Author: Rafał Biegacz, October 2017

Review of ”Game Tree Searching by Min/Max Approximation” article

Introduction

The article proposes a new method of for calculating evaluation functions for Minimizer and Maximizer for tree-based solutions for games. This new method is called “min/max approximation”.

The algorithm presented in the article belongs to family of penalty-based algorithms and it bases its operations on a mean-value operator defined in the article as generalized p-mean. The results achieved for proposed algorithm are compared to MinMax algorithm with alpha-beta pruning (MMBA).

Summary

“min/max approximation” algorithm can be applied in iterative search heuristics while deciding which leaf in a partial game tree to expand. Penalty-based schemes define a concept of edge penalty which is a weight (nonnegative real number) assigned to every edge present in a game tree. Bigger edge penalties show worse moves and on the other hand smaller ones suggest better moves. Edge penalties can be used to calculate penalties between leafs and root of a game tree. Penalty-based algorithms calculate penalties between all leafs and root of a game tree and a leaf with the smallest penalty is the one that will be expanded - “min/max approximation” algorithm follows this pattern. In particular, “min/max approximation” proposes that weights (edge penalties) should be calculated as derivatives of approximation of evaluation function. Evaluation function used by “min/max approximation” is called generalized p-mean and is defined as follows:

$$M_p(a) = \left(\frac{1}{n}\sum_{i=1}^{n} {a_i}^p\right)^{1/p}$$

where p is an arbitrary (chosen heuristically) number (in the article authors used 10 as p value) and weights (edge penalties) are defined as follows:

$$w(x) = -\log\big(D(f(x), x)\big) \; and \; D(x, y) = \frac{\partial V_E(y)}{\partial V_E(x)}$$

where: f(x) is a

In general:

- $V_E(x)$ is approximation of evaluation function for a node x (please, refer to the article to learn how approximation function was defined) and evaluation function is generalized p-mean as it was defined earlier
- f(x) is a father of x which is a node that is a root for a game subtree to which x belongs to [please, refer to the article for more rigorous definition of a father notion).

The idea behind using derivatives of approximation function is that leafs with least penalties which will prefer leafs with biggest values of D(x, y) values.

Generalized p-mean function have very useful features. Generalized p-mean function is a very good approximation of: maximum function if p is reasonable big positive number and minimum function if p is reasonable negative number. There is also another valuable property of generalized p-mean compared to min/max operators – min/max functions pick only single values as their results while generalized p-mean function is able to “notice” that for example there is more than 1 big value having impact on generalized p-mean value which is helpful because in tree-based games is doesn’t eliminate other game options which are close to the best one (chosen in a given moment).

Unfortunately, calculating values of generalized p-mean function and its derivatives imposes additional computational cost on the algorithm.

Conclusions

The “min/max approximation” seems to be very interesting algorithms but it requires further research. It is more computationally demanding than for example MinMax algorithm w/ alpha-beta pruning and it means that it is slower (the results presented in the article show that it is 4 times slower) than MMAB in terms of CPU time required. The “min/max approximation” has also higher memory footprint compared to MMAB – this aspect might not be that important in case of simpler games but it can be significant in case of very complex games producing big game trees. Memory footprint aspect should be evaluated to bigger extent because nowadays computer memory is pretty chip so this aspect seems to be less important than aspect related to higher CPU time required by “min/max approximation”.

On the other hand, if we take into account the number of times “min/max approximation” or MinMax algorithm need to call their “get_move” function to get the next optimal move in a game, “min/max approximation” function wins compared to MinMax algorithm w/ alpha-beta pruning. It is possible to imagine that to speed up calculations done by “min/max approximation” one could use GPGPU or other specialized compute devices (e.g. FPGA-based) to make calculation time smaller and in this way making “min/max approximation” more competitive compared to MMAB algorithm.

Recapitulating, it would be interesting to explore whether “min/max approximation” couldn’t be combined with other algorithms to produce a more efficient/complete solution for tree-based solutions and look for more optimized implementations of “min/max approximation” algorithms using devices other than regular processors.