Application of Markov Decision Process and Learning in Structure Design

Wanzheng Zheng*
University of Illinois, Urbana Champaign

I. Nomenclature

$\mathcal{A}$	=	Action Set
$\mathcal{S}$	=	State Set
$\mathcal{R}$	=	Reward Set
$\mathcal{P}$	=	Transition Probability Set
$\mathcal{N}$	=	Node Set for a Design
$\mathcal{M}$	=	Truss Element Set for a Design
$\mathcal{O}$	=	Operator Set
a_t	=	Action at Time t
I_{xx}	=	Moment of Inertia
$q(s_t, a_t)$	=	Expected Total Reward of State-Action Pair
$Q(s_t, a_t)$	=	Estimation of Expected Total Reward of State-Action Pair
$r(s_t, a_t)$	=	Reward of State-Action Pair
s_t	=	State at Time t
s_T	=	Terminal State
$u(s)$	=	Deformation of Design at State s
u^*	=	Ultimate Deformation
w_i	=	Weighting Factor
$W(s)$	=	Weight of Design s
ρ	=	Length-Specific Density
ϵ_t	=	Greedy Factor, Probability that the agent will play exploit over explore at time t
π	=	Policy

II. Introduction

Additive manufacturing are receiving unprecedented attentions in multiple disciplines. In the context of aerospace engineering, additive manufacturing has enabled high performance structures that are unmanufacturable through conventional means[1]. Additive manufacturing also allows for rapid prototyping and manufacturing of components without the requirements of specialized tooling: An unfortunate but highly relevant example is the Russian-Ukraine conflict. Both sides uses Unmanned Aerial Vehicles (UAVs) extensively, and the introduction of 3-D printers allows field modifications and resupply of UAVs based on the ever changing requirements of the battlefields [2].

Design of 3D printed structures is described as synthesis of a lightweight structure than can withstand prescribed loads. This topic has been widely visited from a multitude of disciplines, and with the positive outlook of additive manufactured UAV market, related work would certainly enjoy even widen attention [3]. Goh et. al [4] reviewed state-of-the-art AM technologies and their applications in UAV manufacturing and pointed out that AM can not only provide superior vehicle performance, but also offer solutions to challenges within the UAV life cycle such as logistics and operating cost. Kazakis et. al [5] discussed the use of topology optimization techniques in 3-D printed structures by solving for optimal density distribution of materials in a discretized design space. Feng et.al [6] reviewed the recent techniques used in design of 3-D printed structures. The general approach can be grouped into two categories: optimizing material density in the design domain and optimizing placement of seed structures in the design domain.

*Graduate Research Assistant, University of Illinois, Urbana Champaign

Structures produced by Fused Deposition Modeling (FDM) 3-D printers is a travel paths of the nozzle on which the printer deposit material, giving the structure a natural skeleton-joint nature. In the context of most aerospace applications of 3D printing, the outer mold line is defined by aerodynamics analysis bounding the design space. Therefore a 3D printed structure design problem can be naturally mapped onto a finite graph defined by connections of a set of vertices and edges. It is to the author's surprise that there are few publications of design optimization problems for 3-D printed structures that utilizes this skeletal nature. Ororbia and Warn [7][8] and Lipson [9] discussed the design synthesis of skeletal structures using Markov Decision Process (MDP) and Machine Learning. In Ororbia and Warn's publication, multiple examples are provided on designs of beam-like structures to minimize deformation under loading.

This study aims to recreate the problem formulation of Ororbia and Warn [7] with objectives functions that are more related to 3-D printed structure design and optimization in aviation context, and discuss its scalability toward wider applications beyond beam-like structures.

III. Markov Decision Process Framework

Formulation of this problem follows that proposed by Ororbia and Warn [7]: The design problem of on a finite graph can be described by an MDP: $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$. Where a state $s \in \mathcal{S}$ represents a feasible design, an action $a \in \mathcal{A}(s_t)$ represents a modification taken from s_t to s_{t+1}. $P(s_t, a, s_{t+1}) \in \mathcal{P}$ represents the transitional probability of transitioning from s_t to s_{t+1} with a certain action a. $r(s_t, a_t) \in \mathcal{R}$ represents the performance of the design represented by state s_t. Each element of this MDP problem will be explained in detail in the following subsections in the context of an example.

