# The Intelligent, the Artificial, and the Random

Maastricht University

Inaugural speech

**dr. Mark H.M. Winands**

Faculty of Science and Engineering

# The Intelligent, The Artificial, and the Random

# The Intelligent, the Artificial, and the Random

Inaugural Lecture
Friday, December 16, 2022

*Mark H.M. Winands*
*Department of Advanced Computing Sciences, Maastricht University*

Highly Esteemed Audience!

Today I will deliver my inaugural lecture as professor in Machine Reasoning, at the Faculty of Science and Engineering of Maastricht University. As you might have noticed, my chair will be located at the Department of Advanced Computing Sciences, which performs research in the domains of Data Science, Computer Science, Applied Mathematics, Robotics, and Artificial Intelligence. The latter is the field I am working in and this will be the focus of this lecture. A disclaimer before I start, this lecture has been mainly composed by myself, and not by an AI bot such as ChatGPT [1].

**What's AI?**
Artificial Intelligence (AI) has become a prominent and growing research field in the last couple of years. It concerns building intelligent entities that act effectively and safely in a wide range of new situations completely on their own. We have seen many applications recently ranging from fields as diverse as health care, finance, law, insurance, communication, education, energy, transportation, manufacturing, farming and games.

There are several views on how to create these intelligent systems. Russell and Norvig [2] define them on two dimensions, i.e., thinking vs acting, which both of them can be human-like or rational. This leads to four views on AI, which are all valuable.

The first one is **Thinking Humanly.** The idea is that if we can model the thought process of humans, then we can implement this model in our computer systems, which then will behave intelligently just as we humans do. The catch here is that we need to have a thorough understanding how exactly we think, or how exactly our brain works. These questions belong to different disciplines, and are studied in the fields of Cognitive Science and Neuroscience. Still, this view on AI has inspired several successful methods such as Artificial Neural Networks.

The second view is **Thinking Rationally**. The idea is that if we can make correct inferences according to the laws of logic, intelligent behaviour will emerge. The catch here is that we should be able to represent the world in such a way that it fits a logic notation to perform these deliberations. This requires knowledge of the world that is 100% certain, which is hard to achieve. Next, it assumes that we have sufficient time to perform these careful deliberations in order to proof the optimal course of action. This is not always the case.

Instead of focussing on whether and how machines can or should think, the question should be: can machines behave intelligently? If a system is **Acting Humanly** [3], it is pretty smart, as we humans tend to act smart ourselves. Acting as a human is especially valuable if we would like to talk, to chat with a computer system. When we would like to get a proper understandable explanation when the computer makes a decision, instead of getting the answer: "Computer says no". When we design androids that have to act as a companion, we would like that they behave like us. But acting as a human also means making suboptimal decisions, which might increase the fun in video games and other entertainment, but it is not recommendable in other situations when safety or sustainability are concerned. In the end, for many applications, it does not matter whether the behaviour is humanlike, as long as the job is being done. Airplanes do not fly as birds, but they do fly.

And this leads to the fourth and pragmatic view of **Acting Rationally**. We do not care how the system thinks, and whether it behaves as a human, as long if the system is able to make the right decision according to a metric that we humans have set. This also includes societal awareness. Acting rationally does not mean clairvoyance, or finding the perfect solution. The pragmatic aim of AI is to create autonomous computer systems that are able to make the best decision given the amount of information available, given the limited amount of time, given the limited resources. What is the optimal decision in hindsight does not count. In hindsight, everything is easy.

**AI Subfields**

There are several approaches to implement these views on AI. The 2018 Dutch AI Manifesto [4] recognizes seven main subfields, whereas the Sector Image Computer Science [5] is more coarse grained by limiting it to three. And I will take the latter one as the starting point. The subfield of AI that is getting the most attention nowadays is Machine Learning, which allows systems to learn on their own by recognizing patterns in huge datasets and making decisions based on similar situations. It has had many noticeable successes in areas such as healthcare, financial trading, fraud detection, marketing, and natural language processing.

However, another important component of AI is Machine Reasoning, which is the subfield I am working in. We humans have a thing called gut feeling. It is usually based on past experience. But before making any rash decisions based on some gut feeling, it is wise to think the situation through. Is the situation really as similar as you think it is? Responsible decision-making requires considering the consequences of an action. That is what Machine Reasoning is supposed to do for AI. Machine reasoning takes into account the potential follow-up actions based on the available information, by gathering additional information in order to reduce the uncertainty of an outcome, by looking at viable alternatives and discarding the irrelevant ones. Machine reasoning involves amongst others, methods to search efficiently in general solution spaces. Machine reasoning should also be socially aware, taking into account the involved people and their preferences, which includes explaining the decisions to a human user.

The recent progress in machine learning challenges and stimulates the advancement of the field of machine reasoning, because there are many things AI cannot do very well yet. Machine-learning methods do not understand the world at the same level as we humans do; these methods have difficulties to generalize to new situations. Humans can generalize from a single example by deriving cause and effect, where machine learning struggles when the amount of data is limited.

While machine learning is rather good at pattern recognition, it is not so strong at reasoning on a more abstract and general level. The next step in AI is the ability to apply learned knowledge to new situations. Combining machine-reasoning and machine-learning methods seems to be a very promising direction. Further integration of these methods will increase the quality of automated decision making, regardless whether this is used to make the final decision autonomously, or to provide a human operator with a range of sensible options.

## Game-Playing: The F1 Racing of AI

To test how well AI methods perform, we need to have a clean and safe test domain. Since the dawn of AI, games have been used for this purpose. The advantage of games is that they are widely available and that their rules are well-defined, making them a great benchmark for comparing AI to human intelligence. The question of whether computers can beat humans in games like Chess, Go, or Stratego has been one of the first tasks that an AI system should be able to do. Even if machines surpass human grandmasters, games continue to be a great benchmark for comparing different AI approaches to each other. In games such as Chess or Go the number of possible positions is huge, making it infeasible to prove who wins the game. AI systems have to find the best line of play given the limited amount of time they have. Moreover, games offer all kinds of other challenges, as one has to deal with other players, whose strategies are not known. In some of them it is not possible to observe the full state, or the outcomes of an action are uncertain, or even the environment can change at any time.

