Why Most Decisions Are Easy in Tetris—And Perhaps in Other Sequential Decision Problems, As Well

Özgür Şimşek OZGUR@MPIB-BERLIN.MPG.DE
Simón Algorta ALGORTA@MPIB-BERLIN.MPG.DE
Amit Kothiyal KOTHIYAL@MPIB-BERLIN.MPG.DE
Center for Adaptive Behavior and Cognition
Max Planck Institute for Human Development
Lentzeallee 94, 14195 Berlin, Germany

Abstract

We examined the sequence of decision problems that are encountered in the game of Tetris and found that most of the problems are easy in the following sense: One can choose well among the available actions without knowing an evaluation function that scores well in the game. This is a consequence of three conditions that are prevalent in the game: simple dominance, cumulative dominance, and noncompensation. These conditions can be exploited to develop faster and more effective learning algorithms. In addition, they allow certain types of domain knowledge to be incorporated with ease into a learning algorithm. Among the sequential decision problems we encounter, it is unlikely that Tetris is unique or rare in having these properties.

1. Introduction

Many problems humans encounter are sequential in nature, where a successful outcome depends not on a single decision but on a series of related decisions. Furthermore, typically there is uncertainty regarding the consequences of decisions. These types of problems, known as *sequential decision problems under uncertainty*, remain challenging for machine learning despite successful applications in large, complex domains such as the game of Go (Silver et al., 2016).

One means to developing faster, more effective learning algorithms is to study the natural environments in which sequential decision problems are encountered. It is well known that many aspects of the natural world have regularities. For example, networks observed in nature are very different from random networks. They exhibit homophily (the tendency of connected nodes to have correlated attributes) and they tend to have some nodes with a very high degree. Both of these properties have informed the development of algorithms that can search natural networks much more effectively than they search random networks (Kleinberg, 2000; Watts et al., 2002; Adamic et al., 2001; Şimşek & Jensen, 2008). It is possible that natural sequential-decision environments similarly exhibit regularities that can inform the development of learning algorithms.

We examined one particular environment in depth: Tetris, one of the most well-known and liked video games of all time. Our interest is not in Tetris per se but in using it as a platform for identifying regularities in sequential-decision environments.

We found that most individual problems encountered while playing Tetris are easy in the following sense: One can choose well among the available actions without knowing an evaluation function that scores well in the game. Furthermore, there is an intuitive feature that typically eliminates a substantial number of actions without eliminating the best action. The underlying reasons are three types of regularities in the domain, which we explain in detail in the following sections.

Our analysis is motivated by earlier results in the decision-making literature on single-shot problems. In the following sections, we review this literature and describe our results in Tetris. Our analysis leads to two immediate questions for further research. The first is whether Tetris is unique or rare in having these regularities. The second is how these regularities can be exploited for faster, more effective learning. We discuss both questions in some detail, presenting additional results from the game of backgammon.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

2. Background

Consider the following judgment problem: You are given a number of objects and are asked to determine which object has the highest value on a specified (unobserved) criterion. For example, you may be given a number of stocks and asked to determine which stock will have a higher rate of return in five years, given a number of features on each stock, for example, the type of industry and the return rate in the last 12 months. This problem is called *comparison*.

There are three conditions under which the comparison problem becomes easy if the relationship between the features and the criterion is linear. When these conditions hold, one can decide correctly without knowing the exact linear relationship between the features and the criterion. It suffices to know only the sign of the weights, and in two of the conditions, also the order in which the weights decrease. These conditions are simple dominance (Hogarth & Karelaia, 2006), cumulative dominance (Baucells et al., 2008), and noncompensation (Martignon & Hoffrage, 2002; Katsikopoulos, 2011; Şimşek, 2013).

These conditions were found to be prevalent in a large, diverse collection of natural data sets (Şimşek, 2013). In 51 data sets examined in this study, on average, simple dominance held in 16%, cumulative dominance in 62%, and noncompensation in 85% of the paired comparisons between objects. Furthermore, with principled, probabilistic approximations to dominance, the prevalence of simple and cumulative dominance increased substantially.

Below we explain the three conditions and give a simple illustrative example. Let $y = w_0 + w_1x_1 + w_2x_2 + ... + w_kx_k$ denote the linear relationship between the features and the criterion, where y is the criterion, w_0 is the intercept, $w_1, ..., w_k$ are the weights, and $x_1, ..., x_k$ are the feature values. Let x_i^A denote the value of the ith feature for object A. We assume, for ease of exposition and without loss of generality, that all weights are positive and that the weights are ordered in decreasing magnitude.

2.1. Simple dominance

Object A is said to dominate object B if $x_i^A \geq x_i^B, \forall i$, and $\exists i$ such that $x_i^A > x_i^B$. If object A dominates object B, one can be certain that A has the higher criterion value. We refer to dominance as *simple dominance* to differentiate it from cumulative dominance, discussed next.

