RAPID TRIAL-AND-ERROR LEARNING IN PHYSICAL PROBLEM SOLVING

Kelsey R. Allen,* Kevin A. Smith,* & Joshua B. Tenenbaum
{krallen, k2smith, jbt}@mit.edu
Brain and Cognitive Sciences, Massachusetts Institute of Technology
*These authors contributed equally to the work

ABSTRACT

We introduce a new problem solving paradigm: solving physical puzzles by placing tool-like objects in a scene. The puzzles are designed to explicitly evoke different physical concepts such as support, blocking, tipping, and launching, and are typically solved by people in a handful of trials. We study human participants' problem solving strategies, including what they try first, how they update their actions based on failed attempts, and how many attempts they eventually take to solve the puzzles. We show that a model which incorporates object-based priors to generate hypotheses, mental simulation to test hypotheses, and a system to improve sampling based on experiences across simulations and real-world trials, and show that all three components are needed to explain human performance. We additionally show that a deep reinforcement learning agent, which treats the task as a contextual bandit, fails to learn strategies across randomly generated levels.

Imagine you are going camping. While trying to set up the tent, you realize you have the stakes you must pound into the ground, but no hammer. You might search your environment for a replacement – and out of all possible objects you could use, it is more likely that you would look for a rock rather than a branch or backpack. If you fail to drive the stake in with that rock, you might search for a more suitable (perhaps heavier) rock, or try a slightly different angle of impact. Solving this problem requires sophisticated planning and learning machinery. We do not choose initial objects at random, but instead choose a rock because we know how we could use it to hit the stake (Osiurak & Badets, 2016). If our initial plan fails, we use information from that failure to guide future search.

Reinforcement learning has had extraordinary success explaining how people perform trial-and-error, incremental learning (Sutton, 1992; Whitehead & Ballard, 1991). However, traditional reinforcement learning starts from unbiased, nearly random behavior and, because it assumes little world knowledge, generally requires many attempts to succeed at a task (Sutton, 1992). This is in contrast to the rapid progress (in one or a few trials) based on rich inferences that humans often demonstrate.

We therefore focus on problems that require rich prior knowledge of the world dynamics, and where failed attempts provide useful feedback to intelligently update proposed solutions. As an example class of these problems, we introduce a "physics game" where people are asked to place objects into a scene to accomplish different objectives. These scenes unfold according to physical dynamics, and so people have the ability to predict the outcome of their actions, albeit with some uncertainty and bias (Battaglia et al., 2013; Smith & Vul, 2013). But because searching this space of possible actions is still difficult, and simulations are imperfect, feedback from failed observations can help people plan future actions.

To explain people's solutions, we propose the "Sample, Simulate, Remember" (SSR) model that includes object-based priors, a noisy physical simulation engine to propose actions, and a Bayesian optimization procedure for more efficient exploration. We show that this model solves our physics puzzles similarly to people, and that each of the three components contributes to explaining human data. We further find that solution strategies are not easily learnable from simple action-outcome pairings, and instead require knowledge of the dynamics of the world to rapidly develop.

1 EXPERIMENT

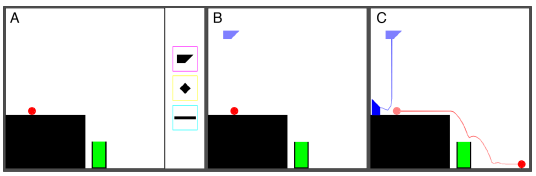


Figure 1: Diagram of a trial. (A) Participants attempt to get a red object into the green goal using one of the three tools on the right. (B) Participants choose a tool and where to place it. (C) Upon placing the tool, physics is turned on and participants see an animation of the scene unfolding. If they fail, they can reset the world and try again.

In this experiment, 94 participants played a game in which they were given a goal state (e.g., "get the red ball into the green goal"; see Fig. 1) that they had to achieve by placing one of three 'tools' into the scene. If their attempt failed, they could reset the level and try again. Solving levels in this game required using the tools in an indirect manner such as dropping them on an object to launch it or putting the tool underneath an object to support it. Because participants could only take one action per attempt, finding solutions required understanding how their actions could cause changes in the scene in the future when they were no longer allowed to intervene.

