*SPS2022*
*A.H.C. Ng et al. (Eds.)*


*doi:10.3233/ATDE220190*

# The Multiple Uses of Monte-Carlo Tree Search

Richard SENINGTON [1]
*Högskolan i Skövde, Högskolevägen Box 408, 541 28 Skövde, Sweden*

**Abstract.** Modern production processes are continuing to move towards more flexible and dynamic conditions, most clearly exemplified by mass customization, but this flexibility can also be seen in technologies like; Human-Robot Collaboration, Automated Guided Vehicle fleets for just in time delivery of parts within factories and reconfigurable manufacturing. Currently, these technologies are developing independently of one another and the supporting industrial software tools such as line balancing optimisation tools, Machine Execution Systems and fleet management tools are similarly developing independently. An alternative to developing individual technologies for each problem is the use of a shared algorithmic framework that can support all of these problem types and future research into general smart factory technology. Monte Carlo Tree Search is a relatively recent Artificial Intelligence algorithm, sometimes described as a general-purpose heuristic, that has been found to be very effective in several theoretical and game-related problems. This paper will review the current growth in research into possible industrial applications of this algorithm and how a framework utilising this algorithm can help to realise the aims of the smart factory vision.

**Keywords.** Artificial Intelligence Real time decision making Flexible production

## 1. Introduction

For the last decade, there has been great interest in what has been called by a variety of names including Industrie 4.0, the Fourth Industrial Revolution, and Smart Industry. The concept has been to move from the mass production model found in industries today, towards a highly flexible production model that can quickly adapt to changing conditions. This is to be supported by the continued increases in the use of robotics of various kinds and a similar increase in the use of IT infrastructure, machine learning, simulation and predictive technologies. The purpose is intended to support; mass customisation where consumers get products that are tailored to their specific wishes, worker productivity and resilience in the production process where recovery from issues is handled quickly and automatically. This is to be done while not sacrificing overall factory productivity and either maintaining or improving the safety and health of the workers.

A general issue in the smart factory will be the ability to change the behavior of production lines and cells to react to current conditions or support changes in demand. For example, in a mass customisation system, each order will be slightly different and

[1]Corresponding Author: richard.james.senington@his.se

the system needs to be able to support all the variations while being extensible in the future and correctly executing all processes. Human-robot collaboration (HRC) provides an alternative example in which the robot would ideally react to the activities that the human coworkers perform. In these examples, the decision-making system needs to be close to real-time to support flexibility and adaptivity within the factory.

Monte Carlo Tree Search (MCTS) is a relatively new Artificial Intelligence (AI) algorithm that has gained significant attention and success in the field of games. The algorithm has been described as a general-purpose heuristic, a search process that can be applied to a wide range of different problems and applications and remain quite effective within all of them. This paper will look at a recent increase in interest in this algorithm from within industrial research. It will look at the topics that recent papers have looked at and several interesting applications that the algorithm can be applied to. The paper will then discuss the possible application of MCTS to enable a learning factory and will conclude with a summary of the discussion presented.

## 2. Literature Review

Monte Carlo algorithms, or algorithms with a stochastic component, are at least as old as modern computers. Monte Carlo Tree Search is a relatively recent addition to this collection. It has its roots in research conducted in the late 1980s into stochastic tree search methods for two-player games [1] and was received its current name 2006 [12]. Researchers using these methods have primarily focused their attention on various games such as Chess and famously Go where the algorithm was a key component in allowing researchers to create a Go AI capable of beating a human grand-master [30].

The application of MCTS to industrial problems has also been growing with the earliest research into potential applications being in the field of industrial planning [10]. More recent interest in planning has looked at logistics [14], flow shop optimisation [22] and job shop optimisation [21] which is also of use in achieving flexible manufacturing systems. A more specific type of planning that is important for future industrial systems is planning related to robots. These applications can take the form HRC [32,29] or of Autonomous Guided Vehicles (AGVs), which introduce the issue of multi-robot planning [18]. Other researchers are examining combining MCTS with other industrial approaches such as Digital Twins [24]. A more complete review of recent applications, including examples of industrial planning, scheduling, optimisation and transport planning can be found in [31].

While this industrial interest must be examined it is also important to note possible limitations of MCTS such as those seen in a comparison with other heuristics methods [25]. This study showed that MCTS suffers as the time made available between decisions decreases, which indicates an issue that needs to be examined in more detail to examine the possible impact on a smart factory environment, though it must be noted that only one problem type (coalition structure generation) was used as the test case.

