Monte-Carlo Tree Search for Artificial General Intelligence in Games

Summary

This thesis investigates how search can be utilized to support *Artificial General Intelligence* (AGI) in games. The aim of AGI is creating Artificial Intelligence (AI) that can perform multiple different tasks in multiple different environments, can autonomously manage itself and possesses the ability to adapt to perform any new task that it might have never performed before. Planning is among the competences that an AGI is expected to possess. Given that games can model a wide variety of computationally hard problems, they can be considered a reasonable subset of all the planning tasks that we would like an AGI to be able to perform. Therefore, *general game playing* (GGP), which aims at creating programs that are able to play many (video) games with different properties without requiring any human intervention, is identified as a suitable domain to test search techniques for AGI. In addition, *Monte-Carlo Tree Search* (MCTS) is presented as a successful search technique for domains like GGP, where no specific domain knowledge is available.

Chapter 1 provides an introduction to games, discusses their relevance for AI and AGI and gives a brief description of the most popular search techniques used by game AI programs, among which is MCTS. The following problem statement guides the research.

> **Problem statement**: *How can the performance of Monte-Carlo Tree Search for general game playing be improved?*

To answer the problem statement four research questions have been formulated. They deal with (1) speeding up the interpretation of game rules written in a declarative language, (2) evaluating the use of global or local information to enhance the selection strategy of MCTS, (3) on-line tuning search-control parameters for MCTS, and (4) investigating the effect of search-control parameter randomization in MCTS.

Chapter 2 gives a formal definition of the games considered in the thesis and defines the terms commonly used when referring to tree search. Furthermore, Monte-Carlo methods are discussed, together with their link to MCTS, which is presented next. Finally, the UCT selection strategy is described both for sequential and simultaneous move games and relevant MCTS enhancements are discussed.

Chapter 3 introduces the test environments used to answer the four research questions: the *Stanford General Game Playing* (Stanford GGP) project and the *General Video Game AI* (GVG-AI) project. The first one focuses on GGP for abstract games, while the second on GGP for arcade-style video games. For each of the two environments the chapter introduces the language used to describe the

corresponding games and describes how the execution of a game run is managed. Moreover, the rules of the competition associated with each environment are presented, together with the common implementation details of the agents that are tested in the subsequent chapters of the thesis.

The Stanford GGP project represents the game rules using a declarative language known as the *Game Description Language* (GDL). This means that agents have to implement a mechanism that interprets the game rules written in GDL and computes all the elements that are necessary to reason on the game (i.e., game states, legal moves, etc.). This interpretation process is in general slow and might reduce the number of simulations that an agent can perform. This might hinder the performance of MCTS, which instead benefits from the more accurate statistics that can be collected with a higher number of simulations. This has led to the formulation of the first research question.

> **Research question 1**: *How can the process of interpreting on-line the game rules written in a declarative language be sped up?*

Chapter 4 answers the first research question by investigating an interpreter based on the representation of the GDL game rules as a Propositional Network (PropNet). Results show that a software implementation of the PropNet performs better than the GGP Base Prover, a custom made interpreter for GDL rules. The software implementation of the PropNet increases the number of game simulations by an average of two orders of magnitude with respect to the GGP Base Prover. Moreover, the speed of the PropNet is further increased by applying four optimizations, which, in order, remove PropNet components with constant truth values, remove PropNet propositions that do not have any special meaning for the game, detect and then remove components that will only assume a constant truth value during the game reasoning process, and remove components that have no output and thus no particular meaning for the game. The use of a cache that memorizes results previously computed by the PropNet is shown to increase the overall speed further. However, its use is recommended only for games with a small search space, like Chinese Checkers with 1 player and Tic Tac Toe, for which it increases the speed already in the first search steps. Furthermore, the use of a PropNet based reasoner enables the MCTS agent to reach a win rate close to 100% against an agent that uses the Prover. Finally, the speed of a PropNet-based reasoner can be further increased by at least one order of magnitude by encoding the PropNet on a Field Programmable Gate Array (FPGA) board. It may be concluded that using a PropNet with an optimized structure to represent game rules written in GDL is beneficial for MCTS-based agents, because they are able to perform more simulations in the given time frame and they can profit from hardware acceleration.

