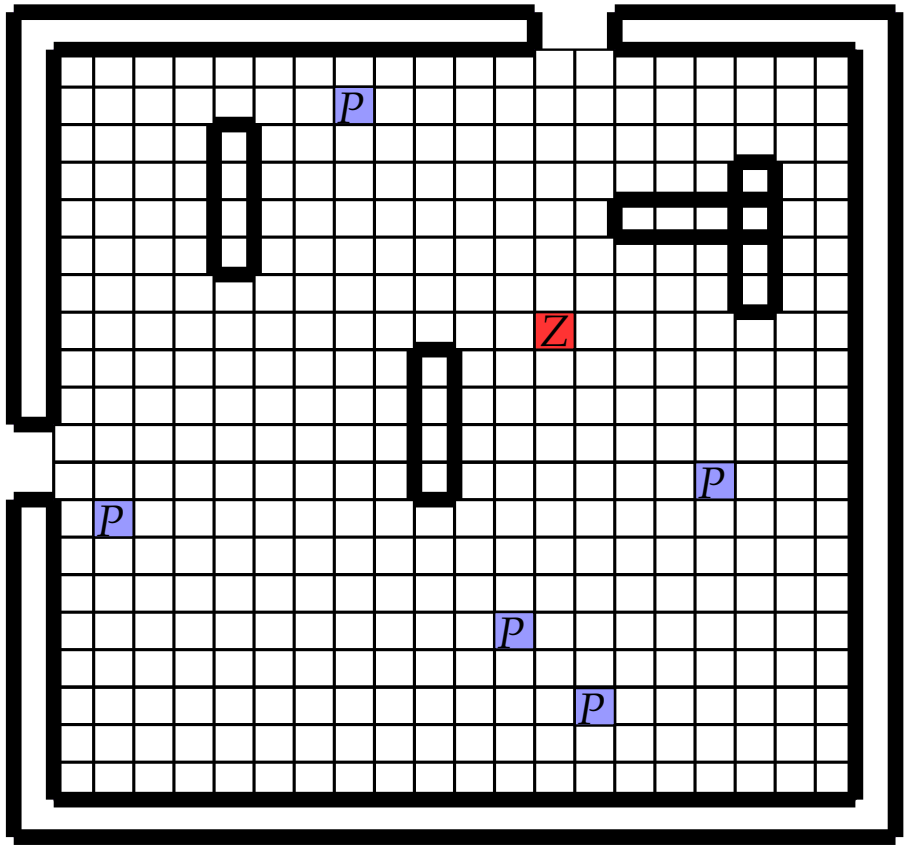
Methods of AI in computer games "Cops and thief" game (Graph search algorithms)

Przemysław Klęsk
pklesk@wi.zut.edu.pl

Department of Methods of Artificial Intelligence and Applied Mathematics

„Cops and thief” — game illustration



On a two-dimensional board with a square-like grid of fields we have 1 thief and 5 policemen. The goal of policemen is to catch the thief. The goal of thief is to remain uncaught as long as possible or to run away out of the board.

Details (1)

1. The board can, in general, be of size $n \times n$. By default: 20×20.
2. There are 4 possible moves are or lack of movement: $\{\leftarrow, \rightarrow, \uparrow, \downarrow, \circ\}$.
3. *The number of policemen* can be a parameter. By default: 5 policemen.
4. In edges of the board there are gates. Parameters: *number of gates* (by default: 2) and *gate width* (by default: 2).
5. If the thief stands onto any of gate fields, it implies its escape out of the board (win of thief).
6. There are rectangular obstacles on the board — walls. Parameters: *number of walls* (by default: 4), *wall length* (by default: 4). Each wall is of dimensions $1 \times$ *wall length*. Wall can be oriented horizontally or vertically.
7. Both gates and walls can move. For all gates the following 2 parameters govern their movement: *probability of movement* P_{GM}, *probability of direction change* P_{GC}. Possible gate movements: clockwise or counter-clockwise. Defaults: $P_{GM} = 0.5$, $P_{GC} = 0.01$. Analogical parameters for walls are: $P_{WM} = 0.75$, $P_{WC} = 0.05$, however walls are allowed to move in all 4 directions. Probabilities of position changes can be associated with velocities.
8. A main loop shall be carried out $t = 0, 1, 2, \ldots, T - 1$ (discrete clock). One may think that executed movements take place in the moments the clock makes a tick.
9. Players should plan k of their movements once per each k time moments (before). By default: $k = 5$. Calculations required to plan the movements must be performed within a time limit of $\Delta = 500$ *ms*.