Game playing is to AI as Formula 1 racing is to the car industry [2]. The competitive pressure has led to new methods and insights in the fields of machine reasoning as well as machine learning. Big Tech companies such as IBM or Google have been using game playing to test and showcase their advances in AI. Therefore, for my own research I am using games as well.

## Heuristic Search

In game playing, the task is to find the best response to the opponent's moves, provided that the opponent does the same. The reasoning method for this is search (see Fig. 1). Here we look at a two-player turn-taking game, which can be represented by a tree, where nodes indicate states, and edges indicate moves. The initial state is the root of tree. Based on the moves you can make, the successor states are generated, in which it is subsequently the turn of the opponent. Because it is impossible to generate the complete game tree, the tree is being truncated to a limited search depth. At the so-called leaf nodes, a heuristic evaluation function is being used that gives an estimate who is ahead. Typically, an evaluation function is a linear weighted sum of features, such as counting the number of different pieces. In the past, these features were usually based on human knowledge and hand-coded by a programmer. Instead of handcrafting one yourself, already in the 1990s Tesauro [6] showed that neural networks, though computationally more intensive, can learn to estimate the value of a state by means of self-play.

Now, after assigning scores to the leaf nodes, the values are being backpropagated by the so-called Minimax principle [7]. At nodes where it is the opponent's turn, minimax selects the move that minimizes the score. The idea is that the worse it is for you, the better it is for the opponent. In nodes where it is your turn, minimax chooses the move that maximizes the result. Based on this analysis, the search chooses the move in the end, and executes it in the game. Playing is nothing more than solving a sequence of these truncated game trees.

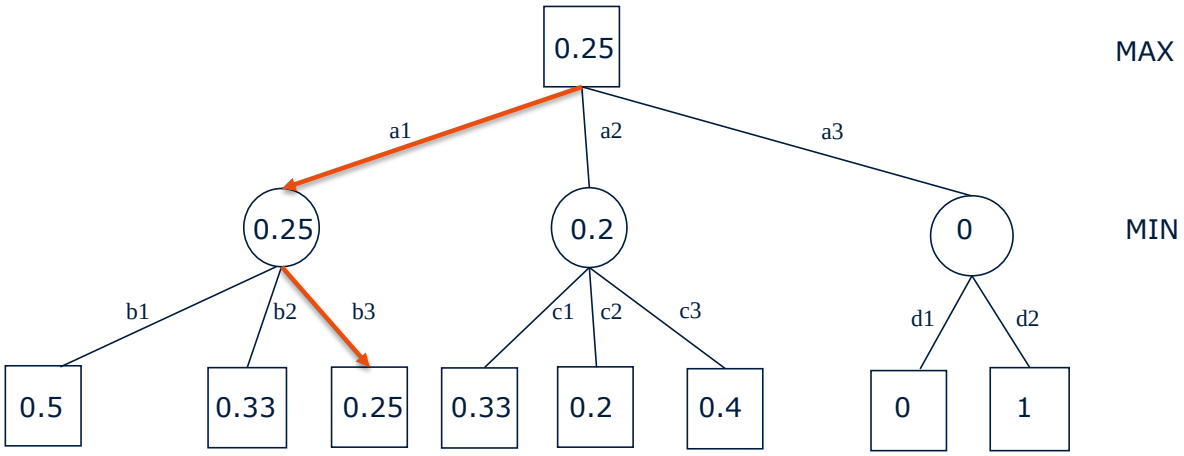


**Figure 1: Minimax Search**

For decades, this kind of heuristic search has been the standard approach used by programs for playing board games such as Chess, Checkers, and many others. Over the years, researchers have proposed many search enhancements for this framework to further enhance its effectiveness. The victories of computer programs against the Human World Champion in Checkers [8] and Chess [9] in the 1990s further sealed the reputation of heuristic search.

## Problem: No Proper Heuristic Evaluation Function

However, this traditional approach has not been successful in some other games. The prime example is the game of Go. This game is full of relatively vague concepts such as life-and-death, territory, influence, and patterns. This implicit human knowledge is hard to turn explicit in a heuristic evaluation function. Without such a function, heuristic search will simply fail. While in many board games the level of play was at least at grand-master level in the 2000s, in Go the machines did not reach even a proper amateur level. Go was the tournament where spectators came to have a good laugh.

Even if we do not care about Go, the general questions we are asking ourselves are:

- What do you do if you lack the knowledge to build an evaluation function?
- What do you do if you do not have the data to generate one automatically?
- What do you do if you do not have the time to generate or build one?
- What do you do if after all your efforts, the evaluation function still does not work?

**Monte-Carlo Evaluations**

What do you do if you have no clue? You are going to gamble, you go to Monte-Carlo. Now, at a leaf node, a number of games are simulated by selecting (semi-) random moves until the end, where they are subsequently scored. The results of these samples are being recorded in order to calculate an average (also called the winning ratio). At the first leaf node of Fig. 2, the resulting score is 75%. In this way, the so-called Monte-Carlo evaluation assesses the merits of a leaf node. The idea is that if there are many lines leading to a win, then there is a good chance that a forced win can be secured. If there are not so many winning scores, then probably the opponent has the upper hand. The same procedure applies for all other leaf nodes, and by applying minimax the best line of play can be chosen.

Now you would wonder why wasn't this done before as Monte-Carlo sampling dates from the 1940s. Well, in the early 1990s, the first attempts were made to use Monte-Carlo evaluations in Chess [10] and Go [11]. Success was rather limited, as the hardware in those days could not support many simulations, resulting in a flat Monte-Carlo search not able to look more than one step ahead.