In recent years there has been substantial growth in interest regarding MCTS and its possible industrial applications. To explore this growth and the interests that the researchers are focusing on several searches of the online libraries IEEE, ACM DL and ScienceDirect were performed. To examine general growth a simple query for the keywords *"Monte Carlo MCTS"*. This was chosen after it was found that MCTS alone would also

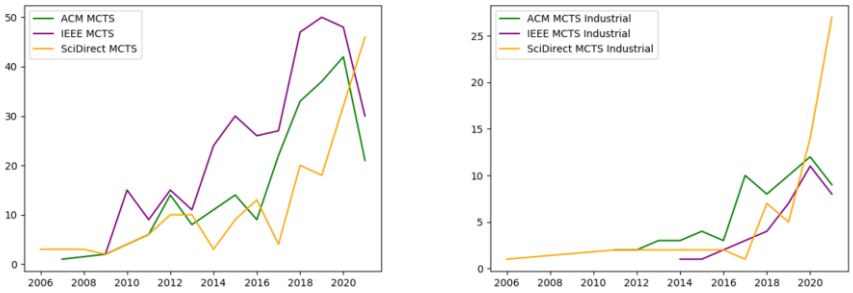


**Figure 1.** Illustrative charts of the number of papers being published in MCTS generally and specifically related to industrial manufacturing. The dip visible in 2021 is noticeable however at the time of writing only half of the year had passed, providing a plausible explanation for this drop.

find papers related to the MCT-4 protein which was not relevant to the interests of this paper. The data for the industrial interest in MCTS was found using two queries the first being "Monte Carlo MCTS industrial" the second "Monte Carlo MCTS robot". Some sanitisation of the data was performed after checking the retrieved paper titles. Since MCTS was only named in 2006 we limited the date range to 2006-2021. All documents needed to at least mention Monte Carlo Tree Search as opposed to monte carlo simulations but not MCTS. Some other sanitisation was performed since some papers on unrelated topics were returned such as some related to radiotherapy and papers with identical titles were not counted twice. The results of these searches, divided by source, can be seen in Figure 1. As can be seen, there is an increase in interest in MCTS generally which is also followed, though more sharply, by the industrial interest.

To examine the interests expressed in these papers the tool VOSviewer [33] was used. The approach taken was to generate a graph based on the keywords attached to each paper. Some mild sanitisation of these keywords was used, such as unifying a number of variations on MCTS into a single keyword. The graph was generated where each keyword's importance was based on the number of papers that refered to it and connections between keywords was based on collocation in an individual papers keyword set. The resulting chart can be seen, using a heatmap visualisation, in Figure 2.

Figure 2 shows a pattern with 5 major areas, the center and 4 branches, however a general observation is that the most common terms appearing are quite varied, though they might refer to similar areas of interest, such as *convolutional neural networks* and the more general *deep reinforcement learning*. Despite this three of the branches appear to coalesce into sub-topics; simulation, coordination/multi-agent systems & planning/explainable planning. The core itself also includes the terms *trajectory* and *robotic exploration* which appear to suggest robotics as a key interest in many of these papers. The final branch is harder to classify including the very specific *bridge maintenance policy* but also the more general *internet-of-things*.

The keywords that are focused on in this diagram can be broadly mapped to topics previously mentioned such as HRC, with the interest in coordination and multi-agent systems, AGVs and multi-robot planning which could include robotic exploration and planning in general. The remainder of this paper will look in more detail at the topics of HRC, AGV fleets and replanning in reconfigurable manufacturing.

