

# Reinforcement Learning and Life Cycle Assessment for a Circular Economy - Towards Progressive Computer Science

Johannes Buchner

IPE Berlin

**Abstract.** The aim of this paper is to discuss the potential of using methods from Reinforcement Learning for Life Cycle Assessment in a circular economy. To give some context, we explain how Reinforcement Learning was successfully applied in computer chess (and beyond). As computer chess was historically called the "drosophila of AI", we start by describing a method for the board representation called 'rotated bitboards' that can potentially also be applied in the context of sustainability.

There exist several techniques for representing the chess board inside the computer. In the first part of this paper, the concepts of the bitboard-representation and the advantages of (rotated) bitboards in move generation are explained. In order to illustrate those ideas practice, the concrete implementation of the move-generator in FUSc# (a chess engine developed at FU Berlin in C# some years ago) is described. In addition, rotated binary neural networks are discussed briefly.

The second part deals with reinforcement learning in computer chess (and beyond). We exemplify the progress that has been made in this field in the last 15-20 years by comparing the "state of the art" from 2002-2008, when FUSc# was developed, with the ground-breaking innovations connected to "AlphaZero". We discuss how a "FUSc#-Zero" can be implemented and what is necessary to reduce the number of training games necessary to achieve a good performance. This can be seen as a test case to the general problem of improving "sample efficiency" in reinforcement learning. We then move beyond computer chess, as the importance of sample efficiency extends far beyond board games into a wide range of applications where data is costly, difficult to obtain, or time consuming to generate. We review some application of the ideas developed in AlphaZero in other domains, e.g. the "other Alphas" like AlphaFold, AlphaTensor, AlphaGeometry and AlphaProof.

In the final part of the paper, we discuss the computer-science related challenges that changing the economic paradigm towards (absolute) sustainability poses and in how far what we call 'progressive computer science' needs to contribute. Concrete challenges include the closing of material loops in a circular economy in order to optimize for (absolute) sustainability, and we present some new ideas in this direction. Finally we discuss the potential of such methods for ecological economic planning by changing the current mostly 'linear' and profit-driven way of organizing the economy towards a more circular, democratic and sustainable mode of production.

## 1 Introduction

The aim of this paper is to describe some methods that were successful in computer chess like (Rotated) Bitboards ([34]) and Reinforcement Learning ([35]), and to discuss their potential for solving problems in other domains. We note that since the introduction of "AlphaZero" ([37]) that was able to achieve superhuman performance in chess, trained only by reinforcement learning from games of self-play, similar approaches lead to spectacular success in other domains, e.g. the "other Alphas" (all created by DeepMind) like AlphaFold [43], AlphaTensor [42] and AlphaProof [44]. In this spirit, we propose some new ideas how to tackle the problem of closing of material loops in a circular economy in order to optimize for sustainability, which is part of a research program that we call 'Progressive Computer Science' in order to respect planetary boundaries (see [26]) and avoid climate tipping points [27]. Finally we elaborate on the potential of artificial intelligence methods for (ecological) economic planning in order to achieve economic democracy within planetary boundaries, as outlined e.g. in [40] or [41].

The first two sections of this paper are essentially a compressed version of the (unpublished) paper [22]. In the first part, our implementation of rotated bitboards in FUSC# is discussed in detail. FUSC# is the chess program that was developed by the "AG Schachprogrammierung" at the Free University in Berlin ([5], see also <https://www.chessprogramming.org/FUSCsharp>). For more background of FUSC#, see [33]. The second part deals with reinforcement learning in computer chess and beyond. After briefly discussing AlphaZero and its sample efficiency, we review some application of the ideas developed in AlphaZero in other domains. In addition, rotated binary neural networks are discussed briefly.

