

**CAPSTONE PROJECT 1**  
**Planning Document**

**Evaluation of Nature-inspired Optimisation  
Algorithms in Solving Versus Tetris**

by

Yap Wei Xiang  
21067939

Supervisor: Dr Richard Wong Teck Ken

Semester: April 2024

Date:

# Abstract

# Contents

<b>Abstract</b>	<b>i</b>
<b>List of Tables</b>	<b>iii</b>
<b>List of Figures</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem Statement . . . . .	2
1.3 Aim . . . . .	3
1.4 Objectives . . . . .	3
1.5 Project Scope . . . . .	3
<b>2 Literature Review</b>	<b>4</b>
2.1 The Difficulty of Tetris . . . . .	4
2.1.1 Complexity Classes . . . . .	4
2.1.2 Justifying Non-traditional Algorithmic Approaches . . . . .	6
2.2 Some Approaches Taken to Solve Tetris . . . . .	6
2.2.1 Learning by Imitation . . . . .	6
2.2.2 Upper Confidence Bounds for Trees . . . . .	8
2.2.3 Reinforcement Learning . . . . .	9
2.2.4 Meta-heuristic Algorithms . . . . .	9
2.3 Playing Tetris with Nature-inspired Algorithms . . . . .	9
<b>3 Technical Plan</b>	<b>10</b>
3.1 Defining the Rules . . . . .	10
<b>4 Work Plan</b>	<b>11</b>
<b>5 References</b>	<b>12</b>

# List of Tables

# List of Figures

1.1	A typical modern Tetris game where four lines are about to be cleared. The Tetrimino on the left of the matrix is the <i>Hold</i> piece and the pieces to the right of the matrix are collectively known as the <i>Queue</i> . . . . .	1
1.2	A typical game of Versus Tetris. Both players are trying to send lines to each other. The grey blocks are <i>Garbage Lines</i> sent from Player 2 (right) to Player 1 (left). . . . .	2
2.1	Visualisation of the sets P, NP, NP-hard and NP-complete. . . . .	5
2.2	Abstraction of Zhang, Cai, and Nebel's System Components . . . . .	7
2.3	An example of a single node of the matrix state and piece in the planning tree. . . . .	8

# 1 Introduction

Tetris is a popular video game created in 1984 by computer programmer Alexey Pajitnov [1]. It is a puzzle game that requires players to strategically place sequences of pieces known as "Tetriminos" into a rectangular Matrix (refer to Figure 1.1). In the classic game, players attempt to clear as many lines as possible by completely filling horizontal rows of blocks, but if the Tetriminos surpass the top of the Matrix, the game ends.

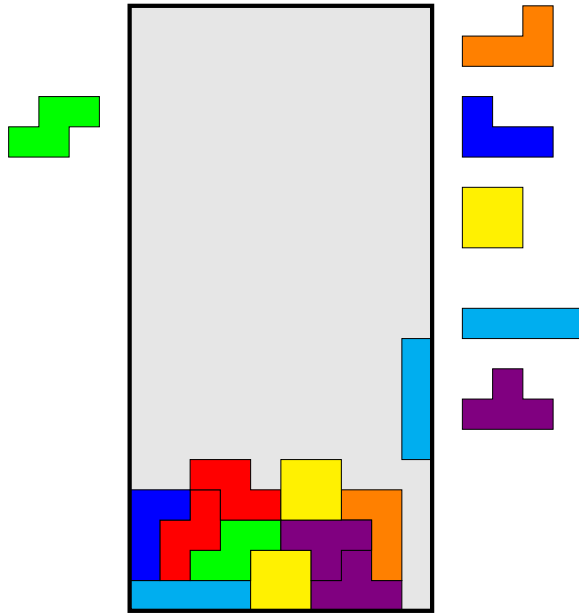


Figure 1.1: A typical modern Tetris game where four lines are about to be cleared. The Tetrimino on the left of the matrix is the *Hold* piece and the pieces to the right of the matrix are collectively known as the *Queue*.

Since its release, mathematicians and computer scientists have been intrigued by the game of Tetris, leading to a diverse array of research endeavours exploring the various facets of the game, including its computational complexity [2], and its possibility of being won [3] [4].

