Learning Curricula for Humans: An Empirical Study with Puzzles from The Witness

Levi H. S. Lelis[1,2,3*] , **João G. G. V. Nova**[3] , **Eugene Chen**[1,2] , **Nathan R. Sturtevant**[1,2] , **Carrie Demmans Epp**[2] and **Michael Bowling**[1,2]
[1]Alberta Machine Intelligence Institute (Amii)
[2]Department of Computing Science, University of Alberta, Canada
[3]Departamento de Informática, Universidade Federal de Viçosa, Brazil

Abstract

The combination of tree search and neural networks has achieved super-human performance in challenging domains. We are interested in transferring to humans the knowledge these learning systems generate. We hypothesize the process in which neural-guided tree search algorithms learn how to solve a set of problems can be used to generate curricula for helping human learners. In this paper we show how the Bootstrap learning system can be modified to learn curricula for humans in a puzzle domain. We evaluate our system in two curriculum learning settings. First, given a small set of problem instances, our system orders the instances to ease the learning process of human learners. Second, given a large set of problem instances, our system returns a small ordered subset of the initial set that can be presented to human learners. We evaluate our curricula with a user study where participants learn how to solve a class of puzzles from the game 'The Witness.' The user-study results suggest one of the curricula our system generates compares favorably with simple baselines and is competitive with the curriculum from the original 'The Witness' game in terms of user retention and effort.

1 Introduction

Tree search algorithms guided by neural networks have achieved super-human performance in challenging domains. For example, AlphaZero learns how to play chess and Go by playing the game with itself [Silver *et al.*, 2018]. In this paper we are interested in generating curricula for human learners (students) that capture the knowledge intelligent systems such as AlphaZero generate. We use a neural-guided tree search algorithm to generate curricula for human learners in the context of a single-agent sequential decision making problem.

A curriculum is an ordered set of problem instances that is designed to ease learning. Students start by attempting to solve the first instance in the curriculum and then they move to the second. A well-designed curriculum gradually introduces novel concepts so that concepts learned while solving earlier instances can be used to support later learning. A curriculum should introduce novel concepts at the right time since students might be bored if instances are too easy and they may become frustrated if instances are too hard [Graesser and D'Mello, 2012]. Consequently, introducing novel concepts at the wrong time could result in increased drop-out rates. Finally, curricula should cover all concepts needed for solving any instance from a class of problems.

We investigate Levin tree search (LTS) [Orseau *et al.*, 2018] and Bootstrap [Jabbari Arfaee *et al.*, 2011] as a model to generate curricula for decision-making tasks. *We hypothesize the order in which LTS solves a set of instances while learning a neural policy can be used as an effective curriculum for human learners.* We evaluate our hypothesis in two settings. The first orders a set of given instances to form a curriculum (**ordering problem**). The second takes a large set of instances and selects an ordered subset from these instances to form a curriculum (**order-and-select problem**).

We tested our hypothesis through a user study with a class of puzzles from the commercial game 'The Witness.' We conducted our between-subjects study (n = 685) via the LabintheWild platform [Reinecke and Gajos, 2015]. Of the five curricula tested, two were automatically generated by our system. One of these generated curricula is a solution to the ordering problem when we provide a small set of puzzles from the original game as input. The second generated curriculum is a solution to the order-and-select problem, where we use a large set of computer-generated puzzles. The user-study results support our hypothesis: a curriculum generated by our system compares favorably with simple baselines and is competitive with the curriculum from the original 'The Witness' game in terms of user retention and effort.

Contributions. We show how a learning system can be modified to generate a curriculum for helping human learners. We perform an extensive evaluation of our system through a user study with a puzzle domain. To the best of our knowledge, we are the first to investigate the transfer of computer-generated knowledge to humans via automatically generated curricula.

2 Related Work

Our work is related to machine teaching [Zhu, 2015; Zhu *et al.*, 2018], where one is interested in generating a training set that allows a learner to efficiently learn a concept. A

[*]Contact Author

challenge in machine teaching is curriculum design: the task of generating a set of problems that is sufficient for learning. Often a curriculum starts with easy instances and gradually increasing difficulty [Bengio *et al.*, 2009; Taylor, 2009]. Others have approached machine teaching from a theoretical perspective [Goldman and Kearns, 1995; Zilles *et al.*, 2011; Zhu *et al.*, 2017]. Our system approximates an optimal training set size for an algorithm, which is similar to the goals of these previous works. In contrast to them, we evaluate a subset of the near-optimal training set as a curriculum for humans.

