Monte-Carlo tree search enhancements for one-player and two-player domains

Summary

This thesis is concerned with enhancing the technique of *Monte-Carlo Tree Search* (MCTS), applied to making move decisions in games. MCTS has become the dominating paradigm in the challenging field of computer Go, and has been successfully applied to many other games and non-game domains as well. It is now an active and promising research topic with room for improvements in various directions. This thesis focuses on enhancing MCTS in one-player and two-player domains.

Chapter 1 provides a brief introduction to the field of games and AI, and presents the following problem statement guiding the research.

Problem Statement: *How can the performance of Monte-Carlo Tree Search in a given one- or two-player domain be improved?*

Four research questions have been formulated to approach this problem statement. Two questions are concerned with one-player domains, while two questions are dealing with adversarial two-player domains. The four research questions address (1) the rollout phase of MCTS in one-player domains, (2) the selection phase of MCTS in one-player domains, (3) time management for MCTS in two-player tournament play, and (4) combining the strengths of minimax and MCTS in two-player domains.

Chapter 2 describes the basic terms and concepts of search in games. It also introduces the two classes of search methods used in the thesis: minimax-based search techniques for two-player games, and MCTS techniques for both one- and two-player games. Enhancements for both minimax and MCTS are explained as far as relevant for this thesis.

Chapter 3 introduces the test domains used in the following chapters. These include the one-player games SameGame, Clickomania, and Bubble Breaker, and the two-player games Go, Connect-4, Breakthrough, Othello, and Catch the Lion. For each game, its origin is described, its rules are outlined, and its complexity is analyzed.

In MCTS, every state in the search tree is evaluated by the average outcome of Monte-Carlo rollouts from that state. For the consistency of MCTS, i.e. for the convergence to the optimal policy, uniformly random rollout moves are sufficient.

However, stronger rollout strategies typically greatly speed up convergence. The strength of Monte-Carlo rollouts can be improved for example through hand-coded knowledge, or by automated offline tuning of rollout policies. In recent years, the topic of improving rollout policies online has received more and more attention, i.e. while the search is running. This leads to the first research question.

Research Question 1: *How can the rollout quality of MCTS in one-player domains be improved?*

Chapter 4 answers this research question by introducing Nested Monte-Carlo Tree Search (NMCTS), replacing simple rollouts with nested MCTS searches. Instead of improving a given set of rollout policy parameters either offline or online, calls to the rollout policy are replaced with calls to MCTS itself. Independent of the quality of the base-level rollouts, this recursive use of MCTS makes higher-quality rollouts available at higher levels, improving MCTS especially when longer search times are available. NMCTS is a generalization of regular MCTS, which is equivalent to level-1 NMCTS. Additionally, NMCTS can be seen as a generalization of Nested Monte-Carlo Search (NMCS), allowing for an exploration-exploitation tradeoff by nesting MCTS instead of naive Monte-Carlo search. The approach was tested in the puzzles SameGame, Clickomania, and Bubble Breaker, with relatively long time settings.

As it is known for SameGame that restarting several short MCTS runs on the same problem can lead to better performance than a single, long run, NMCTS was compared to multi-start MCTS. Level-2 NMCTS was found to significantly outperform multi-start MCTS in all test domains. In Clickomania, memory limitations limited the performance of very long searches, which gave an additional advantage to NMCTS.

Furthermore, level-2 NMCTS was experimentally compared to its special case of NMCS. For this comparison, NMCTS (just like NMCS) used the additional technique of move-by-move search, distributing the total search time over several or all moves in the game instead of conducting only one global search from the initial position. NMCTS significantly outperformed both level-2 and level-3 NMCS in all test domains. Since both MCTS and NMCS represent specific parameter settings of NMCTS, correct tuning of NMCTS has to lead to greater or equal success in any domain.

In *Upper Confidence bounds applied to Trees* or UCT, the most widely used variant of MCTS, the selectivity of the search can be controlled with a single parameter: the exploration factor. In domains with long solution lengths or when searching with a short time limit however, MCTS might not be able to grow a search tree deep enough even when exploration is completely turned off. The result is a search process that spends too much time on optimizing the first steps of the solution, but not enough time on optimizing the last steps. This leads to the second research question.

