Artificial Intelligence for StarCraft 2

Carlos Cardenas-Ruiz, Emilio Tonix-Gleason, and Julia Rodriguez-Abud

Artificial Intelligence

Abstract

The present file show material of the machine learing class. The course compromises three main parts: searching algorithms, bayesian networks, and multilayer perceptrons. The aim is to develop a StarCraft 2 agent, which is based on Pysc2 Python framework. There were scripted 5 agents, the first 3 agents are developed in smaller maps called minimaps. This reduce time and space complexity for each agent. However the following agents are on a map called simple64, that emulates a more real player experience. Most of our agents had good results, but there are not quite perfect. This could be because our realase periods were short, and our learning curve of the Pysc2 was slow. Nevertheless, due to the code is now on github, the learning curve of Pysc2 should be shorter, because the examples and algorithms implementations gives a good reference.

Carlos Cardenas-Ruiz is with the Department of Computater Science, Cinvestav, Guadalajara, México, e-mail: carlos.cardenas@cinvestav.mx.
Emilio Tonix-Gleason is with the Department of Computater Science, Cinvestav, Guadalajara, México, e-mail: xxx@xxx.xxx.
Julia Rodriguez-Abud is with the Department of Computater Science, Cinvestav, Guadalajara, México, e-mail: julia.rodriguez@cinvestav.mx.

Index Terms

CONTENTS

I. Introduction

A. Installation

The easiest way to get PySC2 is to use pip:

python 2.7

```
$ pip install pysc2
```

python 3.X

```
$ pip3 install pysc2
```

For more info, templates and documentation about PySC2, visit the website

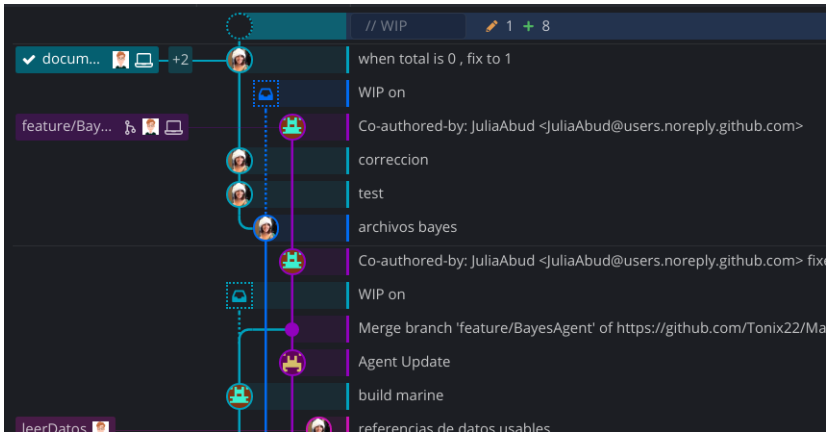
B. About PySC2

It a multiplatform python framework, to read and write data to starcraft II. This allow to implement certain automatic strategies based on learning algorthims. It is a non official deepmind product, use for competitive programing in the machine learning area.

C. Workflow

To develop and produce software we use gitflow strategy. This one allow us to separate tasks as working in batches, which are shown as features of a code implementation. From the git point of view, we use git branching and merging, to do parallel work, and code review. When the task are done, we met to do the branching merge.

Fig. 1. GitFlow



D. Roadmap

We use DevOps methodology. DevOps focuses on bringing the operations lifecycle into the same agile experience as the development teams. When adopting the DevOps philosophy, the team will remain responsible for a release for the entire lifecycle of the product.

1) Select StartCraftII map or minigame
2) Understood the problem
3) Plan a solution with the course tools.
4) Assing task and features
5) Indiviual task and coding
6) Merging and code review
7) Test
8) Release git
9) Show product to teacher for final review and feedback

Our planning is done in a canvan board.