Many have modelled the task of choosing the next activity for a learner as a planning [Chi *et al.*, 2011; Rafferty *et al.*, 2011] or as an optimization problem [Patil *et al.*, 2014; Lindsey *et al.*, 2013]. These methods observe how the learner behaves before deciding the next activity. We develop a curriculum beforehand and do not observe the learner. Often planning and optimization-based methods assume the existence of a learner model while we evaluate a neural-guided tree search algorithm as a learner model.

Khan *et al.* [2011] and Cakmak and Lopes [2012] evaluated strategies humans use for teaching. Khan *et al.* showed that human teachers use a strategy where students are first exposed to easy problems and only later to harder ones. Like this work, we compare the curriculum our system generates with that of a human teacher. While Khan *et al.* focused on classification tasks, we focus on a sequential task. Cakmak and Lopes also deal with sequential tasks, but they focus on training teachers, while we focus on teaching human learners.

3 Problem Definition

$P = (S, A, T, s_0, g)$ is a sequential-decision making problem, where S is a set of states with s_0 and g in S being the start and goal states. A is a function that receives a state s and returns a set of valid actions at s. T is a deterministic transition function that receives a state s and an action a and returns the state reached once a is applied to s. A solution path is a sequence of state-action pairs $\{(s_i, a_i)\}_{i=0}^{m}$ with $T(s_i, a_i) = s_{i+1}$ and $T(s_m, a_m) = g$. A problem instance (or instance for short) is a pair (s_0, g) for fixed S, A, and T.

A learning algorithm L receives an ordered set of instances D and a search algorithm; it returns a heuristic h that is used to guide the search algorithm (e.g., a cost-to-go function or a probability distribution over actions). h is optimal, denoted h^*, if it minimizes the number of states a search algorithm guided by h must visit to find solutions for instances in D. The efficiency of L is measured in terms of the number of states it visits while learning h, which depends on the ordering of D, as we explain in Section 5. The inverse L^{-1} is a function that receives h and returns an ordered set D^* that minimizes the number of states L needs to visit to learn h.

In curriculum learning for sequential-decision making problems, one receives a set of instances D in an arbitrary order and returns an ordered set D^* that minimizes the number of states L visits to learn an approximation of an optimal heuristic h. We hypothesize the D^* that L^{-1} returns can be used as an effective curriculum for human learners.

We consider two variations of the curriculum learning problem. In the first, we are given a small set of instances D_o and we need to provide a total ordering for D_o. The instances are then presented in the order defined by the curriculum to students. We call this variant the ordering problem; the subscript in D_o stands for 'ordering'. The second variant considers a large set D_s and an integer c and the task is to provide a totally ordered subset D' of D_s of size c. The subset D' is used as a curriculum. We call this variant the order-and-select problem; the subscript D_s stands for 'select'.

4 Levin Tree Search (LTS)

We use a modified version of Bootstrap and LTS as the learning algorithm L, which approximates a policy π as the heuristic h that guides the LTS search. A policy $\pi(s, a)$ returns the probability of taking action a at s. Other search algorithms such as A* [Hart *et al.*, 1968] could possibly be used in our experiments. We use LTS because it performs well on the puzzles we use in our experiments [Orseau and Lelis, 2021].

LTS is a best-first search algorithm that uses a policy. LTS grows a search tree in which each node represents a state in S and the tree is rooted at a node that represents s_0. Since each node represents a state, we use the words 'nodes' and 'states' interchangeably. There is a directed edge from n to n' in the tree if $T(n, a) = n'$ with $a \in A(n)$. A node n is expanded when we compute $T(n, a)$ for all a in $A(n)$; a node n' is generated when $T(n, a) = n'$ is computed.

Like A*, LTS uses a priority queue, which we refer to as OPEN. LTS starts by inserting the node representing the start state s_0 in OPEN. In each iteration, it removes from OPEN and expands the node n with lowest cost. The cost function LTS employs is $\frac{d(n)}{\pi(n)}$. Here, $d(n)$ is the depth of n in the search tree and $\pi(n)$ is the probability of n, which is computed as follows. Let $\{(n_i, a_i)\}_{i=0}^{m}$ be the sequence of state-action pairs starting at the root of the tree n_0 and ending at node $T(n_m, a_m) = n$, then $\pi(n) = \prod_{i=0}^{m} \pi(n_i, a_i)$. This cost function allows LTS to focus on nodes n with higher values of $\pi(n)$, while $d(n)$ acts as an exploration term, forcing the search to balance probability and depth values. LTS stops searching once it generates a goal state.