In the final part, we propose some new ideas how to tackle the problem of closing of material loops in a circular economy in order to optimize for sustainability. We also briefly discuss why this is urgently necessary to avoid climate tipping points [27], and achieve "absolute sustainability" [23]. According to a recent "Global Tipping Points Conference" ([46]), there is growing scientific evidence that exceeding 1.5 °C global warming could trigger multiple climate tipping points. We are deeply convinced that these alarming facts from climate science necessarily demand research on what we call 'Progressive Computer Science', as computer science (as well as many other fields) urgently need to contribute to fighting the ecological crisis.

## 2 Rotated Bitboards in FUSC#

The idea for the bitboard representation of the chessboard is based on the observation that modern CPUs are 64bit-processors, i.e. the length of a word in machine language is nowadays mostly 64bit. The 64bit-words correspond to the 64 squares on the chess board, and those "bitboards" (the name that is used for an unsigned int64) are used to represent various information about the position on the chessboard. The advantage of this representation lies in the availability of very fast bit-manipulating operations on modern CPUs: On 64bit-machines, operations like AND, OR, NOT etc. can be executed on a 64bit "bitboard" in only one cycle. It is therefore possible to construct very efficient chess programs on the basis of the bitboard-approach, because, roughly speaking, the CPU operates on all 64bit "in parallel".

The bitboard method for representing a board game appears to have been invented in the mid-1950s, by Arthur Samuel and was used in his checkers program [36]. In the history of computer chess, there were several authors who used variants of the bitboard-representation in their chess engines. As early as in the seventies, Slate and Atkin described the idea of using bitboards in their program "CHESS 4.5" (see [3], chapter 4). Another prominent program that used this technique successfully is the former computer chess champion "Cray Blitz", written by Robert Hyatt, who continued to develop the program as an open-source project called "Crafty" ([9]). DarkThought, developed at the university of Karlsruhe in the late 90s, is also using bitboards. Crafty and DarkThought were also the first programs that used an important refinement of the bitboard-representation called "rotated bitboards". The author of DarkThought Ernst A. Heinz gives an overview of rotated bitboards as used in DarkThought (see [2]), which inspired much of our implementation of rotated bitboards in FUSC#.

## 2.1 The bitboard-approach towards move-generation

The move-generation is used many times during the search-algorithms used in chess programs. Therefore, an efficient move-generation is needed. Based on the bitboard-approach, there exist different strategies for each of the piece-types in chess. One important concept is to compute bitboards of all possible moves (e.g. of a knight) from all the squares beforehand during the initialisation of the program, and store this information in a data-structure that provides efficient access to these pre-computed moves during the move-generation. For non-sliding pieces, this approach works straightforward, but for sliding-pieces some more tricks are needed, because the possible moves for a sliding piece will depend on the configuration of the line/file/diagonal it is standing.

Therefore, the idea for bitboard-move-generation for sliding pieces is to compute all the possible moves for all squares **and** all configurations of the involved ranks/files/diagonals! For example, in FUSC#, the rank-moves for a rook standing on a1 on an otherwise empty chessboard are stored in “rank\_moves[a1][00000001]”, with the second index of the array being the configuration of the involved rank (i.e. 8 bits, with only “a1” being occupied as the rook is standing there itself). This works fine for rank-moves, because the necessary 8 bits for the respective rank can be easily obtained from the bitboard of the occupied pieces (this bitboard consists of 8 byte, and each of those corresponds to one rank). For file-moves of rooks and queens, and especially for the diagonal moves of bishops and queens, things turn out to be much more difficult: the necessary bits about the respective files/diagonals are spread all over the bitboard storing the "occupied" squares, they are not “in order”, as they are for rank-moves. Here the idea of rotated bitboards helps out.