A. Minimal Working Example

Suppose we would want to design the internal structures of an 2D cantilever beam that is clamped at one end and experience a downward force on the other end. A free body diagram of the design problem, as well as the descritized design space is illustrated in Fig 1. The $n_i \in \mathcal{N}$ is the name for each node in design space. A truss is defined by the labeling of its two bounding nodes. For instance the top truss of the cantilever beam is defined by $m_{1,3}$.

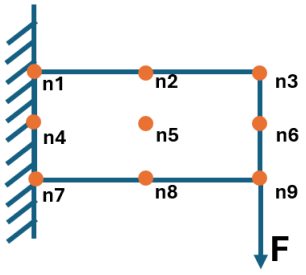


Fig. 1 Free Body Diagram of the Cantilever Beam Design Problem on a 3 x 3 Design Space

$$\begin{aligned} \min_{s} \quad & W(s) = \sum l(\mathcal{M}_s) \times \rho \\ \text{s.t.} \quad & s \in \mathcal{S} \\ & u(m) \leq u^* \quad \forall m \in \mathcal{M}_s \end{aligned} \tag{1}$$

B. States

The state space $\mathcal{S}$ denotes all possible designs. Any s in the state space consists of a collections of trusses and joints where the trusses are connected: $s : (\mathcal{M}_s, \mathcal{N}_s)$, where $\mathcal{M}_s$ is a list of trusses and $\mathcal{N}_s$ is a list of connecting nodes. Ororbia and Warn [7] reported from their experiments that using the list of trusses to express a state results in better performance than using the list of nodes or the combination of two lists. Therefore from now on a state is described by the trusses included within the respective design, and the state space $\mathcal{S}$ is the all the possible combinations of trusses in the design space. For instance, for the initial input, the policy starts at s_0, where only the outer mold line trusses are included: $\mathcal{M}_{s_0} = \{m_{1,3}, m_{1,7}, m_{7,9}, m_{9,3}\}$. The active nodes in s_0 is therefore $\mathcal{N}(s_0) = \{n_1, n_3, n_7, n_9\}$

For formality, termination states s_T is set to represents that all possible trusses and nodes has been utilized. The termination states represents the physical boundary of performance within the design space. The optimum result is definitely not the termination state since the objective is to design a light weight structure. The MDP problem should end, if the problem is well defined, prior to reaching some termination state, the design is at it's minimal weight while still satisfying the strain constraints.

C. Actions

An action transits the agent from previous state to next state, i.e. synthesizing a new iteration of the design. Ororbia and Warn [7] described the action as a tuple of: selection of a node from inactive nodes for a certain state, selection of an existing truss, and selection of a operation from possible operations. The mathematical definition of the action set at a particular state s_t is illustrated in Equation 2.

$$\mathcal{A}(s_t) = \{n_i | n_i \notin N(s_t), m_j | m_j \in s_t, o_k | o_k \in O\{s_t, n_i, m_j\}\} \tag{2}$$

The operations set O was presented by Lipson [9] has two possible operations: **D** and **T**. **D** connects the selected inactive node n_i with the nodes that is defined by the selected edge m_j, adding two extra trusses to the design. **T** connects the selected inactive node n_i with the nodes that is defined by the selected edge m_j, and replace m_j with another edge that is connected from n_i to the closest node in existing node set that does not include the bounding nodes of m_j.

On top of Warns [8] description of action set. Another action that stays at the previous design is added, as any change of the previous design would yield negative rewards. Adding this action allows convergence to a certain state before reaching the termination states.