5 Curriculum Learning with Bootstrap

Bootstrap receives as input a set of instances D and it returns a policy π. In our experiments, Bootstrap starts with a policy π_0 given by a randomly initialized neural network and employs LTS guided by π_0 to try to solve the instances in D; the instances LTS can solve, denoted V_0, are used to train the policy, thus generating π_1. The process is repeated with π_1 for the instances that were not solved in the first iteration. The instances LTS can solve with π_1 form the subset V_1 of D. Policy π_2 is obtained by training π_1 with $V_0 \cup V_1$. Bootstrap stops once it solves all instances, i.e., $\bigcup_i V_i = D$, and it returns the last policy π_n as an approximation of π^*.

Bootstrap uses a schedule for adjusting the budget b allowed for each instance. The search budget is measured in terms of states LTS visits in search and it is fixed for an entire iteration (i.e., LTS tries to solve all instances in D while visiting at most b states per instance). The budget is set empirically to minimize the search effort of learning a policy. For example, Orseau and Lelis [2021] used an initial b of 2,000 for The Witness puzzle we use in our experiments. If LTS is unable to solve any instances in a given iteration, Bootstrap doubles b for the next iteration. Previous implementations of Bootstrap ensure the budget increases monotonically.

The partially ordered set $V = \{V_0, V_1, \cdots, V_n\}$ approximates $L^{-1}(\pi_n)$ because it allows one to learn π_n without trying and failing to solve instances in D. That is, LTS solves the instances in V_0 with π_0, which provides the updated policy π_1. LTS then uses π_1 to solve V_1, and so on. We assume the b value used to solve each V_i during training is stored in memory and is used to learn π_n. The partially ordered set V only approximates $L^{-1}(\pi_n)$ since there could exist other sets that would allow one to learn π_n more efficiently (e.g., use smaller b-values that allow LTS to solve the instances).

We adapt the Bootstrap procedure for learning curricula as follows. We set $b = 1$ and we double it if the the algorithm completes an iteration without solving any of the unsolved instances. We also set $b = 1$ every time the policy is updated. Previous work has used larger b-values that monotonically increase since this tends to reduce the effort of learning π_n. Since in the curriculum learning problem we are not primarily concerned with the efficiency of learning π_n, we use a small b and reset it to 1 whenever the policy updates. The use of smaller b-values produces smaller subsets V_i (for sufficiently large b we would have that $V_0 = D$ and Bootstrap would provide no ordering of the instances). Having smaller subsets V_i is important because all instances in V_i have the same rank in V's partial ordering. Thus, when presented to students, the instances in V_i are presented in an arbitrary order. By minimizing the size of V_i we minimize the number of instances that are presented in an arbitrary order to students.

Ordering Problem The number of instances in the set D_o provided as input to the ordering problem is much smaller than the number of training instances required to train a neural policy. Instead of using only D_o as the input set for Bootstrap, we provide as input a larger training set for which D_o is a subset. Then, we run Bootstrap with this enlarged set. The indexes i of the V_i subsets determine an ordering for the instances in D_o. Let (s, g) and (s', g') be two instances in D_o. The instance (s, g) is considered easier than (s', g') if (s, g) is in V_i and (s', g') is in V_j and $i < j$. Since each run of Bootstrap can result in a different policy due to the random initialization of the network's weights, we ran Bootstrap multiple times instead of only running it once. We determine the ordering of instances using the average index i of the V_i subsets of each instance in D_o across all runs. The multiple runs of the system and our budgeting scheme (i.e., $b = 1$) make it unlikely that two instances will have the same average index, which would imply a partial ordering. In our experiments, our system always returned a total ordering for D_o.