## 2.2 Rotated bitboards

The idea of rotated bitboards is to store the bitboards that represents the “occupied squares” not only in the “normal” way, but also in a “rotated” manner. Therefore, the necessary bits representing files/diagonals are “in-order” in those rotated bitboards, as needed by the move-generation (see previous section). The “rotated bitboards” are updated incrementally during the search, i.e. when a move is done or undone. The following bitboards are maintained in FUSC#:

- board.occ, which represents the occupied squares in the “normal” representation
- board.occ\_l90, the board flipped by 90 °(for file moves)
- board.occ\_a1h8, for diagonal moves in the direction of the a1h8-diagonal
- board.occ\_a8h1, for diagonal moves in the direction of the a8h1-diagonal

The idea that such an “incremental update” of bitboards is possible whenever a move is made (during search) is essential, and this concept might also be useful when applying the idea of bitboards to other domains like sustainability (see below). For more details on our implementation of rotated bitboards in FUSC#, see [22].

## 2.3 Rotated Binary Neural Networks

Here, we shortly discuss Rotated Binary Neural Networks (see [31]) . Similiar to rotated bitboards, the idea is a rotation (here of the full precision weight vector of a neural network) can significantly improve the performance of an algorithm, although of course the details of the implementation are very different (again bit-wise operations increase the speed). This encourages us that the idea of (rotated) bitboards is applicable to other domains, e.g. like sustainability that we discuss below.

Following [31], we observe that Binary Neural Network (BNN) reduce the complexity of deep neural network, but there is a severe performance degradation. One of the major problems is the large quantization error between the full-precision weight vector and its binary vector. One idea is to compensate for the norm gap (this was done e.g. by [32]) , but the angular bias was hardly touched. In the paper [31], the influence of angular bias on the quantization error is discussed and then a Rotated Binary Neural Network (RBNN) is introduced, which considers the angle alignment between the full-precision weight vector and its binarized version.

### 3 Reinforcement Learning in Computer Chess and Beyond

#### 3.1 The Paper “Reinforcement Learning in Chess Engines” (2008)

We quote from the introduction of the paper “Reinforcement Learning in Chess Engines” ([38]) from 2008, where the state of the art at that time was discussed, based on experiments with FUSC#: “The method presented in this paper optimizes the evaluation functions and its coefficients by automating the use of temporal differences and thereby increasing it’s own understanding of chess after each game.” Back then, the understanding was that “the main problem lies in the correct tuning of the coefficients” of the evaluation functions, and that only there Reinforcement Learning could play a useful role in chess engines, not during the search. In the “related work”-section of the same paper [38], a paragraph on "NeuroChess" is interesting to read from the perspective of today: It states that “experiments with other programs showed that a learning strategy based on playing against oneself, does not yield satisfying results”, which was the common sense in 2008, but changed dramatically with AlphaZero, less that 10 years later.

#### 3.2 The Approach of AlphaZero

In December 2017, a paper was uploaded to arxiv with the title “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm” [37]. Its reinforcement learning algorithm, “starting from random play, and given no domain knowledge except the game rules, achieved within 24 hours a superhuman level of play”. So the game of chess was mastered “by tabula rasa reinforcement learning from games of self-play”, in contrary what seemed possible a decade ago. The approach of AlphaZero is very different to classical chess programs and FUSC#: Instead of an “alpha-beta search with domain-specific enhancements”, it uses “a general-purpose Monte-Carlo tree search (MCTS) algorithm. Each search consists of a series of simulated games of self-play that traverse a tree” from root to leaf (compare p.3 of [37]).

But to achieve this impressive result, millions of self-play training games were used for the training: About 20 million games (approx. 4 hours of training time) were necessary to achieve super-human performance, and about 44 million games (approx. 9 hours of training time) were necessary to beat Stockfish, the best available computer chess programm at the time ([37]). We now collect some ideas how to improve "sample efficiency" (i.e. to reduce the number of training games necessary). In “Conclusion”-Secion of the 2008-paper, it is stated that FUSC# considerably improved performance only after 119 (!) games, and also “we estimate that FUSC# requires more than 50.000 training games ...” (p.9 of [38]) - so in a way, there was a much higher “sample efficiency” (of course with limitations w.r.t the performance/chess playing strength, which peaked only at about 2000 ELO). A promising strategy improve sample efficiency is to relate the “piece evaluation heuristics” to “position type” (e.g. opening, middle game, endgame), because the former will greatly vary with the latter (e.g. “king safety” in the middle game vs. “active king play” in the endgame). At first such a structure could be be "given", but the aim is that a “FUSC#-Zero” will “re-discover” such “position-types” itself in a second step, maybe with even more “position types” that just 3 (according to [38], in FUSC# 33 were used).