In the context of the example problem, the inactive nodes are $n \notin \mathcal{N}_{s_0} = \{n_2, n_4, n_5, n_6, n_8\}$, and the active trusses are $\mathcal{M}_{s_0} = \{m_{1,3}, m_{1,7}, m_{7,9}, m_{9,3}\}$. Suppose the action $a(s_0) = \{n_4, m_{1,7}, \mathbf{T}\}$ is taken, $m_{1,7}$ will be replaced by $m_{1,4}$ and $m_{4,7}$, and a new edge will $m_{4,9}$ will be added. $m_{4,9}$ is added instead of $m_{4,3}$ because it would yield higher objective function, since the load is applied on node 9. If $a(s_0) = \{n_4, m_{1,7}, \mathbf{D}\}$ is taken, the truss $m_{1,7}$ is duplicating the new edges $m_{1,4}$ and $m_{4,7}$, therefore this action operator is illegal for this specific choice of inactive node and member. This process is illustrated in Fig. 2

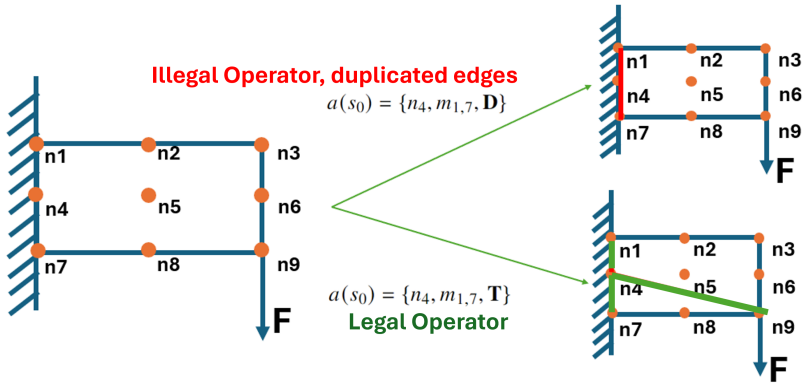


Fig. 2 Examples of Possible Actions from Initial State

D. Rewards

Since the design problem intend to minimize weight while satisfying some strain constraints. It is intuitive to build the reward function of a state-action pair upon those two physical values. Ororbia and Warns publication [7] was interested in minimizing the displacement of the optimal structure and was using the difference in displacement caused by an action as a measure of reward. This problem requires a similar approach in reward function definition but would have to take the constraints into consideration. The generalized reward function for a state-action pair is shown in Equation 3, where w_1 and w_2 are weighting variables to ensure the ratio of weight-specific structure effectiveness is in the same order of magnitude as the constraints.

$$r(s_t, a_t) = w_1 \left(u^* - u_{\max}(s_t, a_t)\right)^2 + w_2 \Delta(W(s_t, a_t)) \tag{3}$$

The first half of the function implies the maximum strain constraint. If a structure have maximum strain equal to the ultimate strain of the material, it means that a truss in the structure is fully loaded and therefore is the most efficient structure: removing another truss will cause strain concentration to fail the maximum strain constraints, and increasing another truss will increase weight. The second half of the reward function refers to the penalty of the weight gain, $\Delta W(s_t)$. An action that leads to higher weight but lower strain concentration yields a higher reward, and the magnitude of such reward is dependent on how far is the resultant design from fully loaded plus penalty of weight gain. It is also noted that an action that increase strain concentration but lower weight is also worth considering during the design iteration.

Finally, to encourage the agent to stay at it's position if all other actions would yield worse results, the reward function of staying in-place is the same as the reward of previous time-step, where an action has brought the agent to this local optimum.

E. Transition Probability

In Ororbia and Warns publication [7] , the transition probability was set to a binary value of either 0 or 1. This infers that a new action will deterministically leads to a new design, or the state pair is physically impossible to achieve.

IV. Problem Setup

A. FEA and Structure Mechanics Definition

An 2D truss element solver in Python, anaStruct[10], is chosen to evaluate the performance of a structure, and from which derive the effectiveness of modifications brought by performing a certain action. The solver takes material properties and geometric locations of the truss elements as input and solve for strain and strain distribution on the system with user defined discretization resolutions.