Order-and-Select Problem We assume the size of D_s for the order-and-select problem is large enough to train a neural policy, so D_s is provided as is to Bootstrap. Similarly to the ordering problem, we also run Bootstrap multiple times and compute the average index i of the V_i subsets of each instance in D_s, which provides an ordering of the instances in D_s. We then select c instances in D_s to form a curriculum as follows. Let $\{o_0, o_1, \cdots, o_m\}$ be a totally ordered set of instances. We select instances with index $\left\lfloor \frac{i \times m}{(c-1)} \right\rfloor$ for $i = 0, \cdots, c-1$ to compose a curriculum. This procedure selects c equidistant instances according to their indexes. For example, if $m = 1000$ and $c = 5$, we select the instances with index $0, 250, 500, 750, 1000$. If Bootstrap provides a partial ordering of D_s, we order the instances with the same rank arbitrarily. This procedure guarantees that the curriculum has instances of varied difficulty, as evaluated by Bootstrap. In particular, it includes the easiest and hardest instances of D_s.

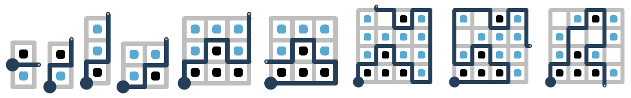
Figure 1: Curriculum from the original game 'The Witness.' The player solves the instances from left to right

6 Experimental Design

We performed a user study to evaluate the effectiveness of the curriculum Bootstrap generates for both the ordering and the order-and-select problems. Our study was performed online with volunteers from LabintheWild (LitW) [Reinecke and Gajos, 2015]. We used LitW because it offers a more demographically and geographically diverse pool of participants than other platforms. Instead of financial compensation, LitW volunteers receive information about themselves and about the experiment following completion. Our study was approved by the research ethics board of our institution.

We used a between-subject design where each participant was randomly assigned a curriculum. Participants started with the consent process, answered a questionnaire, and then attempted to solve the instances in the curriculum before trying to solve a set of test instances. Participants could skip instances or leave the experiment at any time. The next instance in the curriculum (or test set) was made available only after the person either solved or skipped the current instance. After completing the test instances, people could return to skipped instances to try to solve them; we did not analyze these post-study attempts. The graphical user interface of our system allowed participants to see an image of the instances they had previously attempted in the experiment. Participants finished by answering demographic questions.

6.1 Black and White Puzzles

We use the Black and White Squares (BWS) puzzle from the 'The Witness', where the player learns how to solve BWS puzzles by solving them in a carefully designed curriculum. Figure 1 shows 9 instances of the curriculum used in the game, where the player solves the instances from left to right. Initially, only the leftmost instance is available to the player and the next instance becomes available only after the player solves the current instance. We refer to these 9 instances as the **witness-exact curriculum.** In BWS, the player needs to draw a line starting at the circle located at an outer edge of the grid (see bottom-left corner of all instances but the first in Figure 1) and finishing at the small line sticking out of the grid.

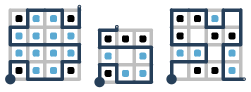
Figure 2: The test instances used in our study. The first instance is from the original game and is fixed for all participants. The other two are representative test instances.

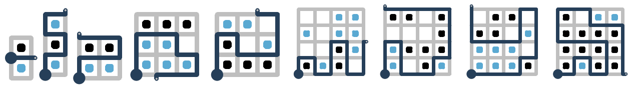
Figure 3: Our system's solution to the order-and-select curriculum learning problem (equidistant curriculum).

The line must partition the grid cells such that no squares of different colors fall in the same partition. If the player draws a line from the start location to the end location that violates the color-separation constraint, the line blinks indicating the player has failed to solve the problem. The game does not instruct the player in the conditions for solving the puzzles (i.e., the start and end points and the need of separating squares of different colors); the player needs to discover the conditions as they solve instances in the curriculum. We follow exactly the same approach in our user study.

6.2 Training the Neural Model

We developed a generator of BWS puzzles and used it to generate 2,360 puzzles: 10 with grid size 1×2; 50 with size 1×3; 300 with size 2×2; 1,000 with size 3×3; 1,000 with size 4×4.

We use the 9 instances from witness-exact as the set D_o for the ordering problem. The union of the 2,360 generated instances and the 9 instances from the witness-exact curriculum form both the augmented data set for the ordering problem and the data set for the order-and-select problem.

We ran Bootstrap with LTS 100 times to compute the average ranking of the instances. Bootstrap obtained exactly the same order used in the original game for the ordering problem with the 9 instances from witness-exact. Under the assumption that the game designer created a near-optimal curriculum for human learners, this result is evidence that Bootstrap with LTS can be an effective model for solving the ordering problem of curriculum generation. Khan *et al.* [2011] also made the assumption that humans design near-optimal curricula and performed a similar experiment, where they compared human-designed curricula to computer-generated ones.