#### 3.3 The “other Alphas”

We review the application of the ideas developed in AlphaZero in other domains, , i.e. the “other Alphas” (all created by DeepMind) like AlphaFold [43], AlphaTensor [42] and AlphaProof. [44]. AlphaFold [43] had spectacular success applying the ideas of AlphaZero to predictions of protein structure. The most recent AlphaFold 3 was announced in May 2024, and it can predict the structure of complexes created by proteins with DNA, RNA, various ligands, and ions. Demis Hassabis and John Jumper of Google DeepMind shared one half of the 2024 Nobel Prize in Chemistry, awarded "for protein structure prediction” with AlphaFold (for more details see e.g. [43]). AlphaTensor [42] was developed to shed "light on a 50-year-old open question in mathematics about finding the fastest way to multiply two matrices”, and discovered new, faster algorithms to do so. AlphaProof and AlphaGeometry2 [44] solved four out of six problems from the 2024 International Mathematical Olympiad (IMO), achieving the same level as a silver medalist.

## 4 Absolute Sustainability, LCA and Progressive Computer Science

In this final part of the paper, we propose some new ideas how to tackle the problem of closing of material loops in a circular economy in order to optimize for sustainability. To the best of our knowledge, we are the first to propose to use methods that originate in computer game research (described in the previous section) to tackle such questions. We were heavily inspired by [41].

We start by briefly discussing why this is urgently necessary to avoid climate tipping points [27], and achieve "absolute sustainability" [23]. According to a recent "Global Tipping Points Conference" ([46]), there is growing scientific evidence that exceeding 1.5 °C global warming could trigger multiple climate tipping points.

### 4.1 Absolute Sustainability and Climate Tipping Points

Absolute sustainability aims to achieve a development that stays within the Earth's environmental limits, ensuring both present and future generations can meet their needs while respecting the planet's carrying capacity, according to [23]. This means that the environmental impacts of our activities must be assessed against established thresholds (with "Life Cycle Assessment", see below), such as planetary boundaries, to determine if they are truly sustainable, in an "absolute" sense (in contrast to "relative" sustainability where the question is e.g. "if product A is more sustainable than product B", without reference to an "absolute" reference frame). Absolute sustainability thus focuses on staying within the safe operating space defined by planetary boundaries, which represent the limits of the Earth's systems (see [26]).

According to [27], exceeding 1.5 °C global warming could trigger multiple Climate Tipping Points (CTPs). As the authors explain, "climate tipping points are conditions beyond which changes in a part of the climate system become self-perpetuating. These changes may lead to abrupt, irreversible, and dangerous impacts with serious implications for humanity.". The paper synthesizes evidence for 16 core and regional-impact tipping elements and finds that exceeding 1.5 °C global warming could trigger multiple climate tipping points. As the authors write in the conclusion, "assessment provides strong scientific evidence for urgent action to mitigate climate change. We show that even the Paris Agreement goal of limiting warming to well below 2 °C and preferably 1.5 °C is not safe as 1.5 °C and above risks crossing multiple tipping points. Crossing these CTPs can generate positive feedbacks that increase the likelihood of crossing other CTPs. Currently the world is heading toward 2 to 3 °C of global warming; at best, if all net-zero pledges and nationally determined contributions are implemented it could reach just below 2 °C. This would lower tipping point risks somewhat but would still be dangerous as it could trigger multiple climate tipping points."