2.2. Cumulative dominance

Let $z_i = \sum_{j=1}^{i} x_j, \forall i$, denote the cumulative profile of an object. Object A is said to cumulatively dominate object B if $z_i^A \geq z_i^B, \forall i$, and $\exists i$ such that $z_i^A > z_i^A$. If object A cumulatively dominates object B, one can be certain that A has the higher criterion value. To check for cumulative dominance, one needs to know the order in which feature weights decrease, which we refer to as *feature order*.

Simple dominance implies cumulative dominance. Simple and cumulative dominance are both transitive: Given three objects A, B, and C, if A dominates B and B dominates C, then A dominates C.

2.3. Noncompensation

Consider the following lexicographic decision rule: Order the features in decreasing magnitude of their weights; identify the first feature that discriminates[1] between the objects; choose the object whose value on this feature is higher. If this decision rule decides correctly, we say that the decision problem exhibits noncompensation.

Noncompensation holds with probability 1 if the features are binary, taking values of 0 or 1, and the weights satisfy the set of constraints $w_i > \sum_{j=i+1}^{k} w_j, i = 1, 2, ..., k-1$. Such weights are called *noncompensatory*. An example is the sequence $1, 0.5, 0.25, 0.125$. If features are numeric, noncompensation may or may not hold, depending on the weights of the linear function and the feature values of the objects being compared.

Both simple and cumulative dominance imply noncompensation.

2.4. An illustrative example

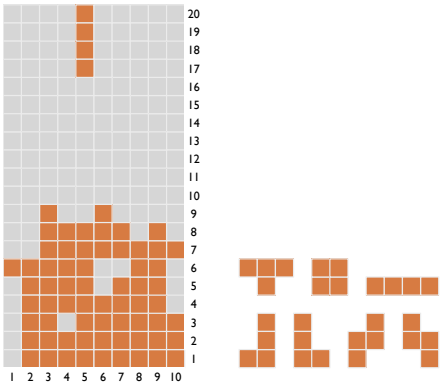
As an illustrative example, consider a pile of U.S. coins, where $\langle x_{\$1}, x_{50¢}, x_{25¢}, x_{10¢}, x_{5¢}, x_{1¢} \rangle$ denotes the number of each coin type in the pile. The dollar value of the pile is described by the linear function $x_{\$1} + 0.5x_{50¢} + 0.25x_{25¢} + 0.1x_{10¢} + 0.05x_{5¢} + 0.01x_{1¢}$.

Pile $\langle 4, 4, 4, 0, 0, 0 \rangle$ simply dominates pile $\langle 3, 4, 2, 0, 0, 0 \rangle$. Even if the dollar value of the coin types are not known, the first pile can be identified as the one with the higher value.

Pile $\langle 6, 4, 2, 0, 0, 0 \rangle$ cumulatively dominates (but does not simply dominate) pile $\langle 3, 5, 1, 0, 0, 0 \rangle$. If the dollar value of each coin type is unknown but the coins can be ranked from most to least valuable, the first pile can be identified as the one with the higher value.

Pile $\langle 4, 4, 4, 0, 0, 0 \rangle$ neither simply nor cumulatively dominates pile $\langle 2, 8, 2, 0, 0, 0 \rangle$. A lexicographic decision rule selects the first pile because it contains more of the highest-valued coin type. It would be correct in this particular case but is not guaranteed to be correct in general—unless each pile has at most one of each coin type (because the weights are noncompensatory).

[1] A feature discriminates between two objects if its value differs on the two objects.



ID	Feature	Weight	Sample value
1	Rows with holes	−24.04	6
2	Column transitions	−19.77	20
3	Holes	−13.08	12
4	Landing height	−12.63	10.5
5	Cumulative wells	−10.49	26
6	Row transitions	−9.22	56
7	Eroded piece cells	+6.60	0
8	Hole depth	−1.61	12

Figure 1. Sample board during a game of Tetris, showing a tetrimino falling from the top of the grid; the seven possible tetriminos; and a table showing the features used by BCTS, their weight in the linear evaluation function, and their value on the sample board if the falling tetrimino is allowed to drop without intervention. The features are described in the text.

3. Objective

Sequential decision problems can be viewed as a series of comparison problems, albeit ones that have complex dependencies. At each decision stage, the agent faces the following comparison problem: Among the actions that are available, choose the one with the highest value. Consequently, the same type of regularities that make judgment problems easy can potentially make sequential decision problems easy as well, if learning algorithms are tailored to exploit them. Our objective is to explore this possibility.

