Machine-specified Ground Structures for Topology Optimization of Binary Trusses using Graph Embedding Policy Network

doi: 10.1016/j.advengsoft.2021.103032

Shaojun Zhu[1,2], Makoto Ohsaki[2], Kazuki Hayashi[2], Xiaonong Guo[1*]

[1] College of Civil Engineering, Tongji University, Shanghai 200092, China

[2] Department of Architecture and Architectural Engineering, Kyoto University, Kyoto-Daigaku Katsura, Nishikyo, Kyoto 615-8540, Japan

Abstract

This paper proposes the concept of machine-specified ground structures for topology optimization of trusses. Unlike general ground structures with dense and regular connectivity, machine-specified ground structures are sparse stable ground structures with a specified number of members designed by machines. Firstly, the generation process of machine-specified ground structures from a given node-set is formulated as a reinforcement learning task. Graph embedding is used to integrate the structural information into a comprehensive feature matrix to describe the state. By establishing the policy network, the probability of each action, i.e., selecting each node in the node-set, is obtained based on the comprehensive feature matrix. The task is solved using a gradient-based algorithm called *REINFORCE*. A randomized 4×4 node-set is used to train the agent. The policy converges with a high average reward, and generates different yet reasonable structures because a stochastic policy is employed. Besides, the agent can handle different-sized node-sets without re-training. Hence, the machine-specified ground structures generated by the trained agent can be utilized to assist the structural topology design. Subsequently, a method for a typical problem with singular optimal solutions, i.e., topology optimization of binary trusses with stress and displacement constraints, is proposed based on machine-specified ground structures. Finally, through different-sized numerical examples, it is demonstrated that the machine-specified ground structures lead to a variety of optimal solutions, and it is more likely to obtain the global optimum than fully-connected ground structures. It is worth noting that machine-specified ground structures can also be applied to other problems without re-training.

Keywords

topology optimization, reinforcement learning, graph embedding, binary trusses, machine-specified ground structures

1 Introduction

The ground structure (GS) method proposed by Dorn *et al*. [1] is an effective approach to limiting the design domain of topology optimization (TO) of truss structures, and has been widely incorporated with a variety of TO methods [2–5]. The GS method generates a sparse optimal topology from a finite number of nodes and members in the design domain specified by the GS. If the GS has sufficiently many nodes and members, an optimal truss close to the analytical one, i.e., the Michell truss [6], can be obtained by eliminating unnecessary members and nodes in the GS. There are two types of GSs, namely the fully-connected GS and the human-specified GS with a small number of members.

For a node-set containing n nodes, the fully-connected GS has $n(n-1)/2$ members. That is to say, the number of members increases by a quadratic function of n. Moreover, when the cross-sectional areas are selected from a predefined set of discrete values, the design domain will expand as an exponential function of the number of possible discrete values. In order to reduce the number of possible combinations, Kawamura *et al*. [7] introduced triangular units to express the connectivity of truss elements for each chromosome in the genetic algorithm (GA). Although this method can avoid undesirable designs, e.g., unstable trusses or trusses with overlapping members, heuristic algorithms like GA may converge to a local optimal solution when the number of variables is enormous. Shakya *et al*. [8] also indicated that when the search space is even larger, almost no optimization method will be able to find the global optimal solution. Particularly, TO problems with stress constraints are facing discontinuity, as the stress constraint suddenly disappears when the cross-sectional area of a member diminishes to zero [9, 10]. The optimal solutions to these problems are called *singular optimal solutions* because some non-existing members may violate the constraint, i.e., the stress constraint is design-dependent. Hagishita and Ohsaki [11] highlighted that the GS with more members could not guarantee a better optimal solution for a TO problem with singular optimal solutions. Therefore, it is reasonable to use sparser human-specified GSs.

A fully-connected GS can be easily generated by connecting all potential members; on the other hand, a human-specified GS needs to be prepared according to the designer's preference or an intuitive knowledge for the global optimal topology. Existing TO methods are often verified by benchmark numerical examples with human-specified GSs [12–19]. However, for a node-set with a large number of nodes, the work of preparing a human-specified GS is highly laborious, and it becomes tough to generate a GS that may lead to the global optimum. An alternative would be generating the members according to the nodal vicinity principle proposed by Beekers and Fleury [20].

When the nodes are uniformly distributed in a rectangular domain, it has been proved that GSs generated by order three and order four vicinity can lead to acceptable optimal solutions at a significantly smaller computational cost compared with a fully-connected GS. However, the method may generate unstable GSs for random node-sets as the definition of nodal vicinity does not consider stability. Besides, the number of members generated by the nodal vicinity method is unpredictable under the circumstance of a non-uniform node-set. Hence, it is hard to estimate the computational cost of the optimization process, and the efficiency of GSs under various order nodal vicinities cannot be evaluated in advance. Hagishita and Ohsaki [11] proposed the growing GS method, which finds an optimal topology from a sparse GS by successively adding nodes and members. The growing GS method can efficiently obtain the singular optimal solution to small-size problems, yet the authors also stated that the method could not tackle large-size problems.

In recent years, machine learning has been extensively applied in structural engineering [21]. Machine learning techniques can be classified into three kinds, namely supervised learning, unsupervised learning, and reinforcement learning (RL). Supervised learning deals with problems regarding classification and regression. It has been successfully applied to solve problems like predicting the non-linear buckling capacity of imperfect spatial structures [22], predicting the shear capacity of reinforced concrete structures with steel fibers [23], assessing the fire-induced progressive collapse potential of steel frames [24], and so on. Unsupervised learning deals with dimension reduction and clustering. It has effectively handled the problems in structural health monitoring and damage detection [25–27]. Unlike the two techniques mentioned above, RL deals with problems that involve the interaction between the agent and the environment. An RL problem aims at training an agent that can maximize the cumulative reward [28]. The training of an agent can be regarded as a trial and error process, and the agent gradually learns how to behave better based on the rewards it receives. Well-known applications of RL include developing autonomous driving [29] and playing the game of Go [30]. Recently, scholars have also incorporated RL in the TO of truss structures. As the image-based recognition method proposed by Gamache *et al*. [31] turns out to be an effective approach to conduct TO on skeletal structures, Sahachaisaree *et al*. [32] used convolutional neural networks [33] to process the image of a truss structure. They trained an agent to generate stable trusses with the shortest total member length under given support and load conditions. Hayashi and Ohsaki [34] indicated that truss structures can be regarded as graphs instead of images so that the graph embedding (GE) [35, 36] may have a higher potential in extracting the structural features. They combined the Q-learning [37] with the edge-embedding in GE and

developed a deterministic agent to conduct TO in which unnecessary members are successively removed from relatively sparse GSs. It is worth noting that the agent can also deal with various-sized problems without re-training.

In order to reduce the computational time and the human workload, this paper proposes the concept of machine-specified ground structures (MGS) for TO of binary trusses based on the GE policy network. Section 2 describes the task of generating MGSs and briefly introduces a general RL problem. The key elements in the current RL task are introduced, including the state, action, reward, the policy network, and the learning method. Section 3 provides an example to illustrate the training process of the RL agent. The generalization ability of the trained agent is also evaluated by testing it in different-sized environments without re-training. Section 4 describes a typical TO problem with stress and displacement constraints and proposes a method based on the MGSs. Different-sized numerical examples are used to verify and illustrate the effectiveness of the proposed method.

2 RL task of the MGS method

2.1 Task description

Here we define the MGS as a *reasonable* GS with the following properties:

1) The whole GS is stable under an appropriate support condition to restrict rigid-body displacements and rotation, as illustrated in Fig. 1;
2) The number of members in the GS is not so large as a fully-connected GS, while there should also be unnecessary members to be eliminated, i.e., the truss should be statically indeterminate. For a GS with n nodes and n_c ($n_c \geq 3$) translational constraints that has property 1), the minimum number of members m_{min} is $2n-n_c$, and the maximum number of members m_{max} is $n(n-1)/2$. Therefore, the actual number of members m should be within the range (m_{min}, m_{max}).

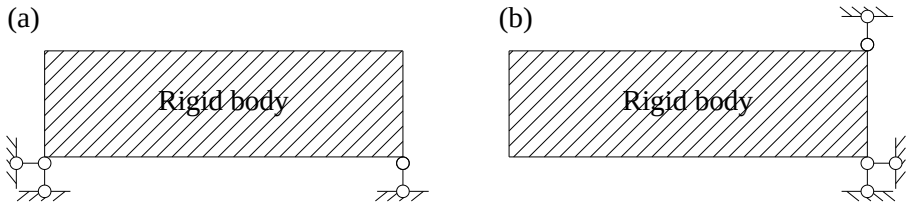


Fig. 1 Different types of support conditions. (a) Appropriate condition. (b) Inappropriate condition.

The process of generating stable GSs is simulated using RL. As explained in the following sections, the agent trained by RL selects an action at each step from the policy obtained by training. Obviously, there are many topologies of GSs with a specific

number of members. The more diverse the GSs are, the more likely it is that one of them leads to the global optimal solution. Hence, MGSs should be generated by a stochastic policy instead of a deterministic one. Besides, if one adds an additional translational constraint or member to an MGS, it will still be reasonable so long as it does not contain too many members. Therefore, we can first generate a stable structure using *favorable* members with the least number of translational constraints, i.e., $n_c = 3$. Here, the term *favorable* means the members are highly possible to exist in the global optimal solution, and are helpful to make the structure stable. Furthermore, for practical application, long members should be avoided in the optimal solutions, and accordingly, in the MGS. For example, we prefer the dotted member in Fig. 2(b) to that in Fig. 2(a) because the former directly makes the GS reasonable, and its length is not large.

Then, the task of generating MGSs can be summarized into two steps:

1) Step 1: For a given node-set, sequentially add members to generate a stable structure using m_0 favorable members under three appropriate translational constraints. In this step, the truss has no isolated node or support, and m_0 should be close to $m_{\min}$. Various topologies are generated using a stochastic policy.
2) Step 2: Randomly add members until the number of members reaches the specified value m.

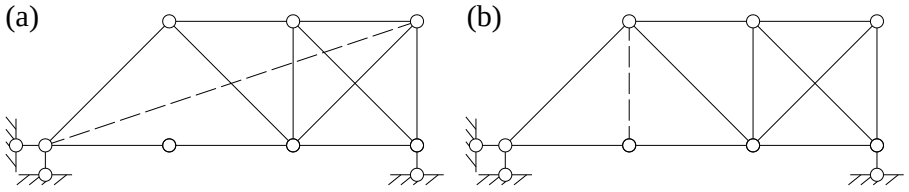


Fig. 2 Potential members to be added in an unstable truss. (a) Unfavorable. (b) Favorable.

Note that since the structure is already stable in Step 1, the members to be added in Step 2 can be selected randomly without considering the stability of the structure. Hence, only the stochastic policy in Step 1 needs to be trained using RL.

2.2 Overview of RL

Fig. 3 illustrates the interaction between an RL agent and the environment. At step t, the agent observes the state s_t and receives the reward r_t. Then, it takes the action a_t by sampling from the policy function $\pi(s_t)$. In the next step, the state transfers to s_{t+1}, and the environment produces a new reward r_{t+1}.

The non-linear relationship between the policy function and the state is usually established using a neural network, which is also known as the policy network. The policy network parameters, namely the weights and biases, are updated according to a gradient-based learning method that aims at minimizing the error computed from the

observed reward at each step.

To summarize, the key elements in an RL task include the state, action, reward, policy network, and the learning method. Details of the current RL task will be introduced in the following sections.

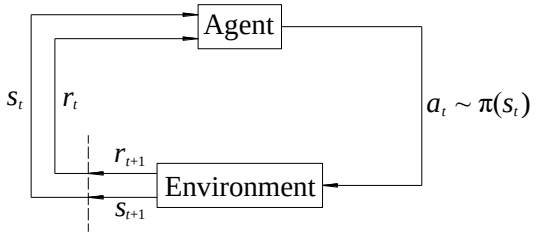


Fig. 3 Schematic diagram of the interaction between an RL agent and the environment.

2.3 Action

Since the number of nodes is significantly smaller than the number of potential members, the action of the current RL task is defined as *node selection*. Hence, the action space $\mathcal{A}$ for a truss with n nodes is

$$\mathcal{A} = \begin{pmatrix} 1 & 2 & 3 & \dots & n \end{pmatrix} \tag{1}$$

Two different nodes need to be selected to generate a member. Let FN and SN denote the first and second nodes, respectively, alternately selected by the agent. Note that $\mathcal{A}$ is fixed during the RL process; therefore, the following infeasible actions may occur:

1) The same node is selected two times, namely FN = SN;
2) FN and SN (FN ≠ SN) are already connected by an existing member.