### 4.2 Life Cycle Assessment and Circular Economy

The environmental impact of products can be traced with "Life Cycle Assessment" (LCA). One example is openLCA (see [openlca.org](https://openlca.org)), a free and open source software for modelling the life cycle of products and sustainability. For a broader discussion of Life Cycle (Sustainability) Assessment, see [24]. Clearly, a sophisticated LCA is necessary for absolute sustainability and for the closing of material loops in order to optimize for sustainability in a Circular Economy. In Germany, the "National Circular Economy Strategy" (NKWS) was adopted in December 2024 ([28]). The recently published survey paper "AI for the Circular Economy - a tool for sustainable transformation?" ([29]) underlines the important role Artificial Intelligence (AI) can play for such a transformation, if employed in the right places and within the proper context.

**Research Challenge** The fundamental problem of trying to understand how to transform the economy to be more "circular" requires moving from individual production processes to a more global view of the economy, understanding all "intermediate" products and following the lifecycle of these products. If one assumes there are multiple ways of creating the same product (and we need

to choose the most “circular” and “sustainable” one), and also accepts that products have common themes with one another (“features”), one can convert parts of the economic problem to a modern AI planning problem (i.e. a game AI problem). By using modern methods pioneered in the fields of reinforcement learning and neural networks (e.g. see Schrittwieser et al. (2020)), combined with certain insights on matrix inversion (Barto & Duff 1994) we can answer questions like “what is the most circular way to produce goods”. Based on these ideas, we propose to use state of the art methods from AI, Reinforcement Learning and Game Tree Search (e.g. Monte Carlo Tree Search that was successfully used in “AlphaZero”, see above) to investigate such questions. As a first step, we introduce some definitions in order to illustrate this idea and to describe the arising challenges more precisely.

**Some definitions of products and life cycle assessment** Assume that we have an economy with  $k$  products and  $m$  raw materials.

**Definition 1 (Product)** A product  $P$  is a tripel  $p = (l, n, \{a_i\}_{i=1}^n)$ , where  $l \in \mathbb{N}$  stands for the “level” of the product,  $n = n_i \in \mathbb{N}$  for the number of other pre-products it is composed of, and a set consisting of  $a_i$  defining those:

$a_i = (q_i, p_i)$ , where  $q_i \in \mathbb{R}$  stands for the quantity of product  $p_i$  that is needed to produce product  $p$ .

**Remark 1** Note that this is a recursive definition, as the product  $p$  appears “on both sides”. Thus, we define “raw materials” as the seed:

**Definition 2 (Raw Material)** Let there be  $\tilde{r} \in \mathbb{N}$  raw materials in the economy.

A “raw material product” is a product of “level 0”, i.e.  $\text{raw} = p = (0, 1, (1, r))$  with  $r \in \{1 \dots \tilde{r}\}$

**Remark 2** By this definition, all products and intermediate “pre-products” can be tracked back to which raw materials they contain.

**Definition 3 (Life Cycle Assessment, LCA)**

Let there be  $m \in \mathbb{N}$  different indicators for LCA measuring the impact of a product on the environment (and society).

For a product  $p$ , define its Environmental Impact over its lifecycle, as a tuple

$\text{lca}(p) = (b_j)_{j=1}^m$ . For simplicity, and following [25], we assume that only 3 indicators are measured for each product

- labour time  $t$ , i.e. the time necessary to produce  $p$
- “climate cost”  $c$ , i.e. the amount of greenhouse-gases (e.g.  $\text{CO}_2$ -equivalents) the production produces
- raw materials  $r = (r_1, \dots, r_{\tilde{r}})$  necessary to produce

**Remark 3** In this simplified setting, it holds that for each product  $p_i$  we have

$$\text{lca}(p_i) = (t_i, c_i, r_i)$$

**Definition 4 (Recursive Definition of LCA for a product)**

Let  $p$  be a product, with  $p = (l, n, \{a_i\}_{i=1}^n)$  and  $a_i = (q_i, p_i)$ , i.e.  $p$  is composed of  $n$  pre-products  $p_i$  with  $i = 1 \dots n$ , as above. Then define the “economic impact” / “lifecycle assessment” as

$\text{lca}(p) = \sum_{i=1}^n q_i \cdot \text{lca}(p_i) = \sum_{i=1}^n (q_i t_i, q_i c_i, q_i r_i)$ , with the last step following from our “simplified setting” with only 3 indicators.