We constructed 20 levels to investigate a range of physical concepts such as 'launching' or 'catapulting' a ball, 'supporting' a table for an object to roll across, or 'preventing' an object from blocking the goal (see Fig. **??**). Of these 20 levels, 12 were constructed in 6 pairs, with small variations in the goals or objects in the scene so that we could test whether subtle differences in stimuli would lead to observable differences in behavior. Participants were asked to solve 14 problems in a random order: 8 unmatched, and one each of the 6 matched pairs.

Participants were given two minutes to solve each problem – if they solved it they could move onto the next level immediately. Otherwise, they could choose to move on any time after two minutes had passed. Within each trial, we recorded all attempted actions: which tool was used, where it was placed, and the clock time when it was placed since the start of the trial. Selected first and last actions for four of the levels can be seen in Fig. 5.

While participants improved in solution rate over the course of the experiment (76% solution rate on the first three trials to 86% on the last three; $\chi^2(1) = 9.7$, $p = 0.002$), there was no evidence that they used fewer attempts later in the experiment ($\chi^2(1) = 2.0$, $p = 0.15$); this suggests that participants were not learning about the game dynamics in a way that they could leverage for more efficient solutions. Instead, we propose that rapid trial-and-error learning relies on pre-existing physical knowledge and makes use of heuristics to perform efficient sampling based on observations. This is in contrast to model based reinforcement learning, which would show improvement based on learning an improved model of the underlying dynamics.

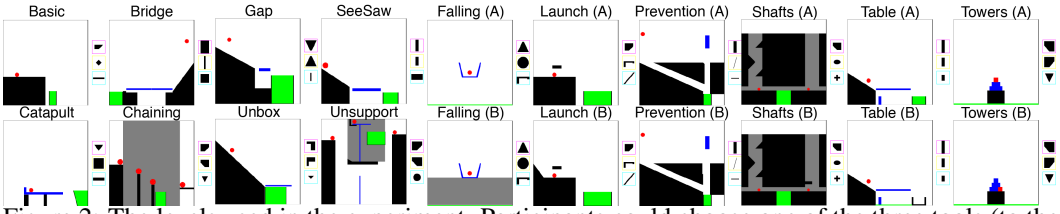


Figure 2: The levels used in the experiment. Participants could choose one of the three tools (to the right of each level) to place in the scene to get a red object into the green goal area. Levels denoted with A/B labels are matched pairs. See `https://bit.ly/2G5ZKgw` for example videos.

2 THE "SAMPLE, SIMULATE, REMEMBER" MODEL

To account for this human behavior, we propose a model incorporating object-based priors (**Sample**), a noisy simulation engine (**Simulate**), and a Bayesian optimization procedure for generating diverse action proposals based on "internal" and "external" observations (**Remember**). We show that each part is necessary to explain human behavior through ablation experiments.

Sample: object-based prior In line with other work on physical problem solving showing that object-oriented and relational priors are important (Hamrick et al., 2018), we incorporate an object-based prior for sampling actions. Since all tools in the game were designed to be unfamiliar to participants, we place a uniform prior over the three tool choices. The position of the tool was sampled according to the factored action distribution $object \sim Multinomial(\{\frac{1}{n_{obj}} | i \in [0, ..., n_{obj}]\})$,

$x, y \sim \mathcal{N}([object_x, object_y], [\sigma_x, \sigma_y])$ where n_{obj} is the number of movable objects in the scene, and *object* is the sampled object. This encourages the model to sample actions that are likely to have an effect on the scene by changing the trajectory of movable objects.

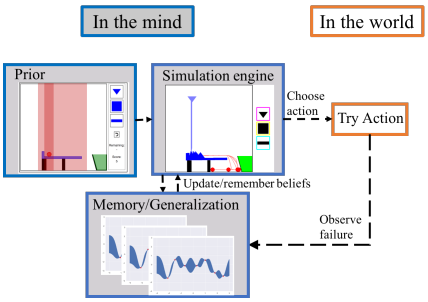


Figure 3: Sample, Simulate, Remember (SSR): a model incorporating an object-based prior, a simulation engine, and a Bayesian Optimization procedure for suggesting new proposals based on observations "in the mind" and "in the world".

