INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022
https://doi.org/10.1017/dsd.2022.000

A REINFORCEMENT LEARNING APPROACH TO PREDICTING HUMAN DESIGN ACTIONS USING A DATA-DRIVEN REWARD FORMULATION

[Authors will be inserted automatically]

[Institutions will be inserted automatically]
[Corresponding author data will be inserted automatically]

Abstract

[Abstract will be inserted automatically]
[---]
[---]
[---]
[---]
[---]

[Keywords will be inserted automatically]

1. Introduction

With the advent of powerful machine learning algorithms, different types of intelligent agents have been developed that can solve challenging problems to reduce human labour. In the design field, although artificial agents are capable of solving well-defined design problems, human heuristics are proved to be more efficient in certain tasks such as abstract decision-making or finding intuitive explanations for design decisions (Raina et al., 2019; Sexton and Ren, 2017). Therefore, there is a significant potential to partner human design thinking with state-of-the-art computational algorithms in a human-AI collaboration framework. In such a framework, realizing an AI that can mimic human behaviours are vital in human-AI interactions because it can predict human design decisions and intervene when necessary to recommend designer to consider different design strategies. Such an AI can improve design quality, shorten design iterations, and be used to train novice designers. However, developing intelligent agents that can mimic human behaviour has many challenges. For example, representing decision-making strategies of designers with theoretical underpinning and interpretation is not an easy task. Additionally, systems design problems require a long design cycle that often involves a large number of actions, thereby identifying beneficial design patterns within such a complex design space is difficult. There are several existing approaches to developing design agents in the literature. For example, deep-learning agents have been developed to predict the next possible design action based on historical data on design actions (Bayrak and Sha, 2020; Rahman *et al.*, 2020, 2021) or on images representing design sequences (Raina et al., 2019). Also, methods have been developed for recommending appropriate design procedures to novice designers (Fuge et al., 2014). To mimic human search strategy from design crowdsourcing data, artificial agent-based inverse learning methods with Bayesian optimization have been developed (Sexton and Ren, 2017). Several other studies have focused on explaining human strategies (Egan and Cagan, 2016; Rahman et al., 2018). These strategies are later used to improve design optimization algorithms (McComb et al., 2016). These previous studies, however, either focus

on using historical data to provide design feedback or extracting strategies from all designers to identify the average design patterns. The design agent developed from these methods could not provide optimal feedback. To build an effective agent for design recommendation and human-AI collaboration, a method is required that can learn beneficial design strategies while mimicking human design behaviours. Reinforcement learning (RL) (Sutton and Barto, 2018), a branch of machine learning, has gained success in developing intelligent agents in many areas, such as video games, self-driving cars, design optimization, etc. However, its potential is not yet fully explored in design behavioural modelling.

The **objective** of this study is to develop a reinforcement learning (RL)-based approach to constructing design agents that can mimic human behaviours in support of human-AI design collaboration. To achieve this, we integrate prior human data into the self-learning capability of RL. More precisely, we utilize Temporal-Difference (TD) based Q-learning method, the most popular method in RL (Jang et al., 2019), which uses a combination of the Monte Carlo method for measuring Q-value through iterations and the advantages of dynamic programming. TD learning uses a mathematical procedure that tries to predict the combination of immediate reward and its own reward prediction at the next step. To integrate human design strategies into our RL agent, we introduce a data-driven reward mechanism utilizing first-order Markov chains. The novelty lies in the integration of this data-driven reward into what is traditionally a self-learning algorithm. The predictive performance of the proposed RL agent is tested in a case study on a solar system design problem.

The rest of the paper is organized as follows: In Section 2, we present the technical background of the RL (i.e., Q-learning method). Section 3 contains the design experiment procedure and data collection method. Section 4 starts with describing the RL model formulation, then it explains the model setup used to evaluate the performance of the RL agent. The result and the corresponding discussions are presented in Section 5. The paper is concluded by discussing possible future work in Section 6.