Polylactic Acid (PLA) is one of the most common materials used in FDM 3D printing, and therefore is chosen to be the material used for this problem setting. The mechanical properties of 3D printed PLA is shown in Table 1 [11] and the respective deformations are plotted in Figure 3. To build in some safety factors, the strain constraints is set to 0.014 instead of 0.017 of the material.

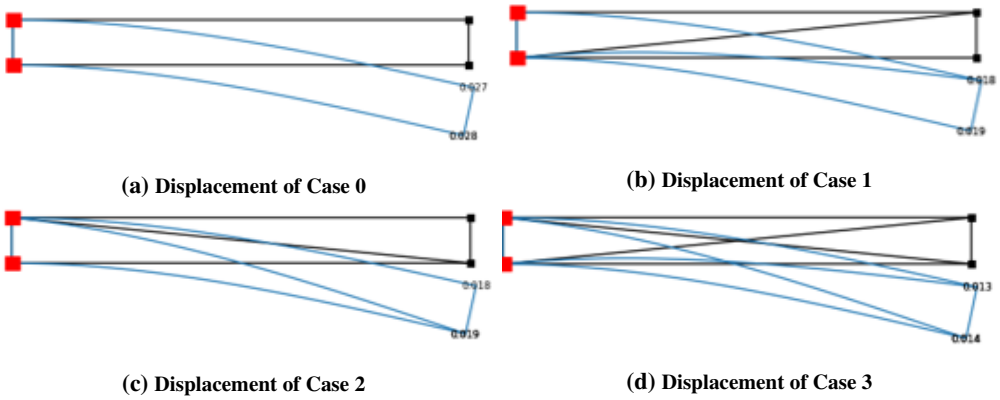
With the materials set, simulations were performed on some modifications of the initial structure, shown in Fig 1, to determine a suitable weight factors for the expected reward and variance. The results are summarized in Table 2 and Figure 3. It is worth noting that these cases are not attainable by the actions defined in previous sections, but are the most intuitive ways of reinforcing the seed structure. It was observed that only Case 3 has satisfies the constraints and that bears the highest reward.

Table 1 Mechanical Properties of 3D Printed PLA

Properties	Value	Units
Ultimate Tensile Strain	1.7%	Characteristic Length
Elastic Modulus	6.1	Gpa
Density	1,120	Kg/m^3
Diameter	0.4	mm
Ixx	6.28E-08	m^-2

Table 2 Simulation Results for Some Possible Designs

Case Number	Fy	Max Deformation	Weight Addition	Reward Function w1 = -1E4, w2=1E3
0	-8	0.02798	0	0
1	-8	0.01932	0.000143	-0.3564
2	-8	0.01868	0.000143	-0.3014
3	-8	0.01437	0.000286	-0.2871



(a) Displacement of Case 0

(b) Displacement of Case 1

(c) Displacement of Case 2

(d) Displacement of Case 3

Fig. 3 Displacement of Some Possible Designs

B. Q-Learning and Dynamic Programming

The expected reward under of a policy under the MDP framework is the summation of all rewards that are collected, shown in Equation4, where γ is the discount factor. For this particular problem, reaching an optimal structure design is not necessary, therefore γ is assumed to be 1, i.e. an undiscounted problem. This problem can be solved to obtain a deterministic policy that maximizes the expected reward at each state, shown in Equation 5.

$$q_\pi(s,a) = \sum_{k=0}^{N} (\gamma^k r_k) \tag{4}$$

$$\pi^* = \underset{a(s_t)\in\pi^*}{\operatorname{argmax}} (q_*(s_t,a_t)) \tag{5}$$

However, the reward functions and states are not effectively known a priori since the state space $\mathcal{S}$ consists of all possible combinations of trusses within the design space, and an analytical solution to Equation 4 cannot be obtained. In this case, Q-learning is used to learn an optimal policy [12]. The expected reward of a state-action pair is first initialized and then updated based on Equation 6. Where Q is an approximation of the expected reward q_π and α is the learning rate.