Fig. 2. Canvan board

II. Uninformed Search

A. Beacon Agent

A map with 1 Marine and 1 Beacon. Rewards are earned by moving the marine to the beacon. Whenever the Marine earns a reward for reaching the Beacon, the Beacon is teleported to a random location (at least 5 units away from Marine).

Initial State

1 Marine at random location (unselected) 1 Beacon at random location (at least 4 units away from Marine)

Rewards

Marine reaches Beacon: +1

End Condition

Time elapsed

Time Limit

120 seconds

B. Mesh

Map is shown in a 64x64 array fashion, this is equivalent to 4K elements to compare and analyse for any recursive algorithm. This means it will significantly underperform the user expirience. It was done a quantisationon of bigger grids as a result of doing less iteration of the algorithms. For example, if we originaly had a grid of 64X64 , then we could transform it to 8x8 grid.

C. Iterative Deepening Search (IDS)

D. Bellman Ford Implicit (BFI)

E. Conclusion

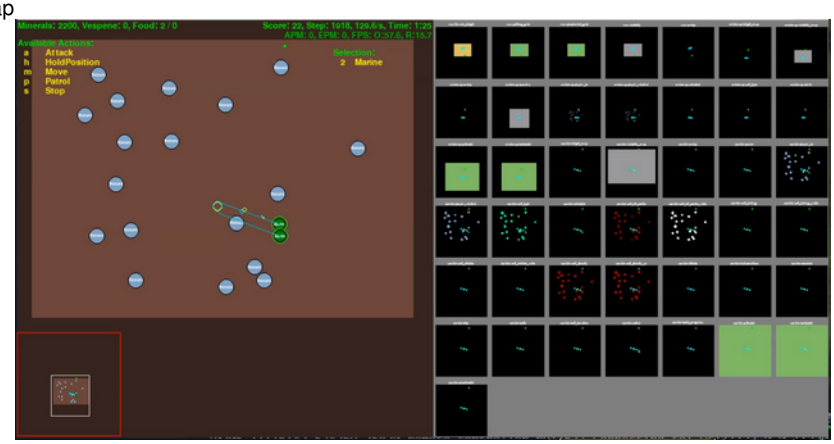
III. Informed Search

A. CollectMineralShards

Description

A map with 2 Marines and an endless supply of Mineral Shards. Rewards are earned by moving the Marines to collect the Mineral Shards, with optimal collection requiring both Marine units to be split up and moved independently. Whenever all 20 Mineral Shards have been collected, a new set of 20 Mineral Shards are spawned at random locations (at least 2 units away from all Marines).

Fig. 3. Collect Minerals Map

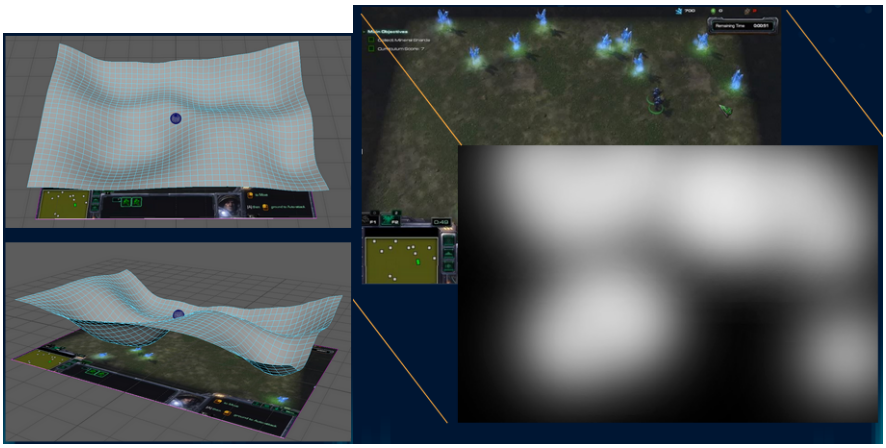


B. Brush

It generates a squared matrix with a Gaussian distribution. We call the output as heightmap. The idea of the heightmap is to generate deep used in the HillClimbing algorithms, as a point on the new mapping is higher, the density of minerals grows too.