## 1.1 Motivation

In their paper, Demaine, Hohenberger, and Liben-Nowell showed that it is NP-complete to optimise several natural objective functions of Tetris [2]. NP-completeness poses a significant challenge in computational problem-solving, as it denotes the absence of polynomial-time algorithms for efficient solutions [5]. Moreover, the discovery of a polynomial-time algorithm for any NP-complete problem implies that any problem in the set of NP, encompassing efficiently verifiable but potentially difficult problems, could be solved in

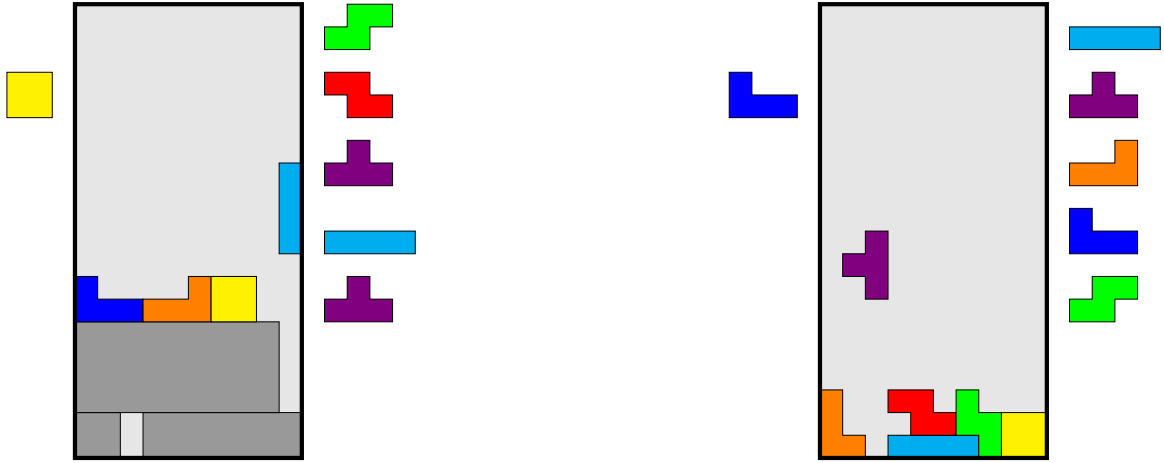


Figure 1.2: A typical game of Versus Tetris. Both players are trying to send lines to each other. The grey blocks are *Garbage Lines* sent from Player 2 (right) to Player 1 (left).

polynomial time [5]. NP-completeness extends beyond Tetris, with real-life instances of NP-complete arising in diverse fields such as route optimisation [6], job scheduling [7], and medicine [8].

To address these challenges, researchers have explored alternative approaches to tackle NP-complete problems, including the use of nature-inspired algorithms [9]. Although they might fail at finding optimal solutions, nature-inspired algorithms are able to return acceptable solutions in shorter running times [10]. In the context of optimising Tetris gameplay, studies have shown the effectiveness of using nature-inspired algorithms in playing the classic single-player game [11] [12]. However, there remains limited research on the effectiveness of nature-inspired optimisation algorithms in the multiplayer versus variant of the game.

## 1.2 Problem Statement

*Versus Tetris* (refer to Figure 1.2) presents a unique challenge in computational gaming due to its complex dynamics and real-time competitive nature. While previous research regarding the use of nature-inspired algorithms for Tetris optimisation have focused on single-player scenarios, the effectiveness of these algorithms in the multiplayer context remains largely unexplored. Despite the demonstrated success of these algorithms in improving single-player Tetris gameplay, their application to the multiplayer variant poses distinct challenges due to a different rule set and differing objectives that require further investigation.

## 1.3 Aim

The aim of this capstone project is to assess the effectiveness of nature-inspired optimisation algorithms in solving the game of Versus Tetris. By integrating insights from nature-inspired algorithms, the project seeks to create a robust and adaptable Tetris-playing software capable of competing against human players or other Tetris-playing programs. Through this endeavour, the project aims to contribute valuable insights into the application of nature-inspired algorithms in addressing computationally complex problems.

