Isolation game analysis

Algorithm variations

The algorithm has the following variations:

- Delete_symmetry: Deletes moves which are symmetrical in an effort to reduce the search space. Two symmetrical moves (moves as well as the board must be symmetrical) will always have the same score and choosing one or another should be the same. Searching for symmetrical moves is expensive in time and computing power so it might not always be the right decision. When there are few moves left it is expected to take too long and have small chances of reducing the search space.
- Reflect: Tries to reflect other player moves once the center is occupied by one player, following Udacity's videos recommendations.
- Check_partitions: Check if there is a partition on the board. If there is one, it is possible to calculate who will be the winner (assuming both players will always choose the longer paths available). Therefore, it can calculate the winner (and return +inf or -inf) as soon as a partition is detected.
- Center_move: Always tries to take the center position. This is not always recommendable, but a first approximation to an optimal algorithm is to always try to take this position.

The following score calculations are used:

- Heuristics 1 & 4 calculate score using:
    - *float(pow(player_1_moves,gain_factor) - pow(player_2_moves, gain_factor))*
- Heuristics 2 & 5 calculate score using:
    - *float((player_1_moves * gain_factor) - player_2_moves)*
- Heuristics 3 & 6 calculate score using:
    - *float(player_1_moves / (player_2_moves +1))*

All of the above increase score if player has more moves and decrease score if the other player has more moves. They all achieve that in different manners, using exponential or linear functions.

Test cases

	reflect = True check_partitions = True center_move = True	reflect = False check_partitions = False center_move = False
Using delete_symmetry	Heuristic_1 test A Heuristic_2 test A Heuristic_3 test A	Heuristic_4 test A Heuristic_5 test A Heuristic_6 test A
Not using delete_symmetry	Heuristic_1 test B Heuristic_2 test B Heuristic_3 test B	Heuristic_4 test B Heuristic_5 test B Heuristic_6 test B

Note: Although reflect, check_partitions and center_move could be used independently, they are tested all true or all false to reduce the test cases.

Expected results

There is no prior knowledge about which score calculation algorithm will be the best (exponential or linear). Delete symmetry, reflection, center move and checking for partitions are expected to increase the player's chances of winning. The time that these variations of the algorithm might be detrimental for the results (and raise more timeouts) but nevertheless it is expected that test cases 1A, 2A or 3A are going to be the best.

Results

The experimental results are the following:

Test	ID_Improved performance	Student performance
Heuristic_1 test A	**75%**	**72%**
Heuristic_2 test A	63%	71%
Heuristic_3 test A	64%	67%
Heuristic_4 test A	69%	66%
Heuristic_5 test A	64%	69%
Heuristic_6 test A	**69%**	**79%**
Heuristic_1 test B	70%	71%
Heuristic_2 test B	69%	67%
Heuristic_3 test B	71%	64%
Heuristic_4 test B	**74%**	**74%**
Heuristic_5 test B	69%	67%
Heuristic_6 test B	69%	73%

All test cases were tested in the following environment:

Conclusions and recommendations

The report makes a recommendation about which evaluation function should be used and justifies the recommendation with at least three reasons supported by the data.

The single best result was given by **Heuristic_6 test A.** The student in this case outperformed ID_Improved by 10%. Heuristic_6 test B (without deleting symmetry when found), also had a good student performance and outperforming ID_Improved as well. Expected results were not achieved which can lead to think that the **cost of applying the algorithm variations was too much for the benefit obtained**.

A **linear score calculation** has proven to be **better** than an exponential one which seems reasonable as the player chances of winning do not improve exponentially with more moves but in a linear manner.

Heuristic 2 & 5 applied a **gain factor to the player's moves** and they did not apply it to the other player's move. This turned out to be **worst** which means that it is as good to have more moves as it is to reduce to opponent's available moves. That is why other heuristics performed better.

Student performed, in average, 2% worst when **symmetry deletion** was turned off (71% against 69%) while ID_Improved performed 3% better (67% vs 70%). It is **not** considered that these results are **conclusive enough** given the enhancements that could be made to the algorithm.

Enhancements could be made if gain factor was the results of A/B testing and not just an arbitrary number. Another variation could be to favour moves that increase the chances of creating a partition that benefits the player. It takes the check_partition part of the algorithm one step further.

Rodrigo Beceiro
March 2017