Previous research has shown that MCTS also profits from enhancing its selection strategy by increasing the amount of information used to guide the search. The Rapid Action Value Estimation (RAVE) strategy and the Generalized Rapid Action Value Estimation (GRAVE) strategy have been shown to be successful enhancements for the selection phase of MCTS. Both of them bias the selection towards actions that seem to perform generally well in the game. However, GRAVE uses more global information than RAVE to bias action selection in nodes that only have a low

number of visits. This strategy has been shown to perform better than RAVE on some variants of Go and a few other games, therefore it might be successful in GGP as well. This has led to the formulation of the second research question.

> **Research question 2**: *What is the effect of using locally or globally collected information to enhance the selection strategy for Monte-Carlo Tree Search?*

Chapter 5 answers the second research question proposing another variant of RAVE, the History Rapid Action Value Estimation (HRAVE) strategy, which biases action selection always using global information about the actions. It then compares the performance of RAVE, GRAVE and HRAVE on a set of games from the Stanford GGP project, both combined with a random play-out strategy and with the more informed MAST play-out strategy. Results show that, when RAVE variants are combined with a random play-out strategy, the performance of GRAVE is, in the worst case, comparable with the one of RAVE, both when using 1s or 10s play-clock. The performance of HRAVE, instead, is more game dependent, sometimes being better than RAVE or GRAVE and sometimes being worse. Moreover, when RAVE variants are combined with the Move Average Sampling Technique (MAST) used as play-out strategy, GRAVE still seems to be overall better than RAVE. However, its advantage is less than when both strategies are combined with random play-outs, and there are a few games where the combination GRAVE-MAST actually performs worse than RAVE-MAST. Additionally, the combination HRAVE-MAST seems to perform slightly less than both RAVE-MAST and GRAVE-MAST. Over all the experiments, the difference in performance between RAVE, GRAVE and HRAVE is not large. However, the overall win rate of GRAVE is never inferior to the one of RAVE and HRAVE, and it seems the most robust among all the RAVE variants. Therefore, it may be concluded that a strategy that starts biasing action selection with global information and uses more local information the more the nodes have been visited is the most suitable to enhance MCTS for GGP. In addition, and advantage of GRAVE is that it can be switched to a pure RAVE or a pure HRAVE strategy by simply modifying one of its parameters. With respect to the other two variants, this makes it more promising to be tuned on-line with the approach presented in Chapter 6.

Together with RAVE, many other enhancements for the different phases of MCTS have been applied successfully in GGP. Often, MCTS and its enhancements are controlled by multiple parameters that require extensive and time-consuming off-line optimization. Moreover, as the played games are unknown in advance, off-line optimization cannot tune parameters specifically for single games. It has to find values that perform overall well on a predefined set of games, with no guarantee that they will perform successfully also on unseen games. An alternative would be to adjust parameter values while playing each new game, therefore in an on-line fashion. This has led to the formulation of the third research question.

> **Research question 3**: *How can search-control parameters for Monte-Carlo Tree Search be tuned effectively on-line?*

Chapter 6 answers the third research question proposing an on-line tuning method for search-control parameters that enables MCTS to be self-adaptive during game play (SA-MCTS). Seven different strategies were introduced to decide how to allocate the available samples to test the parameter combinations: Multi-Armed Bandit (MAB) allocation, Hierarchical Expansion (HE), Naïve Monte-Carlo (NMC), Linear Side Information (LSI), an Evolutionary Algorithm (EA), the N-Tuple Bandit Evolutionary Algorithm (NTBEA) and the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES). The performance of on-line parameter tuning has been tested both on the Stanford GGP and the GVG-AI projects. Results show that among the tested allocation strategies to tune parameters on-line, the ones considering a discrete parameter domain and based on evolutionary algorithms perform best. NTBEA seems to have the best performance overall, but EA is also quite close. Results for the Stanford GGP project show that on-line parameter tuning is beneficial both for simple and more informed agents, when two parameters are tuned. The performance decreases when tuning more parameters. However, when tuning four parameters, the performance is still close to the one obtained using fixed default parameter values. Results for the GVG-AI project show that it is harder to tune parameters on-line with much shorter time settings, even when the number of tuned parameters is small. However, it may still be better to tune parameters on-line when fixed parameter settings might be sub-optimal, such as it is seen in the game Modality. It may be concluded that the proposed approach is useful when off-line parameter tuning is infeasible, or in contexts like GGP, both for abstract and real-time games, where parameters cannot be tuned in advance for each game. Moreover, it is useful when off-line tuned values might be sub-optimal for some games, or off-line tuning incurs in the risk of overfitting the values to the set of games selected for the purpose of tuning. It may also be concluded that on-line parameter tuning is robust against different types of opponents.

