



TITLE:

Graph-based reinforcement learning for discrete cross-section optimization of planar steel frames

AUTHOR(S):

Hayashi, Kazuki; Ohsaki, Makoto

CITATION:

Hayashi, Kazuki ...[et al]. Graph-based reinforcement learning for discrete cross-section optimization of planar steel frames. *Advanced Engineering Informatics* 2022, 51: 101512.

ISSUE DATE:

2022-01

URL:

<http://hdl.handle.net/2433/277035>

RIGHT:

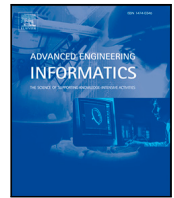
© 2022 The Authors. Published by Elsevier Ltd.; This is an open access article under the CC BY license.



Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei



Full length article

Graph-based reinforcement learning for discrete cross-section optimization of planar steel frames

Kazuki Hayashi*, Makoto Ohsaki

Department of Architecture and Architectural Engineering, Graduate School of Engineering, Kyoto University, Kyoto, Japan



ARTICLE INFO

Keywords:

Machine learning
Reinforcement learning
Graph embedding
Structural optimization
Cross-section optimization
Steel frame

ABSTRACT

A combined method of graph embedding (GE) and reinforcement learning (RL) is developed for discrete cross-section optimization of planar steel frames, in which the section size of each member is selected from a prescribed list of standard sections. The RL agent aims to minimize the total structural volume under various practical constraints. GE is a method for extracting features from data with irregular connectivity. While most of the existing GE methods aim at extracting node features, an improved GE formulation is developed for extracting features of edges associated with members in this study. Owing to the proposed GE operations, the agent is capable of grasping the structural property of columns and beams considering their connectivity in a frame with an arbitrary size as feature vectors of the same size. Using the feature vectors, the agent is trained to estimate the accurate return associated with each action and to take proper actions on which members to reduce or increase their size using an RL algorithm. The applicability of the proposed method is versatile because various frames different in the numbers of nodes and members can be used for both training and application phases. In the numerical examples, the trained agents outperform a particle swarm optimization method as a benchmark in terms of both computational cost and design quality for cross-sectional design changes; the agents successfully assign reasonable cross-sections considering the geometry, connectivity, and support and load conditions of the frames.

1. Introduction

In the design of standard building frames composed of columns and beams, the frame shape is often determined mainly from the building plan, and determining the member cross-sections after fixing the shape has an important role in the structural design process. Therefore, cross-section optimization of frames in which the member cross-sections are handled as design variables is one of the problems of great practical interest. Although the support condition is also an important factor to design steel frames, the supports are not the target of design changes in this study.

If the section sizes can vary continuously, the optimization problem can be formulated as a nonlinear programming problem, and mathematical programming approaches are available. Pezeshk [1] optimized frame cross-sections utilizing sensitivity analysis to minimize the structural weight under displacement constraints considering nonlinear behavior of the structure. Kimura et al. [2] optimized the plate thickness of square steel pipes and the flange-web thickness of I-beams to minimize the roof displacement of the frame, in which the design variables are continuous.

However, considering that most of steel members used for general buildings are standard products manufactured at a factory, it is preferable that the design problem is described as a combinatorial optimization problem in which each member's cross-section is selected from a prescribed list of standard sections [3]. Mathematical programming approaches for continuous design problems cannot be directly applied to a problem with discrete variables because the gradients cannot be analytically obtained. In order to apply mathematical programming to discrete design problems, the variables need to be relaxed to continuous variables during optimization to compute the gradients [4–6].

Metaheuristics are often used to solve combinatorial optimization problems. Liu et al. [7] used genetic algorithm [8], and Balling [9] used simulated annealing [10] to minimize the weight of a steel frame by choosing the cross-sections from discrete standard sizes. Metaheuristic methods require less computational cost to obtain promising results; however, the solutions do not satisfy any theoretically defined optimality criteria, and the quality of the optimal solution strongly depends on the initial solution and hyper-parameters of the algorithm.

* Corresponding author.

E-mail addresses: hayashi.kazuki@archi.kyoto-u.ac.jp (K. Hayashi), ohsaki@archi.kyoto-u.ac.jp (M. Ohsaki).

<https://doi.org/10.1016/j.aei.2021.101512>

Received 1 September 2021; Received in revised form 12 December 2021; Accepted 21 December 2021

Available online 14 January 2022

1474-0346/© 2022 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Apart from mathematical programming and metaheuristic approaches, there is an increasing number of studies on applications of machine learning (ML) to structural engineering problems that require professional knowledge and skills. ML is a term first used by Arthur Samuel [11], and can be regarded as a modeling process of computer algorithms to learn tasks without explicit programming by humans. Because of this automatic self-improvement property, ML is regarded as a subset of artificial intelligence (AI).

Reinforcement learning (RL) is an area of ML, in which rewards for giving feedback to an action taker called an agent are defined, and the agent is trained to take appropriate actions through observed rewards. The training is performed in an environment where a next state and a reward are sequentially observed depending on the current state and an action, which is called a Markov decision process (MDP) [12]. Since RL does not require input–output pairs as training data, RL can be applied to structural design problems where it is difficult to determine the desired optimal solutions in advance. Although previous researches showed that the trained agents overwhelmed the skilled players in Go [13] and some arcade games [14], there are only a few examples of using RL in the field of structural engineering.

The authors have developed a feature extraction method for truss members and implemented RL for topology optimization of trusses [15]. Here, a new feature extraction method categorized as graph embedding (GE) is developed to capture the member features of trusses. This method can handle both node and edge inputs simultaneously, and can extract the member features of trusses while preserving topological information of the structure. Owing to this GE property, the authors successfully obtained a reasonable sparse topology from a densely connected structure. However, this problem deals with binary variables where the variables are either 0 or 1, and its effectiveness for discrete problems with more possible variables has not yet been verified. In addition, the agent in Ref. [15] was trained using only a 4×4 -grid truss due to limitation of the GE operations, which is explained in Section 2.3.

In this paper, we re-formulate the graph-based RL method for discrete optimization of planar steel frames as an extension of the method in Hayashi and Ohsaki [15]. Two types of agents are trained according to their role in the design change; one agent sequentially reduces the size of frame members; the other agent sequentially increases the size of frame members.

It is important to emphasize that this study is positioned as a research to extract the knowledge about the design of the cross-sections of frame members that has been shared among skilled structural engineers. Unlike mathematical programming and metaheuristics, RL methods can learn the causal relationship between variables of a design problem and the objectives and constraints, and can acquire the specialized knowledge in making design changes to cross-sections. By using a model that has learned the causal relationship, it is possible to reach a reasonable solution with a small number of variable changes, which is expected to improve the efficiency of optimization. The knowledge extraction is achieved in a numerical manner through GE and RL where the agent extracts latent features related to the design task and learns to efficiently design the member cross-sections using the features.

For regular shaped 3D frames where torsional deformation is not dominant, planar models with depth direction omitted are frequently used in real-world applications. Therefore, if the proposed method is shown to be effective for planar models, it can be expected to be similarly effective for 3D models.

The main contributions of this study to the formalization of engineering knowledge and practice are summarized as follows. Although the structural design of plane frames is selected as a target of application, the proposed concept and method can be extended to engineering design problems of structures that are modeled as graphs consisting of nodes and edges.

- In order to extract latent features that are shared among data with various connectivity, GE is adopted to capture the connectivity of data expressed as nodes and edges.
- The GE method is formulated to handle both node and edge attributes. As a result, more known information can be processed through nodes and edges, and complex features for knowledge-intensive tasks can be extracted.
- In order to obtain generalization performance for a given decision-making process, agents need to be trained with a variety of data. Therefore, block matrix operation is introduced that can simultaneously handle data with different connectivity.
- The agent's performance is versatile, as the trained agent can handle various data without re-training.
- By training agents using an RL method, any step of the trained agent's decision-making process can be queried, which enables the collaboration between engineers and the agents during knowledge-intensive decision-making processes.

The remainder of this paper is organized as follows. Section 2 is a literature review on ML for structural engineering problems and GE methods. In Section 3, the cross-section optimization problem of planar steel frames under elastic and plastic design constraints is formulated. In Section 4, the optimization problem formulated in the previous section is converted to an RL task so that the proposed method can be applied. In Section 5, the proposed method combining GE and RL is explained in detail. In Section 6, the agent is trained using the proposed method, and the performance and efficiency are comparatively evaluated with particle swarm optimization (PSO). Section 7 is a concluding remark that summarizes the above sections and findings in this study.

2. Literature review

2.1. Machine learning for building engineering problems

One field of ML is supervised learning, in which sets of input–output pairs are given as training data so that the output can be predicted from the input [16]. ML models such as neural networks (NNs) are capable of approximating a highly nonlinear function, and numerous attempts have been made to predict the complex response and performance of structures using supervised learning to reduce the computational cost for structural analysis. Examples of the prediction target in civil engineering are diverse including the shear strength of reinforced concrete beams [17], the compression strength of a concrete material [18], and the ground vibration induced by blasting [19]. Supervised learning is also utilized for data classification, such as the failure mode classification of reinforced concrete columns [20] and the classification of construction documents [21].

By contrast, unsupervised learning solely requires inputs and infers the structure in the inputs. Unsupervised learning is used for clustering, which aims at finding hidden grouping in the data, and dimensionality reduction of inputs. Chow et al. [22] and Pathirage et al. [23] utilized an autoencoder, a method for reconstructing the original input, for detection of concrete defects and structural damage identification, respectively.

Reinforcement learning (RL) seeks to take actions in an environment so as to maximize the cumulative reward. Similarly to unsupervised learning, RL does not require desired outputs as training data; instead, it requires a reward function that evaluates the current state. For instance, Gu et al. [24] utilized RL to train robots to learn a door opening task, in which the robot successfully learned the accurate sequence of the manipulation, which is very difficult to provide explicitly.

There are a few examples of using RL in the field of civil engineering. Nakamura and Suzuki [25] trained an RL agent to choose initial or tangential stiffness for each step of iterative calculation of nonlinear structural analysis for the purpose of accelerating convergence. Chiba et al. [26] learned the control method of an actuator

attached to the steel frame by RL and confirmed response reduction against earthquakes. The authors have also implemented an RL method for cross-section optimization of planar frames using a multi-layer perceptron [27], which is a simplified class of NNs.

2.2. Graph embedding

One of the major difficulties in ML methods is how to capture the structure of the input data. For problems in which regularly aligned data become an input, such as detecting errors in images, there is an established ML model called convolutional neural networks (CNNs) [28], which can extract features that take into account the positional relationships between pixels. On the other hand, it is difficult to apply CNNs to structures such as trusses and frames, where linear members have complex connectivity to transmit forces. Therefore, in order to capture skilled structural engineers' knowledge on the frame cross-section design through ML, it is necessary to use an ML model that can handle input data with arbitrary connectivity.

Feature extraction methods for graphs called graph embedding (GE) have recently been proposed to deal with non-Euclidean data [29]. Here, a graph is a general term for data consisting of nodes and edges connecting them. GE-based methods have shown remarkable results for prediction of chemical properties of organic compounds [30,31] and classification of users with multiple labels in social networking services (SNS) [32]. GE is beginning to be used in the field of building structures. Ross and Hambleton [33] applied a neural message-passing model (NNConv) [31], which is one of the GE methods, to predict the deflection of 3D lattices; however, they used periodic geometries whose unit cell is composed of only 2 to 16 nodes.

There are a number of methods that combine RL and GE [34–38]; however, they are limited to the problem of selecting nodes.