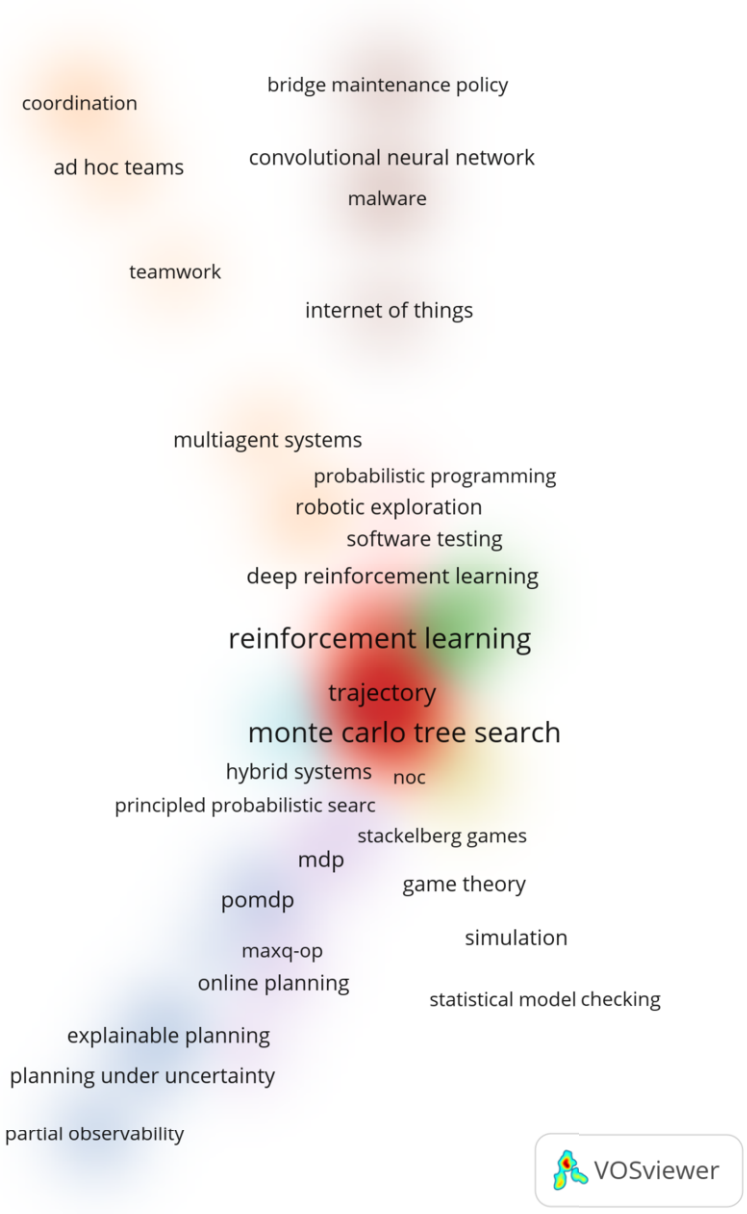


**Figure 2.** A VOSviewer generated graph of keyword pairs and frequency of keyword appearance within recent papers, providing a cluster map of the interests of papers recently published.

## 3. An Overview of the Algorithm

The algorithm for MCTS is, at its core, a tree exploration algorithm. Like other tree search algorithms, it has a root node and a tree of nodes beneath it ending in leaf nodes. Each node of the tree is either fully explored (all possible actions have been tried) or not fully explored yet. For a process that has a definite end, such as a game, some leaf nodes

will not be able to be expanded further. In each iteration of the algorithm, a single node is selected that has not yet been fully explored. A single available action is applied to the node to create a new node. These actions vary depending on the application, it might be the move in a game of Chess, a navigation decision for an AGV or the next step in a factory production process.

Where MCTS differs from traditional tree search methods is that after each node is created the algorithm performs a *simulation* step, where the problem (such as a game of chess) is explored until a terminal condition is reached, however, all actions are applied randomly and during this process. The result of this simulation phase (for instance a win or loss in a game) is then recorded on the new node as an estimate of that nodes quality. The result of the simulation is then added to each parent node going backwards to the root of the tree. In this way, each expansion adds a new simulation and some new information to a series of nodes in the search tree regarding the kinds of results that will happen from taking particular actions. When the time comes to make a decision, it will be the option that leads from the root node to the node that is currently judged best in the root's direct children. An illustration of how an MCTS search develops can be seen in figure 3.