Research Question 2: *How can the selectivity of MCTS in one-player domains be improved?*

Chapter 5 answers this research question by proposing Beam Monte-Carlo Tree Search (BMCTS), a combination of MCTS with the idea of beam search. BMCTS expands a tree whose size is linear in the search depth, making MCTS more effective especially in domains with long solution lengths or short time limits. Like MCTS, BMCTS builds a search tree using Monte-Carlo simulations as state evaluations. When a predetermined number of simulations has traversed the nodes of a given tree depth, these nodes are sorted by a heuristic value, and only a fixed number of them is selected for further exploration. BMCTS is reduced to a variant of move-by-move MCTS if this number, the beam width, is set to one. However, it generalizes from move-by-move search as it allows to keep any chosen number of alternative moves when moving on to the next tree depth. The test domains for the approach were again SameGame, Clickomania, and Bubble Breaker, with relatively short time settings.

BMCTS was compared to MCTS both using one search run per position (single-start), and using the maximum over multiple search runs per position (multi-start). The experimental results show BMCTS to significantly outperform regular MCTS at a wide range of time settings in all tested games. BMCTS was also shown to have a larger advantage over MCTS in the multi-start scenario. In all test domains, multi-start BMCTS was stronger than multi-start MCTS at a wider range of time settings than single-start BMCTS to single-start MCTS. This suggests that the performance of BMCTS tends to have a higher variance than the performance of regular MCTS, even in some cases where the two algorithms perform equally well on average.

This observation led to the idea of examining how BMCTS would perform in a nested setting, i.e. replacing MCTS as the basic search algorithm in Nested Monte-Carlo Tree Search. The resulting search algorithm was called Nested Beam Monte-Carlo Tree Search (NBMCTS), and experiments showed NBMCTS to be the overall strongest one-player search technique proposed in this thesis. It performed better than or equal to NMCTS in all domains and at all search times.

In competitive gameplay, time is typically limited—in the basic case by a fixed time budget per player for the entire game (*sudden-death* time control). Exceeding this time budget means an instant loss for the respective player. However, longer thinking times usually result in better moves, especially for an *anytime* algorithm like MCTS. The question arises how to distribute the time budget wisely among all moves in the game. This leads to the third research question.

Research Question 3: *How can the time management of MCTS in two-player domains be improved?*

Chapter 6 answers this research question by investigating and comparing a variety of time-management strategies for MCTS. The strategies included newly proposed ones as well as strategies described in the literature, partly in enhanced form. The strategies can be divided into semi-dynamic strategies that decide about time allocation for each search before it is started, and dynamic strategies that influence the duration of each move search while it is already running. All strategies were tested in the domains of 13×13 and 19×19 Go, and the domain-independent ones in Connect-4, Breakthrough, Othello, and Catch the Lion.

Experimental results showed the proposed strategy called STOP to be most successful. STOP is based on the idea of estimating the remaining number of moves in the game and using a corresponding fraction of the available search time. However, MCTS is stopped as soon as it appears unlikely that the final move selection will change by continuing the search. The time saved by these early stops throughout the game is anticipated, and earlier searches in the game are prolonged in order not to have only later searches profit from accumulated time savings. In self-play, this strategy won approximately 60% of games against state-of-the-art time management, both in 13×13 and 19×19 Go and under various time controls. Furthermore, comparison across different games showed that the domain-independent strategy STOP is the strongest of all tested time-management strategies.

All time-management strategies that *prolong* search times when certain criteria are met take available time from the endgame and shift it to the opening of the game. All strategies that *shorten* search times based on certain criteria move time from the opening towards the endgame instead. Analysis showed that such a shift can have a positive or negative effect. Consequently, a methodology was developed to isolate the effect of a shift and judge the effect of a given strategy independently of it. Should the shifting effect be negative, it can be counteracted by introducing an explicit shift in the opposite direction.

One of the characteristics of MCTS is Monte-Carlo simulation, taking distant consequences of moves into account and therefore providing a strategic advantage in many domains over traditional depth-limited minimax search. However, minimax with $\alpha\beta$ pruning considers every relevant move within the search horizon and can therefore have a tactical advantage over the highly selective MCTS approach, which might miss an important move when precise short-term play is required. This is especially a problem in games with a high number of terminal states throughout the search space, where weak short-term play can lead to a sudden loss. This leads to the fourth research question.