2. Technical Background

Typical RL approaches rely on the formalism of a Markov Decision Process (MDP) to learn optimal behaviours in sequential decision-making problems. The goal of an MDP is to find the optimal policy for design decisions based on a reward. Q-learning helps to find such a policy by generating a Q-value for each state-action pair that is used to determine the best design action for a given state of the design process. The Q-values of all state and action pairs are typically stored in a Q-table that is learned through multiple iterations in an RL process. At the beginning, we initialize the Q-table with zeros as the state-action dynamics are not known at this phase. In the proceeding steps, the agent selects an action either by exploitation or exploration. At the initial phase, the agent does not have much information from the Q-table and mostly explores the action space by taking random actions. The agent updates the Q-table based on the reward it receives from these random actions. Once the agent performs an adequate number of iterations and collects information about the environment, it begins to exploit. This trade-off between exploitation and exploration can be handled by the epsilon-greedy algorithm. In short, like in a typical design process, at the initial stages, we give more preference to exploration to concepts generation, while in the later stages, exploitation is preferred. Once the agent selects an action, it reaches a new state, S'. In the new state, the design agent selects the best possible action from the maximum Q-value and finds the corresponding rewards. Based on the rewards, the Q-value is updated according to the following equation:

$$Q_t(S,A) = Q_{t-1}(S,A) + \alpha\big(R(S,A) + \gamma \max Q(S',A') - Q_{t-1}(S,A)\big) \quad (1)$$

where, $Q_t(S,A)$ is the new Q-value for the state S and action A for the next iteration t. $Q_{t-1}(S,A)$ is the current Q-values. α is the learning rate - a hyperparameter representing the weight between the new information acquired in the current iteration *vs.* the old information from previous iterations. When α is close to zero, Q-values are never updated; whereas α close to 1 means that learning occurs quickly. $R(S,A)$ is the value of the reward for taking action A at state S. γ is the discount factor that quantifies how important the future rewards are played in updating the Q-value. $\max Q(S',A')$ is the maximum expected future value. The agent will iterate over multiple steps to update Q-table values till convergence. In this paper, we adopt a probabilistic model for action transition. The agent chooses one of the actions with the following probability function based on the Q-values (Wu et al., 2021),

$$\Pr(a|s) = \frac{\exp(\theta \cdot Q(s,a))}{\sum_{a_i \in A} \exp(\theta \cdot Q(s,a_i))}, \quad (2)$$

where A denotes the action space for an agent. The equation takes values from the Q-table and provides a probability for taking each possible action (a) at a given state (s). The hyperparameter $\theta \in [0, \infty)$ determines the agents' decision-making strategy. When θ is zero, the equation provides uniform distribution (i.e., all design actions are equally likely to be selected). When θ goes to ∞, the probability of the action with the highest Q-value (most frequently occurring design action at a given state) approaches to 1. Note that this model is similar to the logit-based choice models commonly used in the design and marketing literature (Gensch and Recker, 1979) where Q-values correspond to the utilities of discrete choices.

3. Experiment and Data Collection

We use the design problem of building a solarized home in Dallas to collect human design behavioural data. The objective of this design problem is to maximize the annual net energy (ANE) while minimizing cost. The overall budget for this design problem is $200,000. In addition, we have set specific design constraints and design requirements as summarized in Table 1. This system design problem involves many design variables with complex coupling relationships among these variables (e.g., designers may want to add many solar panels for higher ANE, however, the distance between solar panels could not be too small so there is a limit for the number of solar panels to be put). For this reason, the design space is large, and designers take different exploitation and exploration strategies during the design process.

To collect data from designers, human subject experiments were conducted in the form of a design challenge. The design task is conducted on Energy3D, a computer-aided design (CAD) software for the solar design problem (Rahman *et al.*, 2019; Xie *et al.*, 2018). Energy3D has several unique features such as interactive visualization, high-fidelity simulation, and built-in evaluation. Additionally, Energy3D collects data in a non-intrusive process where designers are not aware of the data collection. The non-intrusive data collection process could reduce the cognitive bias during an experiment. Energy3D logs design data at a fine-grained level. Particularly, it logs every performed design action and collects design artefacts in every 20 seconds. Therefore, data collected from Energy3D fully capture what designers do (i.e., design actions) throughout the design process.

Table 1. Design requirements

Design variables	Design requirements
Number of floors	1
Height of wall	≥ 2.5 m
Number of windows	≥ 4
Size of window	≥ 1.44 m^2
Number of doors	≥ 1
Size of doors ($width \times height$)	≥ 1.2 $m \times 2\ m$
Distance between edge/ridge and solar panel	≥ 0

A total of 52 designers from the University of Arkansas participated in the design challenge. The participants are indexed according to their registered sessions and the laptop numbers. Sessions are indexed by the letters from A to G and laptops are indexed with numbers. For example, A02 indicates that the participant joined at session A and worked on the laptop number 2.