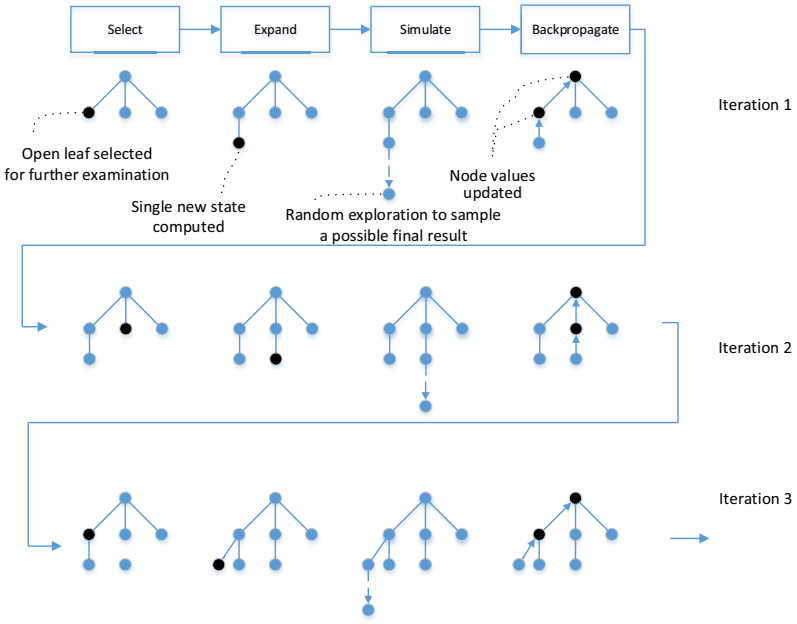


**Figure 3.** A diagram of the process that an MCTS tree grows through. This has been adapted from [11].

As a quick example, let us consider using MCTS on chess. Each node will represent a chessboard with the position of pieces and which player should move next, just as in any other search process. At each stage the algorithm will pick a node that is not fully explored to expand, which involves selecting a piece and moving it, creating a new board and storing this in a new node. The new node will be weighted (white 0, black 0) to indicate that there is no information regarding which player is likely to win initially.

The simulation phase now begins with random moves applied to the chess game until a player wins, or there is a draw. Each node on the path from the root of the tree to the newly created node is then updated, increasing the score of whichever player won in that simulation by 1. Over time bad moves should lead to random simulations that will show the other player winning, and vice-versa. At the time a move must be made in the real game the balance of scores on each node will represent a probability for how the game is likely to turn out if that move is made. MCTS has been called a general-purpose heursitic [8] because can be applied to almost any problem which can be represented as a tree with a branching function and will quickly begin to produce this probabilistic measure of the quality of possible actions.

In much research into MCTS, a key question has been what order to explore nodes in so that the algorithm gives the best possible decisions. Different strategies have been developed including a round-robin strategy, Upper Confidence Bound and the use of Neural Networks [26] to learn how to prioritise the expansion order. How well an expansion strategy will work will depend on the particular domain that the algorithm is being applied to. For a more complete examination of different strategies see [8].

The final issue in implementing MCTS is very similar to those that exist for other tree search algorithms, how to model the problem and how to implement a branching function. For games such as Chess and Go, the model of the game will be the same as for other kinds of tree search such as depth-first and breadth-first search. The first issue in a manufacturing system will be how to easily model the state of the system and the possible actions that can be taken at any time. One approach to this has been called Monte Carlo Action Programming (MCAP), an approach that applies MCTS search over a state action model of a problem [5]. Engineering, Manufacturing and robotics also present other issues, such as continuous environments (an AGV is not just in location A or B but at any point between) and continuous time [2]. Another difference between a game and a manufacturing process is that there may not be a clear endpoint, since it may be possible to begin producing a new product as soon as the current one is complete. This might give rise to an infinite search process. One way to solve the infinite search process issue is to track the expected time of operations, and hence how far the search and simulation have gone, and limit the depth of the search by time as seen in [29]. The precise modelling approach to make use of will vary by application.

This section has presented a brief overview of the algorithm. For those interested in further details of the MCTS algorithm other much more in-depth tutorials and surveys exist [16,8].

## 4. Human Robot Collaboration

Human-Robot collaboration is defined in [4] as performing a shared task as a team and this survey noted several issues that would need to be considered, including; "*decision making, planning, learning*" and that "*Efficient collaboration requires a common plan for all involved partners.*" Other examinations of HRC have proposed a series of levels of the relationship between robot and worker such as in [34] where the levels used are; complete separation, coexistence (shared space), synchronisation, cooperation and full collaboration. However, at present, despite collaborative robots becoming more common in real industrial applications, safety constraints continue to limit how closely humans can work with robots.

There are a range of requirements for a fully safe and collaborative robotic tool including; intention recognition, sensors and flexible automatic planning and decision making. It is this last aspect of HRC that several researchers have begun to investigate using the MCTS algorithm to resolve.

