Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Playout policy adaptation with move features

Tristan Cazenave

PSL, Université Paris-Dauphine, LAMSADE – CNRS UMR 7243, 75775 Paris Cedex 16, France

ARTICLE INFO

Article history:
Received 21 January 2016
Received in revised form 11 June 2016
Accepted 20 June 2016
Available online xxxx

Keywords:
Monte Carlo tree search
Playout policy
Machine learning
Computer games

ABSTRACT

Monte Carlo Tree Search (MCTS) is the state of the art algorithm for General Game Playing (GGP). We propose to learn a playout policy online so as to improve MCTS for GGP. We also propose to learn a policy not only using the moves but also according to the features of the moves. We test the resulting algorithms named Playout Policy Adaptation (PPA) and Playout Policy Adaptation with move Features (PPAF) on Atarigo, Breakthrough, Misere Breakthrough, Domineering, Misere Domineering, Knightthrough, Misere Knightthrough and Nogo. The experiments compare PPA and PPAF to Upper Confidence for Trees (UCT) and to the closely related Move-Average Sampling Technique (MAST) algorithm.

1. Introduction

Monte Carlo Tree Search (MCTS) has been successfully applied to many games and problems [1]. The most popular MCTS algorithm is Upper Confidence bounds for Trees (UCT) [2]. MCTS is particularly successful in the game of Go [3]. It is also the state of the art in Hex [4] and General Game Playing (GGP) [5,6]. GGP can be traced back to the seminal work of Jacques Pitrat [7]. Since 2005 an annual GGP competition is being organized by Stanford at AAAI [8]. Since 2007 all the winners of the competition use MCTS.

Offline learning of playout policies has given good results in Go [9,10] and Hex [4], learning fixed pattern weights so as to bias the playouts. AlphaGo [11] also uses a linear softmax policy based on pattern weights trained on 8 million positions from human games and improved with hand crafted features.

The RAVE algorithm [12] performs online learning of moves values in order to bias the choice of moves in the UCT tree. RAVE has been very successful in Go and Hex. A development of RAVE is to use the RAVE values to choose moves in the playouts using Pool RAVE [13]. Pool RAVE improves slightly on random playouts in Havannah and reaches 62.7% against random playouts in Go.

The GRAVE algorithm [14] is a simple generalization of RAVE. It uses the RAVE value of the last node in the tree with more than a given number of playouts instead of the RAVE values of the current node. It was successful for many different games.

Move-Average Sampling Technique (MAST) is a technique used in the GGP program CadiaPlayer so as to bias the playouts with statistics on moves [5,15]. It consists of choosing a move in the playout proportionally to the exponential of its mean. MAST keeps the average result of each action over all simulations. Moves found to be good on average, independent of a game state, will get higher values. In the playout step, the action selections are biased towards selecting such moves. This is done using the Gibbs (or Boltzmann) distribution.

E-mail address: cazenave@lamsade.dauphine.fr.

http://dx.doi.org/10.1016/j.tcs.2016.06.024
0304-3975/

Please cite this article in press as: T. Cazenave, Playout policy adaptation with move features, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2016.06.024

Predicate Average Sampling Technique (PAST) is another technique used in CadiaPlayer. It consists in associating learned weights to the predicates contained in a position represented in the Game Description Language (GDL).

CadiaPlayer also uses Features to Action Sampling Technique (FAST). FAST learns features such as piece values using TD(λ) [16]. FAST is used to bias playouts in combination with MAST but only slightly improves on MAST.

Playout Policy Adaptation (PPA) [17] also uses Gibbs sampling, however the evaluation of an action for PPA is not its mean over all simulations such as in MAST. Instead the value of an action is learned comparing it to the other available actions for the states where it has been played. PPA is therefore closely related to reinforcement learning whereas MAST is about statistics on moves. Adaptive sampling techniques related to PPA have also been tried recently for Go with success [18].

Later improvements of CadiaPlayer are N-Grams and the last good reply policy [19]. They have been applied to GGP so as to improve playouts by learning move sequences. A recent development in GGP is to have multiple playout strategies and to choose the one which is the most adapted to the problem at hand [20].