$$Q(s_t,a_t) = (1-\alpha)Q(s_t,a_t) + \alpha\left(r(s_t,a_t) + \max_{a\in\mathcal{A}(s_{t+1})} Q(s_{t+1},a)\right) \tag{6}$$

With the MDP and Q-Learning framework, the optimization problem of structure design can be written as a dynamic programming problem in the design space to find the optimum policy π^*. The agent adopts some explore-exploit policy from the seed design until a termination state is reached or further actions does not bring a positive reward, then the $Q(s_T,a_T)$ is set to zero. The set of actions taken to transit from initial seed s_0 to s_T is referred to as an episode. The agent then plays an arbitrary number of episode to learn the optimal policy based on converging Q values. The process of Q-learning is shown in the pseudo code below [7]:

```
Input: Initial State, Learn Rate, Explore-Exploit policy, Number of Episode
Initialize Q for seed
for i in range(Number of Episode):
|   S = Initial State
|   while S!=Terminal State:
|   |   Choose Action based on explore-exploit
|   |   Execute Action, S = New State
|   |   Perform FEA, record r(Old State, Action)
|   |   Update Q(Old_State,Action) based on Equation (6)
|   |   If New State == Terminal State
|   |   |   Break
|   Q(Terminal State,Action)=0
End
```

V. Experiments

The numerical problem follows the minimal working example: find the optimum structure which maximum deformation is less than 1.47% while bearing the minimum weight in a 3 by 3 design space with length of 2 m and height of 0.2 m. Two policies, namely a greedy method and a Q-learning method, were experimented with and the results are shown below. Code for the experiments is uploaded to GitHub [13].

A. Vanilla Greedy Policy

To begin with, a vanilla greedy policy is implemented to set some benchmarking results to the problem of interest. The agent observes the reward of playing actions at a particular state, and always plays the action that has maximum reward. The policy is formally shown in Equation 7.

$$\pi^* = \underset{a(s_t)\in\pi^*}{\operatorname{argmax}}\left(r(s_t, a_t)\right) \tag{7}$$

It has to be noted that the solution obtained, while satisfying the deformation constraints, should not be trusted as a global minimum. The agent cannot only has the information of the immediate neighbours and can easily be trapped into a local optimum due to the greedy policy. This is shown later as the Q-learning policy was implemented.

The iteration from initial seed state to final design is shown in Figure 4. The final structure yields maximum deformation of 0.01366 m and weight of 0.001342 Kg. The weight specific deformation is 10.186 m/Kg.