These infeasible actions can be masked without changing $\mathcal{A}$, which will be introduced in Section 2.5.

2.4 State

Given three appropriate translational support conditions, stability of a binary truss is only related to the nodal locations and the topology. Hence, we can construct a nodal feature vector $\mathbf{v}_i$ (i = 1, 2, …, n) in a node-set as

$$\mathbf{v}_i = \begin{pmatrix} x_i & y_i & \mathrm{Sel}_i \end{pmatrix}^{\mathrm{T}} \tag{2}$$

where x_i and y_i are the x and y coordinates of the ith node, respectively. Sel_i is defined as the selection condition of the ith node, which also provides information on the distance between FN and other nodes when FN has been selected. Sel_i is calculated by

$$\mathrm{Sel}_i = \begin{cases} -1 & \text{if FN} = i \text{ or to be selected} \\ \dfrac{1}{\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}} & \text{if FN} = j\left(j \neq i\right) \end{cases} \tag{3}$$

Let $\hat{\mathbf{v}} = \left(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\right)$ denote the nodal feature matrix. Here we notice that $\hat{\mathbf{v}}$ and $\mathcal{A}$ both have n columns. Hence, it is possible to establish a policy network whose size is independent of n so that the trained agent can be applied to node-sets with various numbers of nodes without re-training. In other words, the policy network is expected to evaluate the ith action based on a vector related to the ith node. However, it is inappropriate to directly use $\mathbf{v}_i$ as the input because the topology information has not yet been included.

Therefore, node-embedding in GE [36] is modified and applied to integrate the topology information by constructing a *comprehensive feature vector* $\boldsymbol{\mu}_i$ for the ith node. Let n_f denote the number of elements of $\boldsymbol{\mu}_i$. Note that n_f is a hyperparameter to be determined and should be significantly larger than the size of $\mathbf{v}_i$. The iteration process of $\boldsymbol{\mu}_i$ to incorporate information of the neighborhood nodes is:

$$\boldsymbol{\mu}_i^{(0)} = \mathbf{0} \tag{4}$$

$$\boldsymbol{\mu}_i^{(T+1)} = \mathrm{ReLU}\left(\mathbf{h}_1 + \mathbf{h}_2 + \mathbf{h}_3^{(T)} + \mathbf{h}_4^{(T)}\right) \tag{5}$$

in which $\boldsymbol{\mu}_i^{(T)}$ is the comprehensive feature vector of the ith node at the Tth iteration. ReLU is an activation function defined for a real value x as

$$\mathrm{ReLU}\left(x\right) = \max\left\{0, x\right\} \tag{6}$$

When the ReLU function is applied to a matrix, we assume the element-wise application of Eq. (6) for simplicity. $\mathbf{h}_1$, $\mathbf{h}_2$, $\mathbf{h}_3^{(T)}$, and $\mathbf{h}_4^{(T)}$ are intermediate vectors calculated by

$$\mathbf{h}_1 = L_1\left(\mathbf{v}_i\right) \tag{7}$$

$$\mathbf{h}_2 = L_3\left\{\mathrm{ReLU}\left[L_2\left(\hat{\mathbf{v}}\mathbf{C}_i\right)\right]\right\} \tag{8}$$

$$\mathbf{h}_3^{(T)} = L_4\left(\boldsymbol{\mu}_i^{(T)}\right) \tag{9}$$

$$\mathbf{h}_4^{(T)} = L_6\left\{\mathrm{ReLU}\left[L_5\left(\hat{\boldsymbol{\mu}}^{(T)}\mathbf{C}_i\right)\right]\right\} \tag{10}$$

where $\hat{\boldsymbol{\mu}}^{(T)} = \left(\boldsymbol{\mu}_1^{(T)}, \boldsymbol{\mu}_2^{(T)}, \ldots, \boldsymbol{\mu}_n^{(T)}\right)$ is the *comprehensive feature matrix* at the Tth iteration. $\mathbf{C}$ is an $n \times n$ adjacency matrix, and $\mathbf{C}_i$ is the ith column of $\mathbf{C}$. The element of

C denoted as $C_{i,j}$ ($i, j = 1, 2, \ldots, n$) is defined as

$$C_{i,j} = \begin{cases} 1 & \text{if a member connects nodes } i \text{ and } j \ \ (i \neq j) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

L_k indicates the kth linear layer:

$$L_k(\mathbf{A}) = \mathbf{w}_k \mathbf{A} + \mathbf{b}_k \tag{12}$$

in which **A** is the input matrix/vector of the linear layer. $\mathbf{w}_k$ and $\mathbf{b}_k$ are the weight and bias of the layer, respectively. The sizes of the linear layer parameters are listed in Table 1.

Table 1 Sizes of the linear layer parameters in the iteration process of the state matrix.

k	$\mathbf{w}_k$	$\mathbf{b}_k$
1	$n_f \times 3$	$n_f \times 1$
2	$n_f \times 3$	$n_f \times 1$
3	$n_f \times n_f$	$n_f \times 1$
4	$n_f \times n_f$	$n_f \times 1$
5	$n_f \times n_f$	$n_f \times 1$
6	$n_f \times n_f$	$n_f \times 1$

Through multiple iterations, $\boldsymbol{\mu}_i^{(T)}$ gradually integrates the information of the ith node and its far-away neighbors. When the maximum number of iterations is specified as $T_{\max}$ ($T_{\max} \geq 1$), it is reasonable to use $\hat{\boldsymbol{\mu}}^{(T_{\max})}$ to represent the state of the structure.

2.5 Policy network

Let $\hat{\boldsymbol{\mu}} = (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_n)$ denote the comprehensive feature matrix at the $T_{\max}$th iteration for simplicity. As briefly mentioned in Section 2.4, the policy network $\mathbf{p} = \pi(\hat{\boldsymbol{\mu}})$ is established as follows:

$$Q_i = \mathrm{ReLU}\left\{ L_9 \left\{ \mathrm{ReLU}\left[L_7(\boldsymbol{\mu}_i); L_8\left(\sum_{i=1}^{n} \boldsymbol{\mu}_i \right) \right] \right\} \right\} \tag{13}$$

$$\tilde{\mathbf{Q}} = \mathbf{S}\mathbf{Q} \tag{14}$$

$$\mathbf{p} = \mathrm{Softmax}(\tilde{\mathbf{Q}}) = \frac{1}{\sum_{i=1}^{n} \mathrm{e}^{\tilde{Q}_i}} \left(\mathrm{e}^{\tilde{Q}_1}, \mathrm{e}^{\tilde{Q}_2}, \ldots, \mathrm{e}^{\tilde{Q}_n} \right) \tag{15}$$

where $[\cdot;\cdot]$ is an operator which concatenates two vectors in the column direction, namely

$$[\mathbf{X};\mathbf{Y}]=\begin{bmatrix}\mathbf{X}\\ \mathbf{Y}\end{bmatrix} \tag{16}$$

in which arbitrary matrices **X** and **Y** have the same number of columns. Q_i in $\mathbf{Q}=(Q_1,Q_2,\ldots,Q_n)$ is the value of the ith action. L_7 to L_9 are linear layers whose parameter sizes are shown in Table 2. **S** is the selection matrix for masking the infeasible actions when selecting SN, defined as

$$\mathbf{S}=\begin{cases}\mathbf{E}_{n\times n} & \text{when FN is to be selected}\\ \operatorname{diag}(S_1,S_2,\ldots,S_n) & \text{when SN is to be selected}\end{cases} \tag{17}$$

$$S_i=\begin{cases}1 & \text{if } C_{i,\mathrm{FN}}=0\\ -10 & \text{if } i=\text{FN or } C_{i,\mathrm{FN}}=1\end{cases}\quad (i=1,2,\ldots,n) \tag{18}$$

where $\mathbf{E}_{n\times n}$ is an $n\times n$ identity matrix. $\tilde{\mathbf{Q}}=(\tilde{Q}_1,\tilde{Q}_2,\ldots,\tilde{Q}_n)$ is the masked action value vector. Softmax is a function that maps the input data into probabilities of selecting the nodes according to their relative magnitudes of action values.

Table 2 Sizes of the linear layers in the policy network.

k	$\mathbf{w}_k$	$\mathbf{b}_k$
7	$n_\mathrm{f}\times n_\mathrm{f}$	$n_\mathrm{f}\times 1$
8	$n_\mathrm{f}\times n_\mathrm{f}$	$n_\mathrm{f}\times 1$
9	$1\times 2n_\mathrm{f}$	1×1

Since Q_i is always non-negative owing to the application of the ReLU function, the masked action value of an infeasible action would be non-positive. After the application of the softmax function, the probabilities of selecting the infeasible actions are expected to be smaller than the feasible ones.

To summarize, the agent observes the state $\hat{\boldsymbol{\mu}}$ and computes the probability vector **p** according to the policy network at each time step. The action is sampled based on **p**, and different actions will be taken at the same state in different episodes or even in the same episode with a different random seed. An episode is defined here as the process of generating a stable truss. Therefore, this type of policy network is called a *stochastic policy*.

2.6 Reward

Let r_p denote the accumulated positive reward, and it is initialized to 0 at the beginning of the episode. The reward and termination condition of an episode is specified as follows:

1) If FN is selected at step t, the reward r_t is 0, and the episode proceeds;
2) If the agent takes an infeasible action at step t, a negative number $-\lambda_\mathrm{in}$ ($\lambda_\mathrm{in} > 0$) is assigned to r_t, and the episode terminates;
3) If a feasible SN is selected and a member is generated at step t, r_t is determined as

$$r_t = \begin{cases} \lambda_i & \text{if the member is } i\text{th-order favorable } (i = 1, 2, \ldots) \\ -\lambda_0 & \text{if the member is unfavorable} \end{cases} \tag{19}$$

where λ_i (i = 0, 1, 2, …) are positive numbers. The 'ith-order-favorable' is to be specified appropriately for each problem. A low-order favorable member is more likely to exist than a high-order favorable member, so $\lambda_{i+1} < \lambda_i$ is recommended when $i > 0$. If $r_t > 0$, update r_p as

$$r_\mathrm{p} \leftarrow r_\mathrm{p} + r_t \tag{20}$$

If $r_t < 0$, proceed with the episode by selecting FN without updating r_p.

4) When the number of members is not smaller than m_min, check the stability. If the truss is stable, i.e., the stiffness matrix is full-rank, update the reward as

$$r_t \leftarrow r_t + r_\mathrm{p} \tag{21}$$

i.e., the previous positive reward is doubled, and the episode terminates. When the number of members is smaller than m_min or the truss is unstable, the reward is not updated, and proceed with the episode by selecting FN.

Here we note that even though the positive reward is doubled when the structure becomes stable, the value of the reward is only dependent on the current state instead of the sequence of the previous actions. In other words, the process maintains the Markov property. Besides, different parameters in the reward policies will result in agents with different behaviors.

2.7 Learning method

Let $\boldsymbol{\theta}$ denote the vector containing all the parameters in linear layers L_1~L_9, and denote the policy network as π_θ. The classical policy gradient method *REINFORCE* [38] is used to update the parameters:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_\theta J(\boldsymbol{\theta}) \tag{22}$$

where α is the learning rate, $J(\boldsymbol{\theta})$ is the expected reward under policy π_θ, and $\nabla_\theta J(\boldsymbol{\theta})$ is the gradient of $J(\boldsymbol{\theta})$ with respect to the parameters in $\boldsymbol{\theta}$:

$$\nabla_\theta J(\boldsymbol{\theta}) = \left(\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_1}, \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_2}, \ldots, \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_{n_\mathrm{p}}} \right)^\mathrm{T} \tag{23}$$

in which n_p is the number of elements in $\boldsymbol{\theta}$, i.e., the total number of parameters in the policy network. Let a *batch* denote a group of episodes under the current policy π_θ, and then the corresponding $\nabla_\theta J(\boldsymbol{\theta})$ can be estimated using the Monte-Carlo sampling method:

$$\nabla_\theta J(\boldsymbol{\theta}) \approx \frac{1}{M} \sum_{i=1}^{M} \left[\sum_{t=0}^{t_{\max}^i - 1} G_t^i \cdot \nabla_\theta \ln \pi_\theta \left(a_t^i \middle| s_t^i \right) \right] \tag{24}$$

where M is the number of episodes in the batch, $t_{\max}^i$ is the maximum number of steps in the ith episode, and a_t^i and s_t^i are the action and state, respectively, at the tth step in the ith episode. G_t^i is the weighted return observed from the tth step in the ith episode calculated by

$$G_t^i = \sum_{t'=t}^{t_{\max}^i - 1} \gamma^{t'-t} r_{t'}^i \tag{25}$$

in which γ is the discount factor of the future reward and $r_{t'}^i$ is the reward of the t'th step in the ith episode.