Research Question 4: *How can the tactical strength of MCTS in two-player domains be improved?*

This research question is answered by proposing and testing *MCTS-minimax hybrids*, integrating shallow minimax searches into the MCTS framework and thus taking a first step towards combining the strengths of MCTS and minimax. These hybrids can be divided into approaches that require domain knowledge, and approaches that are knowledge-free.

Chapter 7 studies three different hybrids for the knowledge-free case, using minimax in the selection/expansion phase (MCTS-MS), the rollout phase (MCTS-MR), and the backpropagation phase of MCTS (MCTS-MB). Test domains were Connect-4, Breakthrough, Othello, and Catch the Lion. The newly proposed variant MCTS-MS significantly outperformed regular MCTS with the MCTS-Solver extension in Catch the Lion, Breakthrough, and Connect-4. The same held for the proposed MCTS-MB variant in Catch the Lion and Breakthrough, while the effect in Connect-4 was neither significantly positive nor negative. The only way of integrating minimax search into MCTS known from the literature, MCTS-MR, was quite strong in Catch the Lion and Connect-4 but significantly weaker than the baseline in Breakthrough, suggesting it might be less robust with regard to differences between domains such as the average branching factor. As expected, none of the MCTS-minimax hybrids had a positive effect in Othello due to the low number of terminal states and shallow traps throughout its search space. The density and difficulty of traps predicted the relative performance of MCTS-minimax hybrids across domains well.

Problematic domains for the knowledge-free MCTS-minimax hybrids can be addressed with the help of domain knowledge. On the one hand, domain knowledge can be incorporated into the hybrid algorithms in the form of evaluation functions. This can make minimax potentially much more useful in search spaces with few terminal nodes before the latest game phase, such as that of Othello. On the other hand, domain knowledge can be incorporated in the form of a move ordering function. This can be effective in games such as Breakthrough, where traps are relatively frequent, but the branching factor seems to be too high for some hybrids such as MCTS-MR.

Chapter 8 therefore investigates three more algorithms for the case where domain knowledge is available, employing heuristic state evaluations to improve the rollout policy (MCTS-IR), to terminate rollouts early (MCTS-IC), or to bias the selection of moves in the MCTS tree (MCTS-IP). For all three approaches, the computation of state evaluations through simple evaluation function calls (MCTS-IR-E, MCTS-IC-E, and MCTS-IP-E, where -E stands for "evaluation function") was compared to the computation of state evaluations through shallow-depth minimax searches using the same heuristic knowledge (MCTS-IR-M, MCTS-IC-M, and MCTS-IP-M, where -M

stands for "minimax"). MCTS-IR-M, MCTS-IC-M, and MCTS-IP-M are MCTS-minimax hybrids. The integration of minimax has only been applied to MCTS-IR before in this form. Test domains were Breakthrough, Othello, and Catch the Lion.

The hybrids were combined with move ordering and k-best pruning to cope with the problem of higher branching factors, resulting in the enhanced hybrid players MCTS-IR-M-k, MCTS-IC-M-k, and MCTS-IP-M-k. Results showed that with these two relatively simple $\alpha\beta$ enhancements, MCTS-IP-M-k is the strongest standalone MCTS-minimax hybrid investigated in this thesis in all three tested domains. Because MCTS-IP-M-k calls minimax less frequently, it performs better than the other hybrids at low time settings when performance is most sensitive to a reduction in speed. MCTS-IP-M-k also worked well on an enlarged Breakthrough board, demonstrating the suitability of the technique for domains with higher branching factors. Moreover, the best-performing hybrid outperformed a simple $\alpha\beta$ implementation in Breakthrough, demonstrating that at least in this domain, MCTS and minimax can be combined to an algorithm stronger than its parts.

Chapter 9 finally concludes the thesis, and gives possible directions for future work. The answer to the problem statement is twofold. First, the performance of MCTS in one-player domains can be improved in two ways—by nesting MCTS searches (NMCTS), and by combining MCTS with beam search (BMCTS). The first method is especially interesting for longer search times, and the second method for shorter search times. A combination of NMCTS and BMCTS can also be effective. Second, the performance of MCTS in two-player domains can be improved in two ways as well—by using the available time for the entire game more smartly (time management), and by integrating shallow minimax searches into MCTS (MCTS-minimax hybrids). The first approach is relevant to scenarios such as tournaments where the time per game is limited. The second approach improves the performance of MCTS specifically in tactical domains.