2.3. Edge embedding

In the initial design phase of building structures, the cross-sections of members are important rather than the design of joints, assuming that the joints hold sufficient strength over the members. Hence, for the problem of designing member sections, it is preferable to embed the edge features rather than node features.

Methods for vector representations of the edges in a graph are called edge embedding. Compared with node embedding methods, there are few studies on edge embedding methods, which are explained below. Bandyopadhyay et al. [39] obtained edge vector representation utilizing a line graph, in which nodes represent the edge of the original graph and two nodes are connected by an edge only if their corresponding edges in the original graph share a common node. Wang et al. [40] developed an edge embedding method that can directly extract edge features. However, these methods are designed for standard edge problems such as edge classification and link prediction, and node and edge attributes cannot be handled as input.

Node and edge attributes are very important in the design of skeletal structures, because various design conditions and structural responses are assigned to nodes and edges. For example, support and load conditions are usually assigned to nodes and axial forces are observed at edges. Since the nodes and edges are assumed to exist in the Euclidean space, geometrical information such as nodal positions and edge lengths are also important factors to capture the structural property. For these reasons, methods that can handle both node and edge attributes are more preferable in this research.

The authors have developed an edge embedding method for truss members and implemented RL for topology optimization of trusses [15]. This method is distinct in that it can handle both node and edge attributes as inputs simultaneously. The RL agent captured the structural property of each member and successfully obtained a reasonable sparse topology from a densely connected structure. Still, the

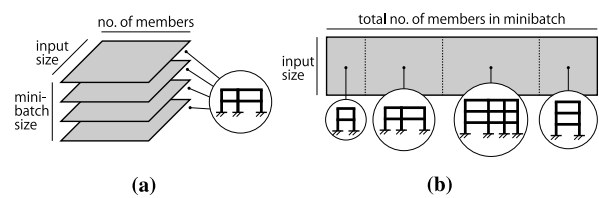


Fig. 1. Difference of input data structure. (a) Ref. [15]. (b) Proposed method in this study.

problem dealt in Ref. [15] was a binary optimization problem where each variable takes a value of either 0 or 1.

In addition, the agent in Ref. [15] was trained using only a 4×4 -grid truss due to limitation of GE operations. More specifically, a three-dimensional array was utilized to assemble the observed transitions for mini-batch training, as shown in Fig. 1(a). The sizes of each dimension are fixed as (1st) the number of observed transitions in the mini-batch, (2nd) the number of members in each data, and (3rd) the size of input feature, respectively. Worse still, only one fixed connectivity matrix could be used when applying GE with this array as an input. For these reasons, all the data comprising a mini-batch had to have the same number of members and connectivity. In order for the agent to acquire more generalized performance to a specific task, it is important to improve the method so as to train the agents over frames of various connectivity simultaneously.

On the other hand, the input data structure is modified as Fig. 1(b) in this study; the input becomes two-dimensional array in which each member input can be concatenated over different frames. The training method using this input is explained in detail in Section 5.4.

3. Cross-section optimization problem of steel frames

In this section, the optimization problem considered in this study is formulated. The total structural volume V of a planar steel frame is minimized under constraints on the stresses, displacements, column-to-beam overstrength factors (COFs), and shear load bearing capacity of the entire structure. A solution that satisfies all the above constraints is feasible, otherwise it is infeasible. The stress, displacement and COF are calculated through elastic structural analysis for 3 static load cases: one applying only vertical load, and the others applying vertical and horizontal loads. The shear load capacity is calculated through inelastic structural analysis for 2 static load cases: both applying vertical and amplified horizontal loads.

3.1. Elastic design

3.1.1. Load condition

The elastic design is to ensure that the stresses and displacements in the members and nodes do not exceed the upper bounds when subjected to long-term and short-term static loads. The long-term loads refer to the self-weight of structural members, dead load and live load, and short-term loads refer to seismic and wind loads in addition to the long-term loads. In this study, wind loads are not considered for simplicity. In order to simulate long-term and short-term loads, there are three load cases to be considered; one is a long-term load case in which only vertical loads are applied, and the others are short-term load cases in which the horizontal seismic loads are further applied in the right and left directions, respectively. The self-weight of structural members is calculated by the product of the steel weight density 77 kN/m^3 and the total structural volume. The dead and live loads are calculated by 1.0 kN/m^2 and 2.4 kN/m^2 per unit floor area, respectively.

The procedure of calculating the short-term seismic loads of a frame is explained below. Let W_T and ψ_i denote the total weight of the

structure and the ratio of the weight above i th story to W_T , respectively. In computing W_T and ψ_i , the reduced live load of 1.3 kN/m^2 instead of 2.4 kN/m^2 is used because seismic forces act in a balanced manner in the total floor area. Let T_f denote the fundamental natural period of the steel frame, which is approximated by the frame height multiplied by 0.03. The seismic shear force acting in i th story is estimated as

$$Q_i = C_B A_i^s W_T \psi_i \quad (1a)$$

$$A_i^s = 1 + \left(\frac{1}{\psi_i} - \psi_i \right) \frac{2T_f}{1 + 3T_f} \quad (1b)$$

where C_B is the base shear coefficient and A_i^s is distribution of the seismic shear force in the height direction of the frame. When implementing structural analyses, the loads are converted into equivalent nodal loads in proportion to the covering area of each node.

3.1.2. Stress and displacement constraints

The structure is designed so that it can continue to be used without critical damage for the long-term load case and the short-term load cases when $C_B = 0.2$. The nodal displacements and stresses are computed by the standard stiffness method with linear analysis. When solving the stiffness equations, the rigid floor assumption is incorporated by constraining all the nodes in the same floor to have the same displacement in the horizontal direction.

Using the yield strength σ_y , long-term allowable stresses of steel members against tension and bending forces are both $\sigma_y/1.5$, respectively. The long-term allowable compression stress f_c is determined using the effective slenderness ratio λ and the critical slenderness ratio Λ as

$$f_c = \begin{cases} \frac{1-0.4(\lambda/\Lambda)^2}{3/2+(2/3)(\lambda/\Lambda)^2} \sigma_y & (\text{if } \lambda \leq \Lambda) \\ \frac{18}{65(\lambda/\Lambda)^2} \sigma_y & (\text{else}) \end{cases} \quad (2)$$

The allowable stresses against short-term loads are 1.5 times of those against long-term loads. Note that these are based on simplified forms of Japanese building standards. The stress ratio $\bar{\sigma}_i$ of member i is computed as the sum of the ratios of the axial stress and bending stress to their allowable values, and must not be larger than 1.0.

In order to ensure safety and serviceability of the building, the inter-story drift ratio should not exceed $1/200$, and the ratio of the center deflection to the beam length should not exceed $1/300$. To unify the notations of relative displacement constraints for columns and beams, the relative displacement ratio of member i computed as the ratio of inter-story drift for a column or the center deflection for a beam to its upper bound is collectively expressed as \bar{d}_i .

3.2. Plastic design

3.2.1. Strong column–weak beam constraint

To avoid the local collapse and ensure the ductility of the frame, it is important to design the frame in accordance with a “strong column–weak beam” principle. If a frame is designed so that plastic hinges are formed at beams rather than at columns, the structure is expected to form a whole collapse mode where hinges at the beams are dominant as shown in Fig. 2(a). If the plastic moment capacity of the columns is smaller than that of the beams, there is a higher possibility of layer collapse as shown in Fig. 2(b). The whole collapse mode is more desirable because it has greater capacity to dissipate seismic energy.

The COF is a strong column–weak beam criterion of each joint to avoid the layer collapse, and calculated for joint j as

$$\beta_j = \frac{\sum M_{pc}}{\sum M_{pb}} \quad (3)$$

where $\sum M_{pc}$ and $\sum M_{pb}$ are the sums of the full plastic moments of the columns and beams connected to the joint, respectively. According to the Japanese building standards law, M_{pb} in Eq. (3) is given as

$$M_{pb} = F Z_{pb} \quad (4)$$

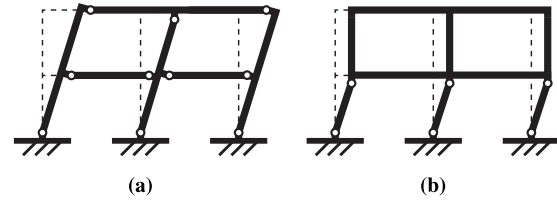


Fig. 2. Typical collapse modes. (a) whole collapse. (b) layer collapse.

where Z_{pb} is the plastic section modulus of a beam element and F is the design strength, which is same as the yield strength σ_y for steel materials. Although M_{pc} for a column is given in a similar way using the design strength F and the column’s plastic section modulus Z_{pc} , the value is adjusted using the axial force ratio η_c , the ratio of the column’s compressive axial stress to the yield stress, as

$$M_{pc} = \begin{cases} \left(1 - \frac{4\eta_c^2}{3} \right) F Z_{pc} & (\eta_c \leq 0.5) \\ \frac{4(1-\eta_c)}{3} F Z_{pc} & (\eta_c > 0.5) \end{cases} \quad (5)$$

Since three static load cases are considered here, three different axial force ratios are obtained, and accordingly, three different COFs are obtained for each node. Frames are designed to satisfy $\beta \geq 1.0$ except nodes on the base and the roof for all the load cases in this study. Note that hinges are allowed to be formed at the upper ends of the top story columns because it is too difficult for only one column to outperform two beams in full plastic moment capacity. Similarly, hinges are also allowed to be formed at the bottom ends of the first-story column, because the bottom nodes are considered to be fixed, and the base beams are not considered.

3.2.2. Horizontal load bearing capacity constraint

Constraints are further imposed to prevent the building from collapsing by an extremely strong earthquake. In this phase, since the design assumes that some of the members will yield, a bi-linear steel material is assumed; the stress–strain relationship is linear with the tangent of elastic modulus E until reaching the yield strength σ_y , and the post-yield tangent is reduced to $0.01E$.

The required shear load bearing capacity $Q_{un,i}$ of i th story against the extremely strong earthquake is obtained by the following equation:

$$Q_{un,i} = D_s F_{es} Q_{ud,i} \quad (6)$$

where D_s is the discount factor depending on the energy dissipation performance of the structure, F_{es} is the surcharge factor depending on the structural shape, and $Q_{ud,i}$ is a set of shear forces at each layer induced by the earthquake computed by Eq. (1a) with $C_B = 1.0$. The value of D_s is determined by the collapse mode and horizontal load bearing ratio of braces and walls. Since the whole collapse mode is expected owing to the COF constraints and the effects of braces and walls are not considered in this study, D_s is set to be 0.3. F_{es} is 1.0 because only frames with regular shape are considered here.

The shear force capacity of i th story $Q_{u,i}$ must be larger than $Q_{un,i}$. In other words, given that $C_B = 0.2$ in the elastic design and $C_B = 1.0$, $D_s = 0.3$ and $F_{es} = 1.0$ in this plastic design, the frame cross-sections are designed not to reach an inter-story drift ratio of $1/100$ when subject to the short-term load whose horizontal components are multiplied by $1.0 \times 0.3 \times 1.0/0.2 = 1.5$.

3.3. Optimization problem

Consider a discrete cross-section optimization problem of a frame with n_m members and n_{st} stories to minimize the total structural volume under the constraints explained above. The column has a square hollow section and the beam section has a wide-flange I section, as shown

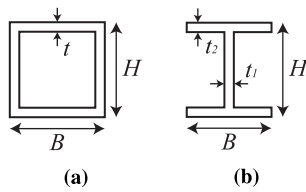


Fig. 3. (a) Column cross-section. (b) Beam cross-section.