The benefits of using MCTS to provide planning for cooperation, or the solving of problems using more than one agent, has been of general interest to MCTS researchers [35,13] and HRC is an example of such a cooperative multi-agent system where agents are either human or robot. To provide collaboration between humans and robots MCTS can be utilised to provide the planning and decision-making system for either the robotic/mechanical components of a cell responding to human actions via sensors or include humans directly via an instruction system. One approach to implementing the MCTS algorithm is to model the state of the components of the cell, the possible actions that can be utilised under various conditions and how long different actions will take. It is important that the actions are composable and ideally that some actions can be reused in various chains, such as moving a robot arm from one location to another. This follows the concept of modelling using an action programming system and then searching using MCTS as seen in [5]. Such an approach has previously been shown on a single robot [32] and in a test cell designed for one robot and one or two human co-workers [29]. It should also be mentioned that another successful approach to HRC has utilised Partially Observable Markov Models [7] and that MCTS can also be used as a search strategy within such as system.

A system utilising MCTS would be expected to run a search process, growing the MCTS tree, for as long as it can between actions being performed. Several actions could be running at the same time, on the condition that the agent (robot or worker for example) is not the same for each action and that other parts of the system, modelled as stateful variables, are not shared inappropriately. For a practical system to be implemented it must also respond to *events*. These might be sensor events, command events or most commonly the end of a running operation. An event interrupts the MCTS search process and causes some state variables to be changed and choices to be made based on the current data harvested from the tree. A small example of how such a state-action model might look is given in Fig 4.

| State Variable | State Set |
|---|---|
| Gripper | {Open, CarryPtA, CarryPtB, Closed} |
| Rob | {LocA, LocB, LocC} |
| RobAvailable | {True,False} |

(a) State Variables

| Action | Time | Preconditions | Effects |
|---|---|---|---|
| Open Gripper | 3s | Gripper=Closed & RobAvailable | Gripper→Open |
| TakePartA | 9s | Rob=LocA & Gripper=Open & RobAvailable | Gripper→CarryPtA |

(b) Robot Actions

**Figure 4.** A small illustrative example of how possible actions in a robot cell might be represented using a simplified state-action modelling system.

## 5. Autonomous Guided Vehicle Fleets

Another task in the future smart factory will be automated warehousing and delivery of required parts to the workers. This task is expected to be handled by autonomous robots and Amazon Warehouses can be looked at for a current industrial example of such a system. One issue in such environments is studied under the title of Conflict-Free Routing, where the task is to give instructions to all the AGVs such as to avoid deadlock, collisions and optimise the system. This is one example of such an algorithm [17].

MCTS has been examined for use in multi-robot path planning [18] and more generally, as with HRC, an MCTS based system would be expected to adapt to changing conditions (such as unexpected blockages) in the environment or changing tasks. This sort of dynamic planning has been recently considered for AGVs, particularly when navigating around other independent agents (potentially human workers) [15]. A similar use case is found in the issue of planning actions for automatic driving [20]. In future this research will either be found in autonomous vehicles providing transport between sites or AGVs within factories. This approach used a state-action system to model possible actions and outcomes.

While implementing the MCTS system for an AGV fleet could make use of a pure state-action system such as that looked at in the section on HRC, the number of possible states and actions would be expected to increase quickly as the size of the environment is increased. An alternative approach used by [28] is to combine a state-action system for requirements like the current load of the AGVs but to make use of a graph for navigation in the environment. It is easy to imagine such a model growing to manage heterogenous AGVs, each with different capabilities, their recharging requirements and the need to adapt to a shared environment with human workers. The use of an MCTS based planner also allows for the possibility of adding new actions to the search process at runtime and having them automatically incorporated into the planning cycle.

## 6. Software for Reconfigurable Manufacturing

Another issue that smart factories often discuss is that of mass customisation [27], the requirement for small lot sizes or even every item being unique. It has been suggested that simply allowing for more variations in an otherwise traditional production line will not be enough to support these demands upon the industry [19]. Hence research has been ongoing into Reconfigurable Factories, where machines will be easily replaced, moved, adapted and reprogrammed to support the current desired output of the company.

There are several issues to consider before fully reconfigurable factories can be achieved [6] and of these two issues that computer systems are used for are production planning and process planning. Various methods exist including those that address well-known problems such as flow and job shop optimisation, or simulation-based methods. In the previous HRC example the robot, and indeed the cell, was planning in direct response to the current conditions of the production process. This can be seen as a form of process or production planning, depending on the scale the MCTS system is being applied to.