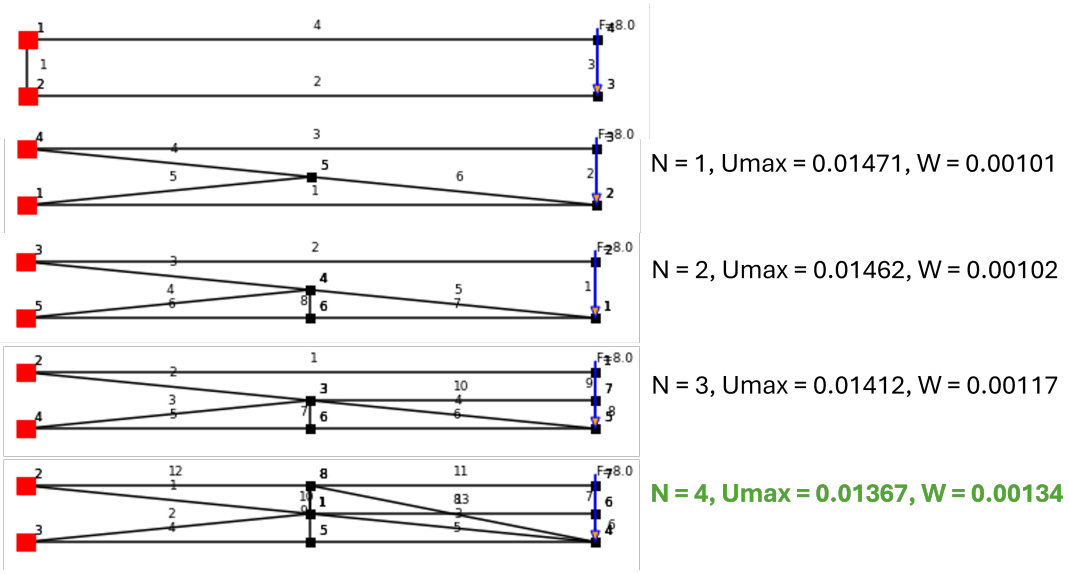


Fig. 4 Convergence History of Vanilla Greedy Policy

B. Q-Learning Policies with ϵ_t-Greedy

Q-learning is implemented on the same problem with expected reward update algorithms shown in Equation 6. Q-learning has shown promising convergence to an optimal solution if the agent slowly favors exploitation over exploration as episode number increases, or a large number of episode is played such that the agent have explored all possible state-action pairs [12][8].

Previous literature [7] and [8] used ϵ-Greedy policy to determine the explore-exploit behaviours of agent and has shown satisfactory convergence behaviour. With the conditions of convergence, Q-learning with ϵ_t-Greedy policy is experimented on in this study. ϵ_t-Greedy policy start with zero probability of exploiting, and the possibility of exploiting increases per episode until the last episode, where $\epsilon_t = 1$ and the agent will always exploit based on the Q-values learned in previous episodes. With a large enough episode number, this explore-exploit policy will satisfy both conditions that ensures convergence. For this experiment the ϵ_t is set to linearly increase from 0 for first episode to 1 at last episode. For ϵ_t-Greedy policy, the agent will always exploit based on previous learning results on the last episode.The learning rate α is set to 0.8.

A number of simulations are performed for episode number ranging from 50 to 500. Convergence history of weight specific deformation of Q-learning with ϵ_t-Greedy explore policy is shown in Figure 5. Some intermediate designs of the learning process is shown in Figure 6. The optimal result obtained from this policy has a deformation of 0.01273 m and bears weight of 0.00117 Kg, the weight specific deformation is 10.880 Kg/m. It was observed that Q-learning indeed converge really well with ϵ_t-Greedy explore policy and resultant structure converges to the optimal design for episode number larger than 300. It was also observed that starting from Episode number 200, the resultant structure is already lighter than the result of vanilla greedy method shown in the previous section.