Several areas of future research are indicated. These include (1) applying NMCTS to non-deterministic and partially observable domains, (2) enhancing BMCTS to guarantee optimal behavior in the limit, (3) testing various combinations of time management strategies, and (4) examining the paradoxical observations of weaker performance with deeper embedded searches in MCTS-minimax hybrids, as well as studying which properties of test domains influence the performance of these hybrids.

Samenvatting

Dit proefschrift houdt zich bezig met het verbeteren van de *Monte-Carlo Tree Search* (MCTS) techniek om zodoende beslissingen te nemen in abstracte spelen. De laatste jaren is MCTS het dominante paradigma geworden in allerlei speldomeinen zoals Go. Tevens is het succesvol toegepast in verschillende optimaliseringsproblemen. MCTS is tegenwoordig een actief en veelbelovend onderzoeksdomein met ruimte voor verbeteringen in verscheidene richtingen. Dit proefschrift richt zich op het verder ontwikkelen van MCTS voor één- en tweespeler domeinen.

Hoofdstuk 1 geeft een kort introductie over de rol die spelen vervullen in de kunstmatige intelligentie. De volgende probleemstelling is geformuleerd om het onderzoek te sturen.

Probleemstelling: *Hoe kunnen we de prestatie van Monte-Carlo Tree Search verbeteren voor een gegeven één- of tweespeler domein?*

Vier onderzoeksvragen zijn geformuleerd voor het aanpakken van deze probleemstelling. Twee vragen houden zich bezig met éénspeler domeinen, terwijl de andere twee zich richten op tweespeler domeinen. De vier onderzoeksvragen adresseren (1) de Monte-Carlo simulatiefase van MCTS in éénspeler domeinen, (2) de selectiefase van MCTS in éénspeler domeinen, (3) tijdmanagement voor MCTS in tweespeler toernooien, en (4) het combineren van de kracht van minimax en MCTS in tweespeler domeinen.

Hoofdstuk 2 beschrijft de basisbegrippen en concepten voor het zoeken in spelen. Het introduceert ook twee klassen van zoekmethoden die worden gebruikt in het proefschrift: minimax technieken voor tweespeler domeinen, en MCTS technieken voor één- en tweespeler domeinen. Uitbreidingen voor zowel minimax als MCTS worden uitgelegd zover ze relevant zijn voor het proefschrift.

Hoofstuk 3 introduceert de gebruikte testdomeinen in het proefschrift. Deze bevatten de éénspeler domeinen SameGame, Clickomania en Bubble Breaker, en de tweespeler domeinen Go, Vier op 'n rij, Breakthrough, Othello en Catch the Lion. Voor elk spel wordt de achtergrond beschreven, de regels geschetst en de complexiteit geanalyseerd.

De sterkte van MCTS wordt mede bepaald door Monte-Carlo simulaties. Tijdens deze fase wordt zetten geselecteerd op basis van een simulatiestrategie. Een dergelijke strategie kan gebaseerd zijn op gecodeerde expertkennis of automatisch zijn verkregen tijdens het zoeken. Dit heeft geleid tot de eerste onderzoeksvraag.

Onderzoeksvraag 1: *Hoe kan de kwaliteit van de Monte-Carlo simulaties in MCTS worden verbeterd voor éénspeler domeinen?*

Hoofdstuk 4 beantwoordt deze onderzoeksvraag door Nested Monte-Carlo Tree Search (NMCTS) te introduceren. Het vervangt de Monte-Carlo simulaties door geneste aanroepen van MCTS. Onafhankelijk van de kwaliteit van de Monte-Carlo simulaties op het basis niveau, zorgt dit recursief gebruik van MCTS voor een betere kwaliteit van de simulaties op de hogere niveaus. Dit geldt vooral als er veel zoektijd beschikbaar is. NMCTS kan gezien worden als een generalisatie van de reguliere MCTS en de Nested Monte-Carlo Search (NMCS). Deze aanpak is getest met relatief hoge tijdsinstellingen in SameGame, Clickomania en Bubble Breaker.

Er wordt aangetoond dat NMCTS beter presteert dan multi-start MCTS in alle testdomeinen. Verder wordt NMCTS vergeleken met NMCS. Het blijkt dat NMCTS significant beter presteert dan NMCS in alle testdomeinen. Aangezien MCTS en NMCS specifieke instanties zijn van NMCTS, leidt het correct afstellen van NMCTS altijd tot betere dan wel gelijke prestaties in elk domein.