Table 1
Dimension list of column sections.

J_i	$H \times B \times t$ [mm]	A_i [cm ²]	I_i [cm ⁴]	Z_i [cm ³]	$Z_{p,j}$ [cm ³]
200	200 × 200 × 12	85.3	4860	486	588
250	250 × 250 × 12	109.3	10 100	805	959
300	300 × 300 × 16	173.0	22 600	1510	1810
350	350 × 350 × 19	239.2	42 400	2420	2910
400	400 × 400 × 22	307.7	69 500	3480	4220
450	450 × 450 × 22	351.7	103 000	4560	5490
500	500 × 500 × 25	442.8	159 000	6360	7660
550	550 × 550 × 25	492.8	217 000	7900	9460
600	600 × 600 × 25	542.8	288 000	9620	11 400
650	650 × 650 × 28	656.3	407 000	12 500	14 900
700	700 × 700 × 28	712.3	518 000	14 800	17 600
750	750 × 750 × 32	866.3	717 000	19 100	22 800
800	800 × 800 × 32	930.3	884 000	22 100	26 200
850	850 × 850 × 32	994.3	1 070 000	25 300	29 900
900	900 × 900 × 36	1177.0	1 420 000	31 500	37 300
950	950 × 950 × 36	1249.0	1 680 000	35 500	42 000
1000	1000 × 1000 × 36	1321.0	1 990 000	39 700	46 900

in Fig. 3. A set of cross-section sizes $\mathbf{J} = \{J_1 \dots J_{n_m}\}$ is assigned from Table 1 for the columns and Table 2 for the beams.

By assembling the elastic and plastic design constraints described above, the optimization problem is formulated as

$$\text{minimize } V(\mathbf{J}) \quad (7a)$$

$$\text{subject to } \begin{cases} J_i \in \{200, 250, \dots, 1000\} & (i = 1, \dots, n_m) \\ \tilde{\sigma}_i \leq 1.0 & (i = 1, \dots, n_m) \\ \tilde{d}_i \leq 1.0 & (i = 1, \dots, n_m) \\ \beta_j \geq 1.0 & (j \in \Omega_\beta) \\ Q_{u,k} \geq Q_{un,k} & (k = 1, \dots, n_{st}) \end{cases} \quad (7b)$$

where Ω_β is a set of indices of nodes except on the base and the roof.

4. Conversion to a reinforcement learning task

The optimization problem (7) is transformed into an RL task that is trainable by the proposed method. Here, two agents are separately trained for increasing and reducing the member cross-sections, respectively. We consider a decision-making process in a discrete time step, where reward r and next state s' are observed when action a is taken in state s at each step. The goal of RL is to acquire a policy $\pi(s) = a$ that maximizes the expected future rewards. As a prerequisite, RL tasks are generally modeled as an MDP, and in the following, state s , action a and reward r , which are the elements of the MDP, are defined.

4.1. State s

In general, the state of a frame structure can be expressed using numerical data of nodes $\hat{\mathbf{v}} = [\mathbf{v}_1, \dots, \mathbf{v}_{n_n}] \in \mathbb{R}^{n_{fn} \times n_n}$, those of members $\hat{\mathbf{w}} = [\mathbf{w}_1, \dots, \mathbf{w}_{n_m}] \in \mathbb{R}^{n_{fm} \times n_m}$, and the connectivity of the nodes and members, which is expressed as the connectivity (or incidence) matrix $\mathbf{C} \in \mathbb{R}^{n_m \times n_n}$, such that each element C_{ij} is provided as follows: $C_{ij} = -1$ when member i leaves node j , $C_{ij} = 1$ when member i enters node j , and $C_{ij} = 0$ otherwise. Therefore, the frame state s is equivalent to a

Table 2
Dimension list of beam sections.

J_i	$H \times B \times t_1 \times t_2$ [mm]	A_i [cm ²]	I_i [cm ⁴]	I_i^y [cm ⁴]	Z_i^z [cm ³]	$Z_{p,j}$ [cm ³]
200	194 × 150 × 6 × 9	38.11	2630	507	271	301
250	244 × 175 × 7 × 11	55.49	6040	984	495	550
300	294 × 200 × 8 × 12	71.05	11 100	1600	756	842
350	340 × 250 × 9 × 14	99.53	21 200	3650	1250	1380
400	400 × 200 × 9 × 19	110.0	31 600	2540	1580	1770
450	450 × 200 × 9 × 22	126.0	45 900	2940	2040	2750
500	500 × 250 × 9 × 22	152.5	70 700	5730	2830	3130
550	550 × 250 × 9 × 22	157.0	87 300	5730	3180	3520
600	600 × 250 × 12 × 25	192.5	121 000	6520	4040	4540
650	650 × 250 × 12 × 25	198.5	145 000	6520	4460	5030
700	700 × 250 × 12 × 25	205.8	173 000	6520	4940	5580
750	750 × 300 × 14 × 28	267.9	261 000	12 600	6970	7850
800	800 × 300 × 14 × 28	274.9	302 000	12 600	7560	8520
850	850 × 300 × 16 × 28	297.8	355 000	12 600	8350	9540
900	900 × 300 × 16 × 28	305.8	404 000	12 600	8990	10 300
950	950 × 300 × 16 × 28	313.8	458 000	12 600	9640	11 100
1000	1000 × 300 × 16 × 28	321.8	515 000	12 600	10 300	11 900

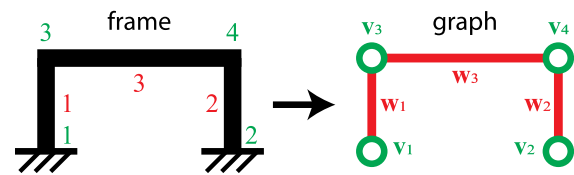


Fig. 4. Conversion of frame properties into a graph.

tuple of $\{\hat{\mathbf{v}}, \hat{\mathbf{w}}, \mathbf{C}\}$ and thus can be expressed in the form of graph as illustrated in Fig. 4.

However, the tuple is difficult to handle because the three inputs are matrices of different sizes and properties. For this reason, the member feature vectors $\hat{\boldsymbol{\mu}} = [\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{n_m}]$ of the same size are extracted from the tuple using GE and trainable parameters θ , and are regarded as an approximation of state s written as

$$s = \{\hat{\mathbf{v}}, \hat{\mathbf{w}}, \mathbf{C}\} \approx \hat{\boldsymbol{\mu}}(\hat{\mathbf{v}}, \hat{\mathbf{w}}, \mathbf{C}; \theta) \quad (8)$$

The detail of GE operations and trainable parameters θ will be explained in Section 5.1, and only node inputs $\hat{\mathbf{v}}$, and member inputs $\hat{\mathbf{w}}$ are focused here.

It is desirable for a state to include numerical information of frame as much as possible, such as the geometry, the property of the nodes and members, and the loading and support conditions, which can be the elements of inputs $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$. The dimension of node inputs \mathbf{v}_k ($k \in \{1, \dots, n_n\}$) can take an arbitrary positive integer depending on the problem definition, and the same is true for the dimension of member inputs \mathbf{w}_i ($i \in \{1, \dots, n_m\}$). Note that the inputs $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ should prevent taking excessive values in order to enhance the RL agent's performance. This is because the larger values will have a higher contribution to the output error, and the error reduction algorithm for the RL agent will neglect the information from the small-valued variables [41]. Therefore, the desirable situation is when all inputs are in the same order of magnitude [42].

In this study, $n_{fn} = 4$ node input items are defined, and the detail of each item is explained in Table 3. In order to consider the support positions of the frame, an input that is 1 at the support and 0 otherwise is assigned to the first component. In addition, the second and third components respectively identifies the nodes at the top where the COF constraints are not necessary, and the nodes at the side ends where the COF values differ greatly due to the difference in the number of beams to be connected. The COF is considered in the fourth component of the node input as the reciprocal value of COF, and the value is scaled using the tanh function not to exceed 1.0.

Similarly, $n_{fm} = 13$ member input items are defined, and the items are described in Table 4. \bar{L} is the upper bound of member length during

Table 3
Detail of node input v_k .

Index	Input
1	1 if fixed, 0 otherwise
2	1 if at the top, 0 otherwise
3	1 if at the side ends, 0 otherwise
4	$\tanh(1.0/\beta_k)$

training, and A'_i , I'^z , I'^y and Z'^z are the properties of the updated cross-section if the design change is applied to member i . \bar{A} , \bar{I}^z , \bar{I}^y and \bar{Z}^z are the upper-bounds (i.e., the values when $J_i = 1000$). Binary inputs representing the member type (column or beam) are assigned separately for indices 1 and 2. The other member inputs consist of the member length, assigned properties of the current cross-section, properties of the updated cross-section if the design change is applied to the member, stress ratio, and displacement ratio. Note that the inputs are scaled to take a value in the range 0 to 1.

4.2. Action a

An action in this study is defined as the smallest design changes applied to a member of the frame. If one action is assigned to each member, there are at most n_m actions that an agent can take at each step. In practice, in order to reduce the action space to be explored, the action is selected from Ω_a which represents a set of possible actions that excludes clearly inappropriate actions such as an action of and increasing/reducing a member size exceeding the upper/lower bounds.

We train the following two agents separately depending on the task of reducing and increasing member sizes: Agent(-) is responsible for changing member size J_i one step smaller by taking an action; Agent(+) is responsible for changing member size J_i one step larger by taking an action.

4.3. Restriction of actions

Furthermore, we introduce the following restriction of actions to exclude obviously inappropriate state in which lower-level columns are slenderer. If the lower columns on the same axis become thinner than the upper column as a result of the action, the columns' cross-sections are modified as follows; for the task of reducing sizes, the cross-section of the upper columns is corrected so that it becomes equal to the lower column; for the task of increasing sizes, the cross-section of the lower columns is corrected so that it becomes equal to the upper column. This restriction scheme is beneficial in boosting learning efficiency because the variable space to be explored can be reduced.

4.4. Reward r

The reward is obtained after each design change of the member sizes. Let $\bar{\sigma}'_i$, \bar{d}'_i and β'_k denote the stress and displacement ratios of member i and the COF at node k at the next state, respectively.

When training an agent to reduce member sizes, the process starts with a redundant design that satisfies all the constraints. If an unnecessarily large cross-section is reduced, a larger positive reward is assigned; on the other hand, if the cross-section of a member that is likely to violate the constraints is reduced, a smaller positive reward is assigned. A negative reward (i.e., penalty) is assigned for the termination state in which the structural design violates the constraints. Following this scheme, the reward function for reducing member sizes is defined as

$$r^- = \begin{cases} \frac{\tau^- \sqrt{|\Delta V|}}{(\max_i \bar{\sigma}'_i - \min_i \bar{\sigma}'_i)} & \text{(if feasible)} \\ \ln((1.0 - r_1^-)(1.0 - r_2^-)) & \text{(else)} \\ -1.0 & \text{(else)} \end{cases} \quad (9a)$$

Table 4
Detail of member input w_i .

Index	Input
1	1 if column, 0 otherwise
2	1 if beam, 0 otherwise
3	L_i/\bar{L}
4	A_i/\bar{A}
5	I_i^z/\bar{I}^z
6	I_i^y/\bar{I}^y
7	Z_i^z/\bar{Z}^z
8	A'_i/\bar{A}
9	I'^z/\bar{I}^z
10	I'^y/\bar{I}^y
11	Z'^z/\bar{Z}^z
12	$\tanh(\bar{\sigma}_i)$
13	$\tanh(\bar{d}_i)$

$$r_1^- = \min\left\{\frac{\max_i \bar{\sigma}'_i}{\max_i \bar{\sigma}_i}, 0.99\right\} \quad (9b)$$

$$r_2^- = \min\left\{\frac{\max_i \bar{d}'_i}{\max_i \bar{d}_i}, 0.99\right\} \quad (9c)$$

where τ^- is a scaling factor, which is 0.1 in this study.