It was no surprise that sooner or later the idea would pop up again at the moment hardware would have advanced such that more simulations per second could be executed. It was Bouzy and Helmstetter [12] who tested this approach at the Computer Olympiad of 2003. Their program performed reasonable, but it was not the breakthrough as they had hoped.

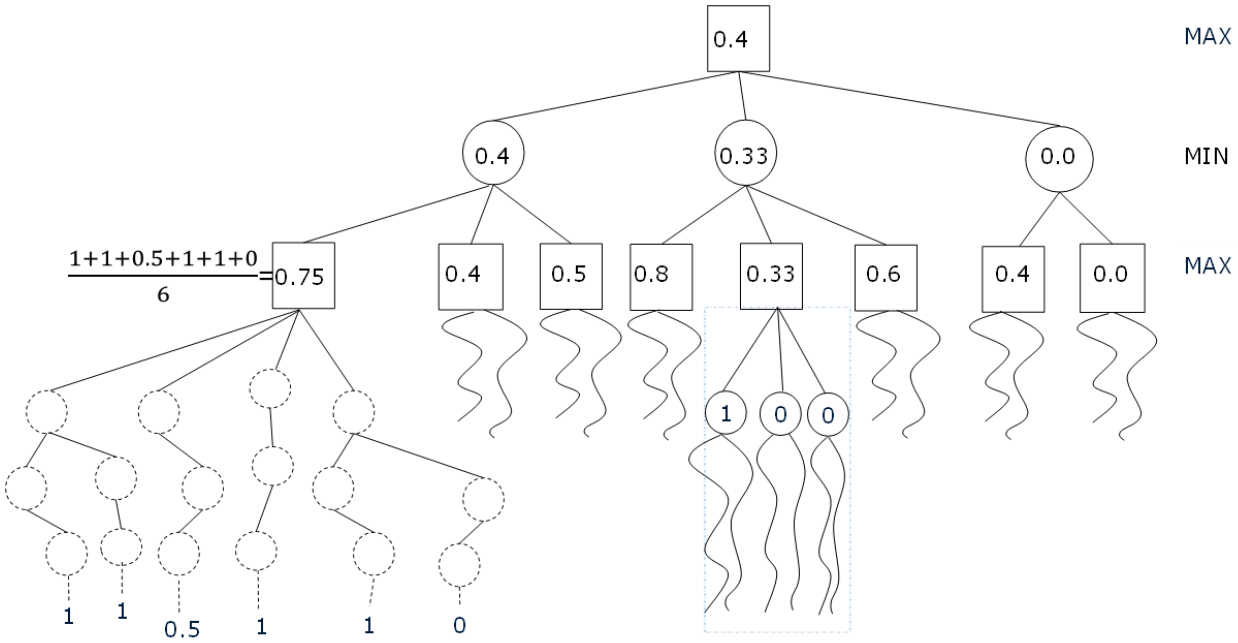


**Figure 2: Monte-Carlo Search**

Why does this Monte-Carlo search not really work? At the moment tactics starts playing a role in a game, such an approach would easily start misinterpreting positions. Let us have a look in Fig. 2 at the leaf node where we scored 33%. If we would zoom in here, we would see that for the first move we are scoring only wins, whereas for the other two, we are only scoring losses. Under the assumption of rational play, we would simply avoid these bad moves, leading to an overall score of 100%, which would eventually lead to a different decision. How to solve this? You could say "let us search one step further", but this would lead to an increase of leaf nodes and therefore an increase in the number of simulations, which costs time. Time that we do not have. You could mitigate this by decreasing the number of samples per state evaluation, but the quality of the assessment of a game state would then deteriorate.

## Monte-Carlo Tree Search

How to solve this trade-off? On the one hand, we would like see as many states as possible. On the other hand, we would like to sample a state as much as possible. How to integrate Monte-Carlo with search?

The answer is simple. Instead of generating a tree first and then using Monte-Carlo simulations to assess the leaf nodes, flip it. Start with the simulations, and build the tree subsequently.

Let random be your guidance!

This concept is called Monte-Carlo Tree Search (aka MCTS) [13]. From the root you start with your simulation, the first state that has not been seen before is added to the tree. Next, the simulation is played out, scored and recorded in the tree. In the example, we see that a tree is gradually generated in memory, and steadily becomes better at estimating the values of the nodes. This enables that 1) from the root a promising line is selected using the previously recorded information, 2) which leads to a node expansion, 3) after which the simulation is played out, and 4) statistics are updated accordingly. This cycle is repeated as long as there is time left [14] (see Fig. 3).