## 1.4 Objectives

The objectives of this project are as follows:

1. Formulate the problem of Versus Tetris for game AI.
2. Research and implement a variety of nature-inspired optimisation algorithms to determine their suitability for optimising gameplay strategies in Versus Tetris.
3. Design a comprehensive framework for objectively evaluating and comparing the performance of the algorithms.
4. Develop a playable game of Tetris that simulates gameplay and training.
5. Using the game, do comparative analyses with the designed framework to assess the effectiveness and efficiency of each algorithm.
6. Summarize findings from the comparative analyses.
7. Share the software with Tetris players of varying aptitudes to find the level of play for each algorithm.

## 1.5 Project Scope

This project will focus specifically on the evaluation of nature-inspired optimisation algorithms in the context of multiplayer versus Tetris. It will entail the development of a playable Tetris game capable of simulating gameplay and the training of algorithms. This simulation environment will facilitate in the analysis and evaluation of these algorithms' performances. The scope includes the exploring of a range of nature-inspired algorithms to address the unique challenges inherent in Versus Tetris.



## 2 Literature Review

### 2.1 The Difficulty of Tetris

In their article, Demaine, Hohenberger, and Liben-Nowell [2] proved that optimising several natural objectives of Tetris is NP-complete, even with a deterministic finite piece sequence. A deterministic finite piece sequence refers to a game at which the player knows every single piece in the sequence, and where there are finite pieces in the sequence, i.e. the game can end without the player losing. The authors defined the natural objectives of the game as follows [2]:

1. maximising the number of rows cleared;
2. maximising the number of piece placed;
3. maximising the number of Tetrises - clearing four lines on the same move;
4. minimising the height of the highest filled grid square.

In 2020, Asif, Coulombe, Demaine, *et al.* [13] demonstrated that playing any game of Tetris with a matrix of eight or more columns, or four or more rows, is NP-complete, further showcasing the difficulty of the game. Both of these papers aim to highlight the difficulty of the game, but what does difficulty actually entail?

This section aims to elucidate some of the key concepts of computational complexity, which involves the study of intrinsic difficulties of computational problems [14], and attempt to justify the use of non-traditional algorithmic approaches to play the game.

#### 2.1.1 Complexity Classes

The study of computational complexity asks questions about the intrinsic difficulty of computational problems [14]. Complexity classes are usually defined by referring to computation models and by putting suitable restrictions on them **uniform-cc**.

The class  $P$  encompasses all decision problems that are polynomial time solvable using a deterministic model of computation [15]. In this model, for any given input, the machine's computation follows a single predetermined path [5].

The complexity class  $NP$ , on the other hand is the class of all decision problems that can be solved in polynomial time by a nondeterministic algorithm [16]. The nondeterministic model of computation allows for guessing correct solutions out of polynomially many options in constant time [17], and solutions can be verified in deterministic polynomial

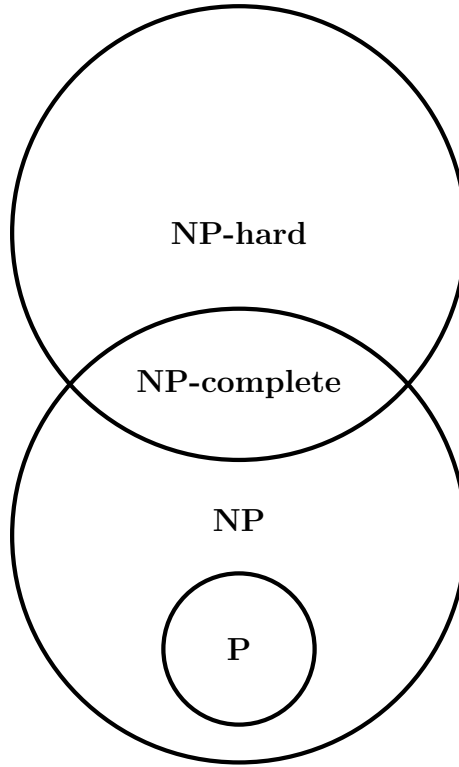


Figure 2.1: Visualisation of the sets  $P$ ,  $NP$ ,  $NP$ -hard and  $NP$ -complete.

time [5].

If all problems in  $NP$  can be reduced to some problem  $X$ ,  $X$  is said to be *NP-hard* [5]. Reductions are useful in showing the relationship between computable problems, as a reduction from problem  $A$  to problem  $B$  tells us that problem  $B$  is at least as hard as problem  $A$  [17].