When training an agent to increase member sizes, the process starts with an infeasible and slender member design that violates the constraints. If a positive reward is given to increase the member sizes, the agent may learn an inefficient strategy in an attempt to take steps as long as possible. Since it is preferable to reach a feasible design that satisfies all constraints in fewer steps, negative rewards are given until satisfying the constraints, and a positive reward is assigned only to the terminal state that satisfies the constraint. The reward function for increasing member sizes is defined as

$$r^+ = \begin{cases} \tau^+ \sqrt{|\Delta V|} (\max_i \bar{\sigma}'_i - \min_i \bar{\sigma}'_i) \cdot \ln((1.0 - r_1^+)(1.0 - r_2^+)(1.0 - r_3^+)) & \text{(if infeasible)} \\ 1.0 & \text{(else)} \end{cases} \quad (10a)$$

$$r_1^+ = \begin{cases} \min\left\{\frac{\max_i \bar{\sigma}'_i}{\max_i \bar{\sigma}_i}, 0.99\right\} & \text{(if } \max_i \bar{\sigma}'_i > 1.0\text{)} \\ 0.0 & \text{(else)} \end{cases} \quad (10b)$$

$$r_2^+ = \begin{cases} \min\left\{\frac{\min_k \beta'_k}{\min_k \beta_k}, 0.99\right\} & \text{(if } \min_k \beta'_k < 1.0\text{)} \\ 0.0 & \text{(else)} \end{cases} \quad (10c)$$

$$r_3^+ = \begin{cases} \min\left\{\frac{\max_i \bar{d}'_i}{\max_i \bar{d}_i}, 0.99\right\} & \text{(if } \max_i \bar{d}'_i > 1.0\text{)} \\ 0.0 & \text{(else)} \end{cases} \quad (10d)$$

where τ^+ is a scaling factor, which is also 0.1 in this study. Note that the shear force capacity constraint is not considered in the reward definition, but is included in the terminal state criterion.

5. Graph-based reinforcement learning

5.1. Graph embedding to extract member features

A graph is a data structure composed of nodes and edges. Here, the frame's joints are regarded as nodes, and members are regarded as edges. The feature of each member is represented as a vector using GE. First, we define three matrices C_A , C_1 and C_2 to be used for GE from the connectivity matrix C . The non-oriented connectivity matrix $C_A \in \mathbb{R}^{n_m \times n_n}$ is obtained by taking the absolute value for each element of C . $C_1 = (C_A - C)/2$ and $C_2 = (C_A + C)/2$ are further obtained to identify the nodes that each member leaves and those that each member enters.

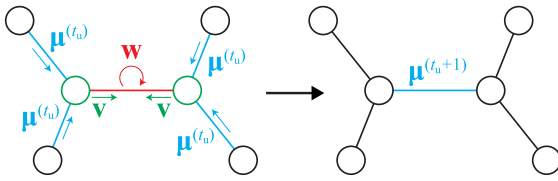


Fig. 5. Illustration of Eq. (11) (adaptation from Ref. [15]). It aggregates numerical data of a member, its end nodes and the other members connecting them.

Let n_f denote the size of the feature vector of each member. n_f should be determined through a careful adjustment with trial-and-error for better performance, because a size that is too small leads to inaccuracy in expressing the features, and a size that is too large requires redundant computation time in training and application of the agent after the training. The aim of the proposed GE operation is to extract the feature vector of each member $\mu_i \in \mathbb{R}^{n_f}$ ($i = 1, \dots, n_m$). In the following, the features of all members $\{\mu_1^{(t_u)}, \dots, \mu_{n_m}^{(t_u)}\}$ are combined to $\hat{\mu}^{(t_u)}$, in which t_u is the iteration counter of GE operation and $\hat{\mu}^{(0)} = \mathbf{0}$.

The trainable parameters $\theta_1 \in \mathbb{R}^{n_f \times n_{fm}}$, $\theta_2 \in \mathbb{R}^{n_f \times n_{fn}}$, and $\theta_3 \in \mathbb{R}^{n_f \times n_{fr}}$ are introduced for weighting the inputs in accordance with their importance to extract features integrating the structural property of neighbor nodes and members. Using $\{\theta_1, \theta_2, \theta_3\}$ and the connectivity matrices, $\hat{\mu}^{(t_u)}$ is updated as follows:

$$\hat{\mu}^{(t_u+1)} = \hat{\mathbf{h}}_1 + \hat{\mathbf{h}}_2 + \hat{\mathbf{h}}_3^{(t_u)} \quad (11a)$$

$$\hat{\mathbf{h}}_1 = \varphi(\theta_1 \hat{\mathbf{w}}) \quad (11b)$$

$$\hat{\mathbf{h}}_2 = \varphi(\theta_2 \hat{\mathbf{v}}) \mathbf{C}_A^T \quad (11c)$$

$$\hat{\mathbf{h}}_3^{(t_u)} = \sum_{k=1}^2 \varphi\left(\theta_3 \left(\mathbf{C}_k \mathbf{C}_A^T \hat{\mu}^{(t_u)T} - \hat{\mu}^{(t_u)T}\right)^T\right) \quad (11d)$$

where $\hat{\mathbf{h}}_1$, $\hat{\mathbf{h}}_2$ and $\hat{\mathbf{h}}_3^{(t_u)}$ are the elements that make up the feature, and φ is the Leaky rectified linear unit (Leaky ReLU) function, which is one of the activation functions, with 0.2 for the multiplier for negative inputs. To simplify the expression, the same notation φ is used irrespective of the size of the input vector. The operation in Eq. (11) is illustrated in Fig. 5. For the sake of clarity, this figure focuses on the operation with respect to only one member, but in the actual calculation of Eq. (11), the feature values of all members are updated simultaneously. The numerical data $\hat{\mathbf{h}}_1$ for a member, $\hat{\mathbf{h}}_2$ for its two end nodes and $\hat{\mathbf{h}}_3^{(t_u)}$ for members connecting to them are aggregated into the feature vector of each member through a single operation. Accordingly, $\hat{\mu}^{(t_u)} = \{\mu_1^{(t_u)}, \dots, \mu_{n_m}^{(t_u)}\}$ is the set of feature vectors incorporating the connectivity of frame.

The operation of Eq. (11) should be iterated more than once in order to capture the features of distant members that are not directly connected to the specific member. The maximum number of iterations T_u should be carefully selected because it determines the training difficulty and how distant nodes and members are considered. If T_u is set closer to 1, the computational cost is cheaper and the risk of divergence of trainable parameters $\{\theta_1, \theta_2, \theta_3\}$ during the training is lower; however, the extracted features cannot capture the property of distant nodes and members. In contrast, if a large positive integer is assigned to T_u , the extracted features are capable of considering the properties of distant nodes and members, but the computational cost and the risk of divergence of the trainable parameters become higher. Therefore, T_u is set to be 3 in this study. It should be noted that the embedded feature vectors $\mu_i^{(T_u)}$ ($i = 1, \dots, n_m$) have the same size n_f regardless of the connectivity. Owing to this property, all members can be evaluated based on the same measure.

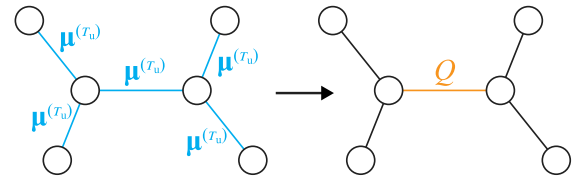


Fig. 6. Illustration of Eq. (12). It converts the extracted member features into the scalar action value.

5.2. Action values computed from the member features

Action value $Q(s, a)$ is the expected return for selected action a in state s , and an important notion that represents the relationship between state, action and overall reward. The agent estimates that the larger the action value, the larger the return to be obtained from the current state. Thus, the member with the largest action value is prioritized to change its size.

Using $\hat{\mu}^{(T_u)} \in \mathbb{R}^{n_f \times n_m}$, the action values $\hat{Q}(\hat{\mu}^{(T_u)}) \in \mathbb{R}^{n_m}$ of applying the design change to a member in the current state is approximated using trainable parameters $\theta_4 \in \mathbb{R}^{2n_f}$ as follows:

$$\hat{Q}(\hat{\mu}^{(T_u)}) = \theta_4^T \left(\begin{bmatrix} \hat{\mu}_\Sigma^{(T_u)} \\ \hat{\mu}^{(T_u)} \end{bmatrix} \right) \quad (12)$$

where $\hat{\mu}_\Sigma^{(T_u)} \in \mathbb{R}^{n_f \times n_m}$ is the matrix in which a column vector $\sum_{i=1}^{n_m} \mu_i^{(T_u)}$ is repeated in the row direction n_m times. The operation in Eq. (12) is illustrated in Fig. 6. In a similar manner as Figs., 6 illustrates the operation with respect to one member. Using Eq. (12), the vectors representing the member features are converted to a scalar representing the action value to choose the member. Although properties of neighborhood nodes and members are integrated, the feature $\hat{\mu}^{(T_u)}$ is the local information about a member, Therefore, the term $\hat{\mu}_\Sigma^{(T_u)}$ is further arranged to capture the global information about the entire structure. The term $\hat{\mu}_\Sigma^{(T_u)}$ also plays an important role in estimating appropriate action values for structures of various scales.

Since $\hat{\mu}$ is computed using $\{\theta_1, \theta_2, \theta_3\}$, the set of action values $\hat{Q}(\hat{\mu}^{(T_u)})$ depends on $\theta = \{\theta_1, \dots, \theta_4\}$.

5.3. Tuning problem of trainable parameters

The training method for tuning the parameters θ is described. The parameters θ are tuned using a method based on 1-step Q-learning, which is frequently implemented as an RL method.

Among a number of RL methods, Q-learning is a great fit for this problem, because the design variables are discrete and the change of each cross-section is described with either of two actions: increasing and reducing the size. More specifically, at each step, Q-learning methods determine an action in which the associated action value is expected to be the largest among possible actions, i.e. $a = \arg \max_a Q$. This implies that the required computational cost to achieve a certain level of performance is greatly dependent on the action space. By assigning one type of action to each member and using GE to calculate the action values of all members together, all the action values can be represented as a compact vector with a size of n_m , which is relatively easy to approximate using Q-learning.

If the action space is continuous or discrete but too large, Q-learning is not a reasonable option and other policy-based RL methods such as DDPG [43] should be used; however, adopting such a method requires a more complex architecture for learning.