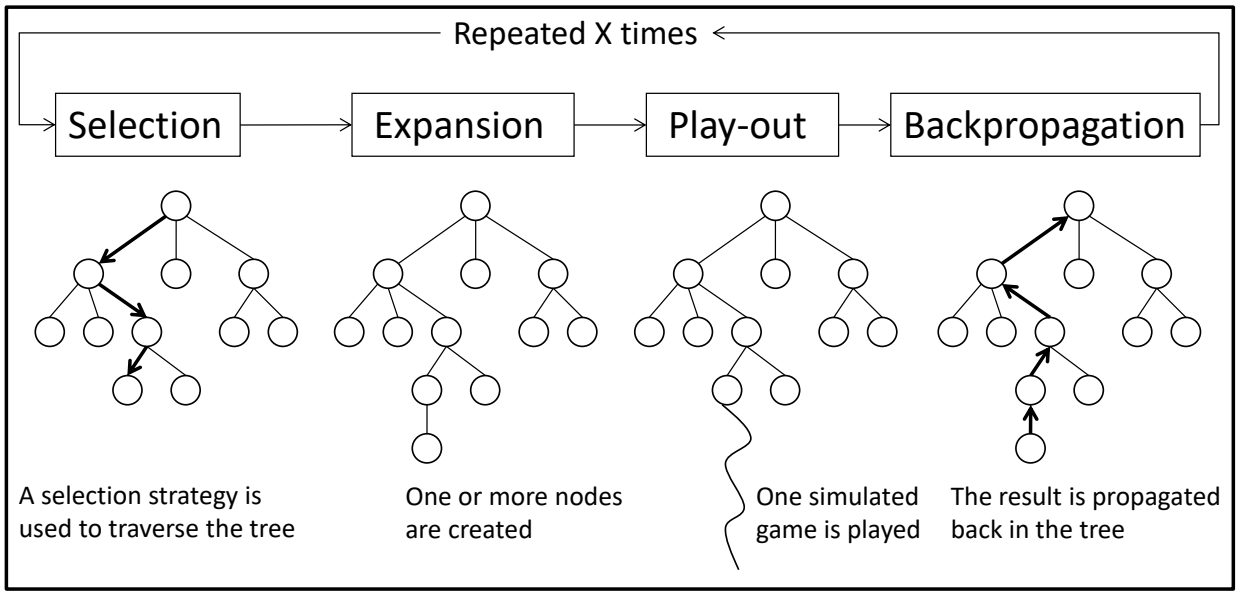


**Figure 3: MCTS Scheme adapted from [14] [15]**

In 2006, Coulom demonstrated the strength of using this MCTS approach by winning the 9×9 Go tournament at the 12th Computer Olympiad [16], causing quite a stir in the game AI community.

Still MCTS was not without its problems. If we have a look at Fig. 4, we see that if we select the moves in a greedy way, by always going for the one that has scored best so far, we will not have a second look at the other root moves. This is the trade-off of exploitation and exploration. You would like to spend most of your time in promising lines, but still to put some effort in the seemingly less promising ones in order to prevent missing a strong one.

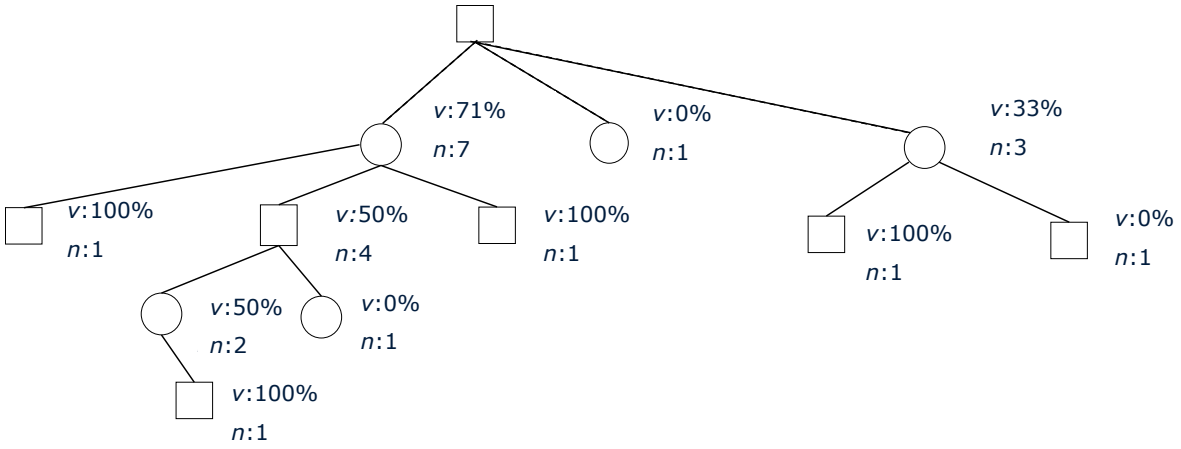


**Figure 4: Greedy MCTS**

## Multi-Armed Bandit

Search is like investing in stocks, you put your money in the potential winners, though investing in some risky ones might pay off. A loser can actually be a winner. Investing in stocks is in a certain way gambling. Where do people know how to gamble? In Monte-Carlo, and what do they have there? Casinos. What do you find in casinos? Slot-machines, also called the one-armed bandit. The goal is to find among the slot-machines, the one-armed bandit, that is giving you the best pay-off as soon as possible. Because every time you pull an arm, it will cost you. This is called the multi-armed bandit problem, and essentially the same problem we are facing in MCTS.

This problem has been well studied in the literature. Kocsis and Szepesvári [17] proposed to use the multi-armed bandit algorithm **U**pper **C**onfidence **B**ounds [18] applied to **T**rees (aka UCT) for selecting moves in the tree.

$$k \in \arg\max_{i \in I} (v_i + C\sqrt{\frac{\ln n_p}{n_i}})$$

The first part of their formula focusses on exploitation by taking the score achieved so far into account. The second part deals with exploration, by taking into account how many times an arm, a move, is being chosen relative to the total number of samples. The parameter *C* balances the exploitation and exploration tradeoff. This clean mechanism has become the way to go in MCTS.

**UM Add Ons**

At UM we also went for MCTS, and focussed the last 15 years to push this framework even further. In these years, I have supervised several bachelor, master, PhD students, and Postdocs who introduced several enhancements to strengthen MCTS. Here I will briefly mention seven of them.

Starting in the tree, the first issue we tackled is that a multi-armed bandit selection mechanism such as UCT has no clue when only a few simulations have been conducted at a node. To resolve this, we introduced Progressive Bias [14], which guides the selection mechanism by adding a (small) bonus $H_i$ based on some offline generated domain knowledge. In case such knowledge is not available, we have suggested Progressive History [19] that takes into account how well moves performed elsewhere in the tree.

These techniques do no solve the problem if we have hundreds of moves in a state. In such a case, it takes a while before every move is sampled at least once, which would mean that MCTS is not able to have a deep look ahead. We introduced Progressive Widening [14] to deal with domains with a huge branching factor. This technique initially limits the moves that MCTS can consider but gradually increases the number of moves as the node is visited more, and therefore seems to be promising.