3 Numerical examples of generating stable trusses

3.1 Training

A 4×4 node-set, as shown in Fig. 4, is used to train the agent. Note that N_i (i = 1, 2, …, 16) represents the ith node number, which is shuffled at the beginning of each episode. The nodal location is also randomized with a maximum amplitude of 0.2 m along the x and y directions. The coordinates of node N_i are

$$\begin{cases} x_{N_i} = \mathrm{floor}\left(\dfrac{i-1}{4} \right) + 0.2\xi_{x,i} \\ y_{N_i} = i - 1 - 4 \times \mathrm{floor}\left(\dfrac{i-1}{4} \right) + 0.2\xi_{y,i} \end{cases} \tag{26}$$

where i is the subscript of N_i ranging from 1 to 16, which indicates the relative location of node N_i. The floor() function returns the largest integer not greater than the input. $\xi_{x,i}$

and $\xi_{y,i}$ are random numbers within the range of [−1, 1]. In every episode, the stable support condition is guaranteed by a horizontal translational constraint at N_1 and two vertical translational constraints at N_1 and N_{16} as illustrated in Fig. 4.

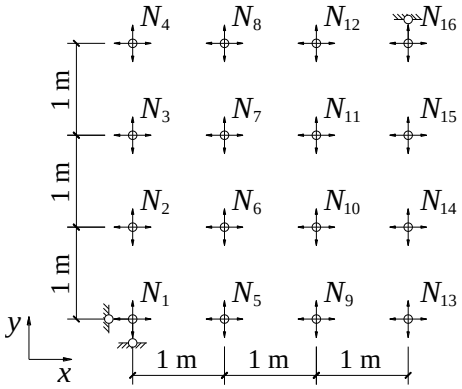


Fig. 4 The randomized 4×4 node-set used for training.

Let l_{ij} define the length of the member connecting nodes N_i and N_j. Two orders of favorable members are defined as

1) 1st-order: $0 < l_{ij} \le \sqrt{\left(1+2\times 0.2\right)^2 + \left(2\times 0.2\right)^2}$;

2) 2nd-order: $\sqrt{\left(1+2\times 0.2\right)^2 + \left(2\times 0.2\right)^2} < l_{ij} \le \sqrt{\left(1+2\times 0.2\right)^2 + \left(1+2\times 0.2\right)^2}$.

The reward parameters λ_{in}, λ_0, λ_1, and λ_2 are 10, 5, 20, and 10, respectively.

The maximum number of GE iterations $T_{\max}$ is 4, and the number of elements in the comprehensive feature vector n_{f} is 100. The adaptive gradient optimizer RMSprop [39] is used to update the parameters, where the learning rate α is 1×10^{-4}, the discount factor γ is 0.99, and the batch size M is 20. The total number of batches is 10000.

The training of 200000 (= 20×10000) episodes takes about 34.2 h on a laptop computer with a CPU of Intel(R) Core(TM) i7-7700 @3.60 GHz and a GPU of NVIDIA GeForce GTX 1080. Both the CPU and GPU have participated in the computation. The RL task is solved in Python 3.8 environment. Fig. 5 plots the training history of the agent when the random seed equals 100, 200, and 300. The vertical axis indicates the average reward of the episodes in a batch. It can be seen that all of the 3 curves converged after 4000 batches of training, and the average reward remained approximately at 1000. The highest average reward of 1123.5 occurs at the 9708th batch when the random seed is 100. Its corresponding policy is regarded as the optimal policy, denoted as π^*.

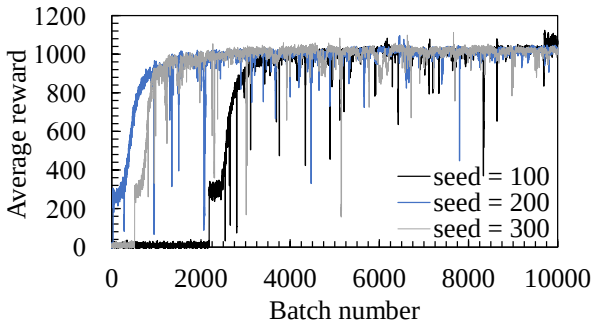


Fig. 5 Training history of the agent under various random seeds.

3.2 Testing

The optimal policy π^* is tested using the training node-sets. The trusses generated by policy π^* are as shown in Fig. 6. Note that the support conditions are the same as stated in Section 3.1, and they are not plotted for simplicity. Since the agent acts based on a stochastic policy, different trusses are generated under different random seeds. The trusses are all stable, and the majority of members are favorable.

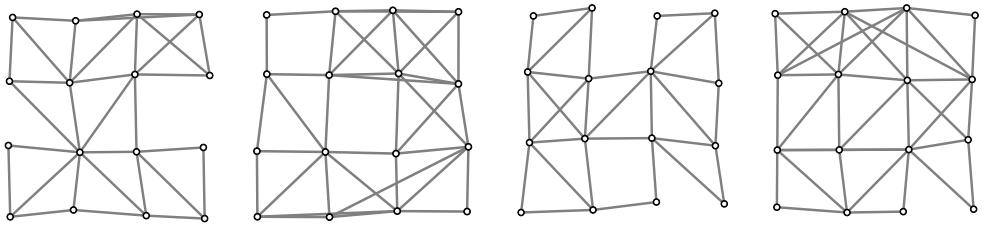


Fig. 6 Trusses generated by policy π^* using the training node-sets.

The generalization ability of the agent is tested using the following node-sets without re-training:

1) Node-set A: A regular 4×4 node-set, i.e., $\xi_{x,i}$ and $\xi_{y,i}$ equal to 0 in Eq. (26);
2) Node-set B: A regular 5×4 node-set shown in Fig. 7(a);
3) Node-set C: A regular 6×3 node-set shown in Fig. 7(b).

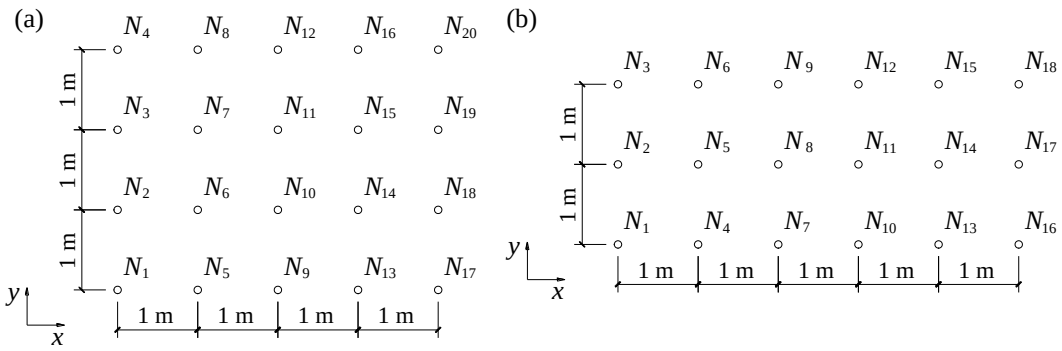


Fig. 7 Testing node-sets. (a) Node-set B. (b) Node-set C.

Trusses generated by π^* using node-sets A, B, and C are shown in Fig. 8, Fig. 9, and Fig. 10, respectively. It can be observed that the trusses generated by the trained

agent are all stable, and favorable members account for a large percentage. Therefore, it can be concluded that the agent is able to handle different-sized problems effectively without re-training.

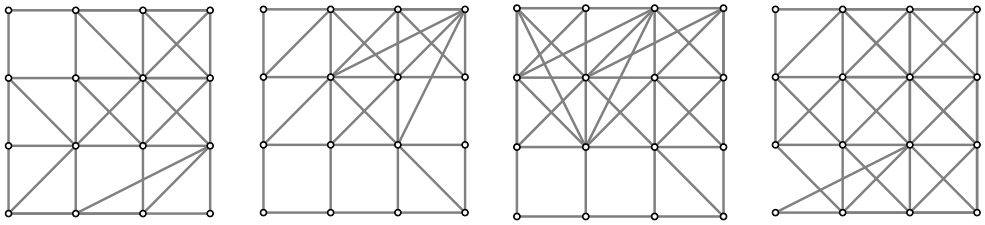


Fig. 8 Trusses generated by policy π^* using node-set A.

Fig. 11(a) and (b) illustrate the first and last four steps, respectively, of the generation process of a truss from node-set C. The probabilities of the actions are also plotted above the nodes, and the node highlighted in red corresponds to the action taken by the agent in the current step. Note that there are 39 members in step 78, including 3 unfavorable long overlapping members. The probability of each action when choosing the first node is uniform. This is because the reward of choosing the first node is 0, and the difference between the rewards of favorable and unfavorable members is not significant; therefore, the action scores Q_i are not yet updated to a positive value. Due to the existence of the ReLU function, Q_i is 0 for all actions. As the topology generated by π^* is reasonable, Fig. 12(a) and (b) show part of the generation process of another agent trained under $\lambda_0 = 40$ and other reward parameters unvaried. Let π^{**} denote its policy function. It can be seen that the probabilities of selecting FN are not uniform, while it also illustrates that different reward parameters lead to agents with various behaviors. Since the trained agent can provide the probability of taking each action, the trained agent can also assist engineers in designing the topology based on a given node-set.

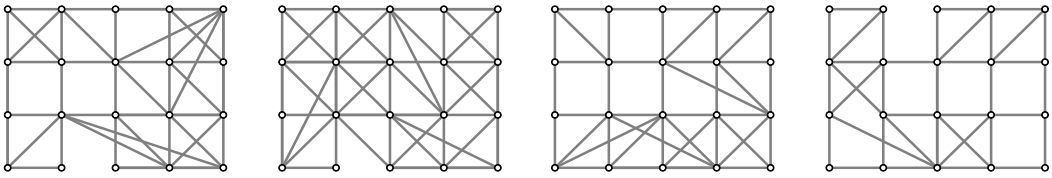


Fig. 9 Trusses generated by policy π^* using node-set B.

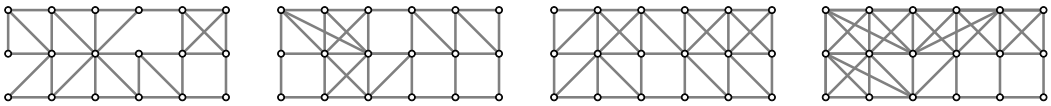


Fig. 10 Trusses generated by policy π^* using node-set C.