4. Method

We examine Tetris, one of the most well-known and liked video games of all time. The game is played on a two-dimensional grid, initially empty. Pieces of different shapes fall from the top of the grid, one at a time, piling up on each other. As each piece falls, the player controls where and how the piece lands by rotating the piece and moving it horizontally, to the left or to the right, any number of times. When a row becomes entirely full, the row is deleted, creating additional space on the grid. The game ends when there is no space at the top of the grid for the next piece. The standard grid is 20 cells high and 10 cells wide. The pieces, called *tetriminos*, each occupy four cells and have seven different shapes. Figure 1 shows the standard board during a game and the seven tetriminos.

It is known that there are tetrimino sequences that terminate the game, no matter how well they are placed (Burgiel, 1997). Finding the optimal placement of tetriminos is NP-complete even if the whole sequence of tetriminos is known in advance (Demaine et al., 2003). Tetris has approximately 1.6×10^{60} states.

Artificial players can learn to play very well, removing on average hundreds of thousands of rows. The best known player is BCTS (which stands for Building Controllers for Tetris Systems), a controller developed by Thiery & Scherrer (2009), who won the 2008 Reinforcement Learning Competition using a variant of BCTS. The controller was developed using the cross-entropy algorithm, following earlier work by Szita & Lörincz (2006). More recently, reinforcement learning methods have succeeded in learning faster but not in learning a better policy (Gabillon et al., 2013; Scherrer et al., 2015).

BCTS uses a linear evaluation function with eight features. For each tetrimino falling from the top, the controller evaluates all possible legal placements on the board using this linear evaluation function and selects the placement with the highest value. The features are described below and listed in Figure 1, along with their weights and values on the sample board if the falling tetrimino is allowed to drop without intervention.

Two of the key concepts used in the features are holes and wells. A *hole* is an empty cell that has one or more full cells placed higher than itself in the same column. A *well* is a succession of empty cells in a column such that the immediate cells on the left and the right are full (where the outside of the grid is assumed to be full).

Rows with holes is the number of rows that have at least one hole.

Column transitions are determined by examining each column from one end to the other and counting how many times there is a transition from a full cell to an empty cell or the reverse. It is assumed that the outside of the grid is full at the bottom but empty at the top.

Holes is the number of holes.

Landing height equals $(y_1 + y_2)/2$, where y_1 and y_2 are the lowest and the highest cell height, respectively,

occupied by the current tetrimino, before any rows are cleared.

Cumulative wells is the cumulative depth of the cells in a well, summed for all wells. For example, for a well of depth 3, the cumulative depth of its cells is $1 + 2 + 3 = 6$.

Row transitions are determined by examining each row from one end to the other, counting how many times there is a transition from a full cell to an empty cell or the reverse. It is assumed that the outside of the grid is full (both on the left and on the right).

Eroded piece cells is the number of lines cleared by the placement multiplied by the number of cells of the current piece cleared along with the line(s). The maximum value of this feature is 16, observed when four lines are cleared at once.

Hole depth is the sum, for all holes on the board, of the number of full cells directly above the hole, in the same column.

Our first objective was to examine the states encountered during the game to identify how often the following conditions hold: simple dominance, cumulative dominance, and noncompensation. Below we describe our implementation of the game and how we collected data.

4.1. Our implementations of Tetris

We developed two implementations of Tetris, one for collecting data from human subjects and one for performing simulation experiments with artificial players. Both implementations had the standard board size of 10×20. The next piece was selected uniformly at random. A reward of $+1$ was given for every deleted row. The total reward received in a game is called the *score*.

Human subjects played with an implementation of the game similar to the standard video game. Tetriminos fell from the top one piece at a time, at a comfortable speed. The game terminated when the next piece did not fit the board.

On the other hand, the implementation for artificial players required only the high-level decision of where, and in what orientation, to place the piece. In addition, the game was slightly simplified as is typical in the literature (Szita & Lörincz, 2006; Thiery & Scherrer, 2009; Scherrer et al., 2015). The top four rows were used solely to display the next piece. The legal moves were those that could be performed by rotating and translating the piece (to the left or the right) as desired in this display area and then dropping it. The game terminated when the player was not able to fit the next piece in the lower 16 rows of the grid.

4.2. Data collection

The states encountered during the game depend on the policy followed by the agent. We generated three different data sets that differed on how the agent played the game. One sample in each data set consisted of the current board configuration and the identity of the next piece to be placed. The data sets are described below.

`BCTS` was obtained by playing 20 games following the BCTS policy. Game scores ranged between 49,013 and 3,381,366 (median=709,636). From each game, we randomly selected 10,000 samples.

`Random` was obtained by selecting actions uniformly at random, playing as many games as necessary to generate 200,000 samples. A random policy clears lines very rarely, hence the games were very short, lasting 12–36 moves (median=22), with scores ranging from 0 to 5 (median=0).