In general, it holds the more simulations you can run, the better Monte-Carlo Search performs. Therefore, you should squeeze as much juice as the computer hardware can provide you. As modern hardware contains several cores nowadays, search algorithms should be parallelized such that they can make maximum use of the available resources. Here, we investigated how so-called Tree Parallelization [20] can explore different parts of the search space simultaneously in an effective way. It turns out that MCTS scales very well compared to other search algorithms [21]. It is no problem if sometimes a result is recorded incorrectly due to racing conditions, as MCTS is handling noise rather well anyways.

Next, we looked into how we could get more out of the simulations. The default is to select the moves randomly, but if more sensible moves are played, the predictive power of a simulation increases. How do you know what is sensible? Use Monte-Carlo. Here we proposed the N-Gram Selection Technique [22] [23] to bias moves on how successful a move pattern was during the other simulations. The tricky part is that if you make such a strategy too "smart" by removing the random element, the performance of MCTS will go down. This can be resolved by using an epsilon-greedy approach. It means sometimes selecting the move randomly, sometimes selecting the move that has been the most successful so far.

In addition, it is not necessary to play out the simulation until the very end. Although MCTS does not need a heuristic evaluation function, if you have one, you can cut the simulation short. If you know who is winning, you can stop the simulation, and give the win to the one that seems much ahead [24]. The better the evaluation function understands the position, the earlier you can stop the simulation.

After performing the search, the MCTS engine commits to a move in the game. In the next turn or time-step, MCTS has to search again for a follow-up move. Instead of starting from scratch relevant information of the search can be reused [25] [26] [27]. However, this is not trivial as in dynamic environments the world changes all the time, the information stored in the tree is no longer in line with the current state of the game. Still, it turns out to be useful to reuse this information to a certain extent, as MCTS is built to deal with noisy information.

Many more enhancements for MCTS have been proposed successfully by us and others. The few I mentioned here already create a Christmas tree. These add-ons are controlled by multiple parameter settings that require extensive and time-consuming offline tuning, which is not always feasible. Moreover, as the strategies of the opponents are unknown or even the domain is unknown, the search engine has to be flexible. We proposed Self-Adaptive MCTS [28] that uses a multi-armed bandit algorithm to find the right search control settings online. We learn how to search when we are searching. Machine reasoning in general has to be adaptive, as circumstances in which decisions must be made may change rapidly [4].

## Machine Reasoning and Machine Learning

Simultaneously with the developments in MCTS, deep neural networks emerged as a powerful tool to perform machine learning. Now the question arises, can you combine this machine-learning technique with this reasoning technique? The Google DeepMind team took up this challenge in order to defeat the strongest human Go player. The procedure is a follows. A deep neural network learns from human Go game records in order to predict expert moves, and to assess the value of a state. This neural network is then used to bias the selection mechanism, and to improve the quality of the simulations. This approach is called AlphaGo [29] and defeated the top Go player, Lee Sedol, in 2016, 10 years after the dawn of MCTS.

Could this also work if you do not have vast amount of data in the form of human game records? If you have zero knowledge, zero information, zero data. The answer is: use MCTS to generate the data to train the neural networks [30]. By letting MCTS play against itself, game records are generated representing a decent level of play. As the neural network learns from these games, and is integrated in MCTS, the level of play of MCTS will increase, which means better games are played, better data is generated, the neural network learns more, and MCTS becomes even stronger. In the end, it leads to a neural network that already plays at grandmaster level of around 3000 Elo points. Nevertheless, putting MCTS on top of this leads to an additional increase of 2000 Elo points. MCTS corrects the mistakes of the neural network. This approach called AlphaGo Zero outperforms the original one as well. Besides Go, this procedure has also been used in Chess and Japanese Chess, achieving a superhuman level [31]. This is a prime example of the breakthrough that can be achieved when a machine-reasoning technique such as MCTS and a machine-learning technique such as deep neural networks are combined together. The MCTS taught the neural networks a lesson. AI that teaches another AI.

## MCTS Game Domains

So far, we saw applications in two-player turn-taking games such as Chess and Go. At UM we investigated applications of MCTS in a wide variety of game domains. A natural first step is to look at puzzle domains [32], aka single-agent search. There we showed that MCTS outperforms classic search algorithms as A* in complex puzzles such as SameGame.

On the other side of the spectrum, there are games with three or more players, so-called multi-player games. These domains are challenging as the optimal strategy depends on the preferences of the other players. In many of these domains, MCTS outperforms other multi-player search algorithms as it plays a mixed strategy [33].

Also, MCTS is successful in domains with uncertainty due to partial observability, or chance. A successful application has been in the domain of Scotland Yard, where our MCTS engine outperformed the commercial Nintendo DS bot [34]. As a testimony of its strength, this MCTS engine has been used as benchmark to test Google DeepMind's Player of Games bot [35].

The biggest challenge for search algorithms are real-time games. These are noisy environments as the world changes while you are pondering your next action. Classic video games such as Super Mario or Pacman are great benchmarks as one has to make a decision every 40ms. Here MCTS is also a feasible technique, as shown by our MCTS Pac-Man agent [25], which achieved the first place out of 36 competitors in the IEEE 2012 Pac-Man Versus Ghost Competition.

**General Game Playing**

AI in the end should be able to perform different tasks in different environments, should be able to manage itself and possess the competence to perform any new task that it has never performed before [36], and this includes playing multiple games. Since 2005, several general-game playing competitions have been organized where bots have to play several unknown games on the spot without any human intervention. This means that programs cannot rely on game-specific and prior knowledge and have to adapt to each new game, and therefore have to adjust their planning. MCTS engines have been performing relatively well in these competitions, even in the general video game AI competition, where bots have to play Arcade games. Though the amount of thinking time is limited, our MCTS bot [27] won this competition in 2016.