6.3 Curricula and Test Instances for the Study

We used the **witness-exact** curriculum to represent both the curriculum of the original game and the Bootstrap solution to the ordering problem. We also evaluated two baseline curricula that present different orderings of these 9 instances: one presents the reverse ordering (**witness-reverse**) and the other the instances in random order (**witness-random**). For the witness-random, the 9 instances are shuffled independently for each participant. These baselines allow us to evaluate whether the ordering chosen by the game designer and found by the Bootstrap system is superior to other alternatives.

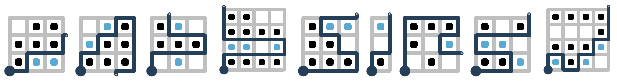
Figure 4: Example of all-random curriculum.

We also evaluate the Bootstrap's solution to the order-and-select problem, which is shown in Figure 3. We use $c = 9$ to allow for a fair comparison with the curricula based on the instances from the original game. We refer to this curriculum as the **equidistant curriculum** because it selects equidistant instances according to Bootstrap's ordering. Finally, we evaluate a curriculum that randomly selects 9 instances from the 2,369 instances used in our experiments. We refer to this curriculum as the **all-random curriculum**. The 9 instances are randomly and independently selected for each participant who was assigned the all-random curriculum. Figure 4 shows an example of the all-random curriculum.

We are primarily interested in comparing (i) witness-exact (our system's solution to the ordering problem) to the other witness-based curricula and (ii) equidistant (our system's solution to the order-and-select problem) to the strong witness-exact baseline. We considered including curricula based on features of the game such as size of the grid and number of colored cells. However, these features do not provide useful information to the curriculum learning problem. Since most of the puzzles are of size 4×4, grid size would not be helpful in distinguishing puzzles of the that size. The number of colored and empty cells provide little to no information about the difficulty of the puzzle, e.g., the hardest instance in witness-exact has several empty cells (Figure 1), while the hardest instance in equidistant has only one empty cell (Figure 3).

Test Instances Figure 2 shows a representative set of the test instances used in our study. The first instance in the test set (p10) was identical for all participants. Instance p10 was also extracted from the original game. In the game, after finishing solving the 9 instances shown in Figure 1, the player solves other types of puzzles before encountering p10. We use p10 in our test set because, in the original game, the player is able to unlock a door after solving it, as if the game uses p10 to test the knowledge the player acquires by solving the initial 9 instances. The other two puzzles shown in the figure can vary for each participant; we control for their solution length (12 and 20 steps for the second and third instances).

6.4 Metrics

We evaluate the curricula described using proxies that indicate how the curriculum relates to student persistence, effort, and performance. For persistence, we use participant drop-out rate (i.e., user retention), where a participant has dropped-out if they quit the experiment before attempting the first test instance. We also count the number of puzzles each participant solves before dropping out. For effort, we count the number of times a participant attempted to solve each instance. An attempt is completed when the path is cleared so the participant can restart solving the instance. Clearing happens when participants push the 'reset' button or connect the start and end locations without separating the squares of different colors. We also report the number of backtrack steps each person performed. A backtrack step is when someone undoes a segment of the path on the grid. For performance, we report the number of puzzles solved. For all metrics we report their average (M) and standard deviation (SD).

The metrics of effort and performance should not be analyzed individually because they are computed for participants who did not drop out the experiment, which can lead to a survivor bias effect [Wald, 1980]. Due to such an effect, curricula with high drop-out rate tend to have larger number of test puzzles solved and fewer number of attempts and backtracks than curricula with low drop-out rate. In our analysis, we consider the drop out rate while discussing puzzles solved, attempts, and backtracks.

Statistical Tests Since the data is binomial, for the drop-out results we use the Chi-Squared test for all curricula and Fisher exact paired tests for comparing two averages. For the other tests we use the non-parametric Kruskal-Wallis test for all curricula and the non-parametric Mann-Whitney test for comparing pairs of averages. We say the difference is statistically significant if the p-value for the test comparing all curricula is less than .05 and if the paired test is also less than .05. We use superscript letters to denote statistically significant differences. If two numbers in the same column have different superscript letters, then their difference is statistically significant. For example, in Table 1, the difference in drop-out rate between witness-exact and witness-reverse is statistically significant, while that between equidistant and witness-exact is not. We do not use superscript letters if the p-value of the statistical test of all averages is larger than .05.