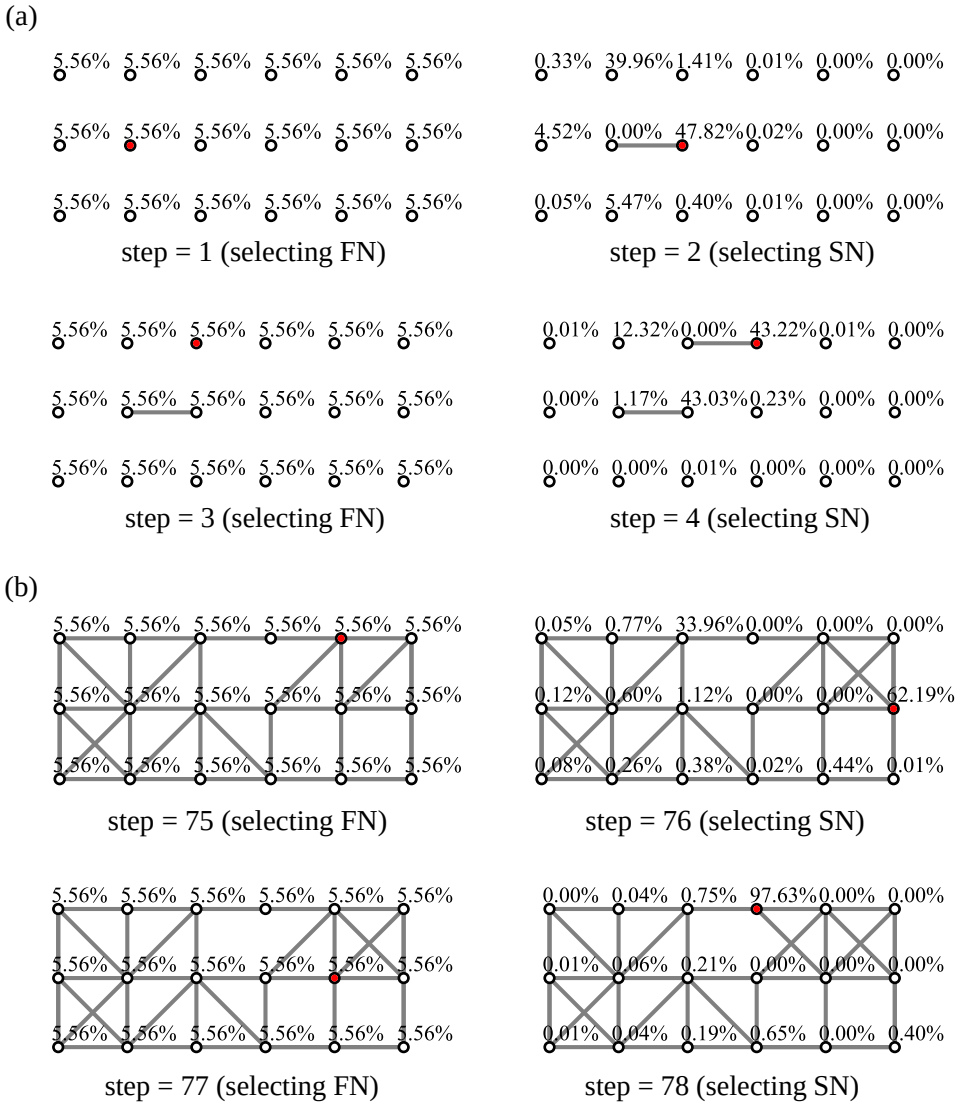
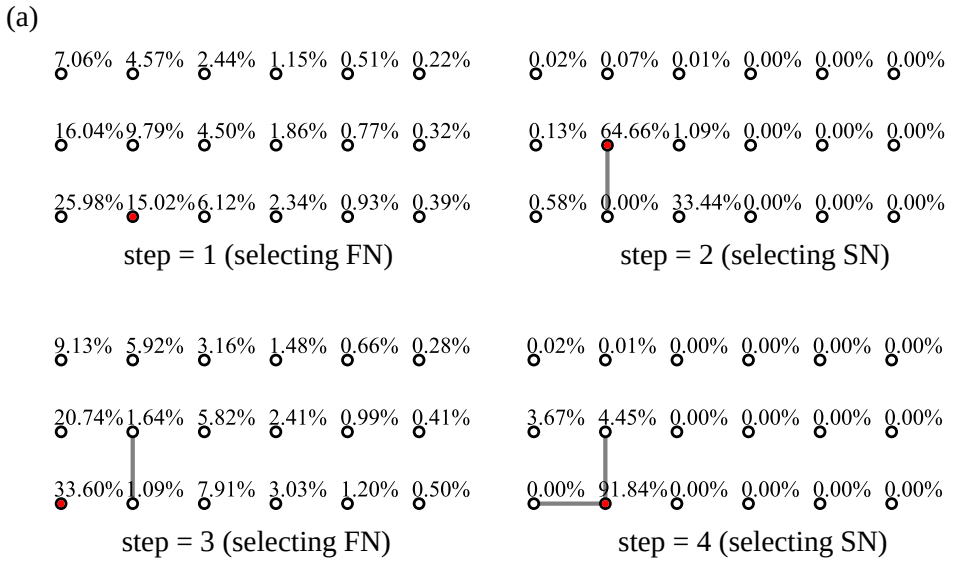


Fig. 11 Generation process of a truss using node-set C under π^*. (a) First four steps. (b) Last four steps.



(b)

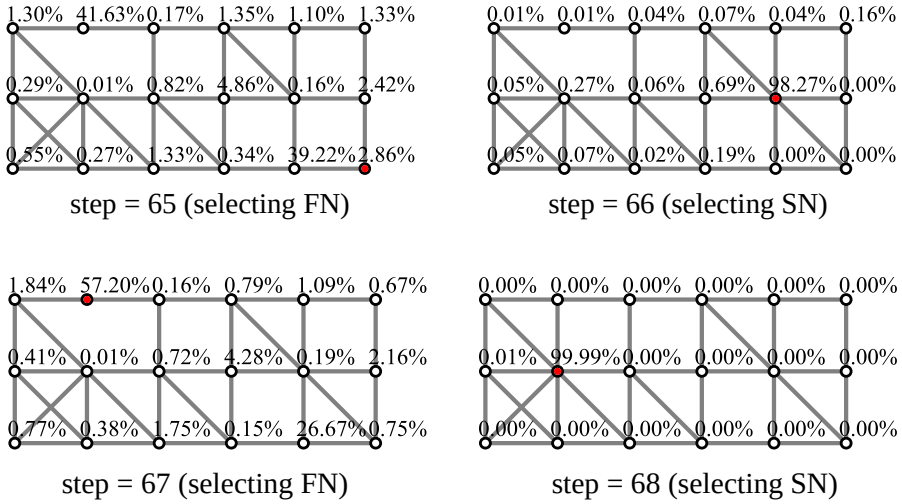


Fig. 12 Generation process of a truss using node-set C under π^{**}. (a) First four steps. (b) Last four steps.

4 Solving TO problems with singular optimal solutions with MGS

In this section, a framework is proposed for solving a TO problem with singular optimal solutions using the MGS method. Effectiveness of incorporating MGSs is demonstrated through numerical examples.

4.1 TO with stress and displacement constraints

The following TO problem with stress and displacement constraints is a typical TO problem with singular optimal solutions as stated in Section 1:

$$\begin{cases} \text{find} & \mathbf{A} \\ \text{min.} & V(\mathbf{A}) \\ \text{s.t.} & \max\limits_{i\in\Omega_{\mathrm{m},j},\, j\in\{1,2,\ldots,n_{\mathrm{L}}\}} \left(\dfrac{\left|\sigma_{i,j}(\mathbf{A})\right|}{\bar{\sigma}} \right) \le 1 \\ & \max\limits_{i\in\Omega_{\mathrm{dof},j},\, j\in\{1,2,\cdots,n_{\mathrm{L}}\}} \left(\dfrac{\left|u_{i,j}(\mathbf{A})\right|}{\bar{u}} \right) \le 1 \end{cases} \tag{27}$$

where $\mathbf{A} = (A_1, A_2, \ldots, A_m)$ is the vector of cross-sectional areas, and $V(\mathbf{A})$ is the total structural volume. n_{L} is the number of load cases. $\Omega_{\mathrm{m},j}$ and $\Omega_{\mathrm{dof},j}$ are the sets of indices of existing members and DOFs of existing nodes under the jth load case, respectively. $\sigma_{i,j}$ is the stress of the ith member in $\Omega_{\mathrm{m},j}$, and $u_{i,j}$ is the displacement of the ith DOF in $\Omega_{\mathrm{dof},j}$ under the jth load case. $\bar{\sigma}$ and $\bar{u}$ are the allowable stress and displacement, respectively. The displacement constraint is included to prevent obtaining an obvious and meaningless solution with no existing members for the stress constrained problem because the stress constraint need not be satisfied by a non-existing member. However,

a tight bound is given in Example 5 in Section 4.5 to compare the results with those obtained by existing methods.

The cross-sectional area A_i (i = 1, 2, …, m) is taken from a discrete section library $S = \{S_1, S_2, \ldots, S_{n_s}\}$ $(S_1 < S_2 < \ldots < S_{n_s})$ in numerical examples 1–4 and 6, where n_s is the total number of cross-sections in the library, while the section library can also be a continuous range as exemplified in Example 5.

4.2 TO method using MGSs

Since the MGS method is able to provide various stable GSs, we propose the following method to solve the TO problem presented in Section 4.1:

1) *Step* 1: *Generate the MGS*. Generate a truss with m_0 members using the agent trained by the RL method presented in Section 2 and demonstrated in Section 3. Randomly add members consecutively until the truss has m members. Assign the largest available cross-sectional area S_{n_s} to A_i (i = 1, 2, …, m);

2) *Step* 2: *Remove unnecessary members*. Conduct linear analysis for each load case and pick m_r members with the lowest stress ratios. Regard these members as removed by setting their cross-sectional areas as $10^{-6} S_{n_s}$. For problems with a loose displacement constraint, repeat the process above until the displacement constraint is violated; for problems with a tight displacement constraint, repeat step 2 and the fully-stressed design, i.e., step 3, until the displacement constraint is violated. Note that the stress ratio of the ith member R_i is defined as

$$R_i = \max_j \frac{|\sigma_{i,j}|}{\bar{\sigma}} \tag{28}$$

where $\sigma_{i,j}$ is the stress of the ith member ($i \in \Omega_m^*$) under the jth load case, and Ω_m^* is the set of remaining members.

3) *Step* 3: *Fully-stressed design* (*FSD*). Conduct linear analysis for the GS obtained in Step 2. Assign the cross-sectional area of the ith member as

$$A_i^F = R_i S_{n_s} \tag{29}$$

The cross-sectional area of the ith member can be selected from the library according to

$$A_i^* = \begin{cases} S_1 & \text{if } A_i^{\mathrm{F}} \le S_1 \\ S_j & \text{if } S_{j-1} \le A_i^{\mathrm{F}} \le S_j \ (j \ge 2) \end{cases} \tag{30}$$

If the stress constraint is not satisfied, repeat the following process and the selection of the cross-sectional area, i.e., Eq. (30), until the maximum stress among the existing members becomes smaller than $\bar{\sigma}$:

$$A_i^{\mathrm{F}} \leftarrow R_i A_i^{\mathrm{F}} \tag{31}$$

This way, we can obtain approximate optimal solutions from various GSs that have moderately sparse topologies, and may contain the members existing in the global optimal solution. A simple deterministic FSD is used here to substantiate the effectiveness of using the MGS in comparison to the fully-connected GS. Note that the process of Eq. (30) is omitted if the cross-sectional areas are continuous variables.

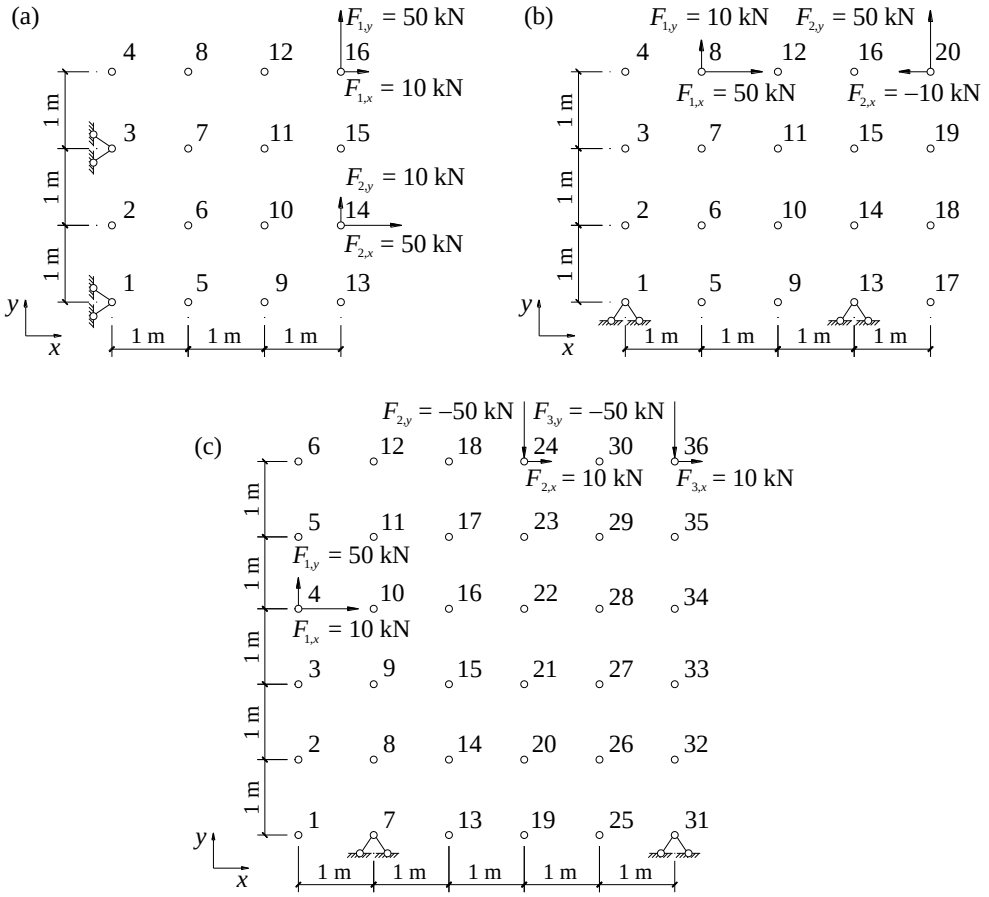


Fig. 13 Support and load conditions of the numerical examples. (a) Numerical example 1. (b) Numerical example 2. (c) Numerical example 3.

