent runs out of either of the nutrients, the agent dies.

The goal is to maximize the sum of agent lifetimes. The lifetime is the number of cycles during which the agent is alive.

In order to live longer, the agents can cooperate. They can exchange information regarding their locations and/or the locations of pieces of food. In addition, when the agents are in adjacent cells, the nutrients are redistributed between them in order to reduce nutritional imbalances in each.

Agents actions consume nutrients of type $M$. Not moving (staying in the current cell) also consumes type $M$ nutrients. The time spent in thinking/deciding will consume type $S$ nutrients. Sending messages also consumes type $S$ nutrients.

Finally, agents have a biological clock. As they get older, they will tend to consume more and more nutrients to do the same actions. Eventually, they will die, but they want to live for as long as possible.

## IV Base code

All base code for this project is in `https://code.ua.pt/projects/tpg-ia-iia-longlife`. This includes code that manages the environment where the agents live as well as an example of a very simple agent. The main entities in the game are represented by classes.

Each group develops an agent in the form of a class derived of `Agent` (as is the case of `Agent1`) and cannot introduce changes to other files, such as `game.py`. You can, however, create new files, folders, etc.

The developed agent must be delivered in a module `studentagent.py` and the class should have the name `StudentAgent`.

By inheritance you can implement all methods of class `Agent`. It is essential to implement the `chooseAction()` method.

A support channel exists in `https://detiuaveiro.slack.com/messages/ai/`, where doubts can be posed and changes to this document may be announced.

Given the novelty of this work/code, some bugs will eventually be found and some adjustments will be introduced. Pay attention to changes in the server (`git pull`) and notifications in *Slack* and *e-learning.*

## V Clarification of doubts

This work will be followed through `http://detiuaveiro.slack.com`. The clarification of the main doubts will be placed here.

1. Fazer `git log` para se manter informado de pequenas alterações que foram ou venham a ser feitas.

2. (2017/11/13) A new version (v1.2) of LongLife is now available in the `tpg-ia-iia-longlife git` repository. The most significant changes are:

   - Execution of actions, message passing, food movements and nutrient redistribution now happen in each player turn.
   - The timeslot, used for computing the "cost of thinking", is now optionally normalised to compensate for differences in processor speed.

3. **Questão**: Como serão avaliados os agentes?

   **Resposta**: Os scores serão obtidos em $N$ execuções do jogo num conjunto de mapas com diferentes complexidades. Todos os agentes serão testados em condições idênticas para minimizar o fator sorte. Na primeira entrega, apenas é avaliado o desempenho. Na entrega final, é avaliado novamente o desempenho, mas também o PowerPoint. Em ambas as entregas, as pontuações do desempenho emergem da distribuição dos desempenhos dos vários agentes. Os desempenhos melhores serão candidatos a notas entre 18 e 20. Os agentes piores terão notas próximas do 10. Notas negativas ficam reservadas para agentes com desempenhos fracos e em que a concepção/implementação deixou de fora um número significativo de aspectos que normalmente terão que ser cobertos num agente destes.

4. (2017/11/17) LongLife v1.3 was released on tpg-ia-iia-longlife. Features changed since 1.2: Redistribution of nutrients only happens between *live* players.

5. **Questão**: Sendo $p$ um ponto de coordenadas $(0, 0)$, se fizer $p - (0, 1)$ fico com $(0, -1)$ ou $(0, 39)$?

   **Resposta**: Os pontos (Points) não conhecem o mundo em que estão, por isso, `Point(0,0) - Point(0,1) == Point(0,-1)`. Isto não é errado: `(0,-1)` significa um ponto que está 1 unidade acima da origem. Dá jeito porque assim podemos usar Points para representar deslocamentos relativos, mesmo que em direções negativas. No entanto, num mundo toroidal de 60x40, não convém usar um ponto como `(0,-1)`. O mundo só aceita coordenadas de `(0,0)` a `(59,39)`. Tudo o resto está "vazio". Terá que se "normalizar"as coordenadas. É para isso que existe o `World.normalize()`. Mas nem deve ser preciso usar esse método diretamente, porque o mundo disponibiliza os métodos `World.point()` e `World.translate()` para criar e fazer translações de pontos num dado mundo (com normalização incluída). Por isso, `p0 = world.point(0,0); p1 = world.translate(p0, Point(0,-1));` dará `p1==Point(0,39)`. Há mais explicações e exemplos no fim de `world.py`.