`People` was obtained by asking 13 acquaintances to play the game as long as they wished. Before their moves were recorded, each participant had a chance to practice playing the game for a duration of their own choosing. The length of play ranged from 172 to 4,230 moves among the participants, with a median value of 554 moves. We gathered a total of 14,006 samples.

5. Results

Before we present the results, we define a few terms. Recall that a *sample* consists of the current board configuration and the identity of the next piece. An *action* is any legal placement of the next piece on the current board. A *consideration set* is a subset of the available actions. We define four different types of consideration sets: legal, distinct, Pareto-simple, and Pareto-cumulative, explained in detail below. An *ideal* placement is one that the linear evaluation function of BCTS ranks as the best. Because multiple placements can have identical feature values, there may be more than one ideal placement.

5.1. Dominance

Figure 2 shows the prevalence of simple and cumulative dominance. The figure displays twelve plots, showing the empirical probability distribution function of the size of four consideration sets under the three policies. By following each column from top to bottom, one can observe the reduction in the size of the consideration set as simple and cumulative dominance are applied as a filter, under a given policy.

`Legal` contains all legal placements of the piece. Its probability distribution has three peaks at 9, 17, and 34. These are the number of possible placements on the board for various pieces, when there is enough room. For instance, the

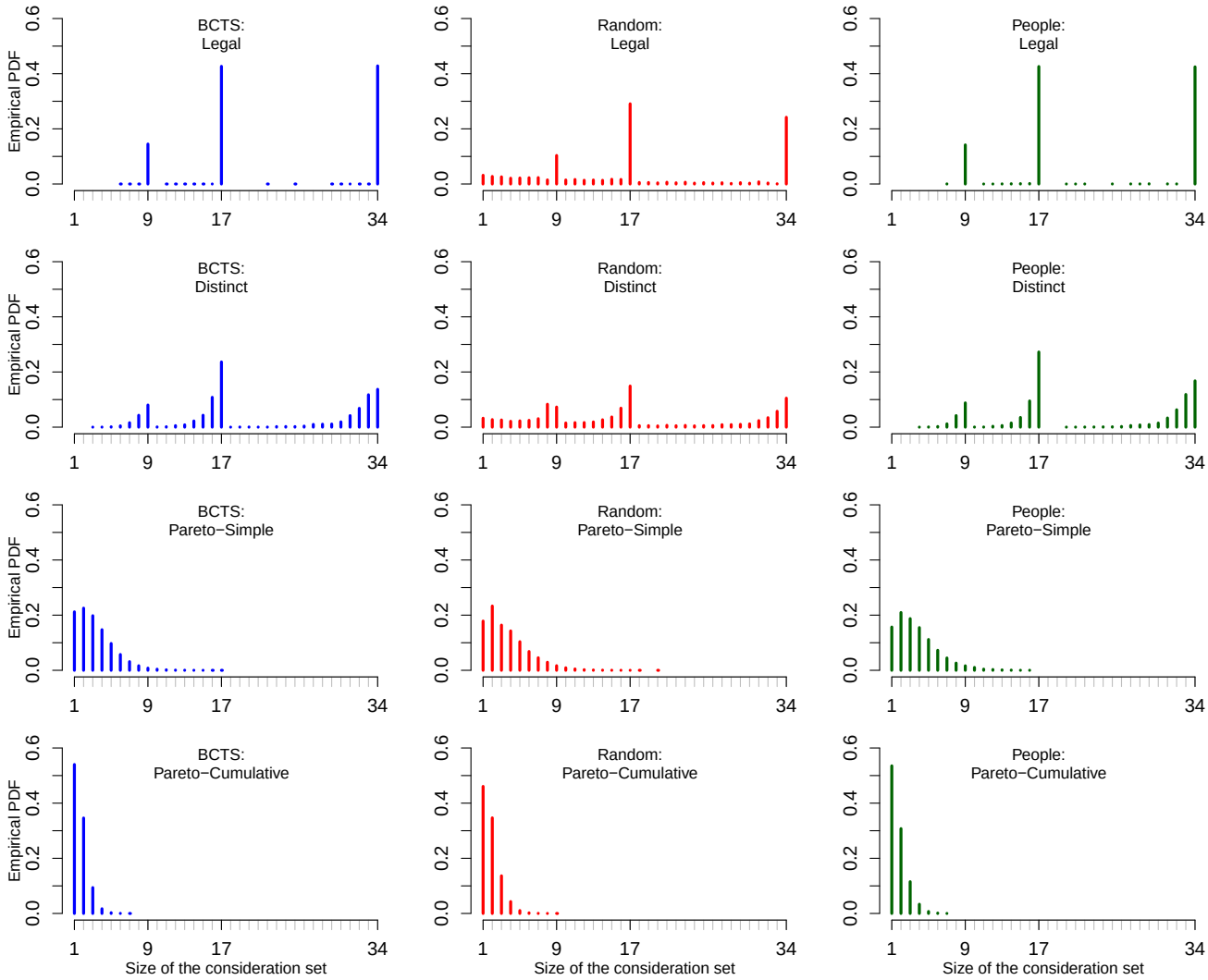


Figure 2. Prevalence of simple and cumulative dominance in Tetris. Each plot shows the empirical probability density function (PDF) of the size of the consideration set while a particular policy was being followed. All plots are drawn on the same scale. Axis labels are shown on the outer plots only. By following each column from top to bottom, one can observe the reduction in the number of placements as simple and cumulative dominance are applied as a filter.