Energy3D collects the design process data as JSON format which includes time-steps, design actions, design artefacts, and simulation results. On average on participant has about 1500 lines of design process data. An example of one line of the design action log is presented below:

{"Timestamp": "2017-12-01 12:58:27", "File": "EnergyPlusHome.ng3", "Edit Door": {"Type": "Door", "Building": 2, "ID": 9, "Coordinates": [{"x": -7.5, "y": -32.5, "z": 1}, {"x": -7.5, "y": -32.5, "z": 15}, {"x": -1.5, "y": -32.5, "z": 1}, {"x": -1.5, "y": -32.5, "z": 15}]}}

In this study, we extract only the design actions related to design objectives, such as "Add wall", "Edit wall", "Edit roof", "Show sun path" etc. We ignore the design actions that have no effect on design outcomes such as "Camera" and "Add tree". This post-processing leads to 115 unique design actions.

4. Research Approach

4.1. Formulating the Reinforcement Learning Model

We define the RL components, i.e., states, actions, rewards, in the context of the design problem as below:

States: State is the current situation in which the agent interacts with the environment. In this study, since our goal is to mimic human design behaviours, we define the states in RL as the state of the designers' thought process during design. Various ontological models have been proposed to represent design processes and interpret design thinking (Gero and Kannengiesser, 2014). In this study, the proposed state representation model is inspired by the function-behaviour-structure (FBS) design process model. We referred to the FBS model because it is a design ontology that can be used to represent a general system design problem. However, the original FBS model is not able to fully describe the design process in CAD environment. Therefore, the FBS model was later extended in the CAD context (Kannengiesser *et al.*, 2009) where a major inclusion is to add the interpretation sub-process. So, we added *interpretation* in our model of representing the design thinking states. In total, we defined six design thinking states in CAD that include Formulation, Reformulation, Synthesis, Interpretation, Evaluation, and Analysis. These states are considered as the states for RL. From the collected design data, we observe that designers can perform the inter-state movement. For example, suppose a designer is in a state where they add a wall. After that, they move to another state where they have various options to perform such as editing the wall, analysing it, assessing the cost, or removing it. Using these options, designers can move from one state to any other state, which produces a fully connected network (Figure 1).

Actions: The actions in our RL problem are the design actions performed by designers. We observe that designers can perform similar categories of actions in each state.

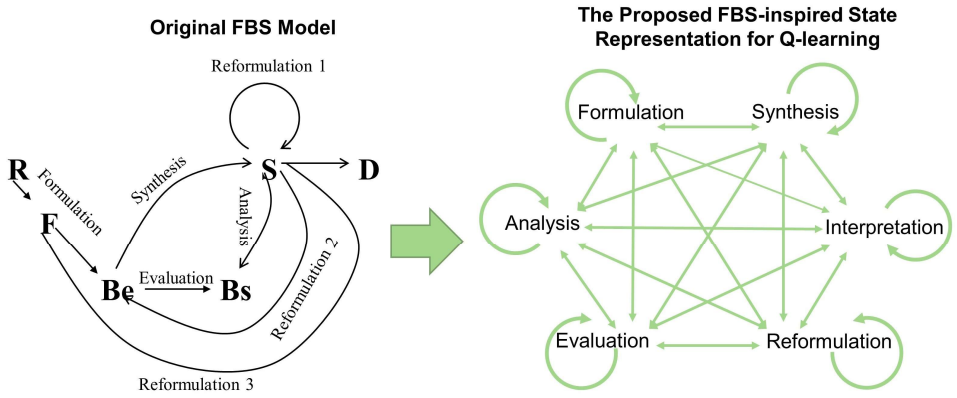

Figure 1. The FBS design process model (Kannengiesser et al., 2009) and the design thinking states defined in the proposed reinforcement learning model

For example, designers can perform different addition tasks (i.e., add a wall, add a window, etc.), edit tasks (i.e., edit wall, edit solar panels etc.), remove tasks (i.e., remove wall, remove solar panel, etc.). So, we group such similar design actions into a few categories. There are three reasons of doing this.