Upper Confidence bounds applied to Trees ofwel UCT is de meest gebruikte variant van MCTS. In deze variant wordt de selectiviteit van het zoekproces gecontroleerd door één parameter, de zogenaamde exploratiefactor. In domeinen met een lange oplossingslengte of een korte zoektijd kan het zo zijn dat MCTS niet diep genoeg komt. Het resultaat is dat het zoekproces te veel tijd spendeert aan de eerste stappen van de oplossing, maar niet meer genoeg tijd heeft voor de laatste stappen. Dit heeft geleid tot de tweede onderzoeksvraag.

Onderzoeksvraag 2: *Hoe kan de selectiviteit van MCTS in éénspeler domeinen worden verbeterd?*

Hoofstuk 5 beantwoordt deze onderzoeksvraag door Beam Monte-Carlo Tree Search (BMCTS) voor te stellen. Het is een combinatie van MCTS met *beam search*. BMCTS laat een boom groeien waarvan de grootte lineair is aan de zoekdiepte. Dit maakt MCTS meer effectief in domeinen met lange oplossingen of korte zoektijden. Wanneer de knopen op een bepaalde zoekdiepte zijn bezocht door een vooraf ingesteld aantal simulaties, worden ze gesorteerd op basis van hun waarde en worden de beste geselecteerd voor verdere exploratie. De testdomeinen zijn wederom SameGame, Clickomania en Bubble Breaker. Echter de zoektijden zijn deze keer relatief kort.

BMCTS is vergeleken met MCTS waar beiden ofwel één zoekproces gebruiken per positie (single-start), ofwel meerdere zoekprocessen gebruiken per positie (multi-start). Voor het eerste scenario laten de experimenten zien dat BMCTS significant beter presteert dan MCTS in alle domeinen voor een bepaalde reeks van zoektijden. Voor het tweede scenario is multi-start BMCTS sterker dan multi-start MCTS voor een grotere reeks van zoektijden dan in de voorgaande vergelijking.

Deze positieve resultaten hebben aanleiding gegeven tot het idee om BMCTS te combineren met NMCTS. Het heeft geresulteerd tot het zoekalgoritme Nested Beam Monte-Carlo Tree Search (NBMCTS). Experimenten tonen aan dat NBMCTS de beste éénspeler zoektechniek voorgesteld in dit proefschrift is. NBMCTS presteert beter dan wel gelijk aan NMCTS in alle domeinen voor alle zoektijden.

Tijd is typisch gelimiteerd in tweespeler toernooien. In de basis variant is er een vaste hoeveelheid tijd voor de gehele partij. Overschrijding van de tijd betekent een onmiddellijke nederlaag voor de betreffende speler. Echter meer bedenktijd voor een zoektechniek zoals MCTS resulteert in het algemeen in betere zetten. De vraag rijst dan hoe de tijd wijselijk te verdelen over alle zetten in de partij. Dit heeft geleid tot de derde onderzoeksvraag.

Onderzoeksvraag 3: *Hoe kan het tijdmanagement van MCTS in tweespeler domeinen worden verbeterd?*

Hoofstuk 6 beantwoordt deze onderzoeksvraag door het onderzoeken en vergelijken van een verscheidenheid van tijdmanagementstrategieën voor MCTS. De strategieën zijn ofwel nieuw, ofwel beschreven in de literatuur, ofwel verbeterd. De strategieën kunnen worden onderverdeeld in semi-dynamische strategieën, die de hoeveelheid zoektijd bepalen voor elke zoekproces voordat het wordt gestart, en dynamische strategieën die de duur van elke zoekproces beïnvloeden, terwijl die al wordt uitgevoerd. Alle strategieën zijn getest in de domeinen van 13×13 en 19×19 Go. De domein onafhankelijke strategieën zijn tevens onderzocht in Vier op 'n rij, Breakthrough, Othello en Catch the Lion.