square tetrimino can be placed in nine different ways on an empty board. When the board is almost full, the number of possibilities is smaller. Random policy experiences a full board much more often than the other two policies, which is why the probability distribution is slightly different for this policy.

Occasionally, two different legal placements yield the same feature values, which means that the BCTS controller—and any other feature-based controller using the same features—would not distinguish between these placements, assigning them equal value.

`Distinct` filters the legal set by removing all but one of those legal placements whose feature values are identical. Consequently, every placement in this set has a distinct set of feature values.

`Pareto-simple` filters the distinct set by further eliminating placements that are simply dominated by one or more other placements.

`Pareto-cumulative` filters the Pareto-simple set by further eliminating placements that are cumulatively dominated by one or more other placements (recall that simple dominance implies cumulative dominance). Mathematical properties of dominance ensure that both Pareto-simple and Pareto-cumulative sets contain at least one ideal placement.

Figure 2 shows that both simple and cumulative dominance reduced the size of the consideration set substantially. This is true for all three policies.

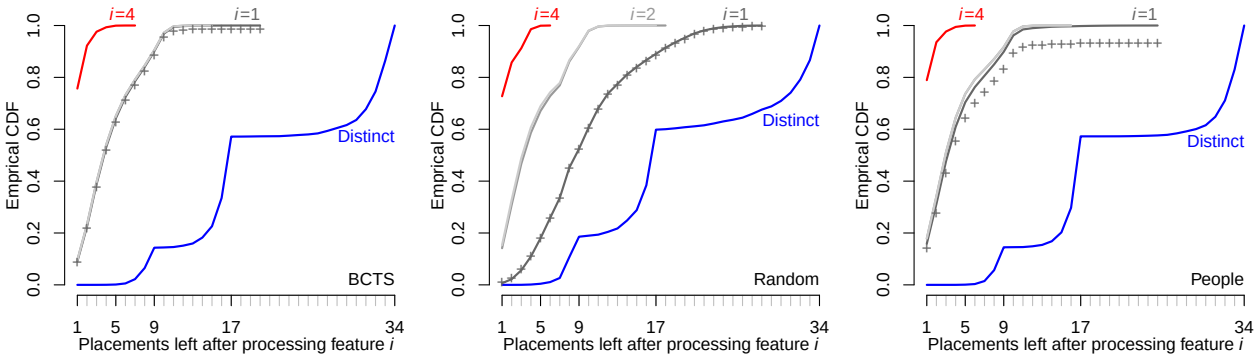


Figure 3. Elimination of alternative placements via the lexicographic decision rule. Solid lines show the empirical Cumulative Distribution Function (CDF) of the number of alternatives remaining after applying each of the first four features sequentially. Notice that on two data sets (BCTS and People), lines belonging to features 1–3 are almost identical. Plus signs show the proportion of cases where the ideal alternative remained under consideration after processing feature 1: if the proportion is 1, the plus sign is on the solid gray line.

Under the random policy, simple dominance reduced the median number of alternatives from 16 to 3, while cumulative dominance further reduced it to 2. In other words, ordinarily, the player had on average 16 distinct placements to consider. By simply knowing the sign of the weights (or guessing them correctly), the player can reduce this number to 3. And if the feature order is also known, this number can be reduced to 2.

Under the policies followed by BCTS and people, the reduction in the median size of the consideration set was larger: from 17 to 3 with simple dominance, and to 1 with cumulative dominance. This is remarkable. A consideration set of size one means that there is no decision to make: only one placement remains and it is known with certainty that this is an ideal placement.

5.2. Noncompensation

To each of the three data sets, we applied the lexicographic decision rule as follows. We started with the distinct set of placements; we used each feature sequentially, one at a time, to reduce the number of placements further and further until there was only one element left. With each feature, we kept only the placements that had the best (highest or lowest, depending on the sign of the feature weight) value for this feature, eliminating all the rest. We used the features in the order listed in Figure 1, which corresponds to the feature order of the BCTS controller.

The rate of noncompensation is the proportion of samples in which the lexicographic decision was an ideal decision. This number was 68.1, 69.0, and 52.0% under BCTS, people, and random controllers, respectively.