Simulate: a noisy simulation engine An "intuitive physics engine" is used to evaluate proposed actions as being likely to succeed or not. Humans characteristically have noisy predictions of how collisions will resolve (Smith & Vul, 2013). We therefore incorporate two sources of collision noise into our model: collision elasticity, and collision direction.[1] In order to decide if a proposed action is worth attempting in the real world, it is stochastically simulated five times to form a set of hypotheses about how that action might affect the world. For each hypothesis, the minimum distance between the goal area and one of the objects that must get into that area is recorded; these values are averaged across the simulations and saved to memory. Since low values are indicative of actions that almost achieve the goal, if the average is below a threshold ε, the model takes that action "in the world." If the model considers more than T different action proposals without acting, it takes the best action it has imagined so far.

Remember: Bayesian Optimization for efficient sampling We implement memory via a Gaussian Process (GP) that maps actions to estimated rewards (minimum distance between the goal and relevant objects, scaled by this distance when no action is taken) in order to guide the internal sampling procedure towards promising regions and away from actions which are likely to be unsuccessful. The model samples diverse but effective action proposals from this memory using Bayesian Optimization with the Expected Improvement acquisition function (Jones et al., 1998). This acquisition function balances taking actions that are near ones already known to be promising, and exploring areas with high uncertainty. This memory incorporates both simulations "in the mind" and observations "in the world" to guide future proposals from all information.

The SSR model will continue to propose actions until one is successful; however, participants could 'fail' a level if they did not complete it within two minutes. We therefore allowed the model to take 10 actions before considering it a failed trial, in line with the number of actions participants used on unsuccessful trials (mean: 9.6, median: 10).

2.1 Alternate models

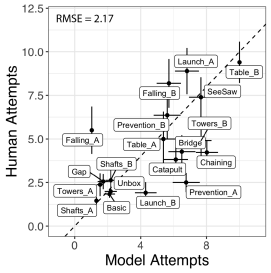
We propose that "trial-and-error" problem solving requires (1) an object-based prior, (2) a simulation engine, and (3) memory with generalization. We therefore considered ablations of the SSR model that lack these various pieces to determine their relative contributions. We find that removing any of these three modules negatively affects the model's performance.

The only tuned parameters were: ε, the threshold for acting, σ_x and σ_y, the standard deviations for the prior, and the collision noise parameters. The proposal threshold T was always fixed to 5. These were fit to maximize the likelihood of participants' first placements under the full model; this likelihood was relatively insensitive to reasonable choices for all other parameters.

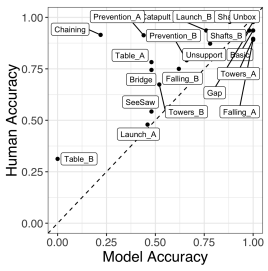
2.2 Results

Our main comparison between these models is whether they solve each level as quickly and as often as our participants did. We therefore compare the average number of attempts participants took on each level with the average model attempts using root mean squared error (RMSE) to capture both bias and variability in these comparisons. We additionally compare the model's solution rate to the human solution rate across all trials.

[1] Sensitivity analyses demonstrate that only these two sources of noise make a significant difference to the model fits.



(a) Comparison of average number of human participants' attempts for each level with average number of attempts for the full SSR model.



(b) The full SSR model's average solution rate (accuracy) for each trial compared to human solution rates. There is a high correlation between model and human accuracy: $r = 0.75$.

Model	*Avg. Pl*	$RMSE_{Pl}$	r_{Acc}
Human	4.48	-	-
Full	4.88	**2.17**	**0.75**
P+S	4.86	**2.35**	0.59
P+M	6.46	2.74	0.65
S+M	5.83	2.81	**0.70**
Prior	8.2	4.6	0.60
Guess	8.85	4.87	0.67

(c) Ablation experiments for average placements, RMSE of placements vs. people, and correlation between model and human accuracy. Model components are denoted by letters: **P** indicates prior, **S** indicates simulation, and **M** indicates memory. Bold numbers indicate the best models on each metric that cannot be distinguished based on 95% bootstrapped confidence intervals.

Figure 4: Model comparisons

The SSR model explains the patterns of human behavior across the different levels well. It uses a similar number of attempts on each level as people do ($r = 0.66$; 95% CI $= [0.56, 0.72]$; mean empirical attempts across all levels: 4.48, mean model attempts: 4.88; see Fig. 4a). It also solves each of the levels at similar rates to participants ($r = 0.75$; 95% CI $= [0.65, 0.80]$; see Fig. 4b).