4.3 Numerical examples 1–3: Comparison of MGSs and fully-connected GS

Effectiveness of the method proposed in Section 4.2 is demonstrated by numerical examples of trusses with node-sets corresponding to 4×4, 5×4, and 6×6 regular node-sets, as shown in Fig. 13. For each numerical example, the truss has two pin-supports.

$F_{i,\beta}$ indicates the β-directional ($\beta \in \{x, y\}$) load of the ith load case.

The elastic modulus E is 2.0×10^5 MPa, the allowable stress $\bar{\sigma}$ is 200 MPa, and the allowable displacement $\bar{u}$ is taken as 100 times the maximum displacement in the GS to avoid meaningless solutions. The cross-section library is set to be **S** = {200, 400, 600, 800, 1000} (unit: mm^2), i.e., $n_s = 5$.

The node-set of numerical examples 1, 2, and 3 contain 16, 20, and 36 nodes, respectively. The number of translational constraints is 4. m_{min} and m_{max} are determined as listed in Table 3. The number of members m selected in the MGS for each numerical example is also listed in Table 3.

Table 3 Number of members in the numerical examples.

Numerical example	m_{min}	m_{max}	m
1	28	90	65
2	56	190	90
3	68	630	200

The agent with the optimal policy π^* obtained in Section 3 is used to generate MGSs without re-training for each node-set. In order to obtain 20 different MGSs, the random seeds are set as integers ranging from 0 to 19. The optimal solutions of numerical examples 1, 2, and 3 using MGSs are shown in Fig. 14, Fig. 15, and Fig. 16, respectively. Note that only the 3 optimal solutions with the least total structural volumes are exhibited for each numerical example. The optimal solutions using the fully-connected GS are also shown in Figs. 17–19 for comparison. Note that the fully-connected GS is also optimized according to Steps 2 and 3 in Section 4.2. In Figs. 14–19, *Max SR* indicates the maximum stress ratio of the existing members. Since the removed members are assigned a small cross-sectional area, n_v indicates the number of removed members that violate the stress constraint. A positive n_v indicates that the optimal solution is singular. The structure on the left is the GS, and the structure on the right is the corresponding optimal solution. In the optimal solution, the width of each member is proportional to its cross-sectional area. The statistical data of the CPU time, the total structural volume, and the maximum stress ratio for the 20 solutions under different random seeds in each numerical example are tabulated in Table 4, where 'Std.' denotes the standard deviation. The comparison of the structural volumes of the optimal solutions are tabulated in Table 5.

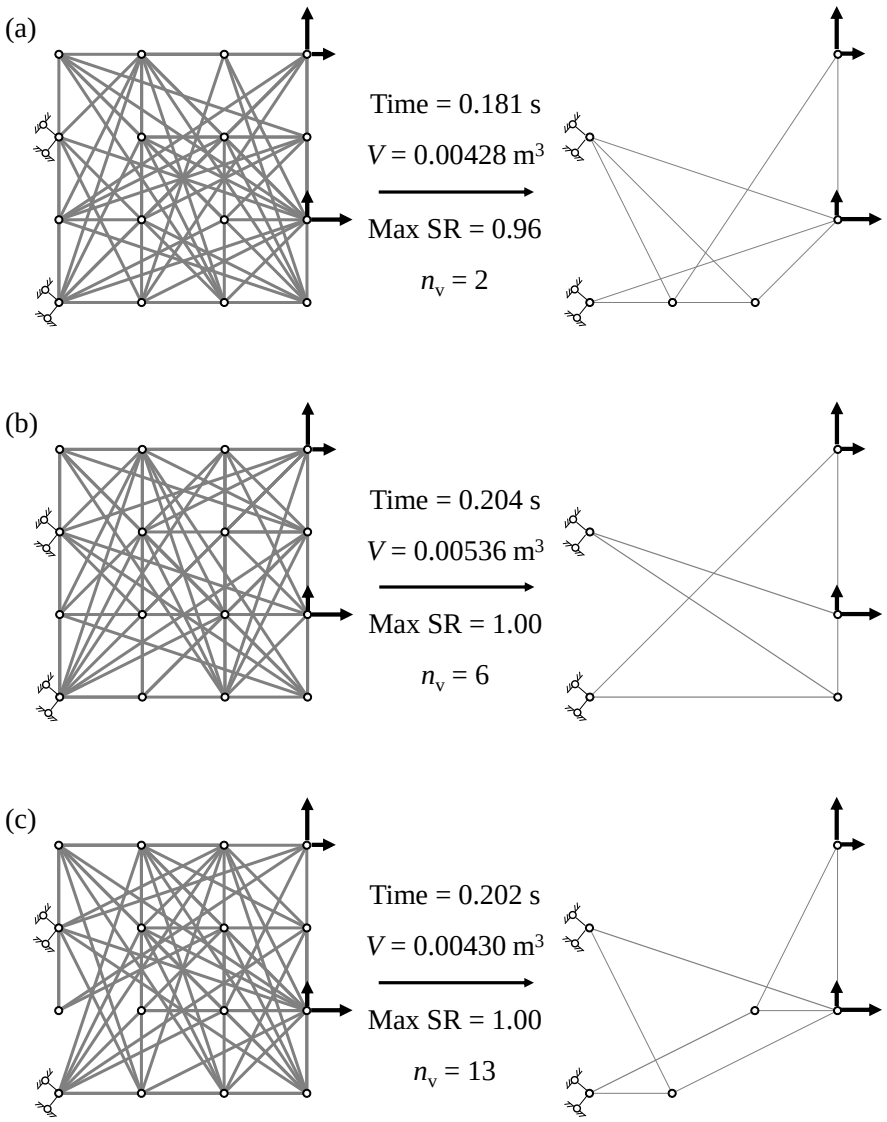
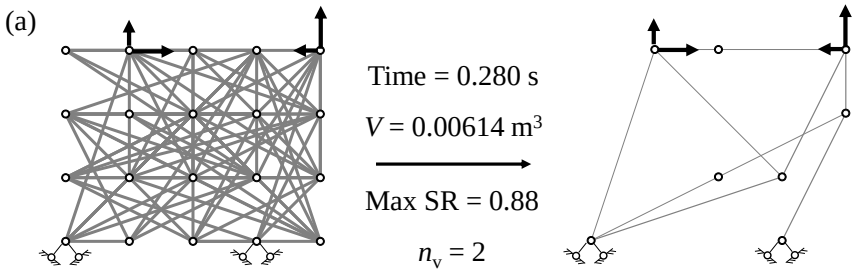


Fig. 14 Optimal solutions of numerical example 1 using the MGS method. (a) Seed = 1. (b) Seed = 14. (c) Seed = 17.



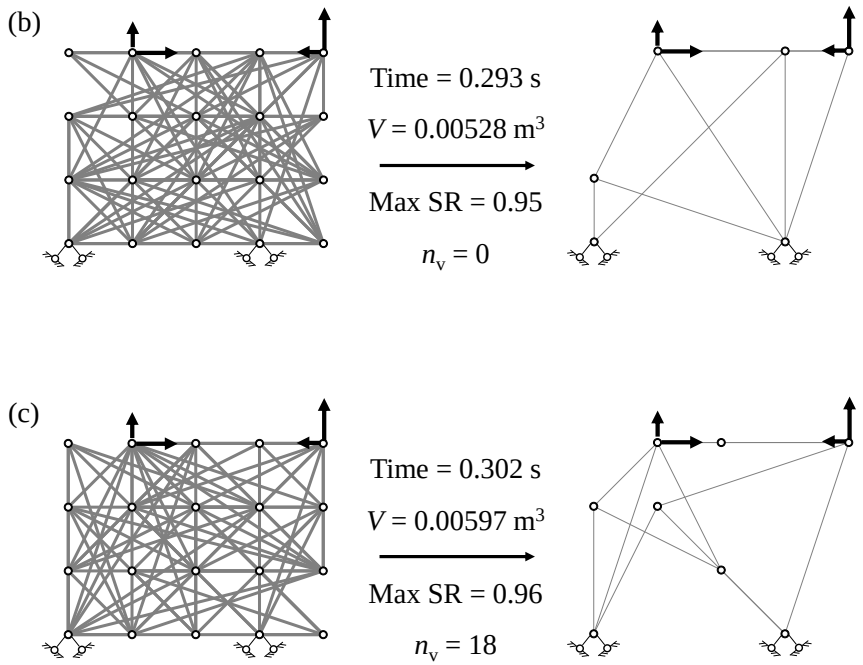
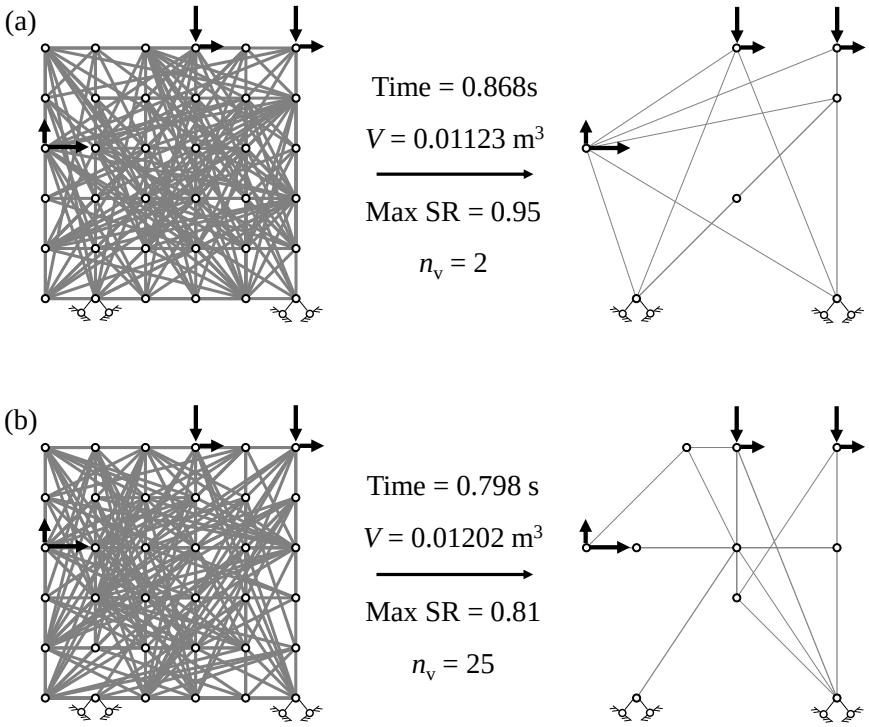


Fig. 15 Optimal solutions of numerical example 2 using the MGS method. (a) Seed = 7. (b) Seed = 14. (c) Seed = 16.



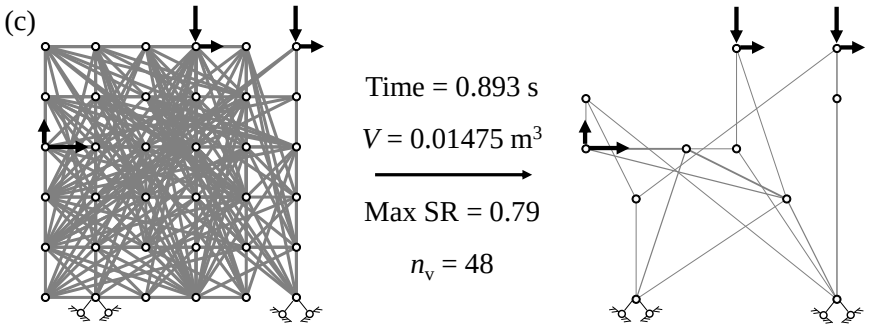


Fig. 16 Optimal solutions of numerical example 3 using the MGS method. (a) Seed = 5. (b) Seed = 7. (c) Seed = 11.

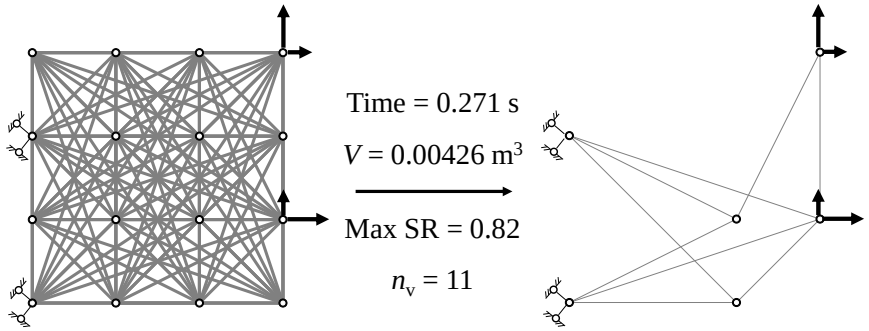


Fig. 17 Optimal solution of numerical example 1 using the fully-connected GS.