**Research Ideas** Now we are in a position to formulate concrete challenges on how to close material loops in a circular economy and to find the most sustainable way to produce a product. In order to do so, we assume the following (making these assumptions more precise is part of the challenge to formulate the problem in a way that modern AI-methods can be implemented):

- Assume different “production processes” are defined, which result in products with the same “features”, but different environmental impact...
- ... which also means that products with the same “features” are composed by different pre-products, and the challenge is to find the most “sustainable” and “circular” way to produce all goods in the economy (the latter meaning that material loops are closed where possible, i.e. that the unused output/“waste” from one production process is used as “input” for as many production processes as possible).

**Remark 4** Let  $p$  be a product as above, and let  $x := p_1$  be a pre-product of  $p$ . Assume that a new production process allows us to replace  $x$  by a new pre-product  $y$ , and denote this new product involving  $y$  by  $\tilde{p}$ . Then the environmental impact / LCA of  $p$  changes in the following way:

$$lca(\tilde{p}) = lca(p) - lca(x) + lca(y)$$

**Definition 5** (Challenge: The Game of a Circular and Sustainable Economy)

Observe that searching this “space of possibilities” for an economy with many such possible “pre-product replacements” has similarities with a search tree in computer games like chess, as discussed above. With this analogy in mind, define

- a “move” in the game of a “Circular and Sustainable Economy”:  $\tilde{p} = move(p, x, y)$ , i.e. such a “pre-product replacement” is considered a move in the game.
- as “evaluation function”, the environmental impact of all products produced in the economy needs to be calculated - and according to the principle of “absolute sustainability”, it must be within the “planetary boundaries”

**Remark 5** Even “bitboards” that have been used successfully in many board games may play a role in this setting, because the “incremental update” of “sustainability bitboards” (that can be defined according to the concrete optimization question at hand) by bit-wise logical operations might save a lot of computing time, as with “classical” board games.

### 4.3 Progressive Computer Science

There are arguably many possible definition of “Progressive Computer Science”, depending how “societal progress” is defined. Here, we put the focus on preserving the planet for future generations, as urgent action is necessary in order to avoid the climate tipping points and to respect the planetary boundaries, i.e. to reach absolute sustainability (see above).

**Definition 6** (“Progressive Computer Science”)

We define “Progressive Computer Science” as an umbrella for research in computer science that contributes to transforming the economic system towards absolute sustainability. This involves changing the rules how the economy works in a way that ensures planetary boundaries are respected.

One example that falls in this category of “Progressive Computer Science” is research on ecological economic planning with computer science methods, which is the topic of the next section.

### 4.4 Ecological Economic Planning and AI

Recent developments in AI (e.g. in the field of reinforcement learning) have enabled economic planning at large scales e.g. in multinational companies (see e.g. Phillips and Rozworski (2019)). In the EU, a “Circular economy Action Plan” [48] was adopted, and one central part of it is to improve the availability of digital product data. The “EU digital product passport” (see e.g. [47] or [49]) is designed to provide information about each product’s origin, materials, environmental impact, and disposal recommendations. With this European-wide product data infrastructure for all products sold and used in the EU, new forms of ecological economic planning will become possible. For a recent discussion of such methods and institutions regarding Artificial Intelligence and economic planning, see [40], for a broader perspective see [50] or [41]. This opens up new possibilities to change the current mostly ‘linear’ and profit-driven way of organizing the economy towards a more circular, democratic and sustainable mode of production (see e.g. [25]).