First, categories allow capturing the context-independent essence of design actions and provide better generalizability. Second, these categories significantly decrease the number of possible state-action pairs and reduce the computational burden during the training of RL agents. Finally, in our previous human-computer interaction experiments (Rahman *et al.*, 2020), where we train a deep learning model to recommend design actions to designers, we have found that designers feel interrupted if they were provided with a detailed suite of actions. Therefore, grouping design actions into a small number of categories provide a more condensed set of design recommendations in a potential extension of this study to human-computer collaboration. In this study, we identify six unique categories of design actions in the experimental data. These categories and the corresponding actions are listed in Table 2

Table 2. Design action categories and their corresponding actions

Design action category	Action Category	Example
Addition of any components	Add	Add wall, Add solar panel, etc.
Edit of any components	Edit	Edit door, Edit wall, etc.
Environmental check	Show	Show Helidon, Show sun path, etc.
Evaluation of cost	Cost	Cost
Removal of any components	Remove	Remove window, Remove roof, etc.
Analysis of annual net energy	Analysis	Energy Annual Analysis

Reward: The reward is the feedback from the environment. The RL agent aims to maximize the total reward that is calculated by summing all the immediate rewards. However, this sum can potentially grow indefinitely. Therefore, a discount factor (γ) is included in the reward function to reduce the contribution of the future rewards. The reward can be expressed as follows:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots \quad (3)$$

Traditional RL is a self-learning method that uses a reward from the environment. As our target is to build an agent that mimics human designers, we use the data containing designers' actions in the solar system design experiment (e.g., those shown in Table 2) to generate a reward table. Combining this data-driven reward with the self-learning capability of RL is a unique aspect of this study. Figure 2 shows our overall approach to training an RL agent to mimic a human designer. We employ the first-order Markov chain model on the designers' sequence to construct the reward table. From the first-order Markov chain, we obtain the transition probability matrix for each designer. Then, we average all the transition probability matrices from all designers to obtain a final reward table. This Markov chain-based reward mechanism reinforces most frequently appearing action pairs during training.

Table 3. An example of the reward table

	Formulation	Analysis	Reformulation	Synthesis	Evaluation	Interpret
Formulation	22.47	1.25	4.15	21.25	1.83	1.06
Analysis	3.28	9.37	3.91	17.67	13.88	3.89
Reformulation	16.61	2.48	17.28	12.11	2.54	0.97
Synthesis	6.02	3.20	2.70	35.24	4.15	0.69
Evaluation	3.97	14.35	3.85	21.46	3.31	2.05
Interpret	5.39	4.62	2.30	10.48	2.057	18.14

Table 3 shows an example of a reward table obtained from the data. Each entry indicates the reward for transitioning from one state to another state. For example, if designers stay in Formulation state between two actions, the corresponding reward value is 22.47. From the reward table, a Q-table can be obtained from Equation (1). Recall that we do not use the highest value in the reward table to predict next design action but rather use all the reward values to produce a Q-table. Based on the Q-table and a given θ value, we create probability distributions to predict design actions using Equation (2).

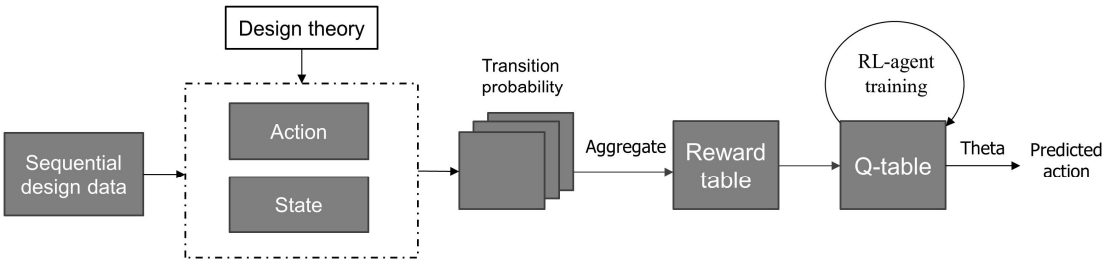

Figure 2. The overall research approach

4.2. Model setup and evaluation

Baseline: We choose the first-order Markov chain model as our baseline model to compare with the Q-learning agent. From the first-order Markov chain, we obtain a transition probability matrix (Rahman *et al.*, 2018) for each designer in the dataset. We aggregate transition probability matrices from $(n-1)$ designers and predict on the n^{th} designer's sequence. By iterating this process, we obtain the prediction accuracy for all designers in the data and use the average as the final prediction accuracy. The average prediction accuracy is about 41%.

Cross-validation: We use the k-fold cross-validation technique to evaluate the Q-learning agent. In this method, we split the dataset into k partitions. Then, k rounds of training and testing are performed in a way such that in each iteration, $(k-1)$ partitions are used to obtain the reward table and train the Q-learning agent. The rest of the partition is used to test the Q-learning agent. In this study, 6 designers are used as test data and the rest of them as training. This process iterates through the entire dataset. In this way, we perform a total of 11 rounds of training and testing.