According to the classical Q-learning method proposed by Watkins [44], the action value is updated using state s , chosen action a , observed next state s' and reward r as

$$Q(s, a) = Q(s, a) + \alpha \left(r(s') + \gamma \max_a Q(s', a) - Q(s, a) \right) \quad (13)$$

where $\gamma \in [0, 1]$ is a discount factor to adjust the importance of long-term rewards; the immediate reward is only considered and the

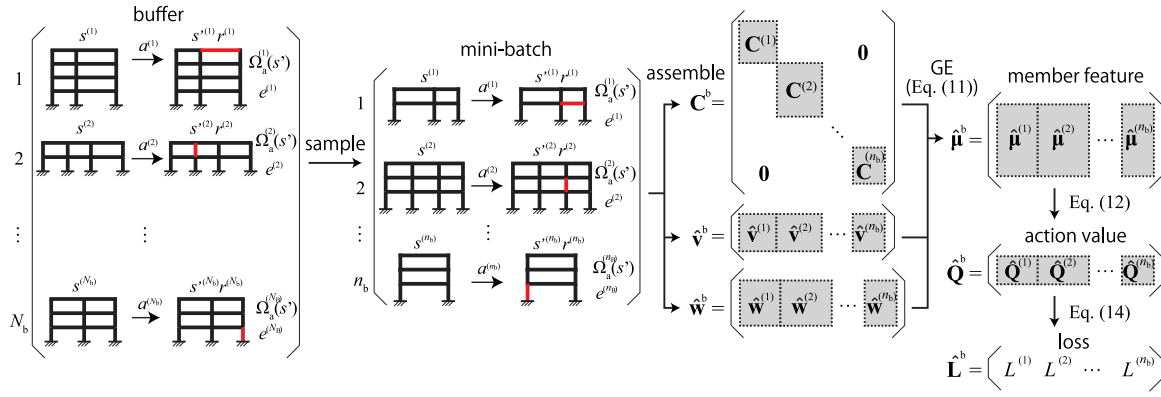


Fig. 7. Mini-batch operation to compute member features of multiple frames using block matrices.

following future rewards are disregarded when $\gamma = 0$, and the rewards at any future step are treated with the same importance as the immediate reward when $\gamma = 1$. In Eq. (13), the action value is updated so as to minimize the difference between the sum of the observed reward and estimated action value at the next state $r(s') + \gamma \max_a Q(s', a)$ and the estimated action value at the current state $Q(s, a)$. Following this scheme, the parameters are trained by minimizing the following loss function [14]:

$$L(\Theta) = \left(r(s') + e\gamma \max_{\tilde{a} \in \Omega_a(s')} Q(s', \tilde{a}; \Theta) - Q(s, a; \Theta) \right)^2 \quad (14)$$

where $\Omega_a(s')$ is a set of actions available at state s' , and e is a binary indicator that takes 0 when s' is the terminal state and 1 otherwise. In Eq. (14), the training can be stabilized by using previous parameters Θ obtained during the training for an estimation of action values at the next state s' [14]. $\tilde{\Theta}$ synchronizes with Θ once every $n_{sy} \in \mathbb{N}^+$ episodes. In the same manner as NNs, the inputs are repeatedly weighted by the trainable parameters and processed through activation functions. Therefore, an arbitrary back-propagation method can be used for solving Eq. (14). Among a number of back-propagation methods proposed so far, RMSProp [45] is adopted as an optimization algorithm for tuning Θ in this study.

5.4. Mini-batch training

Mini-batch training is a method to update trainable parameters using multiple samples. By introducing mini-batch training, the agent is able to avoid sample bias and the training efficiency is improved through parallel computation. Whereas Eq. (14) calculates the loss function for a single sample of transition $\{s, a, s', r, \Omega_a(s'), e\}$, the loss function for n_b samples are calculated together using block matrices as shown in Fig. 7, which is explained below.

As a prerequisite of the implementation, it is necessary to prepare a buffer to store the observed transitions. This buffer can store up to the latest N_b sets of transitions. Let n_b denote the number of transitions randomly sampled from the buffer to constitute one mini-batch. In Fig. 7, a superscript (i) indicates i th sample in the mini-batch, e.g., $\{s^{(i)}, a^{(i)}, s'^{(i)}, r^{(i)}, \Omega_a^{(i)}(s'), e^{(i)}\}$.

The connectivity of all frames in the mini-batch can be expressed with a single matrix C^b in which each frame's connectivity matrix is placed as a sub-matrix in the diagonal direction. Similarly, the inputs of all nodes and edges in the mini-batch can be expressed as \hat{v}^b and \hat{w}^b , respectively, by concatenating each input in the row direction. This way, all member features in the mini-batch $\hat{\mu}^b$ can be simultaneously computed using Eq. (11) without interaction between the samples.

In a similar manner as computation of member features $\hat{\mu}^b$, the action values of all members in the mini-batch can be computed using Eq. (12); note that in the calculation of $\hat{\mu}_\Sigma$, the summation is implemented for members in each sample, not for all members in the

mini-batch. Furthermore, the losses $L(\Theta)$ for n_b samples in the mini-batch can be simultaneously obtained using Eq. (14). The mean squared L2 norm of n_b losses is computed as

$$L_{\text{MSE}}(\Theta) = \frac{1}{n_b} \sum_{i=1}^{n_b} (L^{(i)})^2 \quad (15)$$

Instead of $L(\Theta)$ computed by Eq. (14) from single sample, $L_{\text{MSE}}(\Theta)$ in Eq. (15) is chosen as the function to be minimized by RMSProp in this study.

5.5. Training workflow

The whole training workflow is described in Fig. 8. First, trainable parameters are randomly initialized with a normal distribution with the mean of 0 and the standard deviation of 0.05. At the beginning of each episode, structural conditions are randomly initialized so that the agent learns various design conditions. At each step, the agent selects an action to apply the design change to a member using the ϵ -greedy policy, in which the member with the largest action value is selected with a high probability of $1 - \epsilon$, but a member is randomly chosen with a low probability of ϵ . Here, the ϵ -greedy policy instead of the greedy policy is employed to encourage the agent to explore a variety of member designs. After taking an action, the agent observes the next state s' , reward r , a set of possible actions at the next state $\Omega_a(s')$, and the terminal state indicator e . Between the steps, the set of trainable parameters Θ is updated using RMSProp for the sampled mini-batch data.

Once in 10 episodes of the training, agent's performance is tested for a prefixed condition. The cumulative reward, which is the sum of observed rewards until reaching the terminal state, is recorded using the greedy policy without randomness (i.e. the ϵ -greedy policy with $\epsilon = 0$) during the test. If a larger cumulative reward is obtained compared with the previous best score, Θ at that step is saved. The most recently saved Θ after the n_{ep} -episode training is regarded as the best parameters.

6. Numerical examples

In this section, the graph-based RL algorithm proposed in the previous section is demonstrated through numerical examples. The trained agent's performance and computational efficiency are evaluated comparing the results with those by PSO.

6.1. Training setting

Detail of the training implementation is explained in this section. The elastic modulus of the members is set to be 2.05×10^5 N/mm². The base nodes are rigidly supported against both translation and rotation.

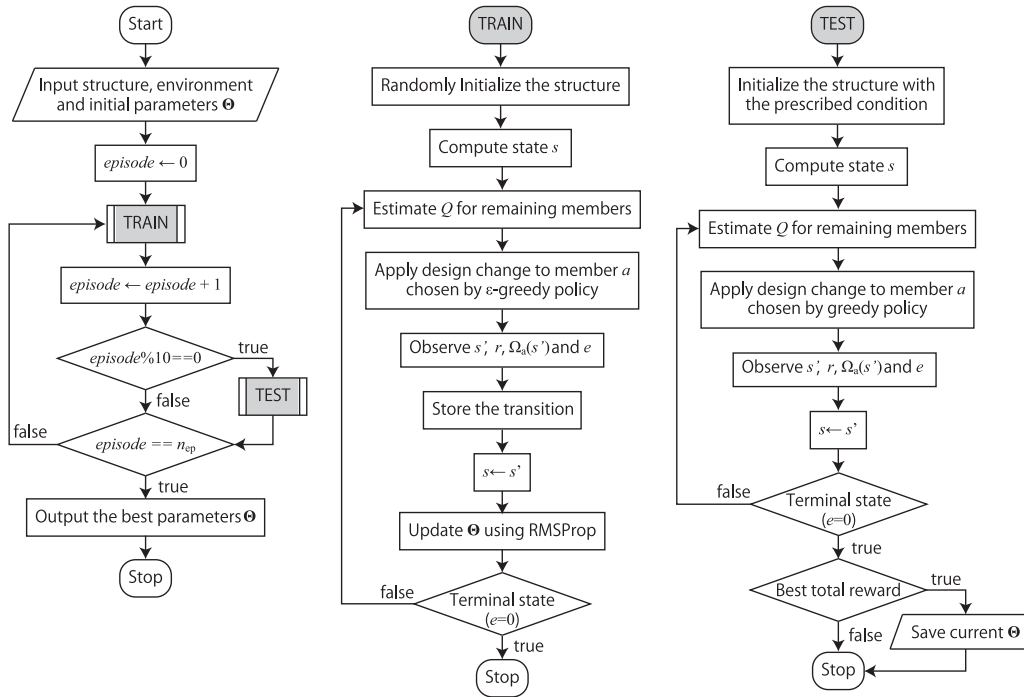


Fig. 8. Training workflow utilizing GE and RL.
Source: Adaptation from Ref. [15].

The agents are trained using various planar frames. At the beginning of each episode, the geometry and connectivity of the frame is initialized as follows. The number of spans is given randomly in the range 2 to 5 and the number of stories in the range 2 to 10. Each span is randomly initialized in the range of 5 to 15 m. The floor heights above the 2nd floor should be equal, and they are randomly initialized in the range 3.0 to 4.0 m. The first floor height is set 0.5 m higher than the other floor heights. When training the agent responsible for reducing member sizes, all initial cross-sections are set to the maximum of $J_i = 1000$ ($i = 1, \dots, n_m$); when training the agent responsible for increasing member sizes, all initial cross-sections are set to the minimum of $J_i = 200$.

Once the shape and cross-sections are determined, the loads are calculated according to the procedure explained in the subsection “Load condition”. Since the live loads and the seismic loads are given in proportion to the floor area, 0.75 times the average of the in-plane spans is assumed for the out-of-plane span for calculating covering area of the frame.

The cross-section is changed, and the long-term and short-term loads are also updated at each step until reaching the terminal state. The training episode is stopped when the terminal state is reached, and a new episode starts with the frame whose geometry is re-randomized.

The number of episodes n_{ep} is 5000, and the size of member feature n_f is 100. The training is implemented with the learning rate $\alpha = 1.0 \times 10^{-5}$ and the discount rate $\gamma = 0.99$. We use Intel(R) Core(TM) i9-7900X CPU @ 3.30 GHz as a central processing unit (CPU). In order for the agent to obtain reasonable solutions, it is necessary not only to perform numerous simulations and observe the structural analysis results, but also to repeatedly update the trainable parameters so as to minimize the loss defined in Eq. (15); in particular, the latter requires large-scale matrix operations. In this study, a graphics processing unit (GPU) GeForce(R) RTX 3090, which is suitable for large-scale operations, is further introduced for computing L_{MSE} and tuning Θ to accelerate the training. Note that the training time would be several times longer if the same training process is performed without the GPU.

6.2. Training result

6.2.1. Training result of reducing sizes

Fig. 9 shows the history of obtained cumulative reward in the test simulations recorded every 10 episodes using the fixed frame model as shown in Fig. 16(a). The cumulative reward obtained by the agent has increased as the number of training episodes increases. It took about 25.4 h for the training. As Fig. 9 indicates, during the 5000 episodes of training, the agent tested at the final episode earned the highest cumulative reward and is considered to be the best agent.

The sequence of design changes of the cross-sections in the test simulation by the best agent is illustrated in Fig. 10 for a 8-story 3-span frame. In the early phase of the episode, the column sizes are intensively reduced. After that, the cross-sections of the columns and beams of the entire frame are reduced in a well-balanced manner without concentrating on the cross-section of a specific member.