The success of on-line search-control parameter tuning on some of the tested games might be partially due to the randomization introduced by exploring different parameter combinations. This might be introducing diversification in the search process, making it explore different parts of the tree that would not be explored keeping the parameters fixed. Moreover, previous research has shown that adding randomization to certain components of the search might increase its diversification and improve its performance. In a domain like GGP, that deals with many games with different characteristics, adding more randomization might be a good strategy for some games. This has led to the formulation of the fourth research question.

> **Research question 4**: *What is the effect of randomizing search-control parameters for Monte-Carlo Tree Search?*

Chapter 7 answers this research question evaluating four different strategies that randomize search-control parameters for MCTS in GGP: randomization per game, per turn, per simulation and per state. Moreover, search-control parameter randomization is compared with fixed parameter settings and with on-line parameter tuning both in the framework of the Stanford GGP project and in the framework of the GVG-AI project. For the Stanford GGP project, results show that the randomization strategy that performs best is the one that randomizes parameter values

before each simulation, selecting such values within a predefined reasonable interval. Moreover, results show that for some games randomizing per simulation the value of a single parameter is better than keeping a good value fixed for the whole game. Furthermore, results show that the effect of parameter randomization depends on multiple factors, such as the game being played, which and how many parameters are being randomized and the type of opponent. It may be concluded that, although not always the best solution for all games, randomization within a given set of values is still beneficial in GGP for games where the fixed parameter settings optimized off-line on a predefined set of games are actually performing poorly. Moreover, parameter randomization might be a valid alternative to on-line parameter tuning when the number of parameters to tune is high and time settings are limited, because the problem of tuning them on-line becomes too hard due to the combinatorial complexity.

Chapter 8 concludes the thesis and gives indications for future research directions. The answer to the problem statement is based on the answers to the research questions given above. First, the process of interpreting the game rules written in a declarative language can be sped up by using a PropNet representation of these rules, optimizing the structure of such PropNet, and embedding the PropNet structure on an FPGA. By speeding up the process of interpreting the game rules, the number of simulations that can be performed by MCTS can be increased. Second, the selection strategy of MCTS can be enhanced using both locally and globally collected information about the available actions. In this case, the best approach is using a mix of global and local information, the first for states that have been visited less and the second for states that have been visited more, like the GRAVE selection strategy does. Third, search-control parameters can be tuned on-line and adapted to each new game being played, using the NTBEA strategy to allocate samples for evaluating parameter combinations. Fourth, randomizing search-control parameters within a predefined set of values before each simulation can be used as an alternative to on-line parameter tuning when the number of parameters to tune is high and the time settings are limited.

All the approaches presented in the thesis have been shown to enhance the performance of MCTS for a wide variety of games, without relying on game-specific pre-coded information. Although evaluated only on a subset of all the planning tasks that AGI is aiming to tackle, the games considered in this thesis present a wide variety of characteristics. They include abstract games, video games, deterministic and non-deterministic games, games with a discrete or continuous game flow, with sequential or simultaneous moves, with constant-sum or variable-sum payoffs, and with different numbers of players. Therefore, the presented MCTS enhancements are promising to also support search and planning for AGI.

The research presented in the thesis indicates several areas for future research. These include specific recommendations such as (i) further speeding up the PropNet reasoner and its implementation on an FPGA, (ii) further testing the RAVE variants, using different formulas to compute their parameters and combining them with different play-out strategies, (iii) improving on-line parameter tuning by testing other allocation strategies, increasing their adaptability to each played game and applying them to other domains, and (iv) testing parameter randomization with longer

time constraints and adding a mechanism that decides when and which parameters to randomize. More generic recommendations for future research suggest to test MCTS for AGI on more challenging domains and to focus on designing dynamic approaches that automatically adapt different aspects of the search and of the agent to each game or to each category of games being played.