6. **Questão**: Os agentes conseguem passar de um lado do mapa para o outro pelas extremidades, mas a visão também passa?

   **Resposta**: Sim, o mundo tem topologia toroidal (como um doughnut): não há qualquer limite correspondente aos lados do retângulo, nem para a visão. Por isso, se o agente estiver em (59,10), conseguirá ver comida em (0,10), por exemplo.

7. (2017/11/19) O prazo da primeira entre foi adiado para 2017/11/27. Rectificado para 2017/11/28.

8. Devemos ter em atenção becos sem saída?

   **Resposta**: O mundo pode ter becos sem saída, quer seja gerado aleatoriamente, quer seja lido de ficheiro. Devem ter isso em consideração.

9. Podemos assumir que os testes serão feitos com custos/limites de recursos iguais aos atuais?

   **Resposta**: Para a primeira entrega podem admitir que os parâmetros serão idênticos, a menos dos "mapas"usados, claro. Para a entrega final é provável que mudemos alguns parâmetros, avisando com antecedência.

10. (2017/11/23) Longlife v1.4 was released on `tpg-ia-iia-longlife`. Read the header comments on `game.py` or do `git log` to see what changed. These changes have no impact on your agent code.

11. É permitido pré-processarmos no construtor do agente ou deveríamos processar apenas na função chooseAction?

    **Resposta**: Sim, podem fazer um pré-processamento no construtor. Não estamos a prever qualquer custo pelo processamento no construtor, desde que dentro de limites razoáveis.

12. O TPG vai ser avaliado em mapas com paredes geradas aleatoriamente ou vai ser testado nos mapas que foram incluidos no repositorio?

    **Resposta**: Para garantir igualdade de condições, os mapas serão fixos (não aleatórios), e não serão apenas os que constam no repositório Alguns mapas serão parecidos com os gerados aleatoriamente pelo jogo, mas "congelados"para serem iguais para todos. Outros serão mapas desenhados "à mão", parecidos com alguns dos fornecidos no repositório.

13. (2017/12/04) Check out your agent scores in the `LongLife-Dashboard`:

    `https://docs.google.com/spreadsheets/d/1ULhc3Zqk_w0Ur56TyxCswVGFIDQsdtO92X2F48BlPNg`

    `/edit?usp=sharing`

    The scoreboard includes some comments explaining what it means. Point to green areas to see the comments. NOTE: *these are not grades*. Grades will be derived from these scores, after appropriate selection and normalization.

14. Algumas notas adicionais:

    A primeira coluna (corresponde ao campo `agent`) inclui o número mecanográfico do aluno que submeteu o trabalho.

    A coluna `time` mostra o tempo real (em segundos) da execução completa do teste, que deve ser próximo do tempo total de processamento. Não será relevante para a nota, mas pode servir para diagnosticar problemas. Um agente que gasta muito tempo, mas obtém um score baixo, talvez precise de "pensar"menos, ou investir numa forma mais eficiente de pensar.

15. (2017/12/05) `LongLife v1.5` is now in the `tpg-ia-iia-longlife` repository. This is the version used to score the agents on the first submission. In this the version, an agent that takes too long to `chooseAction()` is interrupted. It would die anyway, by running out of nutrients. Also, all maps were moved into a folder. We added the two "frozen random"maps that were used in the tests.

16. (2017/12/07) O prazo extendido para a entrega final do TPG termina em 2017/12/15 pelas 23h55. Os parâmetros do jogo serão os mesmos que na primeira entrega.

17. Em relação ao Powerpoint, quais serão os pontos mais importantes a incluir?

    **Resposta**: Devem explicar os aspectos principais da concepção do agente desenvolvido, nomeadamente arquitectura, algoritmos e eventuais resultados que considerem revelevantes. Devem procurar realçar as qualidades do trabalho de forma sintética, e em estilo de apresentação, como se espera num Powerpoint. Podem incluir diagramas, figuras e gráficos.