7 User Study Results

We report the data collected between December 2018 and January 2022 through the LitW platform: 777 people consented to participate. We excluded the data from those who had played 'The Witness' before (n = 92), resulting in a total of 685 participants. The average age was 28.6 (SD = 11.3), with their ages ranging from 18 to 99. Participant gender was relatively balanced: 272 identified themselves as female, 219 as male, 9 as other, and 185 withheld gender information. Most (n = 555) had received or were pursuing post-secondary training, and 130 had completed primary or high school. We also asked two questions about their experience playing games and solving puzzles. We used a 7-point, Likert-type scale from 1, "never play games/solve puzzle", to 7, "always play games/solve puzzles." The median response for playing games was 5 (IQR = 2) and for solving puzzles was 4 (IQR = 2). Most participants reported they play games and solve puzzles occasionally. The demographics distribution is similar across curriculum conditions.

7.1 User Retention

A large drop-out rate could be due to the curriculum failing to balance the difficulty of its instances. The middle column of Table 1 shows the drop-out rate of each curriculum. For example, 39% of participants in the witness-exact curriculum dropped out before attempting the test instances. The last column shows the average and standard deviation of the number of puzzles each participant solved before dropping out. For example, out of the 9 curriculum puzzles, participants of the witness-exact curriculum solved 3.38 puzzles on average (SD = 1.99) before dropping out.

Curriculum Condition	Dropout	# Solved Before Dropping Out
witness-exact	$39\%^a$	3.38^a (1.99)
witness-random	$50\%^a$	1.32^b (1.83)
witness-reverse	$63\%^b$	0.38^c (1.32)
all-random	$63\%^b$	0.73^d (1.69)
equidistant	$48\%^a$	2.47^e (1.87)

Table 1: The drop-out rate and number of puzzles solved, as M (SD), per participant before dropping out.

Curriculum	Number of Puzzles Solved		
Condition	Curriculum	Test	First Test
witness-exact	8.04^a (1.43)	2.08 (0.87)	0.38 (0.49)
witness-random	7.61^a (1.91)	1.92 (0.95)	0.33 (0.47)
witness-reverse	6.67^b (2.23)	1.72 (0.97)	0.22 (0.41)
all-random	7.40 (2.4)	2.09 (1.03)	0.44 (0.50)
equidistant	7.42 (1.73)	2.03 (1.01)	0.44 (0.50)

Table 2: Number of puzzles solved as M (SD).

The curriculum with best user retention was witness-exact (39% drop-out) while the two worst were all-random (63%) and witness-reverse (63%). Participants also solved far more puzzles before dropping out in witness-exact. These results suggest that order matters and that the order chosen by our system in the ordering problem encourages superior user retention to the witness-reverse and witness-random baselines.

While retention was not measurably different between equidistant and witness-exact, participants solved more puzzles before dropping out when they were given the equidistant curriculum than for all baseline curricula but witness-exact. Also, the equidistant drop-out rate was significantly lower than witness-reverse and all-random. These results suggest equidistant retains participants longer than the simpler baselines and is competitive with the witness-exact.

7.2 Number of Instances Solved

There were 321 non-drop-out participants (128 female, 114 male, 5 'other', and 74 did not specify their gender). The average age was 29 (SD = 10.2). The distributions of frequency with which participants played games and solved puzzles and participant education for the non-drop-out participants were similar to that reported for the entire pool of users.

Table 2 shows how many curriculum (maximum is 9) and test instances (maximum is 3) the participants solved. It also shows the average number of p10 puzzles solved per participant (maximum is 1). They solved significantly fewer curriculum instances in the witness-reverse condition than they did in witness-exact or witness-random. This result reinforces those based on retention: the puzzle order influences the number of puzzles solved. We only compared witness-exact, witness-random, and witness-reverse because these curricula

Curriculum	Attempts		
Condition	Curriculum	Test	First Test
witness-exact	2.80 (1.57)	3.95^a (3.86)	7.40^a (7.66)
witness-random	3.82 (1.86)	3.76^a (3.44)	6.77^a (6.08)
witness-reverse	3.66 (3.69)	4.13^a (3.72)	7.70^a (5.98)
all-random	3.94 (2.17)	3.30^a (2.84)	$7.02^{a,b}$ (6.87)
equidistant	3.19 (1.80)	2.26^b (2.29)	4.92^b (5.63)

Table 3: Number of attempts, as M (SD), for the curriculum instances, test instances, and the first test instance.