**MCTS Real-World**

MCTS can also be used beyond games. For example, together with Aucos, we have developed an MCTS planner that optimizes the production times for surface plating jobs [37] (for instance, racks and baskets for kitchen cabinets). Such a Job-Scheduling Problem may involve 6,000 jobs for a production line with 110 machines. The goal is to plan thousands of jobs such that the production time is optimized. This is a challenging problem, not only because of the sheer number of possibilities, but also because tasks can change every day, re-scheduling has to be fast.

This MCTS planner uses several ideas originating from Game AI, such as Single-player MCTS (SameGame), Progressive History/N-Grams (Chinese Checkers, Havannah), Tree Reuse (Ms. PacMan), Tree Parallelization (Go) and combining Neural Networks with MCTS (Go). This is not an academic exercise, the resulting planning system has been implemented and been operating in a production plant's controller since December 2018.

There have been many other MCTS applications in a wide range of domains. For example, a potential application that we investigated together with RWTH Aachen is looking for suitable combinations of bearing elements in the domain of structural engineering [38]. The European Space Agency showed that MCTS is an order in magnitude more efficient on interplanetary trajectory planning than other methods [39]. Other applications are planning chemical syntheses [40], patient admissions scheduling [41] [42], real-time path planning for unmanned aerial vehicles (UAVs) [43], autonomous robot exploration gathering [44], energy management [45] and parking utilization [46] amongst others. Claiming that Game AI is nonsense, is claiming that AI is nonsense.

**Artificial General Intelligence**

I have a confession to make, I misled you in giving you the impression that we have a general-purpose planning algorithm that can play any game, schedule any job, plan any task, and outperforms humans and other software. There is no free lunch. Generic MCTS engines, even with their domain-independent add-ons and self-adaptiveness, might perform well in some domains, but are subpar in others. Many of the MCTS engines I mentioned are specifically built for their specific problem and exploit the underlying domain-specific structure. Moreover, to outperform other search techniques, many of these successful MCTS engines are hybrids, integrating components from other search approaches in order to perform as strong as possible. MCTS is simply a building block. In many domains, MCTS is a contender, but not a winner. Classic search algorithms such as A* or alpha-beta search are still going strong in certain domains, other Monte-Carlo search approaches are in the lead elsewhere such as Nested Monte-Carlo Search.

But then again the dream is to have Artificial General Intelligence that is able to do any task, which can build a car, act as a judge, write a poem in Italian, tell a story, farm insects, and is able to plan for any domain. Should we accept that this broad AI is always outperformed by narrow AI that has been designed for a specific task, for a specific search problem? Or should it do what we do if we face a task that is too intellectually challenging: building a narrow AI to deal with the problem? AI that builds AI, AI that creates search engines. Designing search engines consists mostly of combining existing concepts and tuning them, which is a mechanical process and can be automated.

As a first step, we propose the Adaptive General Search Framework [47], which would consist of a portfolio of existing search techniques. They are subsequently decomposed into their building blocks, which then are encoded in some formal description. This enables the AI to investigate all kinds of combinations to generate a (new) search engine for the problem at hand. Of course, it would not be able to think out of the box, fundamentally new concepts would still be provided by AI researchers for the time being. This may not be a free lunch, but it might be a cheap one.

**Explainable Search**

There is one question that has not been answered, which has been bothering me for the last 20 years.

Why did the search make this move? What is happening?

Answering these questions is already hard for a deterministic search engine that uses a classic human-designed evaluation function. It is even a bigger challenge for a modern method like MCTS. Simply comparing winning ratios between moves will not tell you much. Looking at their main lines will reveal that the scores are all over the place, as the deeper you go down the tree, the less the states have been sampled. These scores are coming from Monte-Carlo simulations where moves have been chosen by some epsilon-greedy strategy, and these simulations have been cut short by a state evaluator based on a deep neural network. And we got there in the first place by a biased UCT mechanism, that has to take into account parallelization as well. Also, the parameters controlling the search could have been tuned automatically online by some self-adaptive mechanism. Even the search itself could have been designed by another AI system. In the future, MCTS could even run on a quantum computer, which is a challenge in itself.

Is someone still following this? I am not. Even if AI behaves in a rational way, it has to be humanlike in order to explain what is going on, in order to be transparent and trustworthy, which contributes in accepting its solutions. These aspects are studied in the field of Explainable AI, which usually focusses on explaining the decisions of machine-learning methods, as some of the successful ones are black boxes. The outcome of a search is not trivial to explain as well, as it is a complex tree consisting of many promising lines. The subfield of Explainable Search aims to overcome the gap between the search result and human understanding. The human user would like to get answers to the following questions [48]: Why? What exactly could happen next? Which possible outcomes were explored? How were they interpreted, compared, and selected from?

At UM [49] we are also investigating Explainable Search, and in our view, it consists of two parts: 1) a mining part that aims to extract the relevant information from a search tree and 2) a storytelling part that aims to create an explanation (either story or dialogue) based on the information extracted. As the search can be executed in an online setting, the explanation should be provided together with the decision without additional delay.

The biggest challenge here is answering the question: Why not? As you cannot analyse everything in the limited time one has, how can you explain something that you could not investigate properly? This is the trade-off between explaining and doing the right thing. If we aim for the best result, we have to compromise on the explanation, and the other way around. What will this trade-off look like? How can we extract relevant information from the search such that we can generate a proper explanation? These topics will be addressed during my farewell speech in 2045.

## Reflections

Now I would like to share my last thoughts. In the past, I was the programme director of the nowadays-called bachelor's in Data Science & Artificial Intelligence. In this programme, we have so-called Project Centred Learning (PCL) where students work together on a challenging task. In the Netherlands, every 6 years programmes are being assessed for their re-accreditation. Here we received the following praise from the assessment panel in 2020 [50] that I would like to share.

*"The programme explicitly aims for its students to acquire an international academic orientation. It combines the PCL approach with an international classroom, in which different backgrounds are deliberately mixed in project groups."*

What were these deliberations? Taking into account that time and resources are limited at a Dutch university, which policies were in place so that such an international classroom could be achieved? What do you do if you do not have the data? What do you do if you do not have the time? What do you do if you do not have a clue? You go to Monte-Carlo, you randomize, and that was what we did.