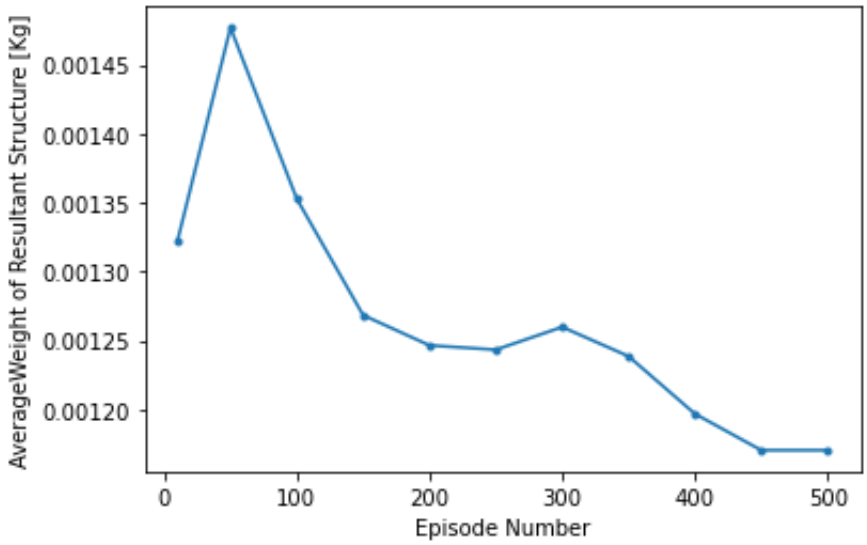


Fig. 5 Average Weight of Optimal Structure Weight under Different Episodes

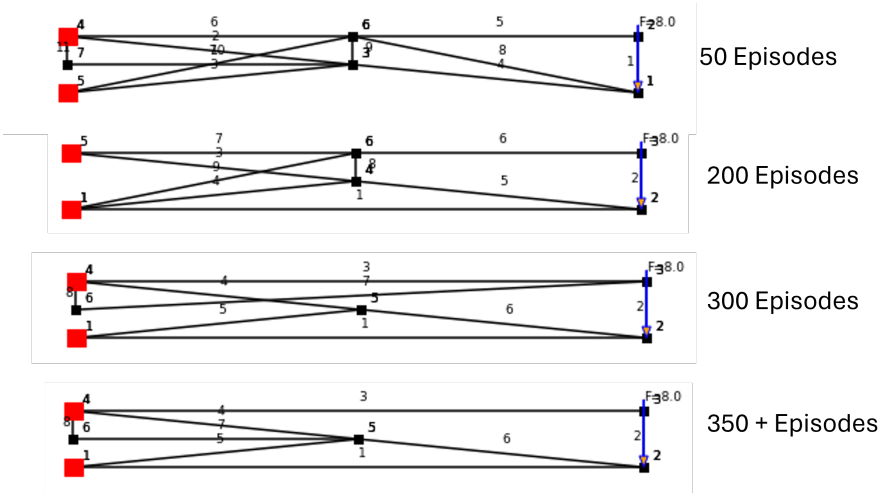


Fig. 6 Expected Result Structure for ϵ_t-Greedy Policy with Different Episode Numbers

VI. Conclusions

A. Discussion of Experiment Results

This study framed structural design optimization problem onto an MDP and used Q-learning to synthesize an optimum structure. The results in numerical experiments show that Q-learning with ϵ_t-Greedy explore-exploit strategy has promising convergence behaviour and would result in a better performing structure compared to a vanilla greedy policy. In summary, the vanilla greedy policy leads to an over-engineered structure with weight of 0.001342 Kg, whereas the Q-learning policy leads to a global optimal design of 0.00117 Kg. The Q-learning policy results in a structure with 12% weight reduction compared to that by the vanilla greedy policy!

This contrast matches intuition in that Q-learning gives the option to explore all possible state-action pairs and can bypass local optimums. If the agent is given enough episodes and explores all-state action pairs, then $r(s_t, a_t) \in \mathcal{R}$ is fully known and Equation 4 can be solved analytically for a global optimum. Whereas the vanilla greedy method only have information for a horizon of 1 and therefore would be tricked to take an action that yields high immediate reward instead of actions that has lower immediate reward but results in better solutions in the long run. It was also observed that if the agent was not given enough episode to learn the possible rewards of all state action pairs, the learned policy would lead to some other local optimum, and yet still outperforms the result of vanilla greedy policy.

B. Considerations of Scalability

For a 3x3 design space with 5 inactive nodes, it took an average of 3 actions to reach a terminal state per episode while the maximum actions to take is 5. This is partly because of the strict deformation constraints imposed in the physical problem. If the deformation constraints was relaxed from 0.014 to 0.016, the average actions to reach terminal state drops to 2. With that this framework can be easily scaled to structural design optimization problems with a larger design space on complex geometries and loading criterion, for instance structural design of a wing rib under aerodynamic loading.

With the same MDP formulation and upgrading the FEA solver, this framework can also expand the design space to 3-D. A truss dominant structure can always be mapped to a graph by a set of nodes in the design space and a connectivity matrix between the nodes, no matter if the structure lives in 2-D or 3-D space.

C. Possibilities of Future Work

This study has shown that using Q-learning in structure design is indeed possible and effective in exploring the design space. Although the current framework has only been validated on a minimal working example, more elements could be added to this framework to solve more exciting problems that has real-life applications.

To begin with, applied and experimental study can be performed on this framework to further validate the practicality of using Q-learning and MDP on structural design. For instance this framework can be applied on structural optimization on 3D printed airfoils. Experiments can be conducted to compare the performance of Q-learning synthesized structures versus manual or automatic structures synthesis.