At step 447, the displacement ratio of the 4th story exceeded 1.0 due to the size reduction of a beam between 3rd and 4th stories, and the structure reached the terminal state. Therefore, the 446th step just before the terminal state is regarded as the sub-optimal solution, as illustrated in Fig. 11. The cross-section of the sub-optimal solution is a reasonable design that balances the constraints of stress, displacement, COF, and shear load capacity without over-designing or under-designing any particular member size.

In Fig. 9, the training result without the restriction described in Section 4.3 is also plotted. Note that the performance cannot be simply compared through cumulative reward due to the difference in the number of members whose size changes in each step. Although the cumulative reward increases by relaxing the restriction, the time required for the training was 37.1 h, which is almost 50% larger than the case of imposing the restriction. Furthermore, the obtained cross-sections in this case are not reasonable at some members because some lower columns in the same axis are slenderer than the upper ones, as shown in Fig. 12. Therefore, introducing the restriction not only increases the learning efficiency, but also contributes to improving RL agent’s performance of design changes. In the following, only the results imposing the restriction are provided.

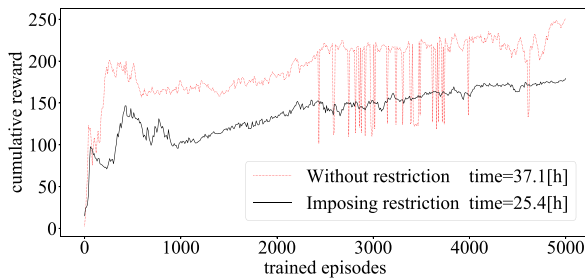


Fig. 9. History of the cumulative reward obtained in each test measured every 10 episodes (reducing member sizes).

6.2.2. Training result of increasing sizes

Next, the result trained for increasing member sizes is explained. The training took approximately 28.5 h. Fig. 13 shows the histories of the cumulative reward in the test simulations recorded every 10 episodes. Similarly to the case of reducing member sizes, RL agent's performance steadily improves as the number of training episodes increases.

The RL agent that has learned 4300 episodes obtains the highest cumulative reward; therefore, it is considered to be the best performing agent. The agent acquires a reasonable strategy to intensively increase the size of columns and beams in the lower layer in the early stage of the test simulation, as seen in Fig. 14.

Because the member sizes are sequentially increased from an infeasible solution until reaching a feasible solution, the feasible cross-sectional design at the 296th step, which is the terminal state, is regarded as the sub-optimal solution. The cross-sections are designed so that the members bear forces in a balanced manner. In addition, larger cross-sections are assigned to long beams than short beams, which is an efficient cross-sectional design (see Fig. 15).

6.2.3. Investigation of generalization performance

The main feature of the proposed method is that the trained agent can be used without re-training for frames of various geometry and topology. Here, the trained agents are applied to other frames: a 4-story 5-span frame, a 12-story 6-span frame, and a L-shaped frame, as shown in Fig. 16.

Among metaheuristic approaches, PSO is chosen as a benchmark for the following reason. First, the PSO algorithm contains fewer parameters [46]. Since the performance of metaheuristic approaches is greatly influenced by the parameter settings, it is preferable for the benchmark methods to have fewer parameters. Second, the PSO algorithm occupies fewer programming resources [47], which is beneficial in avoiding performance differences due to coding and implementation techniques.

The PSO algorithm for comparison is described in Table 5. PSO algorithms utilize a population of candidate solutions, denoted as particles, whose positions are repeatedly updated based on the current position and velocity of each particle, and also the best known positions of the particle and the entire population.

Let $x_{j,i}$ denote the i th variable of particle j . The lower and upper bounds of the variables are uniformly set to be $x^L = 0.0$ and $x^U = 1.0$. The continuous PSO variables are converted to discrete cross-section indices J_i in Tables 1 and 2 using the following equation:

$$J_i = 200 + 50 \times \text{TR} (17x_{j,i} - 1.0 \times 10^{-10}) \quad (16)$$

where TR is a truncation operator. In Eq. (16), a very small value 1.0×10^{-10} is used to avoid the non-existing index $J_i = 1050$ when $x_{j,i} = 1.0$. Note that the negative values are truncated to 0 and accordingly, $J_i = 200$ when $x_{j,i} = 0.0$. To ensure the consistency of the problem definition, if the cross-section of a column is smaller than that of upper columns in the same axis, the cross-section is modified so as to become equal to the maximum cross-section among the upper columns. The

Table 5

Benchmark PSO method.

input: F : cost function, x_j : initial solution of particle j , v_j : initial velocity of particle j , $x^L (= \{1.0, 1.0, \dots, 1.0\})$: upper bounds, $x^U (= \{0.0, 0.0, \dots, 0.0\})$: lower bounds, $n_p (= 10)$: number of particles, $n_1 (= 2000)$: maximum iteration, $\xi (= 0.7)$: momentum factor, $c_1 (= 2.0)$: social coefficient, $c_2 (= 2.0)$: cognitive coefficient

output: x_b : best solution

```

 $x_b \leftarrow x_1$ 
for  $j \leftarrow 1$  to  $n_p$  do
   $x_{pb,j} \leftarrow x_j$ 
  if  $F(x_j) < F(x_{pb,j})$  then
     $x_{pb,j} \leftarrow x_j$ 
 $iter \leftarrow n_p$ 
while  $iter \leq n_1$  do
  for  $j \leftarrow 1$  to  $n_p$  do
     $x_j \leftarrow x_j + v_j$ 
    foreach  $x_{j,i} \in x_j$  do
      if  $x_{j,i} < x^L_i$  then
         $x_{j,i} \leftarrow x^L_i$ 
      if  $x_{j,i} > x^U_i$  then
         $x_{j,i} \leftarrow x^U_i$ 
      if  $F(x_j) < F(x_{pb,j})$  then
         $x_{pb,j} \leftarrow x_j$ 
         $I \leftarrow 0$ 
      if  $F(x_j) < F(x_{pb,j})$  then
         $x_{pb,j} \leftarrow x_j$ 
     $iter \leftarrow iter + 1$ 
  for  $j \leftarrow 1$  to  $n_p$  do
     $v_j \leftarrow \xi v_j + c_1(x_b - x_j) + c_2(x_{pb,j} - x_j)$ 
return  $x_b$ 

```

variables of initial solution of every particle are uniformly given as 0.5. The initial velocity of each particle is provided from a uniform distribution of $[-0.1, 0.1]$.

The cost function to be minimized by the PSO algorithm is defined as

$$F(\mathbf{J}) = V_s(\mathbf{J}) + b_1 C_1(\mathbf{J}) + b_2 C_2(\mathbf{J}) + b_3 C_3(\mathbf{J}) + b_4 C_4(\mathbf{J}) \quad (17a)$$

$$C_1(\mathbf{J}) = \max \left\{ \max_{i \in \{1, \dots, n_m\}} (\tilde{\sigma}_i - 1.0), 0 \right\} \quad (17b)$$

$$C_2(\mathbf{J}) = \max \left\{ \max_{i \in \{1, \dots, n_m\}} (\tilde{d}_i - 1.0), 0 \right\} \quad (17c)$$

$$C_3(\mathbf{J}) = \max \left\{ \max_{k \in \Omega_\beta} (1.0 - \beta_k), 0 \right\} \quad (17d)$$

$$C_4(\mathbf{J}) = \begin{cases} 0.0 & (\text{if } Q_{u,k} \geq Q_{un,k} \ (k = 1, \dots, n_{st})) \\ 1.0 & (\text{else}) \end{cases} \quad (17e)$$

where b_1 , b_2 , b_3 and b_4 are the penalty coefficients for the stress, displacement, COF, and shear load capacity constraints, respectively, which are 1000 in the following results. To consider the variation of solutions due to randomness, the PSO algorithm is executed 10 times and the minimum of $F(\mathbf{J})$ is extracted.

The comparative result is described in Table 6, in which the elapsed wall time t_c [s] using Intel(R) Core(TM) i9-7900X CPU @ 3.30 GHz without GPU and the total structural volume V_s are summarized. Note that the elapsed time t_c for RL+GE includes initializing the frame model, importing the trained RL agent, and simulating the design changes of the members until reaching the terminal state. Owing to the penalty terms, all the solutions obtained by PSO are feasible satisfying all the constraints. t_c measured in PSO is the average of 10 optimizations. The computational cost of RL+GE is lower than PSO, while the proposed RL+GE method achieves better solutions compared with those obtained by PSO in terms of the total structural volume V_s .

According to Table 6, Agent (+) seems to complete the task significantly quicker than Agent (-). As explained in Section 4.2, Agent (+) sequentially increases member sizes starting from the slenderest members, while Agent (-) sequentially reduces member sizes starting from the thickest members. Due to this difference, the time required to complete the task differs even if the frames have the same shape. Since

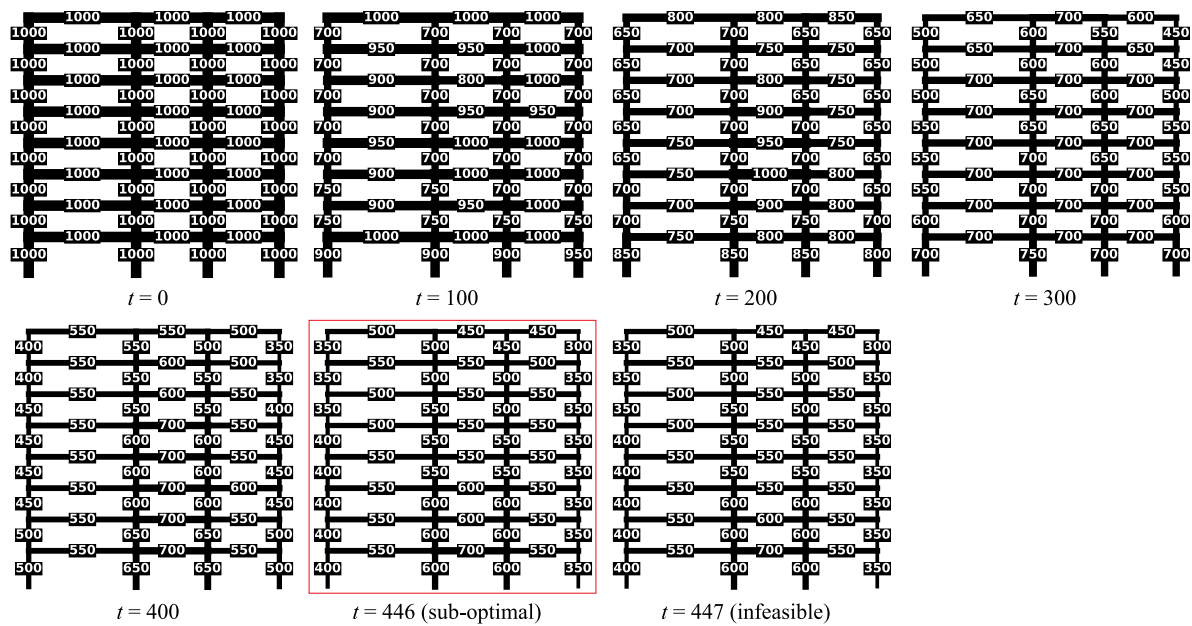


Fig. 10. Sequence of reducing member sizes.