## References

1. Heinz, Ernst A., *Scalable search in computer chess*, Vieweg, 2000
2. Ernst A. Heinz: *How DarkThought plays chess*, <http://supertech.lcs.mit.edu/~heinz/dt/node2.html>
3. Frey, P.W. (editor), *Chess skill in man and machine*, Springer, 2nd edition 1983
4. Plaat, Aske: RESEARCH, RE:SEARCH & RE-SEARCH, Tinbergen Institute, 1996
5. FUSc#-Homepage: <http://www.fuschmotor.de.vu>
6. Download of DarkFUSc# and FUSc#: [http://page.mi.fu-berlin.de/~fusch/download/download\\_de.html](http://page.mi.fu-berlin.de/~fusch/download/download_de.html)
7. Block, Marco, *Verwendung von Temporale-Differenz-Methoden im Schachmotor FUSc#*, Diplomarbeit, Berlin, 2004
8. Dill, Sebastian, *Transpositionstabellen*, Seminararbeit, Berlin, 2005
9. Crafty-homepage <http://www.cis.uab.edu/info/faculty/hyatt/hyatt.html>
10. Homepage of Peter McKenzie on Computer chess: <http://homepages.caverock.net.nz/~peter/perft.htm>
11. Homepage Microsoft .NET: <http://www.microsoft.com/net/default.msp>
12. Homepage of gcc: <http://gcc.gnu.org/>
13. D. Silver et al: "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm", arXiv:1712.01815v1
14. Marco Block et al: "Using Reinforcement Learning in Chess Engines" (2008), <https://page.mi.fu-berlin.de/block/concibe2008.pdf>
15. Nathan J. Robinson: "The The Problem With AI Is the Problem With Capitalism", <https://jacobin.com/2023/03/ai-artificial-intelligence-art-chatgpt-jobs-capitalism>
16. Samothrakis, Spyridon (2021) "Artificial Intelligence inspired methods for the allocation of common goods and services", <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0257399>
17. Samothrakis, Spyridon (2024) Artificial intelligence and modern planned economies: a discussion on methods and institutions. *AI and Society*, 39 (6). pp. 2961-2972. DOI <https://doi.org/10.1007/s00146-023-01826-7>
18. Homepage Announcement of AlphaTensor: <https://deepmind.google/discover/blog/discovering-novel-algorithms-with-alphatensor/>
19. Wikipedia on Alpha Fold: <https://en.wikipedia.org/wiki/AlphaFold>
20. Homepage Announcement of Alpha Proof and Geometry: <https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/>
21. Buchner, Johannes: Critical Mathematical Economics and Progressive Data Science arXiv:2502.06015
22. J. Buchner (2025): *Rotated Bitboards in FUSc# and Reinforcement Learning in Computer Chess and Beyond*, <https://arxiv.org/abs/2503.10822v1>.
23. Absolute Sustainability: <https://www.sciencedirect.com/science/article/pii/S0007850620301372>
24. Life Cycle Sustainability Assessment for Sustainable Bioeconomy, Societal-Ecological Transformation and Beyond, [https://link.springer.com/chapter/10.1007/978-3-031-29294-1\\_8](https://link.springer.com/chapter/10.1007/978-3-031-29294-1_8)
25. Heyer, J., Zeug, W. (2024): *Prokla-Zeitschrift für kritische Sozialwissenschaft* 54 (215), 267 - 286 [10.32387/prokla.v54i215.2116](https://doi.org/10.32387/prokla.v54i215.2116)
26. Planetary Boundaries Rockström et al. (2009) A Safe operating space for humanity nature 461:472. <https://doi.org/10.1038/461472a>
27. Exceeding 1.5 °C global warming could trigger multiple climate tipping points, <https://www.science.org/doi/10.1126/science.abn7950>
28. Bundesministerium für Umwelt, Naturschutz, nukleare Sicherheit und Verbraucherschutz (2024): Nationale Kreislaufwirtschaftsstrategie, <https://www.bmuv.de/download/nationale-kreislaufwirtschaftsstrategie-nkws>
29. Künstliche Intelligenz für die Circular Economy, [https://prosperkolleg.ruhr/wp-content/uploads/2025/02/PK\\_PROSPEKTIVEN\\_PK\\_PROSPEKTIVEN\\_KI-20251.pdf](https://prosperkolleg.ruhr/wp-content/uploads/2025/02/PK_PROSPEKTIVEN_PK_PROSPEKTIVEN_KI-20251.pdf)
30. Phillips, L. and Rozworski, M. (2019): *The People's Republic of Walmart: How the World's Biggest Corporations are Laying the Foundation for Socialism*, Verso Books, London. <https://www.science.org/doi/10.1126/science.abn7950>
31. <https://arxiv.org/abs/2009.13055>
32. Qin, Haotong and Gong, Ruihao and Liu, Xianglong and Shen, Mingzhu and Wei, Ziran and Yu, Fengwei and Song, Jingkuan (2020), Forward and Backward Information Retention for Accurate Binary Neural Networks, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, p.2250–2259
33. Das Schachprojekt FUSc#, <https://refubium.fu-berlin.de/handle/fub188/18583>