A related domain is the learning of playout policies in single-player problems. Nested Monte Carlo Search (NMCS) [21] is an algorithm that works well for puzzles. It biases its playouts using lower level playouts. At level zero NMCS adopts a uniform random playout policy. Online learning of playout strategies combined with NMCS has given good results on optimization problems [22].

Online learning of a playout policy in the context of nested searches has been further developed for puzzles and optimization with Nested Rollout Policy Adaptation (NRPA) [23]. NRPA has found new world records in Morpion Solitaire and crosswords puzzles. Stefan Edelkamp and co-workers have applied the NRPA algorithm to multiple problems. They have optimized the algorithm for the Traveling Salesman with Time Windows (TSPTW) problem [24,25]. Other applications deal with 3D Packing with Object Orientation [26], the physical traveling salesman problem [27], the Multiple Sequence Alignment problem [28] or Logistics [29]. The principle of NRPA is to adapt the playout policy so as to learn the best sequence of moves found so far at each level.

PPA is inspired by NRPA since it learns a playout policy in a related fashion and adopts a similar playout policy. However PPA is different from NRPA in multiple ways. NRPA is not suited for two player games since it memorizes the best playout and learns all the moves of the best playout. The best playout is ill-defined for two player games since the result of a playout is either won or lost. Moreover a playout which is good for one player is bad for the other player so learning all the moves of a playout does not make much sense. To overcome these difficulties PPA does not memorize a best playout and does not use nested levels of search. Instead of learning the best playout it learns the moves of every playout but only for the winner of the playout.

NMCS has been previously successfully adapted to two-player games in a recent work [30]. PPA is a follow-up to this paper since it is the adaptation of NRPA to two-player games. PPA is an online learning algorithm, it starts from scratch for every position and learns a position specific playout policy each time. The PPA algorithm was first described in [17]. In this paper we extend it to use move features so as to have more specific statistics on moves.

The use of features to improve MCTS playouts has also been proposed in the General Game AI settings [31]. The approach is different from the approach in this paper since features are part of the state and are used to evaluate states. Instead we propose to use features to evaluate moves.

As our paper deals with learning action values it is also related to the detection of action heuristics in GGP [32].

We now give the outline of the paper. The next section details the PPA and the PPAF algorithms and particularly the playout strategy and the adaptation of the policy. The third section gives experimental results for various games. The last section concludes.

2. Online policy learning

PPA is UCT with an adaptive playout policy. It means that it develops a tree exactly as UCT does. The difference with UCT is that in the playouts it has a weight for each possible move and chooses randomly between possible moves proportionally to the exponential of the weight.

In the beginning PPA starts with a uniform playout policy. All the weights are set to zero. Then, after each playout, it adapts the policy of the winner of the playout. The way it modifies the weights according to the playout is similar to NRPA. In NRPA the weight of the move of the best playout is increased by a constant α and the weight of the other moves are decreased by a value proportional to the exponential of their weight. PPA does a similar update except that it only adapts the policy of the winner of the playout with the moves of the winner as there is no best playout to follow in PPA.

The NRPA adaptation algorithm is given in Algorithm 1. It reinforces all the moves of the best sequence found so far at a level. The algorithm is given here to highlight the differences with the PPA adaptation algorithm described later. For the sake of completeness we also give the NRPA algorithm in Algorithm 2. It is not suited to multi-player games since players alternate in a game and defining a best sequence is not as simple as in single player games.

The different algorithms we deal with are UCT, PPA, PPAF, MAST and MASTF. The playout code used by all algorithms except UCT is given in Algorithm 3. Each move of a playout is chosen with a probability proportional to its associated weight. The k constant has to be tuned for MAST and MASTF and it is always set to 1.0 for PPA and PPAF.

Algorithm 3 takes as parameters the board, the next player and the playout policy. The playout policy is an array of real numbers that contains a number for each possible move. The only difference with a random playout is that it uses the

Please cite this article in press as: T. Cazenave, Playout policy adaptation with move features, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2016.06.024