Experimentele resultaten tonen aan dat de voorgestelde strategie STOP het meest succesvol is. De strategie is gebaseerd op het idee van het schatten van het resterende aantal zetten in het spel en gebruik te maken van een overeenkomstige fractie van de beschikbare zoektijd. Echter, het MCTS zoekproces wordt gestopt zodra het onwaarschijnlijk is dat de huidige beste zet zal veranderen als men langer zou blijven denken. Er wordt tevens rekening gehouden met deze tijdsbesparingen gedurende de partij. Zoekprocessen vroeg in het spel krijgen relatief meer tijd, zodat niet alleen de latere zoekprocessen profiteren van de opgebouwde tijdwinst. Deze strategie wint ongeveer 60% van de partijen tegen de beste tijdmanagementstrategie tot nu toe,

zowel in 13×13 en 19×19 Go voor verschillende tijdsinstellingen. Bovendien laten experimenten in verschillende spelen zien dat de strategie STOP de sterkste is van alle geteste tijdmanagementstrategieën.

Alle tijdmanagementstrategieën die onder bepaalde voorwaarden de zoektijd verlengen, verschuiven tijd van de latere fasen naar de vroegere fasen van het spel. Alle strategieën die onder bepaalde voorwaarden de zoektijd verkorten verschuiven tijd van de opening naar het eindspel. Verdere analyse toont aan dat deze verschuiving een positief of negatief effect kan hebben. Bijgevolg is een methode ontwikkeld om het effect van deze verschuiving te isoleren en het effect onafhankelijk van de strategie te beoordelen. Indien de verschuiving een negatief effect heeft kan dit worden tegengegaan met een expliciete verschuiving in de tegengestelde richting.

Eén van de kenmerken van MCTS is de Monte-Carlo simulatie, die rekening houdt met de lange termijn. Het geeft daarom in veel domeinen een strategisch voordeel ten opzichte van traditionele minimax zoekmethode. Echter, minimax met $\alpha\beta$ snoeiing beschouwt alle relevante zetten binnen de zoekhorizon. Het kan daardoor een tactisch voordeel hebben ten opzichte van de zeer selectieve MCTS aanpak, die een belangrijke zet kan missen wanneer tactisch spel is vereist. Het is vooral een probleem bij spellen met een groot aantal eindtoestanden verspreid over de gehele zoekruimte, waarbij zwak tactisch spel kan leiden tot een plotseling verlies. Dit heeft geleid tot de vierde onderzoeksvraag.

Onderzoeksvraag 4: *Hoe kan de tactische sterkte van MCTS in tweespeler domeinen worden verbeterd?*

Deze onderzoeksvraag wordt beantwoord door het introduceren en testen van *MCTS-minimax hybriden*, die kleine minimax zoekprocessen in het MCTS raamwerk integreren. Het is daarmee een eerste stap in de richting van het combineren van de sterke punten van MCTS en minimax. Deze hybriden kunnen worden onderverdeeld in aanpakken die domeinkennis vereisen en aanpakken die vrij zijn van domeinkennis.

Hoofdstuk 7 bestudeert drie verschillende hybriden voor de kennisvrije situatie. Er wordt gebruik gemaakt van minimax bij de selectie- / expansiefase (MCTS-MS), de simulatiefase (MCTS-MR), en de terugpropagatiefase van MCTS (MCTS-MB). Testdomeinen zijn Vier op 'n rij, Breakthrough, Othello en Catch the Lion. De voorgestelde variant MCTS-MS presteert aanzienlijk beter dan MCTS in Catch the Lion, Breakthrough en Vier op 'n rij. Hetzelfde geldt voor de voorgestelde MCTS-MB variant in Catch the Lion en Breakthrough, terwijl er geen meetbaar effect is in Vier op 'n rij. De enige uit de literatuur bekende manier om minimax te integreren in MCTS, MCTS-MR, is vrij sterk in Catch the Lion en Vier op 'n rij, maar aanzienlijk zwakker dan de reguliere MCTS in Breakthrough. Dit laatste suggereert dat MCTS-MR minder

robuust is met betrekking tot domeinverschillen zoals de gemiddelde vertakkingsgraad. Ten slotte heeft geen van de MCTS-minimax hybriden een positief effect op Othello door het lage aantal eindtoestanden en ondiepe matsituaties over het gehele zoekgebied. De dichtheid en de moeilijkheidsgraad van de matsituaties zijn goede voorspellers voor de relatieve prestaties van de MCTS-minimax hybriden.