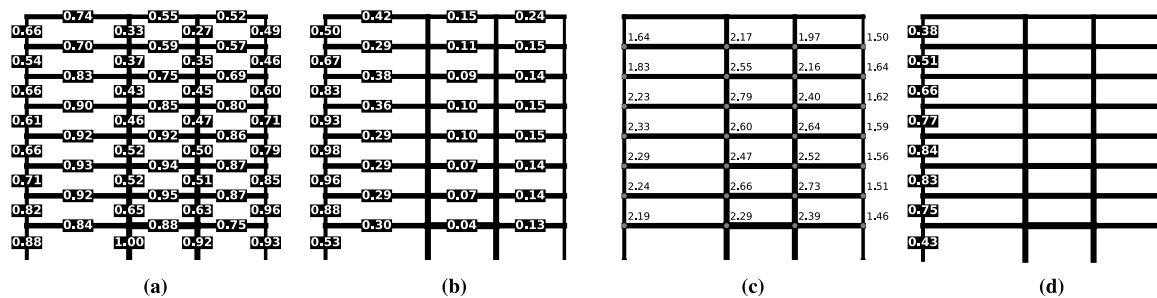


Fig. 11. Result at step 446 (reducing sizes). (a) Maximum stress ratio in each member for the elastic design. (b) Maximum displacement ratio \bar{d}_i in each member for the elastic design. (c) Minimum COF β_k at each node. (d) Maximum inter-story drift ratio divided by the upper bound $1/100$ in each story when loads equivalent to required shear load bearing capacity is applied.

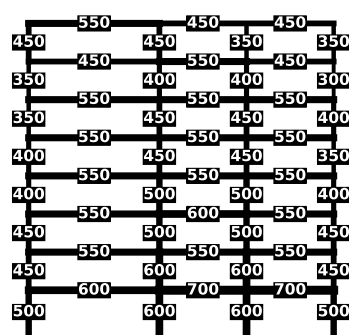


Fig. 12. Sub-optimal design when relaxing the restriction described in Section 4.3.

the optimal solution has many members slenderer than the median, Agent (+) was faster to reach the optimum solution.

Furthermore, RL agents' policy holds consistency in changing the cross-sections as seen in Table 6; the longer the beam, the larger the size of the beam, which is reasonable because a larger bending moment acts on a beam with a larger span. In contrast, consistency is not confirmed in the design of member cross-sections obtained by PSO, because PSO is a method of changing variables stochastically. These results imply

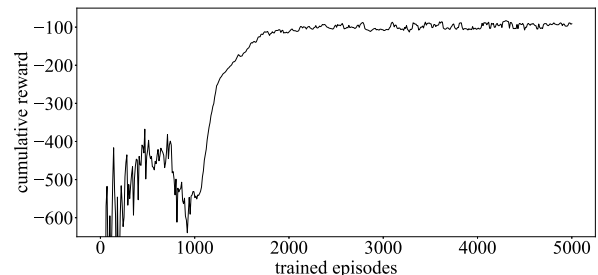


Fig. 13. History of the cumulative reward obtained in each test measured every 10 episodes (increasing member sizes).

that the RL agents are able to change the variables more rationally and efficiently than the PSO algorithm.

7. Conclusion

In this study, a combined method of GE and RL is proposed for the optimal cross-section design of planar steel frames. The cross-sections of steel frames are selected from a prescribed set of standard dimensions so as to minimize the total structural volume while satisfying constraints on the stresses, displacements, COFs, and shear load

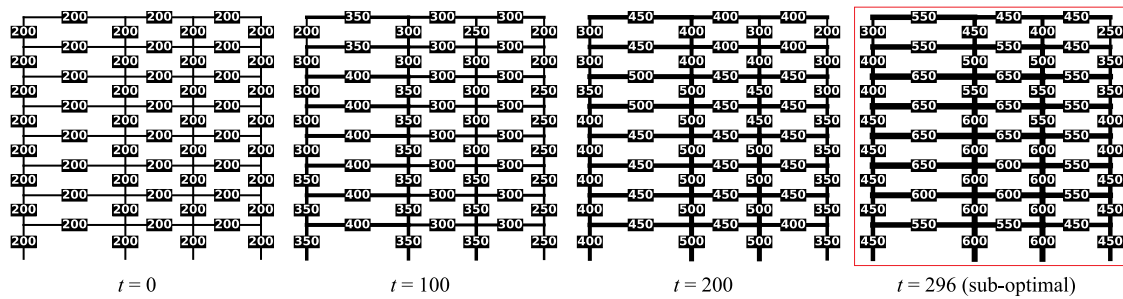


Fig. 14. Sequence of increasing member sizes.

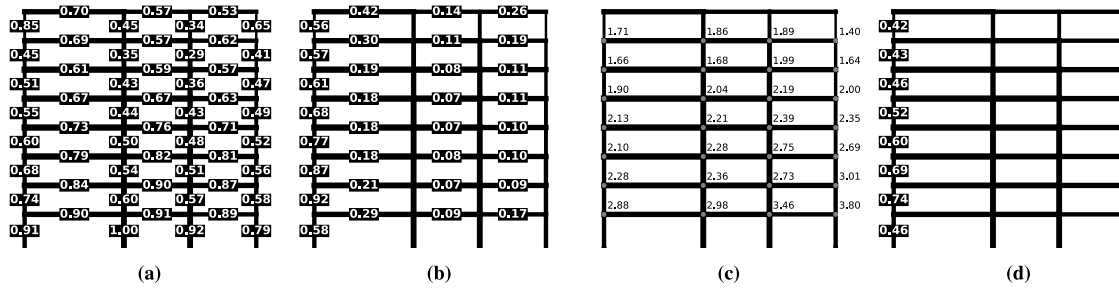


Fig. 15. Result at step 296 (increasing sizes). (a) Maximum stress ratio in each member for the elastic design. (b) Maximum displacement ratio \bar{d}_i in each member for the elastic design. (c) Minimum COF β_k at each node. (d) Maximum inter-story drift ratio divided by the upper bound 1/100 in each story when loads equivalent to required shear load bearing capacity is applied.

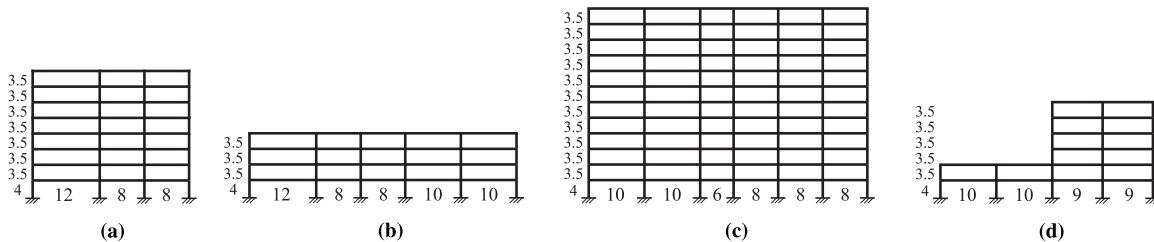


Fig. 16. Frame models used for comparative study in Table 6. The numbers indicate the interval between columns or beams [m].

capacity. The optimization problem is converted to an RL task within an MDP framework, and its components (states, actions, and rewards) are defined.

The cross-sections of beams and columns are selected from the lists of 17 preassigned sections, respectively, and the number of total possible combinations is huge. In order to cope with this difficulty, the number of combinations is reduced by imposing a restriction that the cross-section of columns on the same axis becomes thinner on the upper floors, and by simplifying the process of changing member cross-sections to a process of monotonically increasing or decreasing member sizes. This way, the agent is able to improve its performance in a realistic learning time.

The GE process proposed here is specially formulated to extract member features. Note that the proposed GE method is a generalization of various edge-based graph problems such as link prediction. Owing to the GE operations, the structural property of each member is expressed as a feature vector of the same size considering the status of neighbor nodes and members.

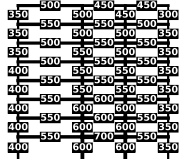

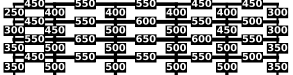

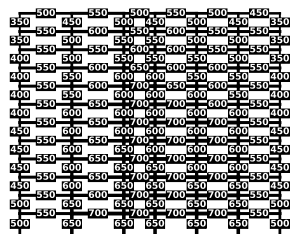
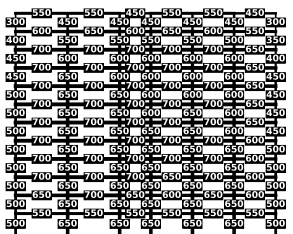
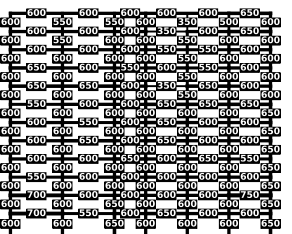
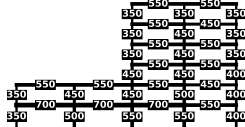
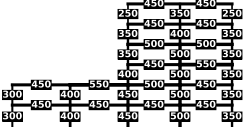
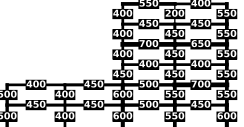
The features are converted to action values of applying design change to each member. To estimate the action values correctly, the loss function to be minimized is formulated based on Q-learning which is one of the RL methods. By using block matrices, mini-batch training using frames with different topology simultaneously is successfully implemented.

The most significant limitation of the proposed algorithm is the huge computational load for training the agent, as with most RL-based

approaches. Despite the use of a GPU to accelerate large-scale matrix computation, the training took a huge amount of time as much as tens of hours. Another limitation is the complexity of implementing the training. For successful learning, the size of the trainable parameters, the learning rate, the discount rate, and other hyper-parameters need to be set by trial and error, and the node and member inputs, v and w , need to be scaled so as to take the value within [0,1]. The reward function is also carefully defined so that the agents can consider the design constraints and the objective of volume minimization in a well balanced manner. These settings may need to be modified when the lists of cross-sections or the frames to be learned changes significantly, which requires a great deal of labor.

Nevertheless, the trained agents exhibited strong performance commensurate with the time and complexity of learning. The trained agent can be used for frames with different geometry, topology and the numbers of nodes and members at a low computational cost. The reduction of computational cost using the RL agent may contribute to boosting the efficiency of structural design process and stimulate the human-computer interaction in the real-world design task. It is shown in the numerical examples that the trained agent is capable of finding reasonable solutions superior to those obtained by PSO. Therefore, the agent is verified to acquire a versatile policy to design the member cross-sections for various frames. Note that the method in this study cannot be directly applied if the frames have irregular shapes or the frames have regular shapes but are designed in 3D; in that case, a new method needs to be developed.

Table 6
Comparison between proposed method (RL+GE) and PSO in view of total structural volume V_s [m³] and elapsed wall time for one optimization t_c [s].

model	RL+GE Agent(-)	RL+GE Agent(+)	PSO
(a)	$t_c = 15.5$ $V_s = 7.806$ 	$t_c = 6.7$ $V_s = 8.355$ 	$t_c = 49.6$ $V_s = 9.034$
(b)	$t_c = 10.6$ $V_s = 6.444$ 	$t_c = 4.1$ $V_s = 6.192$ 	$t_c = 36.4$ $V_s = 7.211$ 
(c)	$t_c = 98.5$ $V_s = 25.213$ 	$t_c = 60.6$ $V_s = 27.358$ 	$t_c = 124.2$ $V_s = 27.540$ 
(d)	$t_c = 7.9$ $V_s = 4.885$ 	$t_c = 2.3$ $V_s = 4.235$ 	$t_c = 32.5$ $V_s = 5.355$ 

The agent trained with the proposed method is expected to become a supporting tool for decision-making in structural design and the first step of developing an agent for automatic structural design. Unlike supervised learning, the trained model learns the sequence towards a final desirable state instead of the desirable state only. Therefore, any step of the sequence towards the sub-optimal design can be queried by structural designers, which may improve the freedom and flexibility in the interactive design between the RL agent and the designers. This improvement is also expected to enhance our design exploration towards better structural designs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability statement

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

Acknowledgments

This study was supported by JSPS KAKENHI No. JP20H04467, Grant-in-Aid for Young Scientists (Start-up) No. JP21K20461 and Grant-in-Aid for JSPS Fellows No. JP18J21456.