While MCTS can be used directly as a planning agent [23] the state-action approach offers a further possible benefit in the application of reconfigurability. In Fig 4 a set of

states and actions was shown. The actions are relatively small and can be composed together. Hence when working with rapidly changing production requirements one action might be used in several ways, where appropriate. Additionally changing the set of states and actions, through adding or removing items, is relatively easy (subject to how easily the underlying operations can be written). Once a change is made the new configuration will be taken into account almost at once in the planning process. This allows the capability of a machine or robot to be changed by adding new actions alone, not describing directly how these actions should be sequenced. When a machine is moved, some old actions might need to be removed, but others might remain and get reused. It will be important when making these changes to check the reachability of states in a process, and the safety and correctness of the possible sequences that the algorithm is capable of finding. While potentially difficult in large systems, checking the capabilities of a set of states and actions is possible using simulation tools.

In conclusion, given the evidence in the AGV and HRC sections of this paper, it seems likely that MCTS can also provide useful capability in the field of reconfigurable manufacturing.

## 7. Learning in the Smart Factory

As was noted earlier in the paper, one variation of MCTS that has been found to be very successful is utilising neural networks to provide the expansion guide heuristic. This provides a combination of strict rules within the underlying MCTS algorithm, with the advantages of machine learning applied to previous experience or simulated experiences. The possible application of this to robotic systems is beginning to be explored [26].

One of the various proposed characteristics of future smart factories is that they will make use of machine learning to automatically adapt how they react to changing conditions. MCTS offers one possible approach to achieving safe decision making combined with learning. The MCTS search process can only consider actions that are possible from a current state. The generation of legal actions would be provided either by human programmers (as in the previous HRC example) or automatically generated by other external programs. As previously discussed these actions would need to be composable but also safe to use under the conditions that allow their activation. In an offline simulation of a factory, a neural network to manage the tree expansion method could then be trained. This training process could use either historical data or randomly generated situations, so long as these would still be within the expected requirements of the factory. Once a set of actions and the related neural network are prepared these would then be downloaded back to the machines of the real factory, completing the learning cycle. This proposal can be seen in Fig 5.

## 8. Summary

The focus of this paper has been the MCTS algorithm and its potential application in the future of smart factory development. There is an increasing interest in this algorithm and researchers are finding applications for it in many tasks that require either machines or entire production processes to react to quick changes in their environment. These changes

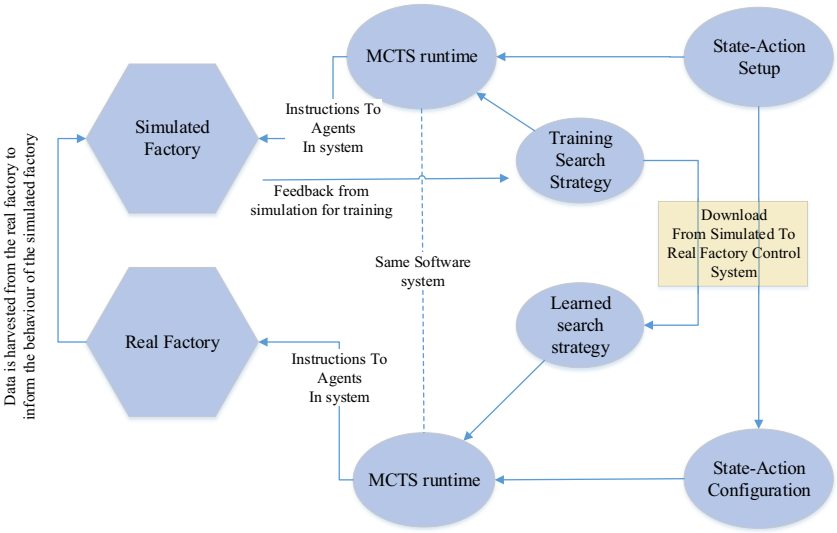


**Figure 5.** A diagram of where the MCTS algorithm, with learning algorithm for the search strategy optimisation, could fit into a future smart factory environment.

can be caused by human coworkers as seen in HRC or vehicle navigation, or demands placed on the factory as seen in mass customisation and reconfigurable manufacturing applications.

Other potential uses that have not been discussed here for lack of space include 3D printing tool path planning [36], offline flow shop optmisation [23], predictive maintenence [9] and recovery from error through automatic replanning. Both predictive maintenance and recovery from issues in a production process aid in supporting the resilience of a manufacturing system. Several of the examples cited here also make use of the combination of machine learning with MCTS to further strengthen their approach. This implies a more general use of MCTS as a useful bridge between the intentions of data gathering, machine learning and automatic improvement and adaptation that is often described as of interest for smart factories.