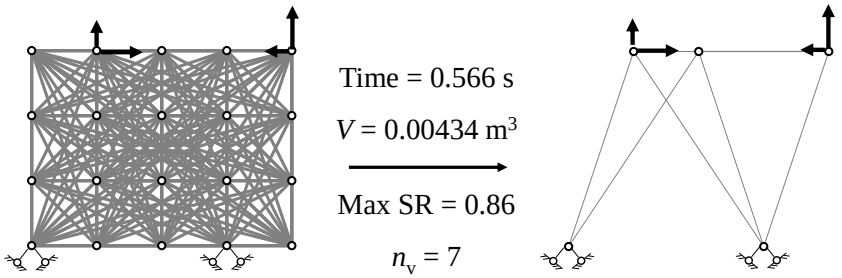


Fig. 18 Optimal solution of numerical example 2 using the fully-connected GS.

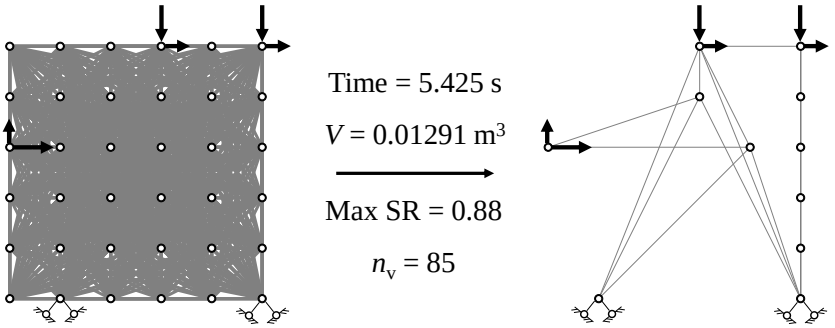


Fig. 19 Optimal solution of numerical example 3 using the fully-connected GS.

Table 4 Statistical data of the 20 results in numerical examples 1–3.

Numerical	CPU time / s				V / 10^{-3} m^3				Max SR			
example	Min.	Max.	Mean	Std.	Min.	Max.	Mean	Std.	Min.	Max.	Mean	Std.
1	0.169	0.412	0.226	0.064	4.28	9.68	6.81	1.43	0.79	1.00	0.92	0.07
2	0.262	0.423	0.302	0.035	5.28	17.16	9.58	3.07	0.55	1.00	0.88	0.12
3	0.634	1.645	0.888	0.202	11.23	23.01	16.36	3.45	0.75	1.00	0.88	0.07

Table 5 Comparison of structural volumes of optimal solutions in examples 1–3, and 6.

Typical solution	V / 10^{-3} m^3			
	Example 1	Example 2	Example 3	Example 6
1	4.28	6.14	11.23	5.74
2	5.36	5.28	12.02	5.57
3	4.30	5.97	14.75	4.99
fully-connected GS	4.26	4.34	12.91	5.63

Comparing the optimal solutions and the computational cost using MGSs and the fully-connected GS, it can be concluded that:

1) Based on the TO method proposed in this paper, various local optimal solutions can be obtained using the different MGSs. On the other hand, the fully-connected GS can provide only a deterministic optimal solution if a deterministic FSD is used;
2) The fully-connected GS does not necessarily lead to the global optimum, as stated in reference [11]. For example, the fully-connected GS of numerical example 3 produces an optimal solution with a volume of 0.01291 m^3. For the same problem, the optimal structural volumes using MGSs under random seeds 5 and 7 are 0.01123 m^3 and 0.01202 m^3, respectively. The optimal solution based on MGSs may, in some cases, be worse than the solution based on the fully-connected GS as shown in Table 5. However, for a TO problem with singular optimal solutions, it is more likely to obtain a solution closer to the global optimum if the MGS method is used;
3) As it takes the longest time to optimize the fully-connected GS of each numerical example, the calculation cost is reduced when a sparser MGS is used. Although the MGS method requires the TO to be conducted multiple times, the computational cost can be reasonably estimated according to the number of members in the MGS. Besides, as can be observed from the solutions of numerical example 3, though the total CPU time for 20 solutions by MGS is larger than the CPU time for the fully-connected GS, the advantage of using

MGS will be significant for a large-scale problem;

4) Although the difference between the maximum and minimum values of the CPU time, the total structural volume, and the maximum stress ratio for the 20 different random seeds are large, their standard deviations are moderately small.

4.4 Numerical example 4: Comparison of MGSs and human-specified GS

Rasmussen and Stolpe [40] used the parallel cut-and-branch method to optimize an L-shaped truss shown in Fig. 20, denoted as numerical example 4. Pinned supports are located at nodes 5 and 15. A vertical load F = 450 kN is applied at node 20. The elastic modulus is 70 GPa. The section library is $\{5, 10\}\times10^{-3}$ m^2. Fig. 21 plots the optimal solutions solved by the reference [40] under various allowable stresses, where the number beside each member indicates the cross-sectional area (unit: 10^{-3} m^2). The structural volumes of structures in Figs. 21 are tabulated in Table 6.

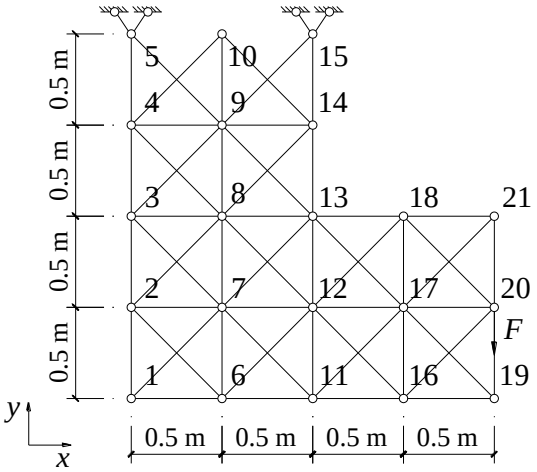


Fig. 20 Support and load conditions of numerical example 4.

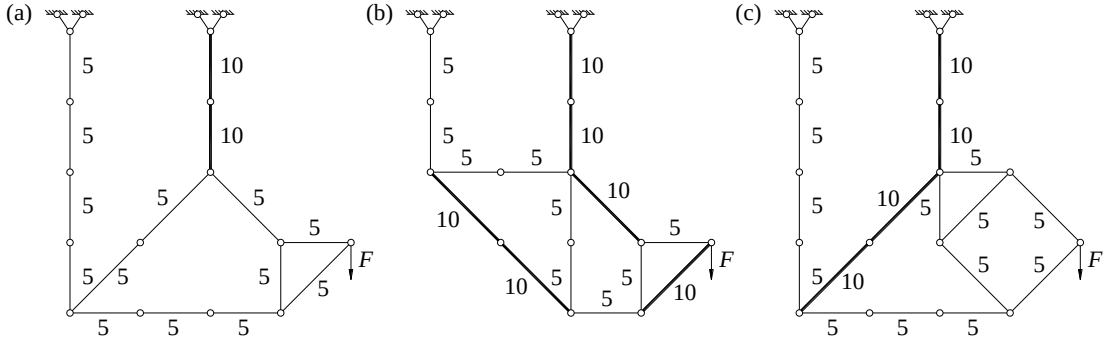


Fig. 21 Optimal solutions of numerical example 4 under various allowable stresses obtained in Ref. [40]. (a) $\bar{\sigma}=170$ MPa . (b) $\bar{\sigma}=130$ MPa . (c) $\bar{\sigma}=90$ MPa .

The node-set in Fig. 20 is used to generate MGSs with m = 100 members, and the agent with policy π^* trained in Section 3 is utilized. Note that there are 21 nodes and 4 translational constraints, so $m_{\min}$ and $m_{\max}$ are 38 and 210, respectively. During the

generation process of the MGS, members outside the L-shaped design domain, emerging due to nonconvexity of the design domain, are simply ignored without assigning additional constraints because the agent acts stochastically.

Table 6 Comparison of optimal solutions of numerical example 4.

$\bar{\sigma}$ / MPa	Total structural volume / m^3	
	Method in Ref. [40]	Proposed method
170	0.0466	0.0432
130	0.0608	0.0519
90	0.0608	0.0564

For each allowable stress, 20 MGSs are generated by varying the random seeds from 0 to 19. The optimal solutions for $\bar{\sigma}$ = 170, 130 MPa, and 90 MPa occur at the 18th, 6th, and 13th MGS, respectively. The optimal solutions and the corresponding MGSs are shown in Figs. 22, 23, and 24, respectively. The structural volumes are also tabulated in Table 6. It can be observed that the structural volumes of solutions obtained by MGSs are smaller than those of the solution obtained by a human-specified GS in reference [40], because longer diagonal members are allowed to exist in the MGSs. It is notable that the agent trained using a regular-shaped truss is also effective for the irregular-shaped truss.

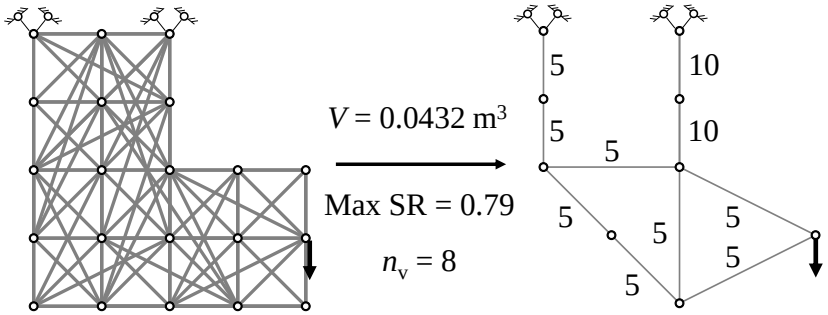


Fig. 22 Optimal solution of numerical example 4 obtained by the 18th MGS ($\bar{\sigma}$ = 170 MPa).

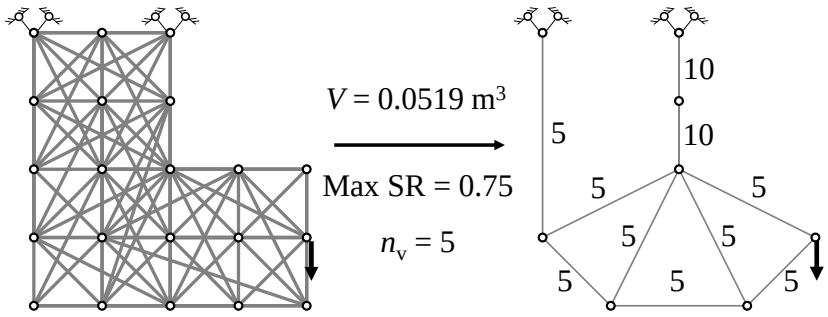


Fig. 23 Optimal solution of numerical example 4 obtained by the 6th MGS ($\bar{\sigma}$ = 130 MPa).

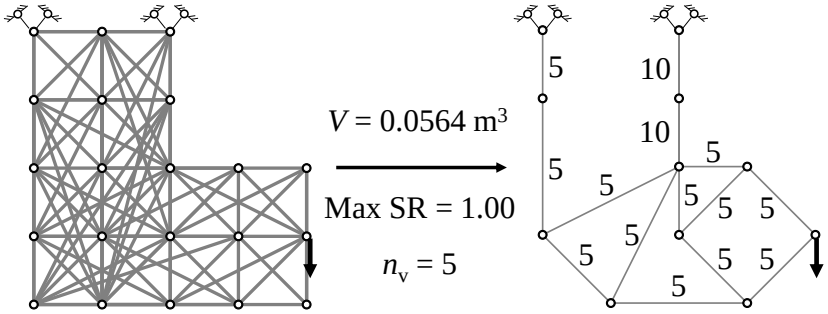


Fig. 24 Optimal solution of numerical example 4 obtained by the 13th MGS ($\bar{\sigma} = 90$ MPa).