Dare to randomize, it seems artificial, but it will make you more intelligent.

## Acknowledgements

Finally, I would like to share my words of sincere thanks. First of all, I would like to thank the Faculty of Science and Engineering for their support and confidence in appointing me as a professor in Machine Reasoning. Next, I would like to express my gratitude to all my colleagues and former colleagues of the Department of Advanced Computing Sciences (formerly known as the Department of Data Science and Knowledge Engineering, formerly known as the Department of Knowledge Engineering, formerly known as the Maastricht ICT Competence Centre, which is a descendant of the Faculteit der Algemene Wetenschappen) for their support over all these years. Especially, I would like to thank everybody who helped me with organizing this event, and providing me with material for this inaugural lecture. I have been honoured to work with so many excellent researchers at this university and elsewhere, providing me with new ideas and insights. It has been a privilege to supervise so many outstanding bachelor's, master's, and PhD students as well as postdocs who explored all kinds of new research directions, which I was not able to do or did not dare to do.

I would like to thank the members of the structure committee and advisory board, and my PhD supervision team for being here. I would like to thank the audience for attending my inaugural lecture and enduring my speech.

Tenslotte wil ik mijn ouders bedanken voor al hun steun, zonder hun had ik hier niet gestaan.

Dixi, Ik heb gezegd, I have said.

## References


[1] OpenAI, “ChatGPT: Optimizing Language Models for Dialogue,” OpenAI, 2022. [Online]. Available: https://openai.com/blog/chatgpt/.

[2] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall: Englewood Cliffs, N.J., 1995.

[3] A. M. Turing, “Computing Machinery and Intelligence,” *Mind,* vol. 59, no. 236, p. 433–460, 1950.

[4] SIGAI, “Dutch Artificial Intelligence Manifesto,” ICT Research Platform Netherlands, 2018.

[5] IPN, “Het fundament onder de digitale samenleving: Beeld van de Nederlandse Informatica,” 2022.

[6] G. Tesauro, “Temporal Difference Learning and TD-Gammon,” *Communications of the ACM,* vol. 38, no. 3, pp. 58-68, 1995.

[7] J. Von Neumann, “Zur Theorie der Gesellschaftsspiele,” *Mathematische Annalen,* vol. 100, p. 295–320 , 1928.

[8] J. Schaeffer, One Jump Ahead: Challenging Human Supremacy in Checkers, Springer, 1997.

[9] M. Campbell, A. J. Hoane Jr and F. Hsu, “Deep Blue,” *Artificial Intelligence,* vol. 134, no. 1-2, p. 57–83, 2002.

[10] B. Abramson, “Expected-Outcome: A General Model of Static Evaluation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 12, no. 2, pp. 182-193, 1990.

[11] B. Brügmann, “Monte Carlo Go,” Max-Planck-Institute of Physics, München, Germany, 1993.

[12] B. Bouzy and B. Helmstetter, “Monte-Carlo Go Developments,” in *Advances in Computer Games 10: Many Games, Many Challenges*, H. J. van den Herik, H. Iida and E. A. Heinz, Eds., 2004, pp. 159-174.

[13] R. Coulom, “Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search,” in *Proceedings of the 5th International Conference on Computer and Games*, H. J. van den Herik, P. Ciancarini and H. H. L. M. Donkers, Eds., Springer, 2007, pp. 72-83.

[14] G. M. J.-B. Chaslot, M. H. M. Winands, H. J. van den Herik, J. W. H. M. Uiterwijk and B. Bouzy, “Progressive Strategies for Monte-Carlo Tree Search,” *New Mathematics and Natural Computation,* vol. 4, no. 3, pp. 343-357, 2008.

[15] M. H. M. Winands, “Monte-Carlo Tree Search in Board Games,” in *Handbook of Digital Games and Entertainment Technologies*, R. Nakatsu, M. Rauterberg and P. Ciancarini, Eds., Singapore, Springer, 2017, pp. 47-76.

[16] R. Coulom and K. Chen, “Crazy Stone wins 9×9 Go Tournament,” *ICGA Journal,* vol. 29, no. 2, pp. 96-97, 2006.

[17] L. Kocsis and C. Szepesvári, “Bandit Based Monte-Carlo Planning,” in *ECML-06*, J. Fürnkranz, T. Scheffer and M. Spiliopoulou, Eds., Springer, 2006, pp. 282-293.

[18] P. Auer, N. Cesa-Bianchi and P. Fischer, “Finite-Time Analysis of the Multi-Armed Bandit Problem,” *Machine Learning,* vol. 47, no. 2–3, p. 235–256, 2002.

[19] J. A. M. Nijssen and M. H. M. Winands, “Enhancements for Multi-Player Monte-Carlo Tree Search,” in *Computers and Games (CG 2010)*, H. J. van den Herik, H. Iida and A. Plaat, Eds., Springer, 2011, p. 238–249.

[20] G. M. J. B. Chaslot, M. H. M. Winands and H. J. van den Herik, “Parallel Monte-Carlo Tree Search,” in *Computers and Games (CG 2008)*, H. J. van den Herik., X. Xu, Z. Ma and M. H. M. Winands, Eds., Springer, 2008, pp. 60-71.

[21] R. B. Segal, “On the Scalability of Parallel UCT,” in *Computers and Games (CG 2010)*, H. J. van den Herik, H. Iida and A. Plaat, Eds., Springer, 2011, pp. 36-47.

[22] J. A. Stankiewicz, M. H. M. Winands and J. W. H. M. Uiterwijk, “Monte-Carlo Tree Search Enhancements for Havannah,” in *Advances in Computer Games. ACG 2011*, H. J. van den Herik and A. Plaat, Eds., Springer, 2012, p. 60–71.