MCTS continues to be explored in a range of other domains and one recent paper [3] points towards further research in using the nature of MCTS to explain the decisions being made. Being a tree search algorithm it is potentially possible to harvest data about the decison making process from it, and use this to provide explanation of the final actions. This could also be of great interest in future industry, where understanding the reason for the behaviour of a smart factory or machine could provide great insight into efficiency, performance, errors and support root cause analysis.

In conclusion this paper proposes that MCTS is an algorithm that is gaining considerable interest at present and appears to have the potential to address a range of tasks in smart industry. This implies that the algorithm is flexible, both in terms of the applications it can be applied to and also how it can be adapted or retrained at runtime. It can cope with stateful environments and react quickly to changing conditions or accept new options and make use of them with limited manual configuration. This paper proposes that the algorithm is deserving of even greater focus within research on smart factories.

## Acknowledgements

The author wishes to thank Marie Schnell, Bernard Schmidt and Carlos Alberto Barrera Diaz for their advice and support in preparing this paper.

## References

[1] Bruce D. Abramson. *The Expected-outcome Model of Two-player Games*. PhD Thesis, Columbia University, New York, NY, USA, 1987.

[2] Seydou Ba, Takuya Hiraoka, Takashi Onishi, Toru Nakata, and Yoshimasa Tsuruoka. Monte carlo tree search with variable simulation periods for continuously running tasks. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 416–423, 2019.

[3] Hendrik Baier and Michael Kaisers. Towards explainable mcts. In *2021 AAAI Workshop on Explainable Agency in AI.*, 2020.

[4] Andrea Bauer, Dirk Wollherr, and Martin Buss. Human-robot collaboration: a survey. *I. J. Humanoid Robotics*, 5:47–66, 03 2008.

[5] Lenz Belzner. Monte Carlo Action Programming, February 2017. arXiv: 1702.08441 [cs].

[6] Marco Bortolini, Francesco Gabriele Galizia, and Cristina Mora. Reconfigurable manufacturing systems: Literature review and research trend. *Journal of Manufacturing Systems*, 49:93–106, 2018.

[7] Nakul Gopalan Brown and Stefanie Tellex. Modeling and solving human-robot collaborative tasks using pomdps. In *Proc. Robot. Sci. Syst.*, pages 1–7, 2015.

[8] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[9] Annelie Carlson and Tomohiko Sakao. Environmental assessment of consequences from predictive maintenance with artificial intelligence techniques: Importance of the system boundary. *Procedia CIRP*, 90:171–175, 01 2020.

[10] G. M. J. B. Chaslot, S. de Jong, J. T. Saito, and J. W. H. M. Uiterwijk. Monte-Carlo tree search in production management problems. In *BNAIC'06: Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence*, pages 91–98. University of Namur, January 2006.

[11] Guillaume Chaslot, Mark Winands, H Herik, Jos Uiterwijk, and Bruno Bouzy. Progressive strategies for monte-carlo tree search. *New Mathematics and Natural Computation*, 04:343–357, 11 2008.

[12] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In H. Jaap van den Herik, Paolo Ciancarini, and H. H. L. M. (Jeroen) Donkers, editors, *Computers and Games*, pages 72–83, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[13] Mohammadreza Daneshvaramoli, Mohammad Sina Kiarostami, Saleh Khalaj Monfared, Helia Karisani, Keivan Dehghannayeri, Dara Rahmati, and Saeid Gorgin. Decentralized communication-less multi-agent task assignment with cooperative monte-carlo tree search. In *2020 6th International Conference on Control, Automation and Robotics (ICCAR)*, pages 612–616, Palo Alto, California, 2020.

[14] Stefan Edelkamp, Max Gath, Christoph Greulich, Malte Humann, Otthein Herzog, and Michael Lawo. Monte-Carlo Tree Search for Logistics. In Uwe Clausen, Hanno Friedrich, Carina Thaller, and Christiane Geiger, editors, *Commercial Transport*, Lecture Notes in Logistics, pages 427–440. Springer International Publishing, 2016.

[15] Stuart Eiffert, He Kong, Navid Pirmarzdashti, and Salah Sukkarieh. Path planning in dynamic environments using generative rnns and monte carlo tree search, 2020. arXiv:2001.11597 [cs].

[16] Michael C. Fu. Monte carlo tree search: A tutorial. In *2018 Winter Simulation Conference (WSC)*, pages 222–236, 2018.