For a problem to be considered *NP-complete*, it must be a member of both the  $NP$  and *NP-hard* classes (refer to Figure 2.1) [17]. Therefore, *NP-complete* problems are some of the hardest problems in  $NP$  [15].

Since all problems in  $NP$  can be reduced to any *NP-complete* problem, if a deterministic polynomial time algorithm can be found for an *NP-complete* problem, it would also mean that the set  $NP$  is polynomial-time solvable, proving  $NP$  and  $P$  are equal sets. The question of whether  $P = NP$  has famously been immortalised as one of the millennium prize problems from the Clay Institute of Mathematics [18]. Most researchers believe that  $P \neq NP$  since years of effort have failed to yield efficient algorithms for *NP-complete* problems [15].

### 2.1.2 Justifying Non-traditional Algorithmic Approaches

It is important to note that even though Demaine, Hohenberger, and Liben-Nowell’s [2] proof is based on a variant of Tetris that is quite different from Versus Tetris, the difficulty of Versus Tetris can be implied from the proof. Versus Tetris introduces additional gameplay factors, such as the competitive aspect where players can send “garbage” lines to their opponents, adding layers of complexity. As such, Versus Tetris is arguably more difficult than its classic variant, or Demaine, Hohenberger, and Liben-Nowell’s variant [2].

Given the NP-completeness of Tetris [2] [13], it is evident that traditional deterministic algorithmic approaches would face significant challenges in efficiently solving the game [14]. The NP-completeness of the game implies that obtaining exact solutions are computationally intractable within a reasonable time frame.

To address this, we may relax the requirements for a solution. This can be done by either broadening the set of acceptable solutions or focusing on average-case scenarios instead of worst-case scenarios [14]. This shift in focus allows the use of more practical, non-traditional algorithms that provide acceptable solutions within a reasonable time, even if they are not optimal.

## 2.2 Some Approaches Taken to Solve Tetris

Now that the difficulty of Tetris has been established, we turn our attention to potential strategies that can be used to handle the game’s complexity. In this section, we explore some of the innovative approaches that have been utilised to tackle the game of Tetris.

### 2.2.1 Learning by Imitation

In 2010, Zhang, Cai, and Nebel [19] employed a learning by imitation approach to the game of Tetris. Imitation learning aims to mimic human behaviour in some given task by learning a mapping between observations of demonstrations [20].

In their approach, played Tetris games were fed into a machine learning model as inputs, where the model would receive positive feedback when a decision matched that of the imitated system, and negative feedback otherwise [19]. The learning is deemed successful if the trained model maintains similarity even when faced with data outside of the training set.

The learning system consists of several components, including filters, pattern calculators, and support vector machines (SVMs) (refer to Figure 2.2) [19]. Because of the time consuming nature of training SVMs, each Tetrimino had its own dedicated filters,

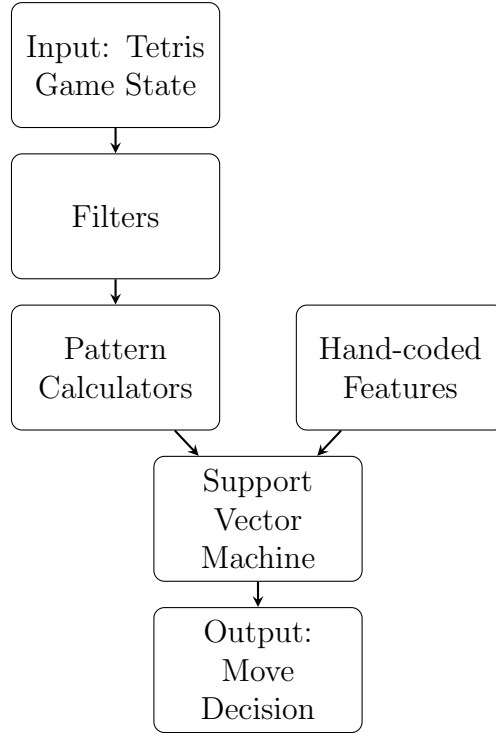


Figure 2.2: Abstraction of Zhang, Cai, and Nebel’s System Components

pattern calculators and SVMs, i.e. seven SVMs <sup>1</sup> are working together in the artificial player. The output of any particular SVM describes how similar a candidate move is to the choice of the imitated player.