As for the current problem, the structure is defined by a set of nodes and a connectivity matrix of these nodes. The element in that connectivity matrix is a boolean: For a certain design, there either exists an between two selected nodes, or there does not exist an edge between that two nodes. This framework can be improved by making the elements in the connectivity matrix to be a number between 0 and 1 to introduce the notion of reliability: With a certain design, if the design asks for an truss between two nodes, that truss has a certain possibility of experiencing manufacturing flaws. This would change the map of structures from a graph to a Markov Chain, but nevertheless the MDP framework of designs as states and modifications as actions, in conjunction with Q-learning, should still apply to this problem.

Finally, it is also possible to investigate action sets that can efficiently modify with none-truss structures and use the current framework to solve structure optimization problems with elements that are sheets or volumetric cells. This would expand the capability of the problem from the skeletal structures discussed in this study.

References

[1] Schiller, G., "Additive manufacturing for Aerospace," *2015 IEEE Aerospace Conference*, 2015, pp. 1–8. https://doi.org/10.1109/AERO.2015.7118958.

[2] Ulrike Franke, J. S., "Star Tech Enterprise: Emerging Technologies in Russia's war on Ukraine," , Sep 2023. URL https://ecfr.eu/publication/star-tech-enterprise-emerging-technologies-in-russias-war-on-ukraine/.

[3] Canis, B., "Unmanned aircraft systems (UAS): Commercial outlook for a new industry," , 2015.

[4] Goh, G., Agarwala, S., Goh, G., Dikshit, V., Sing, S., and Yeong, W., "Additive manufacturing in unmanned aerial vehicles (UAVs): Challenges and potential," *Aerospace Science and Technology*, Vol. 63, 2017, pp. 140–151. https://doi.org/https://doi.org/10.1016/j.ast.2016.12.019, URL https://www.sciencedirect.com/science/article/pii/S127096381630503X.

[5] Kazakis, G., Kanellopoulos, I., Sotiropoulos, S., and Lagaros, N. D., "Topology optimization aided structural design: Interpretation, computational aspects and 3D printing," *Heliyon*, Vol. 3, No. 10, 2017.

[6] Feng, J., Fu, J., Lin, Z., Shang, C., and Li, B., "A review of the design methods of complex topology structures for 3D printing," *Vis Comput Ind Biomed Art*, Vol. 1, No. 1, 2018, p. 5.

[7] Ororbia, M. E., and Warn, G. P., "Design Synthesis Through a Markov Decision Process and Reinforcement Learning Framework," *Journal of Computing and Information Science in Engineering*, Vol. 22, No. 2, 2021, p. 021002. https://doi.org/10.1115/1.4051598, URL https://doi.org/10.1115/1.4051598.

[8] Ororbia, M. E., and Warn, G. P., "Design Synthesis of Structural Systems as a Markov Decision Process Solved With Deep Reinforcement Learning," *Journal of Mechanical Design*, Vol. 145, No. 6, 2023, p. 061701. https://doi.org/10.1115/1.4056693, URL https://doi.org/10.1115/1.4056693.

[9] Lipson, H., "Evolutionary synthesis of kinematic mechanisms," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 22, No. 3, 2008, p. 195–205. https://doi.org/10.1017/S0890060408000139.

[10] Vink, R., "anaStruct 2D Frames and Trusses," , 2018. URL https://github.com/ritchie46/anaStruct.

[11] "MatWeb - The Online Materials Information Resource," , 2023. URL https://www.matweb.com/search/DataSheet.aspx?MatGUID=ab96a4c0655c4018a8785ac4031b9278&ckck=1.

[12] Sutton, R. S., Bach, F., and Barto, A. G., *Reinforcement learning: An introduction*, MIT Press Ltd, 2018.

[13] Zheng, W., "QLearningStrucDesign," , 2024. URL https://github.com/WabalabaKing/QLearningStrucDesign.