Parameter settings: We determine the optimal settings for the hyperparameters of the RL model using trial and error and train the Q-learning agent based on those settings. We choose the learning rate (α) of this study as 0.3 and the discount factor (γ) as 0.6 for updating the Q-value (see Equation (1)). We update the Q-table by 10,000 iterations. To obtain the next possible action from the Q-table with the best accuracy, we also tune the value of θ. Note that the optimal settings for the model parameters will be different in other applications and should be tuned again based on the input data.

Metric for prediction accuracy: We compare the agent's generated sequence with the actual sequence to evaluate the Q-learning agent. The agent generates the next sequence based on the previous action. As the agent chooses the final design action from a probability distribution, the prediction can vary from iteration to iteration. Therefore, to account for the stochasticity, we run a total of 50 realizations to generate each sequence. In each sequence, we compare the predicted actions with the actual decisions in the data, count the number of correctly predicted actions, and divide them by the total length of the sequence.

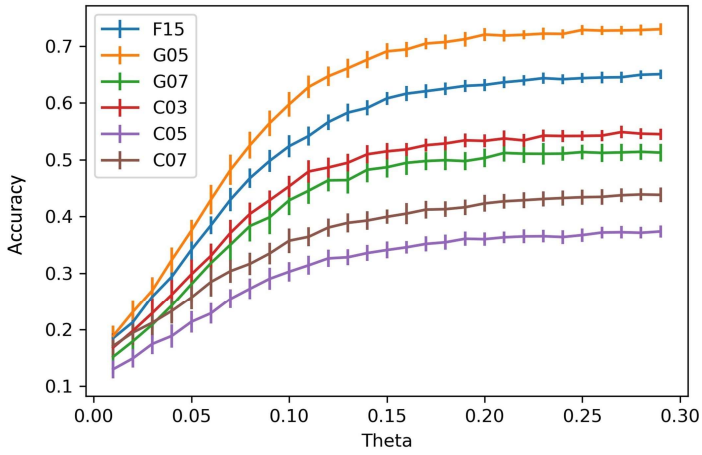

Figure 3. Prediction accuracy over different theta for a fold

Finally, we take the average of the 50 prediction accuracies. In this way, we obtain the final prediction accuracy using the following equation

$$Prediction\ accuracy = \frac{1}{50}\sum_{i=1}^{50}\left(\frac{n_i^{cp}}{L}\right) \quad (4)$$

where n^{cp} is the correctly predicted actions and L is the length of the designer's sequence.

5. Result and discussion:

Results obtained from different folds indicate that the prediction accuracy improves with the increase in θ, as shown in Figure 3. Initially, when θ is close to zero, the RL agent provides uniform distribution to select the next action at a given state, resulting in a random search. However, with the increase in θ, the probability of the reinforced action pairs identified in the data increases. Therefore, the accuracy of predicting the next design action also increases compared to a random search. Figure 3 shows a sample for the prediction accuracy from one of the folds out of 11 for the individuals F15, G05, G07, C03, C05, and C07. The prediction accuracy increases from $\theta = 0$ to 0.25. After that, the prediction accuracy does not increase significantly and saturates to its final value for all the design sequences tested. Among all the designers, G05 achieves the highest prediction accuracy of 73%, higher than the baseline prediction accuracy of 41% achieved by the Markov chain model. We also observe that in several folds, the prediction accuracy obtained from the maximum theta for a few designers is lower than the baseline accuracy. These results indicate that the agent reinforces specific design patterns (i.e., Edit-Edit, Edit-Analysis, etc.) and provides better accuracy for those designers who behaved such patterns but yield lower accuracy for those who did not.

The k-fold cross-validation results inspire us to investigate further the prediction accuracy based on the designers' performance. We define design performance in the following equation:

$$Design\ performance = \frac{ANE \times Budget}{Cost} \quad (5)$$

We pick ten highest and lowest-performing designers and compare their prediction accuracy. We identify all the designer's prediction accuracy in each group by training the rest of the designers. This means when considering the highest performing 10 designers, we train the reinforcement agent on the rest of the 42 designers and test on these 10 designers. We follow the same procedure for the lowest-performing 10 designers.