Figure 3 shows the lexicographic process on each data set. The vertical axis shows the empirical cumulative distribution function (CDF). Feature 1 (rows with holes) substantially reduced the alternatives considered while generally keeping the ideal alternative. Specifically, it reduced the median number of actions from 17 to 4, 9, and 4, while keeping the ideal action in the consideration set in 98%, 99%, and 93% of the cases, in BCTS, random, and people data sets, respectively.

5.3. Strength of overall play

We examined trajectories under three additional policies: selecting randomly among the Pareto-simple set, selecting randomly among the Pareto-cumulative set, and following the lexicographic decision rule. Figure 4 shows the distribution of scores obtained by these policies. To help put these numbers into context, consider that reaching a score of 80 is equivalent to clearing the legal placement area (16 rows) 5 times.

We also computed the probability of making the ideal choice on a randomly chosen sample along the trajectory. This number was 0.38 for the policy that selected among Pareto-simple and 0.75 for Pareto-cumulative.

6. Discussion

Our analysis showed that simple dominance, cumulative dominance, and noncompensation are all prevalent in Tetris. Furthermore, there is a feature (rows with holes) that eliminates a substantial number of actions while only rarely eliminating the best action. This feature is not complex but simple and intuitive.

Our findings raise two immediate questions. First, is Tetris unique or rare among sequential decision problems in having these properties? And second, how can algorithms make use of these properties for learning more effectively? We discuss each question in turn.

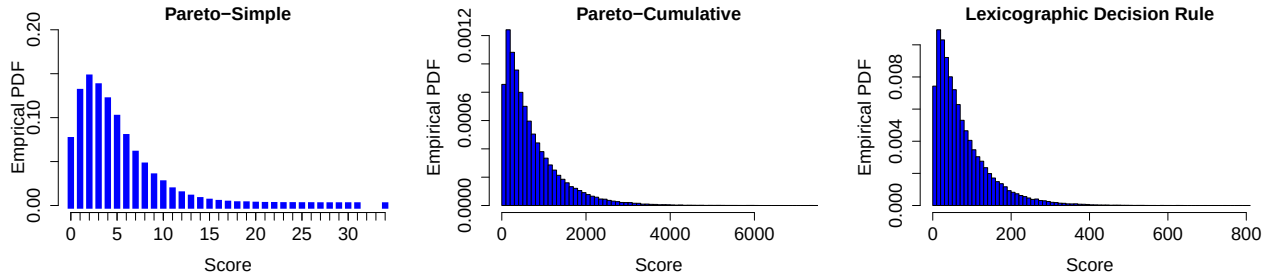


Figure 4. Scores obtained by policies that exploit simple dominance, cumulative dominance, and noncompensation. In contrast, a controller that selects actions randomly almost always ended the game with a score of 0 while BCTS scores ranged between 49,013 and 3,381,366 (median=709,636).

6.1. Is Tetris special?

Our results echo earlier findings on one-shot decision problems in natural environments. Dominance and noncompensation were found to be prevalent when comparing objects (Şimşek, 2013). In addition, presence of a very strong feature, one that can handle much of the workload by itself, was reported in classification and comparison problems (Holte, 1993; Şimşek & Buckmann, 2015). Given similar findings in simpler problems, and given that Tetris is the first sequential decision problem we analyzed, it seems unlikely that Tetris is special. This view is also supported by a limited analysis we performed on the game of backgammon.

Backgammon is one of the oldest board games known, beloved in many regions of the word. It has more than 10^{20} possible board positions. In 1979, a hand-crafted artificial player developed by Berliner (1980) beat the human world champion. Later, Tesauro (2002) developed a better player that learned through self-play using neural networks, one whose level of play surpassed the best human players.

We analyzed 1,838 games of backgammon played at top tournaments in the world, including the Monte Carlo World Championship, in 1973–2011. We obtained the data from an online repository maintained by Hübener to examine the prevalence of simple dominance. We examined positions where the two opponents still have contact—otherwise, the game simplifies considerably and depends mostly on the luck of the roll. This gave us 55,442 samples to examine. We used the following features, in the direction indicated in parenthesis: pip count (−), blot exposure (−), points occupied in home board (+), prime formation (+), pip count of opponent (+), blot exposure of opponent (+).

It should be noted that backgammon is a complex, strategic game. There are different high-level strategies that the players follow, such as a running game, blocking game, or back game, under which the desired movements of the pieces are very different. For instance, while a lower pip count is generally desired, when one is playing a back game, a higher pip count is advantageous. What we present is only a crude analysis that does not take context into account.

Figure 5 shows the results. Backgammon can have a high branching factor, especially if the dice show a double. The number of legal plays of the dice ranged from 0 to 515 (median=13; 95th percentile=64). In contrast, the size of the Pareto-simple set ranged from 0 to 259 (median=5; 95th percentile=22). In 77.8% of the positions examined, the action chosen by the tournament player was in the Pareto-simple set.