Filters are made up of a multitude of patterns, which can be thought of as specific configurations in the playing field that the model uses to recognise and evaluate different game states [19]. Patterns do not take into account the whole matrix, but only a small area in the playing field. Thus, the authors included a list of 19 hand-coded features that takes a more global look at the game, some notable features include the number of holes that will be created after the current placement, and the number of removed lines after the current placement. Both the activated patterns and the features are passed into the support vector machine, and the move that is calculated to be the most similar is played.

To evaluate their approach, Zhang, Cai, and Nebel used a human player as well as Fahey’s artificial player [21] as systems for their model to imitate [19]. The outcome shows that their model can learn different styles of gameplay based on the imitated player, which is to be expected. When pitting the two learned models against each other in their variant of multiplayer Tetris <sup>2</sup>, they found that the human imitated machine performed better than the one imitated from Fahey’s AI. However, The system imitated

<sup>1</sup>Seven SVMs for the seven Tetriminos.

<sup>2</sup>In Zhang, Cai, and Nebel’s variant of multiplayer Tetris, clearing  $n$  lines sends  $n - 1$  attack rows to the opponent, with each attack row containing  $n - 1$  empty cells [19]. Attack rows are added to the bottom of the opponent’s matrix.

from Fahey’s AI outperformed that of the human imitated system in single player games.

### 2.2.2 Upper Confidence Bounds for Trees

Monte-Carlo Tree Search (MCTS) is a best-first search method that is based on a randomised exploration of the search space [22]. The algorithm evaluates states to determine the most rewarding actions, with rewards being calculated from the leaf to the root [23]. MCTS has the added benefit of learning from previous explorations, gradually building a game tree in memory, resulting in more accurate estimation of the most promising moves [22]. In 2006, Kocsis and Szepesvári [24] proposed a method, which was deemed Upper Confidence Bound for Trees (UCT), to improve the performance of the algorithm by applying a bandit algorithm known as UCB1<sup>3</sup> to Monte-Carlo Planning.

The UCB1 formula is employed to balance the exploration-exploitation trade-offs during the search [23]. It helps prioritise actions that are either promising or under-explored, which effectively prunes less promising branches of the search tree. This results in a more efficient search process and a smaller, more focused tree.

In 2011, Cai, Zhang, and Nebel [23] employed this approach to the game of Tetris. In their approach, the authors considered each state of the matrix, along with a given piece as a single node in the planning tree (refer to Figure 2.3). After an action is taken, if any number of rows are cleared, a predefined reward is given to the action and the most rewarding action is selected as the resulting move. The authors also pruned the tree to remove actions that resulted in holes in the field, to further improve efficiency.

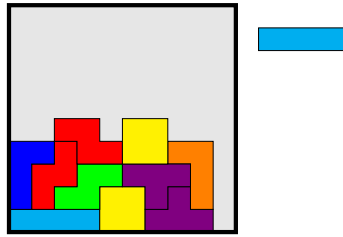


Figure 2.3: An example of a single node of the matrix state and piece in the planning tree.

Experimental results show that the artificial player shows promise, beating Fahey’s AI [21] 91 times out of 100 games using the multiplayer Tetris rules mentioned in subsection 2.2.1 [23]. The results also show that the artificial player outperformed the SVM based pattern player from Cai, Zhang, and Nebel’s previous work, where they approached the game with imitation learning [19].