34. [https://www.chessprogramming.org/Rotated\\_Bitboards](https://www.chessprogramming.org/Rotated_Bitboards)
35. [https://www.chessprogramming.org/Reinforcement\\_Learning](https://www.chessprogramming.org/Reinforcement_Learning)
36. Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development. 1959.
37. D. Silver et al: "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm", arXiv:1712.01815v1
38. Marco Block et al: "Using Reinforcement Learning in Chess Engines" (2008), <https://page.mi.fu-berlin.de/block/concibe2008.pdf>
39. Nathan J. Robinson: "The The Problem With AI Is the Problem With Capitalism", <https://jacobin.com/2023/03/ai-artificial-intelligence-art-chatgpt-jobs-capitalism>
40. Samothrakis, Spyridon (2021) "Artificial Intelligence inspired methods for the allocation of common goods and services", <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0257399>
41. Samothrakis, Spyridon (2024) Artificial intelligence and modern planned economies: a discussion on methods and institutions. AI and Society, 39 (6). pp. 2961-2972. DOI <https://doi.org/10.1007/s00146-023-01826-7>
42. Homepage Announcement of AlphaTensor: <https://deepmind.google/discover/blog/discovering-novel-algorithms-with-alphatensor/>
43. Wikipedia on Alpha Fold: <https://en.wikipedia.org/wiki/AlphaFold>
44. Homepage Announcement of Alpha Proof and Geometry: <https://deepmind.google/discover/blog/ai-solves-imo-problems-at-silver-medal-level/>
45. Buchner, Johannes: Critical Mathematical Economics and Progressive Data Science arXiv:2502.06015
46. [https://www.pik-potsdam.de/en/institute/futurelabs-science-units/ersu/news/copy3\\_of\\_2021-january-26-29](https://www.pik-potsdam.de/en/institute/futurelabs-science-units/ersu/news/copy3_of_2021-january-26-29)
47. European Commission (2019), *The European Green Deal*, COM (2019) 640 Final; European Commission: Brussels, Belgium.
48. European Commission (2020), *Circular Economy Action Plan. For a Cleaner and More Competitive Europe*, COM(2020) 98 final; European Commission: Brussels, Belgium.
49. European Commission (2022), *Proposal for Ecodesign for Sustainable Products Regulation*, COM(2022) 142 final; European Commission: Brussels, Belgium.
50. J. Groos, C. Sorg (Eds.) (2025): *Creative Construction - Democratic Planning in the 21st Century and Beyond*, Bristol University Press.