As can be seen in Table 4c, eliminating either the object-based prior or the physics simulator causes a large decrease in performance, with the ablated models typically requiring many more attempts to solve levels because they are either searching in the wrong area of the action space (no prior), or attempting actions that have no chance of being successful (no physics engine). However, even the 'Prior + Simulation' model does somewhat worse at explaining the pattern of attempts and accuracy across levels, suggesting that the memory module helps guide the search in a more human-like fashion. Specifically, when the memory is ablated, correlations for the overall accuracy between the model and human participants decreases. This suggests that the memory is improving the efficiency and the diversity of the search.

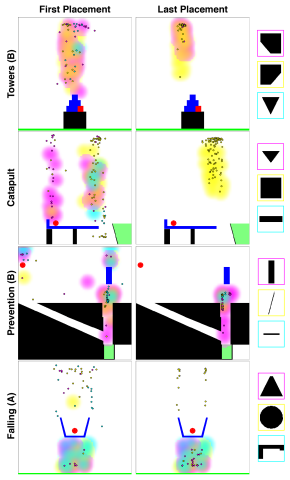


Figure 5: Distribution of predicted model actions (background) versus human actions (points) on the first attempts of the level (left) and the attempt used to solve the level (right) for a selection of four levels.

To investigate whether the model is solving these levels in similar ways to people, we can visualize where both the model and participants decide to try their first action, and what sorts of actions they use to eventually solve the level – see Fig. 5. In general, both people and the model will begin with a variety of plausible actions (e.g., Towers (B) and Catapult). In some cases, both will attempt initial actions that have very little impact on the scene (e.g., Prevention (B)); this could be because people cannot think of any useful actions and so decide to try *something*, similar to how the model can exceed its action threshold T. However, in some cases, the model's initial predictions diverge from people, and this leads to a different pattern of search and solutions. For instance, in Falling (A), the model quickly finds that placing an object under the container will reliably tip the ball onto the ground, but people are biased to drop an object from above. Because of this, the model always solves the level with an object below, whereas a proportion of participants find a way to flip the container from above; this discrepancy can also be seen in the comparison of number of attempts before the solution, where the model finds a solution quickly, while people take longer (Fig. 4a).

2.2.1 Learning strategies with deep reinforcement learning

While the SSR model assumes access to a physical simulator in order to update its actions within a level, we also investigated whether winning strategies could be learned in a "model-free" way using

standard deep reinforcement learning methods. To study this, we created level templates, which could be used to generate random levels for each of five level types: "Basic," "Catapult," "Shafts," "Table (A)," and "Table (B)" (see the Appendix and Fig. 2).

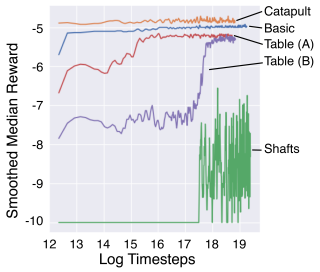


Figure 6: Performance of deep reinforcement learning approach PPO on set of training levels from specific level type.

These levels were associated with tools that were generated randomly by resizing and mirroring tools used in the human experiments. All levels were checked to ensure that they could be solved using a prototypical strategy (e.g., 'drop an object from above the ball') using at least one of the tools. We generated 1000 different random levels for each of level types.

We trained an agent using Proximal Policy Optimization (PPO; Schulman et al., 2017) to learn high reward strategies within a level type. We used the Atari architecture from Schulman et al. (2017), giving the scene and tool images as input. The reward was scaled by a factor of 5 such that a reward of -10 indicated an illegal placement, a reward of -5 corresponded to having no impact on the scene, a reward of 0 indicated success, and rewards between -5 and 0 indicated actions that would move the target object closer to the goal. The training curves for the different level types are shown in Fig.6. While PPO learns to take legal actions in each level type, it does not learn a good, generalizable strategy for that level type even after many observations.