- Map - Matrix with the size of our game screen (initialized with 0s)
- We can stamp out our brush
  - Add the values of our brush over a coord in Map
- If we stamp several times we get our map with concentration point

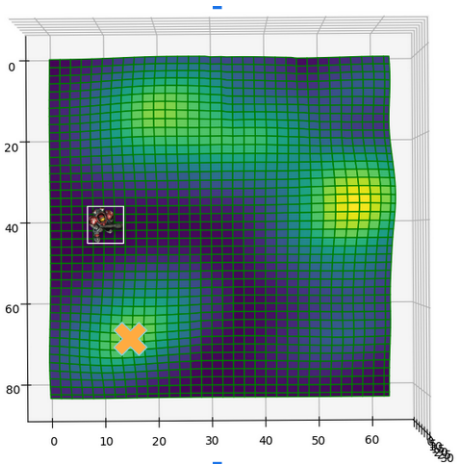
Fig. 4. Height map



C. A*

This version of A* try to find a next first maximum local only using a heuristic based in the value of coordinates of the mesh. The real problem is that needs a lot of calculus when the map is almost empty, because there are a lot of “plain ground” and that make difficult to find the maximum.

Fig. 5. Upper view of heights used in A*



D. HillClimbing and Simulated Annealing

Using the heihgt map, we could procced to find a max height point, as a best option to gather minerals. This is hill climbing algorithm, however a problem is that we could get stuck in a local maxima, so our strategy is to implement a chance given by a probability distribution, in this case we use a temperature boltzmann distribution $e^{\frac{\Delta E}{T}} > rand(0,1)$. The temperature will decres by an alpha factor inside a loop. For tuning the alpha we did the following equations. $T_0 = inital\,temperature$, $a = decreasing\ factor$, $n = iterations$, $T_f = final\,temperature$

$$T_0 * a^n = Tf$$

$$a = e^{\frac{ln(\frac{T_f}{T_0})}{n}}$$

As shown below there is an aplication of the simulated anealing, the chance part is designed with the equation above.

Algorithm 1 Simulated Annealing

```
Algo(simulated annealing):
  for Temp=Tmax to Tmin:
    #************CURRENT***********#
    Energy_current = E(Current) # Function cost at given point
    #************NEXT**********#
    N = Next(c) # go for next neighbour
    Energy_Flanders = E(Next) # Function cost at Flanders
    #************DELTA**********#
    Delta_Energy = Energy_Flanders - Energy_current
    #************UPDATE**********#
    if(Delta_Energy >0): # if positive
      Current = Next
    #************CHANCE**********#
    #Delta was negative, lets give another chance and
    #throw a probabilistic shot, maybe we update current
    else if (e^(Delta_Energy/Temp) > rand(0,1)):
      Current = Next
```

E. Conclusion

IV. Min Max and Probabilistic

A. FindAndDefeatZerglings

1) Description: A map with 3 Marines and an endless supply of stationary Zerglings. Rewards are earned by using the Marines to defeat Zerglings, with the optimal strategy requiring a combination of efficient exploration and combat. Whenever all 25 Zerglings have been defeated, a new set of 25 Zerglings are spawned at random locations (at least 9 units away from all Marines and at least 5 units away from all other Zerglings).

B. Alpha-beta Prunning / Minmax

C. Cost function with Heuristic

D. Conclusion

V. Bayesian Networks

VI. Conclusions

...

Appendix A
First appendix

Citation: [1]

Appendix B
Second appendix

Acknowlegment

bla bla

References

[1] N. H. F. Beebe. (2010, Dec.) TEX user group bibliography archive. [Online]. Available: http://www.math.utah.edu/pub/tex/bib/index-table.html

Replace this box by an image with a width of 1 in and a height of 1.25 in!

Your Name All about you and the what your interests are.

Coauthor Same again for the co-author, but without photo