An Upper Confidence Bounds for Self-Adaptation of Playing Strategies in General Game Playing

N. Radhika Amareshwari\(^1\), S. Ramanjaneyulu\(^2\)
\(^1\),\(^2\)Assistant Professor, Dept. of Computer Science Engineering
Geethanjali College of Engineering & Technology, Cheryal, Hyderabad.
\(^1\)radhika.amareshwari@gmail.com
\(^2\)ramanji.csit@gmail.com

Abstract: The term General Game Playing (GGP) refers to a subfield of Artificial Intelligence which aims at developing agents able to effectively play many games from a particular class (finite, deterministic). It is also the name of the annual competition proposed by Stanford Logic Group at Stanford University, which provides a framework for testing and evaluating GGP agents. In this paper we present our GGP player which managed to win 4 out of 7 games in the 2013 preliminary round and advanced to the final phase. Our system (named MINI-Player) relies on a pool of playing strategies and autonomously picks the ones which seem to be best suited to a given game. The chosen strategies are combined with one another and incorporated into the Upper Confidence Bounds applied to Trees (UCT) algorithm. The effectiveness of our player is evaluated on a set of games from the 2012 GGP Competition as well as a few others, single-player games. The paper discusses the efficacy of proposed playing strategies and evaluates the mechanism of their switching. The proposed idea of dynamically assigning search strategies during play is both novel and promising.

Index Terms: Game Tree Search, General Game Playing, Monte Carlo Methods, Statistical Learning.

1. INTRODUCTION
Designing an artificial agent capable of exhibiting intelligent behavior in a variety of environments is one of the major goals of Artificial Intelligence (AI). General Game Playing (GGP) is a step towards the accomplishment of this long-term goal. In short, GGP domain encompasses finite, deterministic, synchronous, multi-player, perfect-information games, which are defined in the so-called Game Description Language (GDL) a subset of Prolog. Since it is generally assumed in the paper that the reader is familiar with the GGP challenge, we will only briefly recall its basic principles. Readers who are less experienced with GDL-based game descriptions may consult one of the Internet repositories of sample games.

Although official GGP tournaments started in 2005, it is worth recalling, that the history of game-independent play-ing agents dates back to more than 50 years ago with the introduction of Jacques Pitrat’s work. Modern research includes SAL, Hoyle and METAGAMER, the last one having been a direct conceptual predecessor of (modern) GGP agents. The idea of METAGAMER brought together researchers interested in developing intelligent agents capable of playing a predefined class of chess-like games without human intervention. This multi-game environment consisted of the definition of a class of allowed games, a communication protocol, a game generator and resource limitations. In some sense, the goal denoted in was to shift computer game-playing from an engineering back to a research discipline. GGP is currently the main application to multi-game playing.

On a general note, a program able to successfully participate in the GGP contest is a complex piece of software. In addition to the underlying AI methods and concepts, there are several “technical” issues having a great impact on the overall performance. All of them are worth research attention. In this paper, we present a player built on top of the so-called Upper Confidence Bounds Applied to Trees (UCT) method. In GGP framework, UCT is currently a state-of-the-art approach to searching the game tree. It is aimed at providing balance between exploration and exploitation. The player uses strategies (implementing the concept of an informed search) instead of a blind Monte-Carlo Tree Search (MCTS)
method. The novelty of the proposed method is twofold. Most of the MCTS state-of-the-art agents use random playouts (see section IV-A for details) which are complemented with light-weighted playing policies. They are based on some statistical properties rather than elements related to games. Such elements, in turn, are common parts of heuristic evaluation functions in classical methods. Our idea was to consider selected methods of game-state estimation and adapt them to GGP in such a way that they can be used as strategies to guide the subsequently quasi-random Monte Carlo simulations. Secondly, these strategies are dynamically evaluated in terms of being adequate for a given game. Consequently, the tree search is performed in such a way that the highest-evaluated strategies start to dominate and are chosen for most of the simulations. At the same time, worse strategies have a marginalized impact on the search process. Such an adaptation of strategies enables the avoidance of performing inadequate or wasted simulations.