3 Discussion

We have shown that a model incorporating an object-based prior, simulation engine, and efficient sampling procedure captures how participants' attempted strategies evolve from start to finish (see Fig. 5), and the overall difficulty of the different levels. We find that all three components of this model independently help to explain human problem solving. The prior helps the SSR model focus on areas of the action space that are likely to make a difference in the problem, just like people do. The noisy simulation engine is necessary to filter out poor action proposals that people do not choose. The Bayesian Optimization-based memory allows the model to efficiently search the hypothesis space in a more human-like way.

Each of these three components, an object-based prior, a noisy simulation engine, and memory, can be interpreted in light of classic notions of how animals and humans use objects to accomplish goals.

First, affordance-based reasoning has a long history in cognitive science (Gibson, 1979). Here we assumed a very simple notion of affordances through an object-based prior, but without any of the shape or context cues to direct how an object might be used. However, affordances are typically thought to be specific to objects and how they might be used, suggesting that more complex priors as joint probability distributions over tools, object types, shapes, and relational uses of that tool might better describe how people initialize their search.

Second, we assumed a noisy physics engine as the simulator in this model, which requires planning over the full continuous action space. However, work in robotics has developed approaches to planning in more abstract *event* spaces (Toussaint et al., 2018; Kaelbling et al., 2007) that can be grounded to continuous physical states. We believe such mechanisms could provide a way of bridging more qualitative physical simulation (Forbus, 1988) with the physics engines that simulate over continuous spaces used here.

Third, the SSR model only learns *within* levels, not across them. This was a reasonable assumption because object properties (such as density, friction, and elasticity) were identical across levels. To support this assumption, we find little learning across levels in our participants. But in a more complex version of this game where physical properties might vary across levels, memory and generalization will be important not just for learning good actions, but also for learning the changing physical properties.

Overall, we believe this domain will allow for further investigations into the underlying physical representations used for complex object-based planning and manipulation, and how these representations are updated efficiently based on interactions with the world.

REFERENCES

Peter Battaglia, Jessica Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45):18327–18332, 2013.

Kenneth D Forbus. Qualitative physics: Past, present, and future. In *Exploring artificial intelligence*, pp. 239–296. Elsevier, 1988.

James J Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin Co., 1979.

Jessica B Hamrick, Kelsey R Allen, Victor Bapst, Tina Zhu, Kevin R McKee, Joshua B Tenenbaum, and Peter W Battaglia. Relational inductive bias for physical construction in humans and machines. *arXiv preprint arXiv:1806.01203*, 2018.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

L. P. Kaelbling, H. M. Pasula, and L. S. Zettlemoyer. Learning Symbolic Models of Stochastic Domains. *Journal of Artificial Intelligence Research*, 29:309–352, July 2007. ISSN 1076-9757.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr`, 2018.

François Osiurak and Arnaud Badets. Tool use and affordance: Manipulation-based versus reasoning-based approaches. *Psychological review*, 123(5):534, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Kevin A Smith and Edward Vul. Sources of uncertainty in intuitive physics. *Topics in Cognitive Science*, 5(1):185–199, 2013.

Richard S Sutton. Reinforcement learning architectures. 1992.

Marc Toussaint, Kelsey Allen, Kevin Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Robotics: Science & Systems*, 2018.

Steven D Whitehead and Dana H Ballard. Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83, 1991.

4 APPENDIX

4.1 REINFORCEMENT LEARNING MODEL DETAILS

We trained a PPO agent based on the library available from Kostrikov (2018). We have four images for the network: one for each of the tools, and one of the environment. We run the image of the environment through the network used by Schulman et al. (2017) for playing Atari. We use a smaller convolutional network with the same number of layers for each of the tools. The network weights are shared across tools. We perform late fusion on the feature vectors output by the environment and tool networks, and then use two fully connected layers of 100 units each to predict a mean position for placement of a tool. We additionally predict categorical weights on which tool to use with a separate fully connected layer. While we considered alternative parameterizations of the action distribution, this did not appear to affect the results significantly.

The agent was trained with a learning rate of 1e-5, and PPO batch size of 128. We found that the training was sensitive to these hyperparameter choices, with larger learning rates leading to exploding gradients.

The reward was shaped in the same way as the bayesian optimization based procedure (minimal distance to the goal, divided by this distance if no action was taken). This was multiplied by a factor of 5 since it appeared to improve learning efficiency.