[23] M. J. W. Tak, M. H. M. Winands and Y. Björnsson, “N-Grams and the Last-Good-Reply Policy Applied in General Game Playing,” *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 4, no. 2, pp. 73-83, 2012.

[24] M. H. M. Winands, Y. Björnsson and J.-T. Saito, “Monte Carlo Tree Search in Lines of Action,” *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 2, no. 4, pp. 239-250, 2010.

[25] T. Pepels, M. H. M. Winands and M. Lanctot, “Real-Time Monte Carlo Tree Search in Ms Pac-Man,” *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 6, no. 3, pp. 245-257, 2014.

[26] M. J. W. Tak, M. H. M. Winands and Y. Björnsson, “Decaying Simulation Strategies,” *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 6, no. 4, pp. 395-406, 2014.

[27] D. J. N. J. Soemers, C. F. Sironi, T. Schuster and M. H. M. Winands, “Enhancements for Real-Time Monte-Carlo Tree Search in General Video Game Playing,” in *2016 IEEE Conference on Computational Intelligence and Games (CIG 2016)*, IEEE, 2016, pp. 436-443.

[28] C. F. Sironi, J. Liu and M. H. M. Winands, “Self-Adaptive Monte Carlo Tree Search in General Game Playing,” *IEEE Transactions on Games,* vol. 12, no. 2, pp. 132-144, 2020.

[29] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel and D. Hassabis, “Mastering the Game of Go with Deep Neural Networks and Tree Search,” *Nature,* vol. 529, no. 7587, p. 484–489, 2016.

[30] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel and D. Hassabis, “Mastering the Game of Go without Human Knowledge,” *Nature,* vol. 550, no. 7676, pp. 354-359, 2017.

[31] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis, “A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go Through Self-Play,” *Science,* vol. 362, no. 6419, pp. 1140-1144, 2018.

[32] M. P. D. Schadd, M. H. M. Winands, H. J. van den Herik, G. M. J. B. Chaslot and J. W. H. M. Uiterwijk, “Single-Player Monte-Carlo Tree Search,” in *Computers and Games (CG 2008)*, H. J. van den Herik., X. Xu, Z. Ma and M. H. M. Winands, Eds., Springer, 2008, pp. 1-12.

[33] N. Sturtevant, “An Analysis of UCT in Multi-Player Games,” *ICGA Journal,* vol. 31, no. 4, pp. 195-208, 2008.

[34] J. A. M. Nijssen and M. H. M. Winands, “Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard,” *IEEE Transactions on Computational Intelligence and AI in Games,* vol. 4, no. 4, pp. 282-294, 2012.

[35] M. Schmid, M. Moravcík, N. Burch, R. Kadlec, J. Davidson, K. Waugh, N. Bard, F. Timbers, M. Lanctot, Z. Holland, E. Davoodi, A. Christianson and M. Bowling, “Player of Games,” arXiv:2112.03178, 2021.

[36] B. Goertzel and C. Pennachin, Artificial General Intelligence, Springer, 2007.

[37] F. Wimmenauer, M. Mihalák and M. H. M. Winands, “Monte-Carlo Tree-Search for Leveraging Performance of Blackbox Job-Shop Scheduling Heuristics,” arXiv:2212.07543, 2022.

[38] L. Rossi, M. H. M. Winands and C. Butenweg, “Monte Carlo Tree Search as an Intelligent Search Tool in Structural Design Problems,” *Engineering with Computers,* vol. 38, p. 3219–3236, 2022.

[39] D. Hennes and D. Izzo, “Interplanetary Trajectory Planning with Monte Carlo Tree Search,” in *IJCAI 2015*, Q. Yang and M. Wooldridge, Eds., Buenos Aires, Argentina, 2015, pp. 769-775.

[40] M. H. S. Segler, M. Preuss and M. P. Waller, “Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI,” *Nature,* vol. 555, no. 7698, pp. 604-610, 2018.

[41] J. Van Eyck, J. Ramon, F. Guiza, G. Meyfroidt, M. Bruynooghe and G. Van den Berghe, “Guided Monte Carlo Tree Search for Planning in Learned Environments,” in *Asian Conference on Machine Learning*, 2013, p. 33–47.

[42] G. Zhu, D. Lizotte and J. Hoey, “Scalable Approximate Policies for Markov Decision Process Models of Hospital Elective Admissions,” *Artificial Intelligence in Medicine,* vol. 6, no. 1, pp. 21-34, 2014.

[43] J. L. Nguyen, N. R. J. Lawrance, R. Fitch and S. Sukkarieh, “Real-Time Path Planning for Long-Term Information Gathering with an Aerial Glider,” *Autonomous Robots,* vol. 40, p. 1017–1039, 2016.

[44] A. Arora, R. Fitch and S. Sukkarieh, “An Approach to Autonomous Science by Modeling Geological Knowledge in a Bayesian Framework,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, p. 3803–3810.

[45] F. Golpayegani, I. Dusparic and S. Clarke, “Collaborative, Parallel Monte Carlo Tree Search for Autonomous Electricity Demand Management,” in *2015 Sustainable Internet and ICT for Sustainability (SustainIT)*, 2015.

[46] A. Mirheli and L. Hajibabai, “Utilization Management and Pricing of Parking Facilities Under Uncertain Demand and User Decisions,” *IEEE Transactions on Intelligent Transportation Systems.,* vol. 21, no. 5, pp. 2167-2179, 2020.

[47] C. F. Sironi and M. H. M. Winands, “Adaptive General Search Framework for Games and Beyond,” in *2021 IEEE Conference on Games (CoG)*, IEEE, 2021.

[48] H. Baier and M. Kaisers, “Explainable Search,” in *2020 IJCAI-PRICAI Workshop on Explainable Artificial Intelligence*, 2021.

[49] A. Wilbik, *Personal Communication,* 2022.

[50] QANU, “Report on the Bachelor’s Programme Data Science and Knowledge Engineering of Maastricht University,” 2020.