4.5 Numerical example 5: Problem with a strict displacement constraint

In the studies of Deb and Gulati [41] and El Bouzouiki *et al*. [42], a 39-member 12-node truss shown in Fig. 25 is optimized using the GA and the non-uniform cellular automata framework, respectively. We denote this example as numerical example 5. The truss is simply-supported, and the single load case contains three concentrated loads $F = 90.72$ kN. The density of the material is 2768 kg/m^3, and the elastic modulus is 68947 MPa. The allowable stress and displacement are 137.89 MPa and 0.0508 m, respectively. The cross-sectional library is a continuous variable, and the lower and upper bounds are 6.45×10^{-4} mm^2 and 1.45×10^{3} mm^2, respectively. The optimal topologies obtained by references [41] and [42] are shown in Fig. 26(a) and Fig. 26(b), respectively. The cross-sectional areas and the structural weights are tabulated in Table 7. Note that a single cross-sectional area corresponding to two members indicates that the two members have the same cross-sectional area.

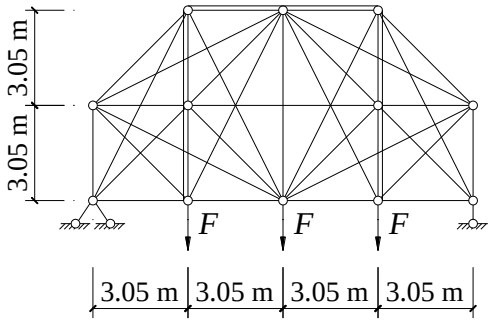


Fig. 25 Support and load conditions of numerical example 5.

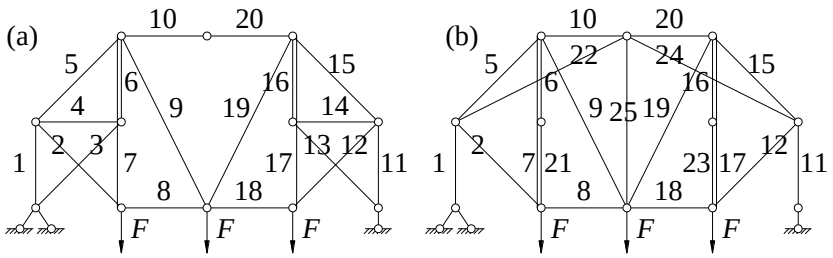


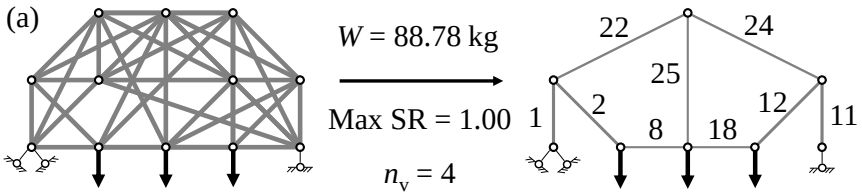
Fig. 26 Optimal solutions of numerical example 5. (a) Ref. [41]. (b) Ref. [42].

The node-set in Fig. 25 is used to generate 50 MGSs with m = 39 members by varying the random seeds from 0 to 49, and the agent with policy π^* trained in Section 3 is utilized. Note that there are 12 nodes and 3 translational displacement constraints, and accordingly, $m_{\min}$ and $m_{\max}$ are 21 and 66, respectively. The optimal solutions based on the 1st and the 26th MGS are plotted in Fig. 27, where W indicates the total structural weight. The cross-sectional areas and the structural weights are also tabulated in Table 7 for comparison. It can be seen that the 1st MGS leads to a solution superior to the solution in reference [41]. Besides, the weight of the solution based on the 26th MGS is higher than yet quite close to that of the solution in reference [42]. However, we can also expect a better solution when the number of MGSs increases.

Table 7 Comparison of optimal solutions of numerical example 5.

Member No.	Cross-sectional area / mm^2			
	Fig. 26(a) [41]	Fig. 26(b) [42]	Fig. 27(a)	Fig. 27(b)
1, 11	969.03	967.74	986.20	986.40, 1005.69
2, 12	684.51	754.97	929.30	931.45, 733.67
3, 13	32.90	–	–	–
4, 14	32.90	–	–	–
5, 15	33.55	472.19	–	–, 688.59
6, 16	33.55	55.61	–	–
7, 17	161.94	55.61	–	–
8, 18	484.52	534.00	657.25	178.92, 189.74
9, 19	360.64	248.84	–	–, 437.29
10, 20	648.39	222.58	–	–, 676.13
21, 23	–	55.61	–	–
22, 24	–	223.55	734.59	731.68, –
25	–	199.35	657.20	284.23
Structural weight / kg	89.152	87.090	88.780	87.280

Therefore, it can be concluded that the MGS method is also available to handle problems when both the stress and displacement constraints are tight, and to obtain favorable solutions.



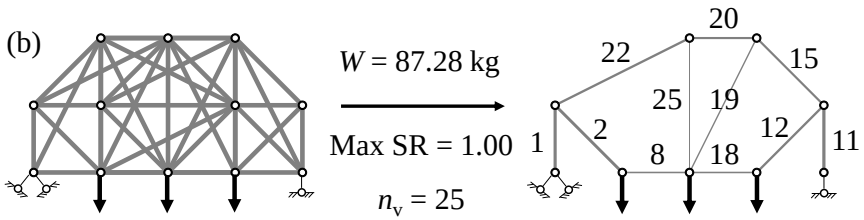


Fig. 27 Optimal solutions of numerical example 5 obtained by the MGS method. (a) The 1st MGS. (b) The 26th MGS.

4.6 Numerical example 6: Problem considering member buckling

In the numerical examples 1–5, the allowable stress of tension and compression is considered to be equal, i.e., the member buckling is ignored. In order to demonstrate the feasibility of the MGS method under member buckling constraints, the same TO problem of numerical example 1 is solved by replacing the constant allowable stress $\bar{\sigma}$ (unit: MPa) as

$$\bar{\sigma} = \begin{cases} 200 & \text{for tension members} \\ \min\{\sigma_{\mathrm{cr}}, 200\} & \text{for compression members} \end{cases} \tag{32}$$

where σ_{cr} is the Euler buckling stress of the member, which is related to the member length and the radius of gyration of the cross-section. For calculation of σ_{cr}, square hollow sections are assigned to the cross-section library $\mathbf{S} = \{200, 400, 600, 800, 1000\}$ (unit: mm^2), as tabulated in Table 8.

Since the Euler buckling stress varies non-linearly with the cross-sectional area, the FSD design, i.e., *Step* 3 in Section 4.2, is not applicable for compression members. Instead, we use the axial force to determine the cross-section of compression members. Given the absolute value of the axial force of the ith member in compression is $N_{\mathrm{c},i}$ (i = 1, 2, …, n_{m}), then its cross-sectional area is determined as

$$A_i^* = \begin{cases} S_1 & \text{if } N_{\mathrm{c},i} \le N_{\mathrm{cr},i,1} \\ S_j & \text{if } N_{\mathrm{cr},i,j-1} \le N_{\mathrm{c},i} \le N_{\mathrm{cr},i,j} \ (j \ge 2) \end{cases} \tag{33}$$

where $N_{\mathrm{cr},i,j}$ (j = 1, 2, …, n_{s}) is the Euler buckling load of the ith member when the jth cross-section is adopted. Note that $N_{\mathrm{cr},i,j}$ is related to the length of the ith member. Therefore, the intermediate nodes connecting only 2 colinear members should be fixed or removed, e.g., the intermediate nodes on the right side of Fig. 19.

Table 8 Cross-sectional properties of the sections in numerical example 6.

Width and height/ mm	Thickness / mm	Area / mm^2	Radius of gyration / mm
50	1.021	200	20.000
52	2.000	400	20.429

54	2.938	600	20.881
56	3.834	800	21.354
58	4.690	1000	21.848

The agent with the optimal policy π^* obtained in Section 3 is used to generate 30 MGSs without re-training, and TO is conducted based on the modified method mentioned above. The 3 typical solutions are shown in Fig. 28. The optimal solution using the fully-connected GS is also shown in Fig. 29 for comparison. Note that the red line indicates that the member is in compression under at least one load case. The structural volumes are tabulated in Table 5 for comparison.

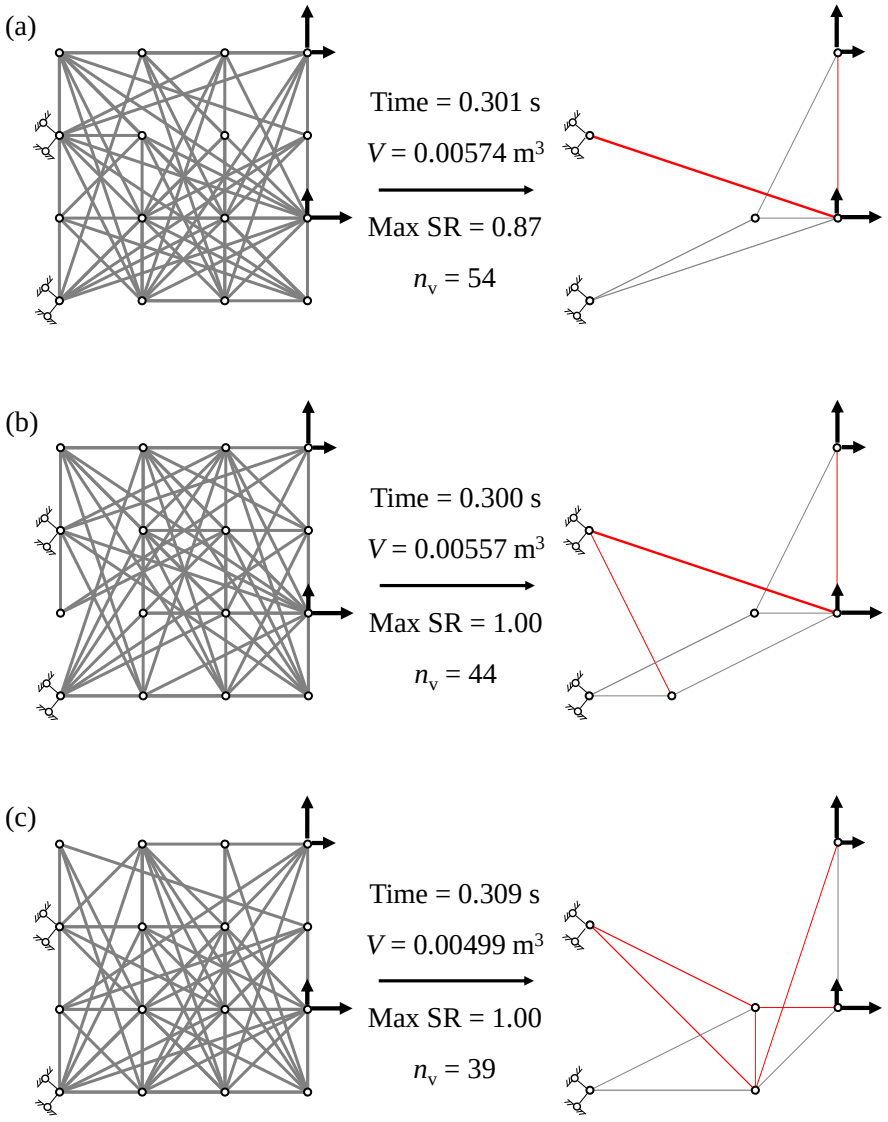


Fig. 28 Optimal solutions of numerical example 6 using the MGS method. (a) Seed = 10. (b) Seed = 17. (c) Seed = 22.

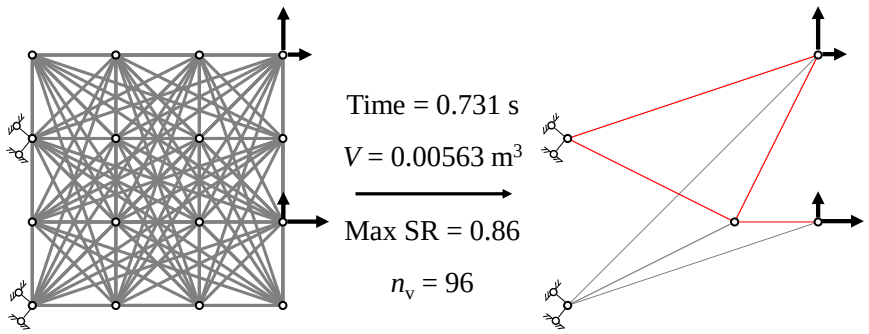


Fig. 29 Optimal solution of numerical example 6 using the fully-connected GS.

Comparing the optimal solutions of numerical examples 1 and 6, it can be concluded that:

1) Member buckling constraint can be incorporated by assigning the Euler buckling stress as the lower bound for the stress;
2) The MGS method can be effectively incorporated with different TO algorithms to obtain various local optimal solutions;
3) Although some of the optimal topologies considering member buckling are the same as those without considering member buckling, i.e., Fig. 14(c) and Fig. 28(b), one can expect an optimal topology with shorter compression members by generating more MGSs, e.g., Fig. 28(c), as the computational cost of generating MGSs is relatively low if a trained agent is used.