[17] Ewgenij Gawrilow, Max Klimm, Rolf Möhring, and Björn Stenzel. Conflict-free vehicle routing. *EURO Journal on Transportation and Logistics*, 1, 06 2012.

[18] Phillip Hyatt, Zachary Brock, and Marc D. Killpack. A versatile multi-robot monte carlo tree search planner for on-line coverage path planning. *ArXiv*, abs/2002.04517, 2020.

[19] Yoram Koren, Xi Gu, and Weihong Guo. Reconfigurable manufacturing systems: Principles, design, and future trends. *Frontiers of Mechanical Engineering*, 13(2):121–136, June 2018.

[20] Karl Kurzer, Chenyang Zhou, and J. Marius Zöllner. Decentralized cooperative planning for automated vehicles with hierarchical monte carlo tree search. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 529–536, 2018.

[21] Kexin Li, Qianwang Deng, Like Zhang, Qing Fan, Guiliang Gong, and Sun Ding. An effective mcts-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem. *Computers & Industrial Engineering*, 155:107211, 2021.

[22] Marco Lubosch, Martin Kunath, and Herwig Winkler. Industrial scheduling with Monte Carlo tree search and machine learning. *Procedia CIRP*, 72:1283 – 1287, 2018. 51st CIRP Conference on Manufacturing Systems.

[23] Marco Lubosch, Martin Kunath, and Herwig Winkler. Industrial scheduling with monte carlo tree search and machine learning. *Procedia CIRP*, 72:1283–1287, 2018. 51st CIRP Conference on Manufacturing Systems.

[24] Marvin Carl May, Leonard Overbeck, Marco Wurster, Andreas Kuhnle, and Gisela Lanza. Foresighted digital twin for situational agent selection in production control. *Procedia CIRP*, 99:27–32, 01 2021.

[25] Fredrik Präntare, Herman Appelgren, and Fredrik Heintz. Anytime heuristic and monte carlo methods for large-scale simultaneous coalition structure generation and assignment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11317–11324, May 2021.

[26] Benjamin Riviere, Wolfgang Hoenig, Matthew Anderson, and Soon-Jo Chung. Neural tree expansion for multi-robot planning in non-cooperative environments, 2021. arXiv: 2104.09705 [cs].

[27] Enrico Sandrin, Alessio Trentin, and Cipriano Forza. Organizing for mass customization: Literature review and research agenda. *International Journal of Industrial Engineering and Management*, 5:159–167, 01 2014.

[28] Konstantin M. Seiler, Andrew W. Palmer, and Andrew J. Hill. Flow-achieving online planning and dispatching for continuous transportation with autonomous vehicles. *IEEE Transactions on Automation Science and Engineering*, pages 1–16, 2020.

[29] Richard Senington, Bernard Schmidt, and Anna Syberfeldt. Monte carlo tree search for online decision making in smart industrial production. *Computers in Industry*, 128:103433, 2021.

[30] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[31] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review on recent modifications and applications. *ArXiv*, abs/2103.04931, 03 2021. Accessed:2022-01-31.

[32] Marc Toussaint, Thibaut Munzer, Yoan Mollard, Li Yang Wu, Ngo Anh Vien, and Manuel Lopes. Relational activity processes for modeling concurrent cooperation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5505–5511, 2016.

[33] Ludo Waltman, Nees Jan van Eck, and Ed C.M. Noyons. A unified approach to mapping and clustering of bibliometric networks. *Journal of Informetrics*, 4(4):629–635, 2010.

[34] Bauer Wilhelm, Bender Manfred, Martin Braun, Peter Rally, and Oliver Scholtz. Lightweight robots in manual assembly – best to start simply! examining companies' initial experiences with lightweight robots. Technical report, Fraunhofer Institute For Industrial engineering Iao, Fraunhofer IAO Nobelstraße 12 70569 Stuttgart, 10 2016.

[35] Piers Williams, Joseph Walton-Rivers, Diego Perez Liebana, and Simon Lucas. Monte carlo tree search applied to co-operative problems. In *2015 7th Computer Science and Electronic Engineering Conference (CEEC)*, pages 219–224, Sep 2015.

[36] Chanyeol Yoo, Sam Lensgraf, Robert Fitch, Lee Clemon, and Ramgopal Mettu. Toward optimal fdm toolpath planning with monte carlo tree search. In *Conference: 2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4037–4043, 05 2020.