Details (2)

1. Walls are transparent to each other — they may overlap (cross) each other.
2. Gates are transparent to each other.
3. Each wall is not allowed to change its orientation (horizontal, vertical) throughout the game.
4. Walls are not transparent to players. If a player chooses a move, which turns out to direct him to a field occupied by a wall, then the player's move does not take place (can be identified with an empty movement $\circ$).
5. Walls bound off edges of the board.
6. Movements are executed "simultaneously" by all objects (players, gates, walls) along with the ticks of the discrete clock, but the priority to occupy fields is given to walls and gates.
7. Policemen are not allowed to occupy gate fields (to skulk in a gate for thief).
8. At corners of board, gates of width 2 allow to go only through 1 of its fields.
9. The thief is considered to be caught when its position coincides with a position of some policeman.

Scoring — payoffs in the game

Payoffs

1. The payoff for policemen equals the negative payoff for the thief.
2. The thief caught at the moment t receives t points.
3. In particular: if the main loop finishes and the thief remains uncaught, the thief receives T points.
4. If the thief runs away through a gate at the moment t, he receives $2T - t - 1$ points. An escape at the last time moment (when $t = T - 1$) is paid off equally as a survival.

Environment — game engine

Algorithm

1. **Initialization.** Pick on random (no conflicts) initial positions for players, gates and walls.
2. **Players instantiation.** Create in memory objects for thief and policemen.
3. **First display.** Display initial state of game.
4. **Main loop.** For $t = 0, 1, \ldots, T-1$ do the following steps:
   1. **Displacement of gates and walls.** For each gate and wall check if the probability of direction change and / or of movement took place. Do suitable movements. Abort a wall movement if this movement leads the wall to an old position of a player.
   2. **Displacement of players.**
      1. If $t \,\%\, k = 0$ then: for each player create a thread being a wrapper around the players object; execute the thread with a time limit Δ; inside the thread execute on the object a method which calculates the plans for the next k moves; when time limit Δ is reached, collect the plans.
      2. Move the players according to planned moves. If a new position of a player coincides with a wall then abort the move.
   3. **Display.** Display current state of game.
   4. **Catch-check.** If the position of thief coincides with a position of some policemen then break the loop — the thief is caught.
   5. **Escape-check.** If the position of thief coincides with a position of a gate then break the loop — the thief escapes out of the board.

Players awareness

- Players are aware of all values set up as the game parameters. They receive them at the moment of being created and are allowed to perform any precalculations or initializations of wanted inner structures within unlimited time before the game starts.
- Players are aware of all game sates at each t. Information about the last k states is given to players from the game engine at the moment the wrapping-thread is being created, just before the method to calculate plans is triggered.
- Players are allowed to memorize historical states, in their own interest, e.g. in order to observe the current movement direction of gates and walls. However this must be carried out within time limits allowed per each move.

Solution suggestions (AI)

Heuristical changeable goal + A^* algorithm

- At each time moment, each player is suggested to execute A^* to calculate the trajectory directed to some target field. The field should be chosen at each time moment according to the proposed heuristics (no to be confused with a heuristics within A^*.
- The thief should try to estimate whether it is more attractive (at current time moment) to head for the gate or for some place distant from policeman.
- Policemen are allowed to should estimate wether to block the gate or to skulk for the thief.
- A good solution can also sample, predict and guess opponent's plants (its target field).

Solution suggestions (AI)

- **Rewarding policemen for local dispersion.** When surrounding and going for the thief, it is sensible to make policemen „head for" different fields. One may also consider perturbing their paths on purpose, whenever they come from a similar direction, in order to avoid a situation when pieces of paths coincide.
- **Avoiding policemen.**
  - The thief may try to establish a path avoiding policemen by treating their position as obstacles having a certain neighbourhood (choice of neighbourhood radius).
  - The neighbourhood around a policeman can be regarded as a *soft obstacle*, which allows to be crossed but at a larger cost (g summand in A^*). One may grade the costs within the neighbourhood by decreesing them along increasing radius.

Solution suggestions (AI)

- **Foreign A^*.** Within their time limits, players are allowed to execute additional A^* algorithms for the hypothetical trajectories of opponents (this requires guessing their target field) and to make their own paths cross such imaginary paths or go close to them.
- **Fast data structures.** In the scope of A^* algorithm it is reasonable to implement the *Open* set by means of a priority queue coupled with a hashmap; this would allow to quicly check if a state is in the *Open* set — $O(1)$ and could also allow for potential deletion of the state. Unfortunately the insertion after update is again of order $O(\log_2 n)$.