References

- [1] S. Pezeshk, Design of framed structures: an integrated non-linear analysis and optimal minimum weight design, *Internat. J. Numer. Methods Engrg.* 41 (1998) 459–471, [http://dx.doi.org/10.1002/\(SICI\)1097-0207\(19980215\)41:3<459::AID-NME293>3.0.CO;2-D](http://dx.doi.org/10.1002/(SICI)1097-0207(19980215)41:3<459::AID-NME293>3.0.CO;2-D).
- [2] T. Kimura, M. Ohsaki, R. Okazaki, Simultaneous optimization of brace locations and cross-sections of beams and columns of steel frames, *J. Struct. Constr. Eng.* 83 (2018) 1445–1454, <http://dx.doi.org/10.3130/aajs.83.1445>, (in Japanese).
- [3] N. Tamura, H. Ohmori, Supporting system for structural design of steel frame structures by using multi-objective optimization method, *J. Struct. Constr. Eng.* 73 (2008) 891–897, <http://dx.doi.org/10.3130/aajs.73.891>, (in Japanese).
- [4] K. Hager, R. Balling, New approach for discrete structural optimization, *J. Struct. Eng.* 114 (1988) 1120–1134, [http://dx.doi.org/10.1061/\(ASCE\)0733-9445\(1988\)114:5\(1120\)](http://dx.doi.org/10.1061/(ASCE)0733-9445(1988)114:5(1120)).
- [5] S. Yoshitomi, M. Yamakawa, K. Uetani, A method for selecting optimum discrete sections of steel frames using two-step relaxation, *J. Struct. Constr. Eng.* 69 (2004) 95–100, http://dx.doi.org/10.3130/aajs.69.95_5, (in Japanese).
- [6] V.K. Srivastava, A. Fahim, An optimization method for solving mixed discrete-continuous programming problems, *Comput. Math. Appl.* 53 (2007) 1481–1491, <http://dx.doi.org/10.1016/j.camwa.2007.01.006>.
- [7] M. Liu, S.A. Burns, Y.K. Wen, Genetic algorithm based construction-conscious minimum weight design of seismic steel moment-resisting frames, *J. Struct. Eng.* 132 (2006) 50–58, [http://dx.doi.org/10.1061/\(ASCE\)0733-9445\(2006\)132:1\(50\)](http://dx.doi.org/10.1061/(ASCE)0733-9445(2006)132:1(50)).
- [8] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, USA, 1998.
- [9] R.J. Balling, Optimal steel frame design by simulated annealing, *J. Struct. Eng.* 117 (1991) 1780–1795.
- [10] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, *Science* 220 (1983) 671–680, <http://dx.doi.org/10.1126/science.220.4598.671>.
- [11] A.L. Samuel, Some studies in machine learning using the game of checkers, *IBM J. Res. Dev.* 3 (1959) 210–229.

- [12] A.A. Markov, N.M. Nagorny, *The Theory of Algorithms*, first ed., Springer Publishing Company, Incorporated, 2010.
- [13] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of go without human knowledge, *Nature* 550 (2017) 354–359, <http://dx.doi.org/10.1038/nature24270>.
- [14] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fiedjland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, *Nature* 518 (2015) 529–533, <http://dx.doi.org/10.1038/nature14236>.
- [15] K. Hayashi, M. Ohsaki, Reinforcement learning and graph embedding for binary truss topology optimization under stress and displacement constraints, *Front. Built Environ.* 6 (2020) <http://dx.doi.org/10.3389/fbuil.2020.00059>.
- [16] M. Mohri, A. Rostamizadeh, A. Talwalkar, *Foundations of Machine Learning*, The MIT Press, 2012.
- [17] D. Prayogo, M.-Y. Cheng, Y.-W. Wu, D.-H. Tran, Combining machine learning models via adaptive ensemble weighting for prediction of shear capacity of reinforced-concrete deep beams, *Eng. Comput.* (2019) <http://dx.doi.org/10.1007/s00366-019-00753-w>.
- [18] J.-S. Chou, A.-D. Pham, Enhanced artificial intelligence for ensemble approach to predicting high performance concrete compressive strength, *Constr. Build. Mater.* 49 (2013) 554–563, <http://dx.doi.org/10.1016/j.conbuildmat.2013.08.078>.
- [19] M. Khandelwal, Blast-induced ground vibration prediction using support vector machine, *Eng. Comput.* 27 (2011) 193–200, <http://dx.doi.org/10.1007/s00366-010-0190-x>.
- [20] D.-C. Feng, Z.-T. Liu, X.-D. Wang, Z.-M. Jiang, S.-X. Liang, Failure mode classification and bearing capacity prediction for reinforced concrete columns based on ensemble machine learning algorithm, *Adv. Eng. Inform.* 45 (2020) 101126, <http://dx.doi.org/10.1016/j.aei.2020.101126>.
- [21] N. Jung, G. Lee, Automated classification of building information modeling (BIM) case studies by BIM use based on natural language processing (NLP) and unsupervised learning, *Adv. Eng. Inform.* 41 (2019) 100917, <http://dx.doi.org/10.1016/j.aei.2019.04.007>.
- [22] J. Chow, Z. Su, J. Wu, P. Tan, X. Mao, Y. Wang, Anomaly detection of defects on concrete structures with the convolutional autoencoder, *Adv. Eng. Inform.* 45 (2020) 101105, <http://dx.doi.org/10.1016/j.aei.2020.101105>.
- [23] C.S.N. Pathirage, J. Li, L. Li, H. Hao, W. Liu, P. Ni, Structural damage identification based on autoencoder neural networks and deep learning, *Eng. Struct.* 172 (2018) 13–28, <http://dx.doi.org/10.1016/j.engstruct.2018.05.109>.
- [24] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, in: *Proceedings 2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Piscataway, NJ, USA, 2017.
- [25] S. Nakamura, T. Suzuki, High-speed calculation in structural analysis by reinforcement learning, in: *The 32nd Annual Conference of the Japanese Society for Artificial Intelligence (JSAI2018)*, 2018, (in Japanese).
- [26] D. Chiba, M. Suzuki, M. Hayashi, K. Watanabe, Development of active response control system using AI (part 2. creation of AI for response control by deep reinforcement learning), in: *Summaries of Technical Papers of Annual Meeting (Structure II)*, 21200, Architectural Institute of Japan, 2018, pp. 399–400, (in Japanese).
- [27] K. Hayashi, M. Ohsaki, Reinforcement learning for optimum design of a plane frame under static loads, *Eng. Comput.* 37 (2021) 1999–2011, <http://dx.doi.org/10.1007/s00366-019-00926-7>.
- [28] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, *Neural Comput.* 1 (1989) 541–551.
- [29] H. Cai, V.W. Zheng, K. Chang, A comprehensive survey of graph embedding: Problems, techniques, and applications, *IEEE Trans. Knowl. Data Eng.* 30 (2018) 1616–1637, <http://dx.doi.org/10.1109/TKDE.2018.2807452>.
- [30] F.A. Faber, L. Hutchison, B. Huang, J. Gilmer, S.S. Schoenholz, G.E. Dahl, O. Vinyals, S. Kearnes, P.F. Riley, O.A. von Lilienfeld, Machine learning prediction errors better than DFT accuracy, 2017, CoRR 1702.05532.
- [31] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, 2017, CoRR 1704.01212.
- [32] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14*, Association for Computing Machinery, New York, NY, USA, 2014, pp. 701–710, <http://dx.doi.org/10.1145/2623330.2623732>.
- [33] E. Ross, D. Hambleton, Using graph neural networks to approximate mechanical response on 3D lattice structures, in: *Proceedings of AAG2020 - Advances in Architectural Geometry*, 24, 2021, pp. 466–485.
- [34] I. Bello, H. Pham, Q.V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, in: *5th International Conference on Learning Representations (ICLR 2017)*, 2017, pp. 1–5.
- [35] H. Dai, E.B. Khalil, Y. Zhang, B. Dilkina, L. Song, Learning combinatorial optimization algorithms over graphs, in: *Proceedings of the 31st International Conference on Neural Information Processing Systems*, in: *NIPS'17*, 2017, pp. 6351–6361.
- [36] J. Jiang, C. Dun, Z. Lu, Graph convolutional reinforcement learning for multi-agent cooperation, 2018, CoRR 1810.09202.
- [37] A. Malysheva, T.T.K. Sung, C. Sohn, D. Kudenko, A. Shpilman, Deep multi-agent reinforcement learning with relevance graphs, 2018, CoRR 1811.12557.
- [38] W. Zheng, D. Wang, F. Song, Opengraphgym: A parallel reinforcement learning framework for graph optimization problems, in: V.V. Krzhizhanovskaya, G. Závodszy, M.H. Lees, J.J. Dongarra, P.M.A. Sloom, S. Brissos, J. Teixeira (Eds.), *Computational Science - ICCS 2020-20th International Conference*, Amsterdam, the Netherlands, June 3-5, 2020, *Proceedings, Part V*, in: *Lecture Notes in Computer Science*, vol. 12141, Springer, 2020, pp. 439–452, http://dx.doi.org/10.1007/978-3-030-50426-7_33.
- [39] S. Bandyopadhyay, A. Biswas, M.N. Murty, R. Narayanam, Beyond node embedding: A direct unsupervised edge representation framework for homogeneous networks, 2019, CoRR [abs/1912.05140](https://arxiv.org/abs/1912.05140).
- [40] C. Wang, C. Wang, Z. Wang, X. Ye, P.S. Yu, Edge2vec: Edge-based social network embedding, 14 (2020). <http://dx.doi.org/10.1145/3391298>.
- [41] J. Sola, J. Sevilla, Importance of input data normalization for the application of neural networks to complex industrial problems, *IEEE Trans. Nucl. Sci.* 44 (1997) 1464–1468, <http://dx.doi.org/10.1109/23.589532>.
- [42] M. Puheim, L. Madarász, Normalization of inputs and outputs of neural network based robotic arm controller in role of inverse kinematic model, in: *SAMI 2014 - IEEE 12th International Symposium on Applied Machine Intelligence and Informatics*, 2014, p. 4, <http://dx.doi.org/10.1109/SAMI.2014.6822439>.
- [43] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, in: Y. Bengio, Y. LeCun (Eds.), *4th International Conference on Learning Representations, ICLR 2016*, San Juan, Puerto Rico, May 2-4, 2016, *Conference Track Proceedings*, 2016.
- [44] C.J.C.H. Watkins, *Learning from Delayed Rewards* (Ph.D. thesis), King's College, Cambridge, UK, 1989.
- [45] T. Tieleman, G. Hinton, Lecture 6e - rmsprop: Divide the gradient by a running average of its recent magnitude, in: *COURSERA: Neural Networks for Machine Learning*, 2012.
- [46] A. Rezaee Jordehi, J. Jasni, Particle swarm optimisation for discrete optimisation problems: a review, *Artif. Intell. Rev.* 43 (2015) 243–258, <http://dx.doi.org/10.1007/s10462-012-9373-8>.
- [47] Y. Li, Y. Peng, S. Zhou, Improved pso algorithm for shape and sizing optimization of truss structure, *J. Civ. Eng. Manage.* 19 (2013) 542–549, <http://dx.doi.org/10.3846/13923730.2013.786754>.