6.2. How can learning algorithms exploit these properties?

One possibility is to reduce the action choices of the agent to Pareto-simple and Pareto-cumulative sets. We give two examples. First, we applied approximate λ-policy iteration to Tetris, using the same set of features and parameters described by Bertsekas & Tsitsiklis (1996). Each iteration used 100,000 samples. Second, we applied AmpiQ, which belongs to the family of approximate modified policy iteration (AMPI) algorithms (Scherrer et al., 2015). We used a rollout-set size of 20,000 and rollout length of 15. We

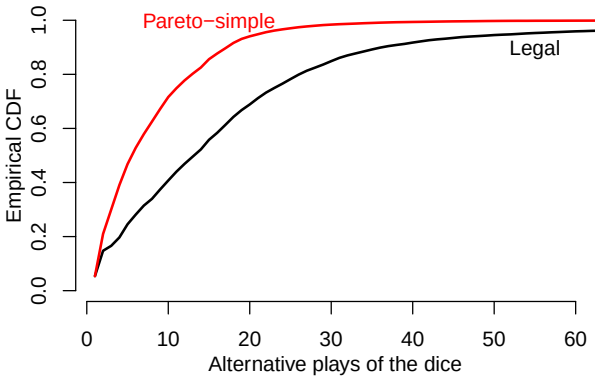


Figure 5. Simple dominance in backgammon.

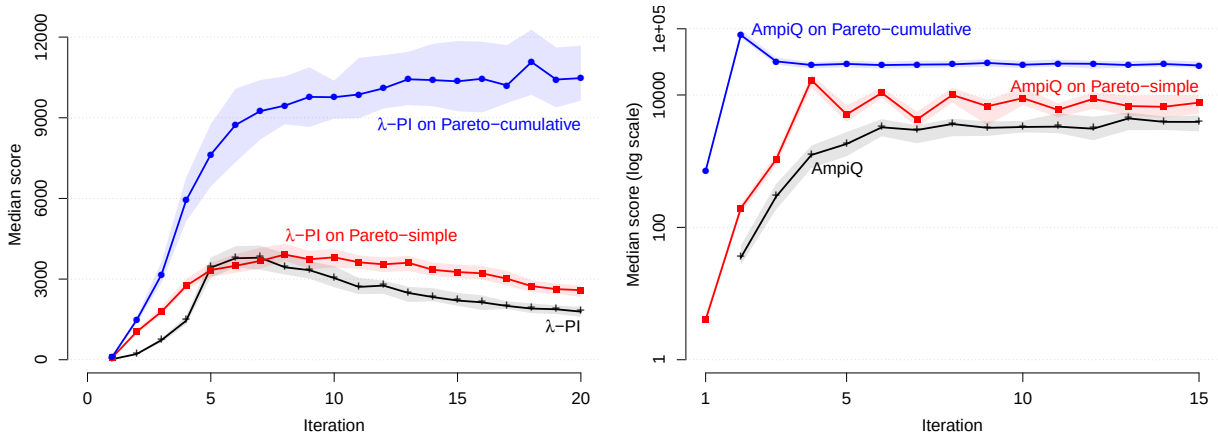


Figure 6. λ-Policy iteration (λ-PI) and AmpiQ applied to Tetris, with and without reducing the available actions to Pareto sets. The figure shows the distribution of the median score at each iteration (100 replications). The solid lines connect the median values, while the shaded areas show the 25th and 75th percentiles. The scores on AmpiQ are shown on a log scale. The first iteration of AmpiQ without Pareto sets returned a median score of zero (not displayed in the figure).

used the features listed in Figure 1 along with those used by Lagoudakis et al. (2002).

Figure 6 shows learning curves with and without first reducing the action set to Pareto-simple and to Pareto-cumulative, using the features listed in Figure 1. The figure shows the distribution of the median score at each iteration (100 replications). The solid lines connect the median values, while the shaded areas show the 25th and 75th percentiles. The learning curves show clear improvements in the learning rate and in the quality of the policies learned.

Another promising research direction is to examine very simple policies. Dominance and noncompensation allow simple rules, such as lexicographic and tallying heuristics (Gigerenzer et al., 1999), to make correct decisions in one-shot problems. Very simple policies inspired by these heuristics may be able to learn reasonably-well policies within short periods of interaction with the environment.

7. Concluding remarks

This work explored a novel direction in learning to solve sequential decision problems, leveraging insights from various research fields such as cognitive psychology and operations research. We examined whether three mathematical properties—which were found to be useful in making one-shot decisions—also hold for sequential decisions. We found that the properties were prevalent and could be embedded in learning algorithms in the games of Tetris and backgammon.