5 Conclusions

This paper proposes the MGS method for TO of binary trusses using RL and GE. Through a numerical example of the training and testing of the RL agent, the following conclusions are drawn:

1) By defining the action for nodes, rather than members, and by specifying a member by the first and second nodes to be selected, the size of action space is drastically reduced compared with the case of defining the action with respect to the member to be added;
2) The policy of the agent converges with a high average reward under different random seeds of training, i.e., the formulation of the RL task is robust against the randomness during the training process;
3) The RL agent can produce different stable trusses under different random seeds since a stochastic policy has been employed;
4) By incorporating GE in extracting the structural information, the agent can handle trusses with node-sets of different sizes from the training ones.

The trained RL agent for generating stable binary trusses is also expected to assist the topology design, as the various stable trusses generated by the agent involving randomness can be used as MGSs for improving the possibility of obtaining the global optimal topology. Based on the MGS method, a framework to solve the TO problem with stress and displacement constraints is proposed as an example. Six numerical examples are solved using the proposed framework. Numerical examples 1–3 compare the optimal solutions obtained based on MGSs and fully-connected GSs; numerical example 4 compares the optimal solution obtained based on MGSs and human-specified GS; numerical example 5 demonstrates the feasibility of using MGSs on TO problems with tight displacement constraints; numerical example 6 deals with TO problems with member buckling constraints. In specific, the following conclusions are obtained:

5) Different MGSs can lead to different optimal solutions, while the fully-connected GS or a specific human-specified GS can only provide a deterministic optimal solution if a deterministic TO method is used. MGSs can lead to better solutions compared with the fully-connected GS and human-specified GSs. Hence, the MGS method is more likely to obtain a global optimal solution for a TO problem with singular optimal solutions;
6) The computational cost of optimizing a sparse MGS is lower than that of optimizing a fully-connected GS. Even if one intends to conduct TO on multiple MGSs, the computational cost can still be reasonably estimated. Therefore, the MGS method has a significant advantage in solving large-scale problems;
7) Member buckling constraint can be incorporated by assigning the Euler buckling stress as the lower bound for the stress. Furthermore, by avoiding the existence of unfavorable long members, the MGS method can effectively handle problems with member buckling constraints, where short members are favored for compression;
8) A GS including all members existing in the global optimal solution can be generated by taking the union of members that are effective for each load case;
9) The MGS method is also effective when a tight displacement constraint is assigned, and can be incorporated with different TO algorithms to obtain various local optimal solutions other than stress and displacement constraints.

Since MGSs are various stable GSs with appropriate statical indeterminacy, the agent trained in this paper can be used for optimization of binary trusses with various design conditions.

Data Availability Statement

Some or all data, models, or codes that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments

The authors gratefully acknowledge the financial support provided by the China Scholarship Council (CSC) during a visit of Shaojun Zhu (No. 201906260152) to Kyoto University. Besides, the first author would like to acknowledge Bolei Zhou for providing an open-access course on reinforcement learning and Qiang Zeng for his suggestions on this research. The second author acknowledges the support by JSPS KAKENHI Grant Number JP20H04467.

References

[1] Dorn W, Gomory R, and Greenberg H. Automatic design of optimal structures. Journal de Mécanique, 1964, 3: 25-52.

[2] Topping B H V. Shape optimization of skeletal structures: a review. Journal of Structural Engineering, 1983, 109(8): 1933-1951. doi: 10.1061/(ASCE)0733-9445(1983)109:8(1933)

[3] Miguel L F F, Lopez R H, and Miguel L F F. Multimodal size, shape, and topology optimisation of truss structures using the Firefly algorithm. Advances in Engineering Software, 2013, 56: 23-37. doi: 10.1016/j.advengsoft.2012.11.006

[4] Finotto V C, da Silva W R L, Valášek M, and Štemberk P. Hybrid fuzzy-genetic system for optimising cabled-truss structures. Advances in Engineering Software, 2013, 62: 85-96. doi: 10.1016/j.advengsoft.2013.04.012

[5] Hayashi K and Ohsaki M. FDMopt: Force density method for optimal geometry and topology of trusses. Advances in Engineering Software, 2019, 133: 12-19. doi: 10.1016/j.advengsoft.2019.04.002

[6] Michell A G M. The limits of economy of material in frame-structures. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 1904, 8(47): 589-597. doi: 10.1080/14786440409463229

[7] Kawamura H, Ohmori H, and Kito N. Truss topology optimization by a modified genetic algorithm. Structural and Multidisciplinary Optimization, 2002, 23(6): 467-473. doi: 10.1007/s00158-002-0208-0

[8] Shakya A, Nanakorn P, and Petprakob W. A ground-structure-based representation with an element-removal algorithm for truss topology optimization. Structural and Multidisciplinary Optimization, 2018, 58(2): 657-675. doi: 10.1007/s00158-018-1917-3

[9] Kirsch U. Optimal topologies of truss structures. Computer Methods in Applied Mechanics and Engineering, 1989, 72(1): 15-28. doi: 10.1016/0045-7825(89)90119-9

[10] Ohsaki M. Genetic algorithm for topology optimization of trusses. Computers & Structures, 1995, 57(2): 219-225. doi: 10.1016/0045-7949(94)00617-C

[11] Hagishita T and Ohsaki M. Topology optimization of trusses by growing ground structure method. Structural and Multidisciplinary Optimization, 2009, 37(4): 377-393. doi: 10.1007/s00158-008-0237-4

[12] Steven G, Querin O, and Xie M. Evolutionary structural optimisation (ESO) for combined topology and size optimisation of discrete structures. Computer Methods in Applied Mechanics and Engineering, 2000, 188(4): 743-754. doi: 10.1016/S0045-7825(99)00359-X

[13] Guo X, Cheng G, and Yamazaki K. A new approach for the solution of singular optima in truss topology optimization with stress and local buckling constraints. Structural and Multidisciplinary Optimization, 2001, 22(5): 364-373. doi: 10.1007/s00158-001-0156-0

[14] Stolpe M and Svanberg K. A note on stress-constrained truss topology optimization. Structural and Multidisciplinary Optimization, 2003, 25(1): 62-64. doi: 10.1007/s00158-002-0273-4

[15] Richardson J N, Adriaenssens S, Bouillard P, and Coelho R F. Multiobjective topology optimization of truss structures with kinematic stability repair. Structural and multidisciplinary optimization, 2012, 46(4): 513-532. doi: 10.1007/s00158-012-0777-5.

[16] Mela K. Resolving issues with member buckling in truss topology optimization using a mixed variable approach. Structural and Multidisciplinary Optimization, 2014, 50(6): 1037-1049. doi: 10.1007/s00158-014-1095-x.

[17] Zhou P, Du J, and Zhenhua L Ü. Interval analysis based robust truss optimization with continuous and discrete variables using mix-coded genetic algorithm. Structural and Multidisciplinary Optimization, 2017, 56(2): 353-370. doi: 10.1007/s00158-017-1668-6.

[18] Pedroza-Villalba M, Portilla-Flores E A, Vega-Alvarado E, Calva-Yáñez M B, Santiago-Valentín E, and Alcalá-Fazio E. Truss topology optimization based on a birth/death element approach. IEEE Access, 2018, 6: 72609-72619. doi: 10.1109/ACCESS.2018.2881609

[19] Tejani G G, Savsani V J, Bureerat S, Patel V K, and Savsani P. Topology optimization of truss subjected to static and dynamic constraints by integrating simulated annealing into passing vehicle search algorithms. Engineering with

Computers, 2019, 35(2): 499-517. doi: 10.1007/s00366-018-0612-8

[20] Beekers M and Fleury C. A primal-dual approach in truss topology optimization. Computers & Structures, 1997, 64(1-4): 77-88. doi: 10.1016/S0045-7949(96)00144-7

[21] Sun H, Burton H V, and Huang H. Machine learning applications for building structural design and performance assessment: state-of-the-art review. Journal of Building Engineering, 2020: 101816. doi: 10.1016/j.jobe.2020.101816

[22] Zhu S, Ohsaki M, and Guo X. Prediction of non-linear buckling load of imperfect reticulated shell using modified consistent imperfection and machine learning. Engineering Structures, 2021, 226: 111374. doi: 10.1016/j.engstruct.2020.111374

[23] Olalusi O B and Awoyera P O. Shear capacity prediction of slender reinforced concrete structures with steel fibers using machine learning. Engineering Structures, 2021, 227: 111470. doi: 10.1016/j.engstruct.2020.111470

[24] Fu F. Fire induced progressive collapse potential assessment of steel framed buildings using machine learning. Journal of Constructional Steel Research, 2020, 166: 105918. doi: 10.1016/j.jcsr.2019.105918

[25] Wang Z and Cha Y J. Unsupervised deep learning approach using a deep auto-encoder with a one-class support vector machine to detect structural damage. Structural Health Monitoring, 2020: 1475921720934051. doi: 10.1177/1475921720934051

[26] Entezami A, Shariatmadar H, and Mariani S. Fast unsupervised learning methods for structural health monitoring with large vibration data from dense sensor networks. Structural Health Monitoring, 2020, 19(6): 1685-1710. doi: 10.1177/1475921719894186

[27] Flah M, Nunez I, Chaabene W B, and Nehdi M L. Machine learning algorithms in civil structural health monitoring: a systematic review. Archives of Computational Methods in Engineering, 2020: 1-23. doi: 10.1007/s11831-020-09471-9

[28] Sutton R S. Reinforcement Learning. Boston: Springer, 1992. doi: 10.1007/978-1-4615-3618-5

[29] Sallab A E L, Abdou M, Perot E, and Yogamani S. Deep reinforcement learning framework for autonomous driving. Electronic Imaging, 2017, 2017(19): 70-76. doi: 10.2352/ISSN.2470-1173.2017.19.AVM-023

[30] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, Baker L, Lai M, Bolton A, Chen Y, Lillicrap T, Hui F, Sifre L, Driessche G, Graepel T, and Hassabis D. Mastering the game of go without human knowledge. Nature, 2017, 550(7676): 354-359. doi: 10.1038/nature24270

[31] Gamache J F, Vadean A, Noirot-Nérin É, Beaini D, and Achiche S. Image-based

truss recognition for density-based topology optimization approach. Structural and Multidisciplinary Optimization, 2018, 58(6): 2697-2709. doi: 10.1007/s00158-018-2028-x

[32] Sahachaisaree S, Sae-ma P, and Nanakorn P. Two-Dimensional Truss Topology Design by Reinforcement Learning//ICSCEA 2019. Springer, Singapore, 2020: 1237-1245. doi: 10.1007/978-981-15-5144-4_122

[33] LeCun Y, Bottou L, Bengio Y, and Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278-2324. doi: 10.1109/5.726791

[34] Hayashi K and Ohsaki M. Reinforcement learning and graph embedding for binary truss topology optimization under stress and displacement constraints. Frontiers in Built Environment, 2020, 6: 59. doi: 10.3389/fbuil.2020.00059

[35] Makarov I, Kiselev D, Nikitinsky N, and Subelj L. Survey on graph embeddings and their applications to machine learning problems on graphs. PeerJ Computer Science, 2021, 7:e357. doi: 10.7717/peerj-cs.357

[36] Dai H, Khalil E B, Zhang Y, Dilkina B, and Song L. Learning combinatorial optimization algorithms over graphs. arXiv preprint arXiv:1704.01665, 2017. arXiv:1704.01665

[37] Watkins C J C H and Dayan P. Q-learning. Machine Learning, 1992, 8: 279-292. doi: 10.1023/A:1022676722315

[38] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 1992, 8: 229-256. doi: 10.1007/BF00992696

[39] Tieleman T and Hinton G. Lecture 6.5 - RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012, 4: 26-31.

[40] Rasmussen M H and Stolpe M. Global optimization of discrete truss topology design problems using a parallel cut-and-branch method. Computers & Structures, 2008, 86(13-14): 1527-1538. doi: 10.1016/j.compstruc.2007.05.019

[41] Deb K and Gulati S. Design of truss-structures for minimum weight using genetic algorithms. Finite Elements in Analysis and Design, 2001, 37(5): 447-465. doi: 10.1016/S0168-874X(00)00057-3

[42] El Bouzouiki M, Sedaghati R, and Stiharu I. A non-uniform cellular automata framework for topology and sizing optimization of truss structures subjected to stress and displacement constraints. Computers & Structures, 2021, 242: 106394. doi: 10.1016/j.compstruc.2020.106394