Curriculum	Backtracks		
Condition	Curriculum	Test	First Test
witness-exact	0.92^a (1.09)	4.91 (5.99)	9,98 (14.78)
witness-random	1.17^a (1.37)	4.83 (5.15)	10.34 (13.32)
witness-reverse	0.91^a (0.87)	3.73 (3.92)	8.53 (10.91)
all-random	1.41^a (2.32)	4.44 (5.08)	9.78 (12.42)
equidistant	2.28^b (2.33)	3.70 (3.99)	7.65 (9.92)

Table 4: Number of backtracks, as M (SD), for instance types by curriculum.

use the same set of instances and differ only in their ordering.

7.3 User Effort

We now turn to the number of attempts and backtracks, our proxies for user effort. The results we report are for the non-drop-out participants, regardless of whether they solved a given instance. Table 3 shows the number of attempts in the test set and the first test instance. There is a statistically significant difference between equidistant and all curricula on the average number of attempts for solving the test instances. While the average numbers of test instances solved are similar (Table 2) for all curricula, equidistant has the smallest number of attempts. A similar pattern emerges for the first test instance. Table 4 shows the number of backtracks. There is a significant difference between equidistant and the other curricula on the numbers for curriculum instances.

7.4 Discussion

The backtrack results suggest equidistant requires more effort from users. The required effort seems to be well balanced with participants' learning because the equidistant drop-out rate is significantly smaller than that of all-random and witness-reverse. The results also suggest equidistant better prepares learners for solving the test instances as they needed fewer attempts to solve about the same number of test instances as those solved by people given the other curricula.

The last instance of equidistant (p9) offers explanations for the user effort results. p9 requires the solution path to separate squares of the same color (rightmost puzzle in Figure 3), which is a rare feature that makes the puzzle hard. We will refer to this feature as the same-color feature. Recall that p9 is the hardest instance according to our system and, while the average number of attempts for equidistant is 3.19, p9 alone has an average number of attempts larger than 12 (not shown in the table). No other instance required more attempts than p9. Despite containing the hardest instance of the study, equidistant has a significantly lower drop-out rate than witness-reverse and all-random. This is likely because equidistant slowly introduces new concepts to the learner, which is a desired property of a good curriculum.

Like p9, p10 also has the same-color division feature (see Figure 2). Perhaps the game designer saved a puzzle with this feature as a test for the player. Interestingly, our system selects an instance with such a rare feature to be part of its curriculum. A good curriculum should cover all important aspects the learner needs to know and equidistant is the only one to include an instance with this feature. One can argue p10 is part of the curriculum found in the original game; the game just enforces that the player solves other types of puzzles after solving the curriculum instances and before solving p10. In this case, the game designer chose p10 to be the last instance of the curriculum, which is similar to our system choosing p9 to be the last instance of equidistant.

While equidistant's drop-out rate is only 9% larger than witness-exact (Cohen-d of 0.19, which points to a negligible effect size), it reduces the average number of attempts performed in the test instances by more than 40% (Cohen-d of 0.51; medium effect size). Given the negligible increase in drop-out rate and the dramatic reduction in attempts, equidistant's reduced user effort in the test instances is unlikely explained by a survivor effect alone, but likely by the effectiveness of the curriculum. By comparison, all-random's drop-out rate is much larger than witness-exact (24%; Cohen-d of 0.49) and it reduces the number of attempts only from 3.95 to 3.30 (Cohen-d of 0.19). All-random has a much larger drop-out rate than witness-exact and yet, a possible survivor's bias effect in terms of number of attempts in the test instances is negligible. These results suggest that equidistant is at least competitive with witness-exact. Equidistant is more challenging than witness-exact as it covers the same-color concept not covered in the latter. As a result, it has a slightly higher drop-out rate, but it better prepares the students for test instances. These results support our hypothesis that LTS and Bootstrap can be used to generate effective curricula for human learners.

8 Conclusions

We showed how to modify the Bootstrap system to learn curricula for humans in a puzzle domain. We performed an extensive evaluation comparing two generated curricula to baselines. The first generated curriculum took a set of instances and ordered them to favor learning if the instances were solved in order. The second took a large set of problem instances and returned an ordered subset to serve as a curriculum (order-and-select problem). We evaluated our curricula and found they compare favorably with simple baseline curricula in terms of user retention and user effort when solving test instances. Our results also suggested that our system's solution to the order-and-select problem is competitive with the curriculum from the original 'The Witness' game: while our system's curriculum is more challenging than the game designer's, it better prepares humans to solve test instances.