Problematische domeinen voor deze MCTS-minimax hybriden kunnen worden aangepakt met behulp van domeinkennis. Enerzijds, kan domeinkennis in de hybride algoritmen worden geïncorporeerd in de vorm van heuristische evaluatiefuncties. Dit kan minimax veel effectiever maken in zoekruimten met relatief weinig eindtoestanden vóór de eindfase, zoals die van Othello. Anderzijds kan domeinkennis in de vorm van een zettenordeningsfunctie worden geïncorporeerd. Dit kan effectief zijn voor spellen zoals Breakthrough, waar matsituaties relatief vaak voorkomen, maar de vertakkingsfactor te hoog lijkt te zijn voor sommige hybriden zoals MCTS-MR.

Hoofdstuk 8 onderzoekt daarom drie algoritmen voor het geval wanneer domeinkennis aanwezig is. Er wordt gebruik gemaakt van heuristische evaluatiefuncties om de simulatiestrategie te verbeteren (MCTS-IR), om Monte-Carlo simulaties voortijdig te stoppen (MCTS-IC), of om de selectie van zetten in de zoekboom te sturen (MCTS-IP). Voor alle drie de aanpakken wordt de waardering van toestanden middels directe aanroepen van de evaluatiefunctie (MCTS-IR-E, MCTS-IC-E en MCTS-IP-E) vergeleken met de waardering middels kleine minimax zoekprocessen gebruikmakend van dezelfde evaluatiefunctie (MCTS-IR-M, MCTS-IC-M, en MCTS-IP-M). Deze laatste drie zijn MCTS-minimax hybriden. De integratie van minimax is alleen door MCTS-IR gebruikt in deze vorm. Testdomeinen zijn Breakthrough, Othello en Catch the Lion.

De hybriden worden gecombineerd met zettenordening en k-best snoeiing om zodoende om te gaan met hogere vertakkingsgraden resulterend in de verbeterde hybride spelers MCTS-IR-Mk, MCTS-IC-Mk en MCTS-IP-Mk. Resultaten tonen aan dat met deze twee relatief eenvoudige $\alpha\beta$ verbeteringen, MCTS-IP-Mk de sterkste MCTS minimax-hybride in alle drie de geteste domeinen is. Omdat MCTS-IP-Mk minimax minder vaak aanroept, presteert het beter dan de andere hybriden bij lagere tijdsinstellingen waar de prestatie van MCTS het meest gevoelig is voor een reductie van het aantal simulaties per seconde. MCTS-IP-Mk werkt ook goed op een groter Breakthrough bord, waaruit de geschiktheid van de techniek voor domeinen met hogere vertakkingsgraden blijkt. Bovendien overtreft de best presterende hybride een $\alpha\beta$ implementatie in Breakthrough, waaruit blijkt dat althans voor dit domein MCTS en minimax kunnen worden gecombineerd in een algoritme dat sterker is dan ieder afzonderlijk.

Hoofstuk 9 sluit het proefschrift af. Het antwoord op de probleemstelling is

tweeledig. Ten eerste kan de prestatie van MCTS in éénspeler domeinen op twee manieren worden verbeterd — door het nesten van MCTS zoekprocessen (NMCTS), en het combineren van MCTS met beam search (BMCTS). De eerste methode is vooral interessant voor langere zoektijden, en de tweede methode meer geschikt voor kortere zoektijden. Een combinatie van NMCTS en BMCTS kan ook effectief zijn.

Ten tweede kan de prestatie van MCTS tevens op twee manieren worden verbeterd — door de beschikbare tijd voor de hele partij op een slimmere manier te gebruiken (tijdmanagement), en door het integreren van kleine minimax zoekprocessen in MCTS (MCTS-minimax hybriden). De eerste aanpak is relevant voor scenario's zoals toernooispel waar de hoeveelheid tijd per partij gelimiteerd is. De tweede aanpak verbetert de prestatie van MCTS specifiek voor tactische domeinen.

Er zijn vier richtingen voor vervolgonderzoek. Deze zijn (1) het toepassen van NMCTS in non-deterministische en gedeeltelijk waarneembare domeinen, (2) het verbeteren van BMCTS om zo in de limiet optimaal gedrag te garanderen, (3) het uittesten van verscheidene tijdmanagementstrategieën, en (4) het onderzoeken van zwakkere prestaties met diepere ingebedde zoekprocessen in MCTS-minimax hybriden, alsmede het bestuderen van welke eigenschappen van de testdomeinen invloed hebben op de prestaties van deze hybriden.