We have tested several strategies and eventually chosen six of them including a purely random search. History Heuristic and Mobility are nowadays a standard in game AI, so we only did a light tuning to adapt them to our approach. The Approximate Goal Evaluation is a concept introduced in GGP in, however our realization of this idea is different (more details are presented in section VIII-B). Statistical Symbols Counting was proposed in our earlier work. The Exploration Strategy is designed from scratch for the purpose of our GGP program. The other novel part, albeit of somewhat lesser importance than the strategy mixing mechanism, is a modified formula applied to choosing a move to make. The aim of this formula is to perform a shallow min-max-type search around the root of the tree. Nevertheless, the experiments performed on games used in the GGP 2012 Competition proved that adaptive strategies have a major impact on the overall strength of our player.

2. RELATED WORK
The underlying idea of our GGP agent is to enhance simulation-based playing by speeding it up, lowering the amount of randomness and introducing mechanisms for dynamic discovery of strong lines of play. These improvements offer significant advantage over vanilla UCT approach. The improved player not only achieved a 28% better score against the test opponent but, most importantly, improved the record of wins from 1=9 to 5=9.

In the following sections the major components of MINI-Player are introduced and discussed. Experimental results are presented in section VII. In section VIII, the main differences to previous related works are outlined. Finally, section IX concludes the paper and discusses future research directions. The General Game Playing Competition has been played eight times since 2005. The first winner, Clune player, employed a state evaluation function along with a min-max tree search method in a way similar to the common approach for two-player zero-sum games. The function operated on a weighted linear combination of predefined features: payoff, control and mobility. Each feature was tested against its stability and correlation with the game score. One of the problems encountered in this approach was a complete lack of stable features detected for some types of games. The following year’s competition winner was FluxPlayer, whose underlying concept was based on the observation that games often possess common elements like boards, order relations, pieces and their quantities. Hence, Fluxplayer’s state evaluation procedure took into account the existence of certain predefined structures. Furthermore, FluxPlayer applied fuzzy logic in order to detect a degree of truthfulness of terminal state conditions. The system used a variant of an iterative deepening depth-first search method to explore the game space. A similar idea was presented in, but instead of semantic structures the authors chose to identify the syntactic ones. In particular, they demonstrated the way of detecting successor relation, boards, counters, markers, pieces and their quantities directly from GDL description. The three most recent winners used an MCTS method which had also been successful in Go playing programs. Instead of using any domain knowledge, these agents play random games until a terminal state is reached and fetch the game results.

In some cases, they offer no improvement or even cause a slight performance decrease, because of computational overhead. All of the proposed improvements are based on some statistical
optimizations without using explicitly any game features. Game playing is approached through a K-armed bandit stochastic simulation, an approach based on the detection of some game features is presented in. The authors analyze differences between the near end-of-game states in the form of GDL fluents. The fluents then become offensive or defensive features depending on whether they lead to a win state or prevent the player from reaching a loss state. This is an attempt at dynamic extraction of domain knowledge, however computationally quite expensive and with limited generalization capabilities, since features correspond to particular (fully grounded) GDL fluents.

3. IMPLEMENTATION
This section covers some preliminary concepts needed to understand the rest of the paper.

A. Monte Carlo Tree Search
The MCTS method became highly popular in the game community after becoming the first successful approach to Go. In the GGP competition it has been used since 2007. The basic idea of the MCTS simulation is to play a game acting randomly in order to reach the terminal state. In this state a goal value (a game result) for a particular player is computed and back-propagated to all states belonging to the respective path of play. This way a value of each state is estimated by the average result of all simulations which visited this state. Simulations are used to build a game tree whose nodes represent game states and edges represent players’ actions.

B. UCT
The UCT stands for Upper Confidence Bounds Applied For Trees. It is the most successful and widely used algorithm aimed at enhancing the selection process in MCTS. It provides a balance between exploration and exploitation.

C. History Heuristic
The history heuristic has been widely applied as a tree-search enhancement since 1989. The general idea is to transfer information about past actions taken in other states into the current state when the same action is available. In GGP, the history heuristic is typically used to affect the probability of choosing an unexplored action during simulations. For each action, the average score of simulations in which the action was played (regardless of the particular state in which it was performed) is stored.