Acknowledgements

This research was supported by Brazil's FAPEMIG, Canada's CIFAR AI Chairs program and NSERC. The research was carried out using computational resources from Compute Canada. We thank the anonymous reviewers for their comments and suggestions as well as Rob Holte for his helpful suggestions in the early stages of this project.

References

[Bengio *et al.*, 2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, page 41–48. Association for Computing Machinery, 2009.

[Cakmak and Lopes, 2012] Maya Cakmak and Manuel Lopes. Algorithmic and human teaching of sequential decision tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, page 1536–1542. AAAI Press, 2012.

[Chi *et al.*, 2011] Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1):137–180, 2011.

[Goldman and Kearns, 1995] S.A. Goldman and M.J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

[Graesser and D'Mello, 2012] Arthur C. Graesser and Sidney D'Mello. Chapter Five - Emotions During the Learning of Difficult Material. In Brian H. Ross, editor, *Psychology of Learning and Motivation*, volume Volume 57 of *The Psychology of Learning and Motivation*, pages 183–225. Academic Press, 2012.

[Hart *et al.*, 1968] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[Jabbari Arfaee *et al.*, 2011] Shahab Jabbari Arfaee, Sandra Zilles, and Robert C. Holte. Learning heuristic functions for large state spaces. *Artificial Intelligence*, 175(16):2075–2098, 2011.

[Khan *et al.*, 2011] Faisal Khan, Bilge Mutlu, and Jerry Zhu. How do humans teach: On curriculum learning and teaching dimension. In *Advances in Neural Information Processing Systems*, volume 24, pages 1–9, 2011.

[Lindsey *et al.*, 2013] Robert V Lindsey, Michael C Mozer, William J Huggins, and Harold Pashler. Optimizing instructional policies. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26, 2013.

[Orseau and Lelis, 2021] Laurent Orseau and Levi H. S. Lelis. Policy-guided heuristic search with guarantees. In *AAAI Conference on Artificial Intelligence*, pages 12382–12390. AAAI Press, 2021.

[Orseau *et al.*, 2018] Laurent Orseau, Levi Lelis, Tor Lattimore, and Theophane Weber. Single-agent policy tree search with guarantees. In *Advances in Neural Information Processing Systems 31*, pages 3201–3211. Curran Associates, Inc., 2018.

[Patil *et al.*, 2014] Kaustubh R Patil, Jerry Zhu, Ł ukasz Kopeć, and Bradley C Love. Optimal teaching for limited-capacity human learners. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, 2014.

[Rafferty *et al.*, 2011] Anna N. Rafferty, Emma Brunskill, Thomas L. Griffiths, and Patrick Shafto. Faster teaching by pomdp planning. In Gautam Biswas, Susan Bull, Judy Kay, and Antonija Mitrovic, editors, *Artificial Intelligence in Education*, pages 280–287. Springer Berlin Heidelberg, 2011.

[Reinecke and Gajos, 2015] Katharina Reinecke and Krzysztof Z. Gajos. Labinthewild: Conducting large-scale online experiments with uncompensated samples. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 1364–1378. ACM, 2015.

[Silver *et al.*, 2018] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[Taylor, 2009] Matthew E. Taylor. Assisting transfer-enabled machine learning algorithms: Leveraging human knowledge for curriculum design. In *The AAAI Spring Symposium*, pages 141–143. AAAI, 2009.

[Wald, 1980] Abraham Wald. A Reprint of 'A Method of Estimating Plane Vulnerability Based on Damage of Survivors. Technical report, Center for Naval Analyses Alexandria VA Operations Evaluation Group, July 1980. Section: Technical Reports.

[Zhu *et al.*, 2017] Xiaojin Zhu, Ji Liu, and Manuel Lopes. No learner left behind: On the complexity of teaching multiple learners simultaneously. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 3588–3594, 2017.

[Zhu *et al.*, 2018] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N. Rafferty. An overview of machine teaching. Technical Report arXiv:1801.05927 [cs.LG], ArXiV, 2018.

[Zhu, 2015] Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

[Zilles *et al.*, 2011] Sandra Zilles, Steffen Lange, Robert Holte, and Martin Zinkevich. Models of cooperative teaching and learning. *Journal of Machine Learning Research*, 12(11):349–384, 2011.