Javier de Velasco Oriol A01202564
Germán Alamilla A01746968
Intelligent Systems
Santiago Enrique Conant Pablos
September 21th 2017

Assignment 3

The blob game can be programmed as an adversarial search problem and then solved via Alpha Beta Minimax optimization.

This game is primarily focused on maintaining the most amount of blobs of the own players colour once either the game closed down or the complete enemy is wiped out. The mechanics of the game are pretty simple, one chooses a direction and all the blobs of the players colour move towards that by one space, eating any enemy if they get the same blob. Furthermore if they step into the outside zone the blobs get deleted. Another feature is that once there are no blobs in a line or in a column then that area gets reduced.

To implement this is quite easy in a computer, as it can be simply put into a matrix, with the direct coordinates where each blob is and the corresponding colour. It can also be organized in a tuple manner. The restrictions about direction and the shrinking zones can also be easily implemented in a simple manner.

They are indeed worth it to consider as the computational cost is quite small, especially with the current computers of today, and if a library like Numpy is used it can be even more efficient.

Some strategies can be implemented in the eval function, but there are also some that can't. For example if they depend on multiple states or if they require more complex information or inferences from the state. Plus the function is limited to mathematical and logical equations.

The weights assigned to evaluation function features can be computed with a meta-heuristic or even a hyper-heuristic algorithm. Multiple options are Bayesian models, Genetic Algorithms, Support Vector Machines, etc.

The evaluation function for the blob game is actually quite simple

$$\alpha len(ownBlobs) - \beta len(enemyBlobs) - \gamma Risk$$

Where risk stands for the amount of opportunities for blobs to be taken as they are one block away from and enemy blob. Alpha, Beta and Gamma are weights which experimentally were chosen as 1, 0.8 and 0.05. This stems from the fact that having many of your own blobs is the most important part of the game, then trying to minimize the enemies and finally trying to avoid putting blobs into risky positions.

For the situation 1 that position has a value of -2.05 for MIN

Situation 2 has a value of 1.90 for MIN

Javier de Velasco Oriol A01202564
Germán Alamilla A01746968
Intelligent Systems
Santiago Enrique Conant Pablos
September 21th 2017

And situation 3 has a value of 2.15 for MAX

The Pic-a-Pix problem can be formulated as a CSP problem, albeit in a not so trivial manner.

The chosen variables for the CSP are every square in the 5x5 matrix. Thus we would have 25 different variables. We will organize them in a 5x5 matrix with the following naming:

$$Var_{xy}: \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{bmatrix}$$

$$where\ x\ stands\ for\ the\ row\ and\ y\ for\ the\ column\ of\ each\ variable.$$

Regarding values, they can take the 3 possible colours or have no colour:

$$D_i(1st\ Problem): \{Red, Green, Yellow, None\}$$

$$D_i(2nd\ Problem): \{Black, Orange, Yellow, None\}$$

For the actual implementation this had to be modified, as the problem was that there was no way to manage to check multiple variables for their values as the algorithm went and checked individually each constraint so it was impossible to do a check for the total number of current colours in a column and a row. Thus values defining the current state were implemented.

And for constraints the major one is that no two adjacent blocks might have the same color, that every row must have the specified number of squares with the defined colours in that order, and that every column must also have the specified colours and order.

$$ColourRow_{xy} = \begin{bmatrix} A_1 & B_1 & C_1 & N_1 \\ A_2 & B_2 & C_2 & N_2 \\ A_3 & B_3 & C_3 & N_3 \\ A_4 & B_4 & C_4 & N_4 \\ A_5 & B_5 & C_5 & N_5 \end{bmatrix}$$

$$where\ x\ stands\ for\ the\ colour\ and\ y\ for\ the\ row, A\ is\ the\ first\ colour, B\ the\ second, C\ the\ third\ and\ N\ None$$

Javier de Velasco Oriol A01202564
Germán Alamilla A01746968
Intelligent Systems
Santiago Enrique Conant Pablos
September 21th 2017

$$ColourColumn_{xy} = \begin{bmatrix} A_1 & B_1 & C_1 & N_1 \\ A_2 & B_2 & C_2 & N_2 \\ A_3 & B_3 & C_3 & N_3 \\ A_4 & B_4 & C_4 & N_4 \\ A_5 & B_5 & C_5 & N_5 \end{bmatrix}$$

$$where\ x\ stands\ for\ the\ colour\ and\ y\ for\ the\ column.$$

$$OrderColumn_x = [C_1 \quad C_2 \quad C_3 \quad C_4 \quad C_5]$$

$$Where\ x\ is\ the\ column, and\ each\ is\ avector\ for\ order{:}\ A, \{B, B\}, B, C$$

$$OrderRow_x = [R_1 \quad R_2 \quad R_3 \quad R_4 \quad R_5]$$

$$Where\ x\ is\ the\ row, and\ each\ is\ a\ vector\ for\ order(Ex){:}\ A, \{B, B\}, B, C$$

Challenge 1 Initial:

$$Var_{xy}{:}\ zeroes$$

$$ColourRow_{xy} = \begin{bmatrix} 2 & 0 & 0 & 3 \\ 1 & 1 & 1 & 2 \\ 4 & 0 & 0 & 1 \\ 3 & 1 & 0 & 1 \\ 2 & 0 & 0 & 3 \end{bmatrix}$$

$$ColourColumn_{xy} = \begin{bmatrix} 2 & 0 & 0 & 3 \\ 1 & 1 & 1 & 2 \\ 4 & 0 & 0 & 1 \\ 3 & 1 & 0 & 1 \\ 2 & 0 & 0 & 3 \end{bmatrix}$$

$$OrderColumn_x = \left[\{\{R, R\}\} \quad \{Y, R, G\} \quad \left\{\{R, R\}, N, \{R, R\}\right\} \quad \{R, G, R, R\} \quad \{\{R, R\}\}\right]$$

$$OrderRow_x = \left[\{\{R, R\}\} \quad \{Y, R, G\} \quad \left\{\{R, R\}, N, \{R, R\}\right\} \quad \{R, G, R, R\} \quad \{\{R, R\}\}\right]$$

Challenge 2 Initial:

$$Var_{xy}{:}\ zeroes$$

Javier de Velasco Oriol A01202564
Germán Alamilla A01746968
Intelligent Systems
Santiago Enrique Conant Pablos
September 21th 2017

$$ColourRow_{xy} = \begin{bmatrix} 0 & 2 & 1 & 2 \\ 2 & 2 & 0 & 1 \\ 0 & 4 & 0 & 1 \\ 0 & 2 & 1 & 1 \\ 0 & 0 & 2 & 3 \end{bmatrix}$$

$$ColourColumn_{xy} = \begin{bmatrix} 0 & 0 & 2 & 3 \\ 1 & 1 & 0 & 3 \\ 0 & 4 & 1 & 0 \\ 1 & 3 & 0 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix}$$

$$OrderColumn_x = \left[\{Y, Y\} \quad \{B, O\} \quad \{\{O, O, O, O\}, Y\} \quad \{O, B, \{O, O\}\} \quad \left\{\{O, O\}, Y\right\}\right]$$

$$OrderRow_x = [\{Y, \{O, O\}\} \quad \{B, O, B, O\} \quad \{\{O, O, O, O\}\} \quad \{Y, \{O, O\}\} \quad \{Y, Y\}]$$