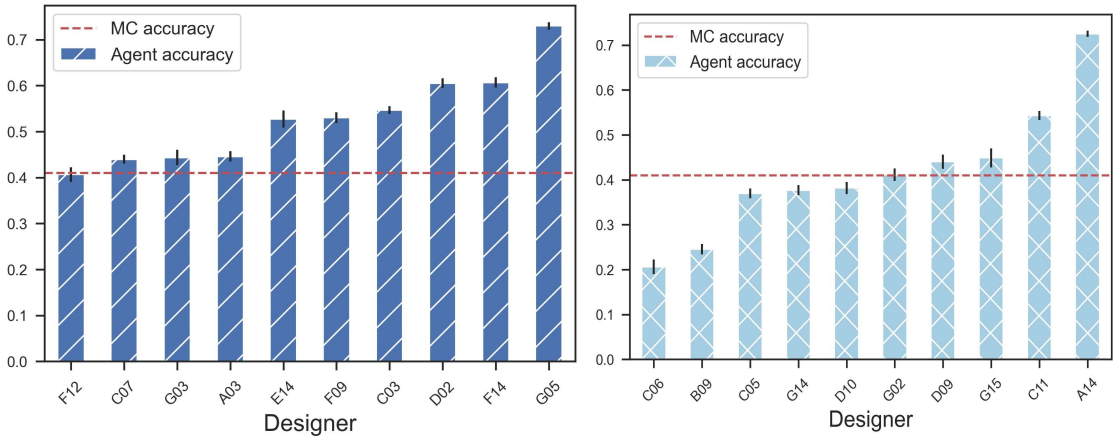

Figure 4. Prediction accuracy of the high-performing design group (left) and the low-performing design group (right)

Figure 4(a) shows the prediction accuracy for the high-performance group. In this group, nine designers out of ten achieve high prediction accuracy (>41%) compared to the Markov chain model. While in the low-performance group, only four designers' prediction accuracy is higher than the Markov chain prediction accuracy. It is worth mentioning that the highest performer among all the designers, G05,

also achieves the highest prediction accuracy. To understand the design strategy, we compare the transition probability matrix of the highest (G05) and lowest performer (F12) in the high-performance group. Figure 5 shows the heatmap of the transition probability matrix. Bigger circles indicate a higher transition probability while smaller circles indicate a lower transition probability. The highest performer G05 follows only specific design patterns and is very focused on a few particular actions while the lowest performer F12 in this group uses a variety of design patterns. For example, the highest three transition probabilities for G05 are "Edit-Edit" (0.82), "Edit-Add" (0.71), "Edit-Analysis" (0.58). The transition probabilities of these patterns for F12 are 0.57, 0.46, and 0.25 respectively. Additionally, G05 did not use 'Analysis-Analysis', 'Cost-Remove', or 'Remove-Show' action pairs at all during the process. However, F12 had these patterns during the design process. Although both designers achieve good design performance as they were both in the good-performing group, G05 finished the design task by exploiting a few specific design patterns while F12 explored different patterns to reach the objective. The RL agent learns a particular set of consistent action pairs during training due to the reinforcement of frequently occurring behaviours, thus the prediction accuracy increases if the designers follow such a consistent design behaviour pattern.

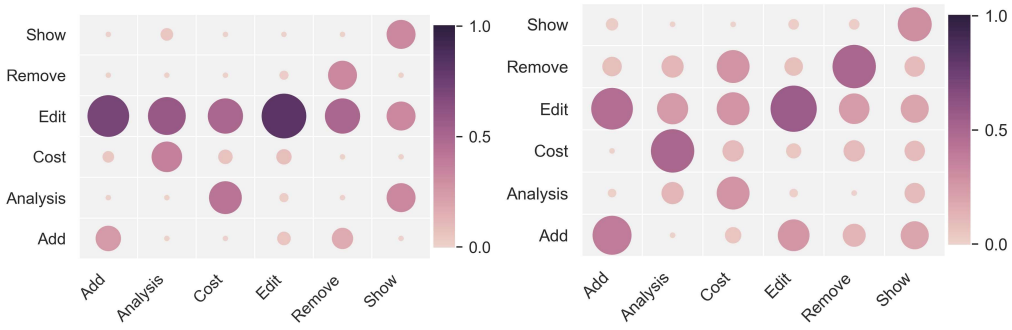

Figure 5. The transition probability of G05 (left) and F12 (right)

In the low-performance group, most of the designers also use a variety of design patterns to explore the design space. The individual A14 achieves the highest prediction accuracy in this group. Similar to the highest prediction accuracy (G05) achieved in the high-performance group, A14 also achieve the highest performance in the low-performance group. Figure 6 shows the transition probability matrix of A14. Similar to G05, A14 also uses specific design patterns during the design process, but the design performance achieved by A14 is lower than G05. This may be attributed to the fact that A14 uses several redundant design action pairs. For example, the transition probability of using "Analysis-Analysis" for this designer is high but not necessary. This is because once "Analysis" (the analysis of ANE) is conducted, there is no need to perform the analysis again in the next action.