---

<sup>3</sup>UCB stands for Upper Confidence Bound

**2.2.3 Reinforcement Learning**

**2.2.4 Meta-heuristic Algorithms**

**2.3 Playing Tetris with Nature-inspired Algorithms**

## 3 Technical Plan

### 3.1 Defining the Rules

## 4 Work Plan



## 5 References

- [1] Tetris Inc., *About Tetris*, <https://tetris.com/about-us>, [accessed Apr. 22, 2024].
- [2] E. D. Demaine, S. Hohenberger, and D. Liben-Nowell, “Tetris is hard, even to approximate,” in *Computing and Combinatorics*, T. Warnow and B. Zhu, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 351–363, ISBN: 978-3-540-45071-9.
- [3] J. Brzustowski, “Can you win at tetris?” Master’s Thesis, University of Waterloo, 200 University Ave W, Waterloo, ON N2L 3G1, Canada, 1988.
- [4] H. Burgiel, “How to lose at tetris,” *The Mathematical Gazette*, vol. 81, no. 491, pp. 194–200, 1997. DOI: 10.2307/3619195.
- [5] M. Sipser, in *Introduction to the Theory of Computation*, Cengage Learning, 2013.
- [6] V. Lesch, M. König, S. Kounev, A. Stein, and C. Krupitzer, “A case study of vehicle route optimization,” *CoRR*, vol. abs/2111.09087, 2021.
- [7] J. D. Ullman, “Np-complete scheduling problems,” *Journal of Computer and System sciences*, vol. 10, no. 3, pp. 384–393, 1975.
- [8] J. Arle and K. Carlson, “Medical diagnosis and treatment is np-complete,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 33, pp. 1–16, Mar. 2020. DOI: 10.1080/0952813X.2020.1737581.
- [9] L. Davis, “Job shop scheduling with genetic algorithms,” in *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 1985, pp. 136–140.
- [10] W. Korani and M. Mouhoub, “Review on nature-inspired algorithms,” *Operations Research Forum*, vol. 2, Jul. 2021. DOI: 10.1007/s43069-021-00068-x.
- [11] J. Lewis, “Playing tetris with genetic algorithms,” 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:17416568>.
- [12] L. Langenhoven, W. S. van Heerden, and A. P. Engelbrecht, “Swarm tetris: Applying particle swarm optimization to tetris,” in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–8. DOI: 10.1109/CEC.2010.5586033.
- [13] S. Asif, M. Coulombe, E. Demaine, *et al.*, “Tetris is np-hard even with  $o(1)$  rows or columns,” *Journal of Information Processing*, vol. 28, pp. 942–958, Jan. 2020. DOI: 10.2197/ipsjjip.28.942.

- [14] O. Goldreich, “Computational complexity: A conceptual perspective,” *SIGACT News*, vol. 39, no. 3, pp. 35–39, Sep. 2008, ISSN: 0163-5700. DOI: 10.1145/1412700.1412710. [Online]. Available: <https://doi.org/10.1145/1412700.1412710>.
- [15] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009, ISBN: 9780521424264.
- [16] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, First Edition. W. H. Freeman, 1979, ISBN: 0716710455.
- [17] MIT OpenCourseWare. “Np-completeness.” 6.046J: Design and Analysis on Algorithms, Massachusetts Institute of Technology. (2015), [Online]. Available: [https://ocw.mit.edu/courses/6-046j-design-and-analysis-of-algorithms-spring-2015/resources/mit6\\_046js15\\_lec16/](https://ocw.mit.edu/courses/6-046j-design-and-analysis-of-algorithms-spring-2015/resources/mit6_046js15_lec16/).
- [18] *The Millennium Prize Problems - Clay Mathematics Institute — claymath.org*, <https://www.claymath.org/millennium-problems/>, [Accessed 15-05-2024].
- [19] D. Zhang, Z. Cai, and B. Nebel, “Playing tetris using learning by imitation,” 2010. [Online]. Available: <https://api.semanticscholar.org/CorpusID:10668262>.
- [20] A. Hussein, M. Gaber, E. Elyan, and C. Jayne, “Imitation learning: A survey of learning methods,” English, *ACM Computing Surveys*, vol. 50, no. 2, Apr. 2017, ISSN: 0360-0300. DOI: 10.1145/3071073.
- [21] *Tetris AI*, [accessed Jun. 23, 2024], 2003. [Online]. Available: <http://www.colinfahey.com/tetris/>.
- [22] G. Chaslot, “Monte-carlo tree search,” English, Ph.D. dissertation, Maastricht University, 2010, ISBN: 9789085590996. DOI: 10.26481/dis.20100930gc.
- [23] Z. Cai, D. Zhang, and B. Nebel, “Playing tetris using bandit-based monte-carlo planning,” Jan. 2011.
- [24] L. Kocsis and C. Szepesvári, “Bandit based monte-carlo planning,” in *Machine Learning: ECML 2006*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293, ISBN: 978-3-540-46056-5.