4.2 RANDOM LEVEL GENERATION FOR PPO

Because PPO must learn strategies from scratch, including how to represent objects, we constructed a training set to determine whether the PPO model could learn within a specific level type. To do so, we selected six levels from those we tested participants on, and built custom random levels generators for each of those levels. These generators allowed properties such as the size, position, or shape of objects to vary in ways that were hand-designed to allow for similar strategies to solve all levels of each type (see Fig. 2a for examples of the randomly generated levels).

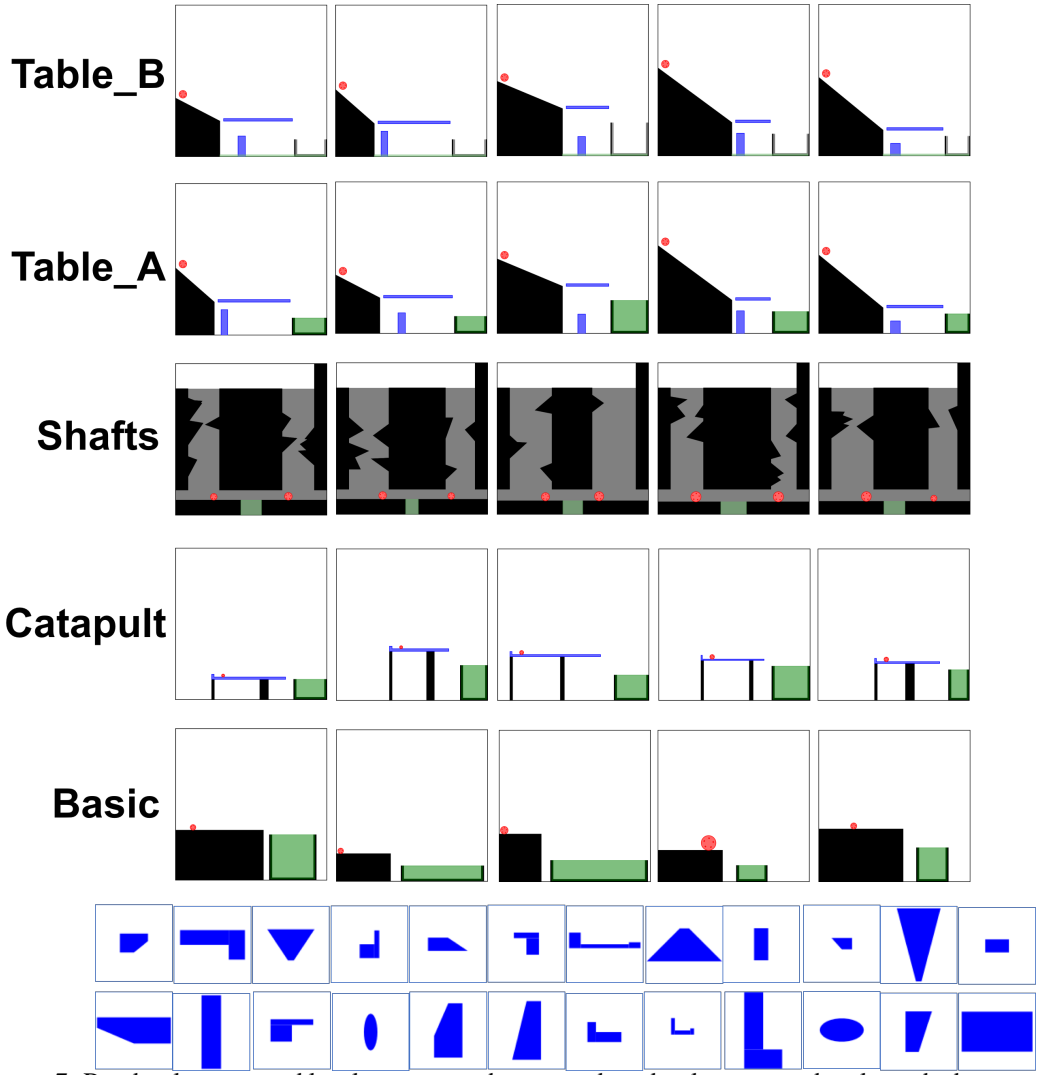


Figure 7: Randomly generated level screens on the top, and randomly generated tools on the bottom. Each row represents a different trial template. All levels are solvable with at least one of the tools available on that level.