Figure 6. The transition probability of A14

Additionally, this designer did not use “Show” at all which indicates that the designer is not interacting with the CAD environment and is not active in learning the solar science concepts underpinning the design problem.

This result indicates that though high-performing and low-performing designers may have similar prediction accuracy, their design strategies could be different albeit consistent throughout the process. Here consistency refers to the behavioural pattern those designers use the same strategies (e.g., a pair of actions) over and over. For example, designers frequently use "Formulation→Edit" and "Edit→Edit", and they follow these strategies consistently even if they may not be improved design objectives. This is also congruent with the findings in the literature. Burnap et al., 2015 showed that both experts and consistently wrong non-experts can present such behaviours. But it is worth noting that our models do not rely only on consistency but also the previous design action data to predict future actions.

Furthermore, to identify the model performance on predicting the entire sequence instead of merely predicting the next immediate action, we tested on G05 by training the model on the rest of the dataset. In this case, we predict the next design actions based on the previously *predicted* actions instead of true actions. Figure 7 shows the prediction accuracy of designer G05 on the entire sequence. The prediction accuracy saturates at about 0.7 after $\theta = 0.25$, which is similar to the prediction accuracy identified from the previous setting. In other words, our model accurately predicts 70 actions on average in a sequence of 100 actions. The result indicates that the model is capable of predicting not only the next design actions but also the entire sequence for designers with prominent design patterns.

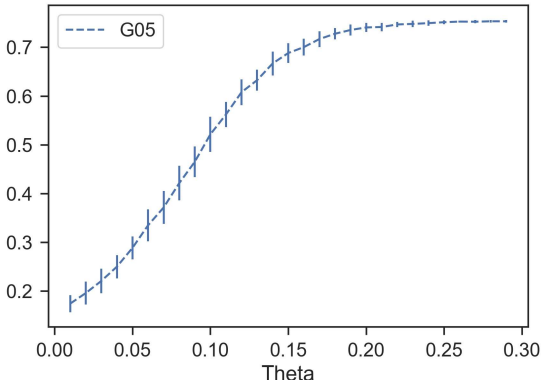

Figure 7. Prediction accuracy of G05 for the entire sequence

6. Conclusion

In this study, a design agent is developed based on Q-learning, which is a commonly used method in the reinforcement learning literature, to mimic human design strategies. The overall contribution of this study is twofold. First, the model trained using a reinforcement learning algorithm is novel in human behaviour exploration. We train this model by leveraging a data-driven reward mechanism based on the first-order Markov chain model. Once the model is trained, it chooses design actions from a probability function controlled by a specific model parameter, θ. To the best of our knowledge, this is the first study using reinforcement learning to understand human design strategies. Second, this model provides several important design behaviours. For example, increasing the model parameter θ also increases the prediction accuracy. Additionally, in most of the cases, compared to the Markov chain baseline model, the agent provides better prediction accuracy for high-performing designers. We also observe that certain strategies/patterns differentiate some of the high-performing designers from low-performing designers. The design agent learns those design patterns and provides higher prediction accuracy for most high-performing designers. There are some limitations of this study. First, this approach does not evaluate design strategies but rather predicts future design actions and can identify beneficial design patterns for improving design objective. To implement this model in a human-AI collaboration architecture, we will add a second layer to assess whether the predicted design action leads to good or bad decision in each step and intervene accordingly. Second, although we train the agent with average design behaviour data, the prediction accuracy is higher only for high-performance design groups when

these designers follow a consistent design pattern. In a future study, we aim to increase the type of design behaviour that is learned by the agent by dividing the design sequence into different stages based on the nature of a design process and training the model in each stage. We will also conduct a comprehensive sensitivity analysis to understand the effect of hyperparameters in the model.

Acknowledgment

The authors gratefully acknowledge the financial support from NSF-CMMI-1842588.

References

Bayrak, A.E. and Sha, Z. (2020), "Integrating Sequence Learning and Game Theory to Predict Design Decisions Under Competition", Journal of Mechanical Design, Vol. 143 No. 5, available at: https://doi.org/10.1115/1.4048222.