As noted earlier, two important future directions are to study additional sequential decision problems and to explore how learning algorithms can further exploit these properties. Additional research directions include: (1) taking context into account when identifying Pareto sets, (2) using principled probabilistic approximations to simple and cumulative dominance (Şimşek, 2013), (3) exploring the use of simple dominance for nonlinear value functions—we have discussed dominance only for a linear function, but simple dominance applies to a much broader set of functions.

Traditionally, it has been difficult to inject domain knowledge into reinforcement learning algorithms. The work described here presents an easy way of doing so for certain types of domain knowledge. The type of knowledge required to construct Pareto-simple and Pareto-cumulative sets, or close approximations of them, may be readily available in many domains. For example, in Tetris, feature directions are all intuitive, which allows the Pareto-simple set to be constructed easily.

Our results make a strong case for the utility of studying natural decision problems. While synthetic domains play an important role in developing learning algorithms, studying the structural regularities of natural problems can yield additional insights for learning effectively.

Acknowledgments

Thanks to Konstantinos Katsikopoulos, Gerd Gigerenzer, Anita Todd, and three anonymous reviewers for comments on earlier drafts.

References

Adamic, Lada A., Lukose, Rajan M., Puniyani, Amit R., and Huberman, Bernardo A. Search in power-law networks. *Physical Review E*, 64(4):046135, 2001.

Baucells, Manel, Carrasco, Juan A., and Hogarth, Robin M. Cumulative dominance and heuristic performance in binary multiattribute choice. *Operations Research*, 56(5):1289–1304, 2008.

Berliner, Hans J. Backgammon computer program beats world champion. *Artificial Intelligence*, 14(2):205–220, 1980.

Bertsekas, Dimitri P. and Tsitsiklis, John N. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

Burgiel, Heidi. How to lose at Tetris. *Mathematical Gazette*, 81:194–200, 1997.

Demaine, Erik D., Hohenberger, Susan, and Liben-Nowell, David. Tetris is hard, even to approximate. In *Computing and Combinatorics*, pp. 351–363. Springer, 2003.

Gabillon, Victor, Ghavamzadeh, Mohammad, and Scherrer, Bruno. Approximate dynamic programming finally performs well in the game of Tetris. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 1754–1762. Curran Associates, Inc., 2013.

Gigerenzer, Gerd, Todd, Peter M, Group, ABC Research, et al. *Simple heuristics that make us smart*. Oxford University Press, New York, 1999.

Hogarth, Robin M and Karelaia, Natalia. "Take-the-best" and other simple strategies: Why and when they work "well" with binary cues. *Theory and Decision*, 61(3): 205–249, 2006.

Holte, Robert C. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–91, 1993.

Hübener, Hardy. Hardy's backgammon pages. http://www.hardyhuebener.de/engl/matches.html. Accessed: 2015-04-24.

Katsikopoulos, Konstantinos V. Psychological heuristics for making inferences: Definition, performance, and the emerging theory and practice. *Decision Analysis*, 8(1): 10–29, 2011.

Kleinberg, Jon M. Navigation in a small world. *Nature*, 406(6798):845–845, 2000.

Lagoudakis, Michail G., Parr, Ronald, and Littman, Michael L. Least-squares methods in reinforcement learning for control. In *Methods and Applications of Artificial Intelligence*, pp. 249–260. Springer, 2002.

Martignon, Laura and Hoffrage, Ulrich. Fast, frugal, and fit: Simple heuristics for paired comparison. *Theory and Decision*, 52(1):29–71, 2002.

Scherrer, Bruno, Ghavamzadeh, Mohammad, Gabillon, Victor, Lesner, Boris, and Geist, Matthieu. Approximate Modified Policy Iteration and its Application to the Game of Tetris. *Journal of Machine Learning Research*, 16:1629–1676, 2015.

Silver, David, Huang, Aja, Maddison, Chris J., Guez, Arthur, Sifre, Laurent, van den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Şimşek, Özgür. Linear decision rule as aspiration for simple decision heuristics. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2904–2912. Curran Associates, Inc., 2013.

Şimşek, Özgür and Buckmann, Marcus. Learning from small samples: An analysis of simple decision heuristics. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 3159–3167. Curran Associates, Inc., 2015.

Şimşek, Özgür and Jensen, David. Navigating networks by using homophily and degree. *Proceedings of the National Academy of Sciences*, 105(35):12758–12762, 2008.

Szita, István and Lörincz, András. Learning Tetris using the noisy cross-entropy method. *Neural Computation*, 18(12):2936–2941, 2006.

Tesauro, Gerald. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, 134(1):181–199, 2002.

Thiery, Christophe and Scherrer, Bruno. Building controllers for Tetris. *International Computer Games Association Journal*, 32:3–11, 2009.

Watts, Duncan J., Dodds, Peter Sheridan, and Newman, Mark E. J. Identity and search in social networks. *Science*, 296(5571):1302–1305, 2002.