D. Mobility
Mobility in games stands for the number of legal actions available for the agent (usually in comparison to other players). Generally speaking, a drastic change in mobility often corresponds to performing strong offensive or strong defensive actions (for many board games capturing pieces is a good example). Having a greater number of actions available to the player is usually considered beneficial. In GGP, mobility was implemented as part of the evaluation function in the first winning program.

Upper Confidence Bounds is a well-known algorithm for maintaining a trade-off between exploration and exploitation. A simple solution might be, for example, to play the currently best arm with the probability 0.5 and a randomly selected other one in the remaining cases. Such an approach, however, is not optimal. UCB offers a statistically justified solution which is optimal in the sense of the maximization of the expected value.

We also tried to order the strategies by their average results and assign to them predefined numbers of simulations based on their ranks. The distribution of the numbers of assigned simulations was optimized manually based on observations made in games we know how to play. In these games we have observed MINI-Player’s performance and the quality of its move-selection mechanism in the case of various distributions of the strategies used in simulations. Since we assign a number of simulations equal to the floor of R, we would not be able to distinguish between a better outcome and a worse one because all of them would become 1. The OF is used to add the fractional part, which is lost in the floor operation, to the next iteration of the allocation process. This enables maintaining appropriate proportions.

This method of allocation works well when there are only two strategies. When there are more of them, the distribution is too narrow, i.e. the quantities of simulations assigned to strategies are too close to
each other. At first we decided to test just the linearly proportional allocations, without the square component applied to the average scores, but in that case the numbers obtained were even closer.

Whether the first or the second method is better depends on the game. During the 2012 GGP Competition, method A was implemented by our agent. Most probably the optimal solution would consist of applying the UCB method with some, statically defined, lower and upper bounds, i.e. a kind of “the best of both-worlds” solution. An investigation into this issue is one of our current research goals.

4. RESULTS
We used the original authors’ implementation. Although the program has most probably been optimized since then, the authors do not claim to have introduced any major changes to their agent. CadiaPlayer is undoubtedly regarded as a state-of-the-art GGP player and is the most renowned prize-winning tournament participant. The second opponent was a clone of our system which did not use any simulation strategies other than a random one, enhanced by the history heuristic, and employed the classical move-selection method based on the nodes’ average scores. We call this player MINI-Player-C (“C” stands for “classic”). In order to ensure sufficient complexity and diversity of games, we used the same set of games which our agent played during the 2012 GGP Competition plus a few single-player ones (games of this kind were not present during the competition). Each game was played 270 times (rounds) with the roles swapped after each game. Role switching is important since in some games there exist favorable starting positions. Inspired by we used a mechanism of aggregating the results similar to CadiaPlayer’s. The game is considered won by a player if its score is greater than that of the opponent. The actual difference does not matter. A final decision which action to play during a match is taken on higher level than inner-simulation choices. Each strategy is too straightforward and not universal enough to be used as a unique, stand-alone tool. The root node in a game tree represents the current state. The selection of a move is technically equivalent to the selection of a direct child node of the root. Even though the whole UCT formula could, in principle, be applied to this task, it is better to use the exploitation part of equation 1 only, i.e. the average action value Q. During a match an agent should answer with the best possible move.

5. CONCLUSION
MINI-Player took part in the official 2012 General Game Playing Competition. Despite some inefficiency in play it managed to reach the final round, winning some games against former champions. Furthermore, the results of experiments evidently show an advantage of the MINI-Player’s playing skills over the plain UCT-based player (MINI-Player-C). Since both players use the same code framework the comparison is straightforward and there are no external factors which might have come into play during the experiment.

Performance in single-player games of both agents is also at a similar level. The results against CadiaPlayer proved that the proposed approach has potential and is worth further investigation and development. Certainly, the exact match-up is hard to judge since the pool of games available in GGP is practically unlimited and the decisive factors for gaining advantage in particular games are generally unknown. Although our assessment of MINI-Player’s play is generally very enthusiastic and favorable, the agent is definitely far from being a perfect player and suffers from several weaknesses. First of all, the efficiency of strategies measured during MC simulations does not always translate into a better play. This can be concluded from MINI-Player vs. MINI-Player-C direct tests, where in three games MINI-Player-C managed to score higher results.

REFERENCES