Burnap, A., Ren, Y., Gerth, R., Papazoglou, G., Gonzalez, R. and Papalambros, P.Y. (2015), "When Crowdsourcing Fails: A Study of Expertise on Crowdsourced Design Evaluation", Journal of Mechanical Design, Vol. 137 No. 3, available at: https://doi.org/10.1115/1.4029065.

Egan, P. and Cagan, J. (2016), "Human and computational approaches for design problem-solving", Experimental Design Research, Springer, pp. 187–205.

Fuge, M., Peters, B. and Agogino, A. (2014), "Machine Learning Algorithms for Recommending Design Methods", Journal of Mechanical Design, Vol. 136 No. 10, available at: https://doi.org/10.1115/1.4028102.

Gensch, D.H. and Recker, W.W. (1979), "The Multinomial, Multiattribute Logit Choice Model", Journal of Marketing Research, SAGE Publications Inc, Vol. 16 No. 1, pp. 124–132, available at: https://doi.org/10.2307/3150883

Gero, J.S. and Kannengiesser, U. (2014), "The Function-Behaviour-Structure Ontology of Design BT - An Anthology of Theories and Models of Design: Philosophy, Approaches and Empirical Explorations", in Chakrabarti, A. and Blessing, L.T.M. (Eds.), , Springer London, London, pp. 263–283.

Jang, B., Kim, M., Harerimana, G. and Kim, J.W. (2019), "Q-Learning Algorithms: A Comprehensive Classification and Applications", IEEE Access, Vol. 7, pp. 133653–133667.

Kannengiesser, U., Gero, J.S., De Smet, C.M. and Peeters, J.A. (2009), "An ontology of computer-aided design", Computer-Aided Design Research and Development, In Computer-Aided Design Research and Development. Hauppauge, NY, USA: Nova Science Publishers.

McComb, C., Cagan, J. and Kotovsky, K. (2016), "Drawing Inspiration From Human Design Teams for Better Search and Optimization: The Heterogeneous Simulated Annealing Teams Algorithm", Journal of Mechanical Design, Vol. 138 No. 4, available at: https://doi.org/10.1115/1.4032810.

Rahman, M.H., Gashler, M., Xie, C. and Sha, Z. (2018), "Automatic clustering of sequential design behaviors", Proceedings of the ASME Design Engineering Technical Conference, Vol. 1B-2018, available at: https://doi.org/10.1115/DETC201886300.

Rahman, M.H., Schimpf, C., Xie, C. and Sha, Z. (2019), "A Computer-Aided Design Based Research Platform for Design Thinking Studies", Journal of Mechanical Design, Vol. 141 No. 12.

Rahman, M.H., Xie, C. and Sha, Z. (2021), "Predicting Sequential Design Decisions Using the Function-Behavior-Structure Design Process Model and Recurrent Neural Networks", Journal of Mechanical Design, pp. 1–46, available at: https://doi.org/10.1115/1.4049971.

Rahman, M.H., Yuan, S., Xie, C. and Sha, Z. (2020), "Predicting human design decisions with deep recurrent neural network combining static and dynamic data", Design Science, Cambridge University Press, Vol. 6, p. e15, available at: https://doi.org/10.1017/dsj.2020.12

Raina, A., McComb, C. and Cagan, J. (2019), "Learning to design from humans: Imitating human designers through deep learning", Journal of Mechanical Design, Vol. 141 No. 11, available at: https://doi.org/10.1115/1.4044256

Sexton, T. and Ren, M.Y. (2017), "Learning an Optimization Algorithm Through Human Design Iterations", Journal of Mechanical Design, Vol. 139 No. 10, p. 101404, available at: https://doi.org/10.1115/1.4037344

Sutton, R.S. and Barto, A.G. (2018), Reinforcement Learning: An Introduction, A Bradford Book, Cambridge, MA, USA.

Wu, H., Ghadami, A., Bayrak, A.E., Smereka, J.M. and Epureanu, B.I. (2021), "Impact of Heterogeneity and Risk Aversion on Task Allocation in Multi-Agent Teams", IEEE Robotics and Automation Letters, Vol. 6 No. 4, pp. 7065–7072, available at: 10.1109/LRA.2021.3097259

Xie, C., Schimpf, C., Chao, J., Nourian, S. and Massicotte, J. (2018), "Learning and teaching engineering design through modeling and simulation on a CAD platform", Computer Applications in Engineering Education, Vol. 26 No. 4, pp. 824–840, available at https://doi.org/10.1002/cae.21920.