buildings

MDPI

Article

Automatic Generation of Precast Concrete Component Fabrication Drawings Based on BIM and Multi-Agent Reinforcement Learning

Chao Zhang [1], Xuhong Zhou [1], Chengran Xu [1,*], Zhou Wu [2], Jiepeng Liu [1] and Hongtuo Qi [1]

[1] School of Civil Engineering, Chongqing University, Chongqing 400044, China; 20211601016@stu.cqu.edu.cn (C.Z.); zhouxuhong@126.com (X.Z.); liujp@cqu.edu.cn (J.L.); hitqht@163.com (H.Q.)
[2] School of Automation, Chongqing University, Chongqing 400044, China; zhouwu@cqu.edu.cn
* Correspondence: chengranxu@cqu.edu.cn

Abstract: Fabrication drawings are essential for design evaluation, lean manufacturing, and quality detection of precast concrete (PC) components. Due to the complicated shape of PC components, the fabrication drawing needs to be customized to determine manufacturing dimensions and relevant assembly connections. However, the traditional manual drawing method is time-consuming, labor-intensive, and error-prone. This paper presents a BIM-based framework to automatically generate the readable drawing of PC components using building information modeling (BIM) and multi-agent reinforcement learning (MARL). Firstly, an automated generation method is developed to transform BIM model to view block. Secondly, a graph-based representation method is used to create the relationship between blocks, and a reward mechanism is established according to the drawing readability criterion. Subsequently, the block layout is modeled as a layout optimization problem, and the internal spacing and position of functional category blocks are regarded as agents. Finally, the agents collaborate and interact with the environment to find the optimal layout with the guidance of a reward mechanism. Two different algorithms are utilized to validate the efficiency of the proposed method (MADQN). The proposed framework is applied to PC stairs and a double-sided shear wall to demonstrate its practicability.

Keywords: precast concrete (PC) component; fabrication drawing; layout optimization problem; multi-agent reinforcement learning (MARL); building information modeling (BIM)

Academic Editor: Flora Faleschini

Received: 31 December 2024
Revised: 6 January 2025
Accepted: 16 January 2025
Published: 19 January 2025

Citation: Zhang, C.; Zhou, X.; Xu, C.; Wu, Z.; Liu, J.; Qi, H. Automatic Generation of Precast Concrete Component Fabrication Drawings Based on BIM and Multi-Agent Reinforcement Learning. *Buildings* **2025**, *15*, 284. https://doi.org/10.3390/buildings15020284

1. Introduction

Prefabricated concrete (PC) structures are widely used in various kinds of practical projects, including residential structures, business buildings, manufacturing plants, and infrastructure constructions. Prefabricated construction is an industrialized construction method in which precast components are produced in a factory setting and then transported to the construction site for assembly [1]. Industrialization has put forward new demands for design, requiring designers to consider not only building function but also manufacturing requirements. PC component fabrication cannot be completed by simply using architectural and structural drawings. The generation of fabrication drawings is the key process link to the design and production of PC components. However, due to the need for high precision and the integration of complex and multidisciplinary systems, producing fabrication drawings for PC components is challenging. Additionally, designers must navigate manufacturing constraints, adhere to strict standards and quality control requirements,

and manage frequent design revisions. The generation of fabrication drawings is the critical and last step in the detailed design of prefabricated components. Effective collaboration and well-organized, three-dimensional (3D) visualizations are essential to ensure accurate, compliant, and manufacturable drawings [2]. Building information modeling (BIM) is an integrated digital tool utilized for the design, construction, and management of buildings. BIM provides collaboration and communication for different stakeholders by developing and managing digital representations of building projects. With the aid of BIM technology, the automated generation of fabrication drawings has become possible by converting the 3D model into two-dimensional (2D) views [3].

In PC components, in addition to concrete and reinforcement bars, there are numerous reserved holes and embedded parts designed to facilitate stacking, hoisting, installation, and connection to the building. In practical projects, the detailed design proposals are presented in the form of 2D fabrication drawings, which contain the instructions and information necessary for the accurate manufacturing and installation of PC components. The detail level of the fabrication drawings affects the quality and efficiency of component production. Therefore, the generation of fabrication drawings is of crucial importance, as PC components possess complex shapes requiring customized designs to accurately convey the design intent.

Currently, designers predominantly utilized prior knowledge and trial-and-error approach to produce the fabrication drawing with the aid of CAD. To clearly and precisely represent the design detail, numerous section views must be drafted, resulting in significant time spent on repeated revisions and reviews of drawing characteristics. The determination of view contents and quantities is largely based on designers' existing expertise, while the annotation of graphical elements (including dimensions and text) heavily depends on human-computer interactions. Additionally, the block layout within the drawing space is governed by rule-based methods, which can limit flexibility and efficiency. Moreover, despite advancements in CAD 2020 software, the drafting drawing process remains labor-intensive and susceptible to human error, particularly in complex projects involving intricate geometries and extensive detail requirements. In addition, although existing software can automatically generate the fabrication drawing of a simple PC component, the drawing of a complicated component still requires customized design. The integration of BIM with CAD has shown promise in enhancing data consistency and collaboration, yet challenges persist in fully automating the transition from 3D models to comprehensive 2D fabrication drawings. Therefore, it is imperative that it needs to develop more intelligent and efficient methods for the generation of fabrication drawings.

As a technology used in the digital representation of building physics and functional characteristics, BIM has been widely used in the generation of fabrication drawings in architecture-, construction-, and manufacturing-related fields due to advanced 3D modeling techniques. Industry Foundation Classes (IFC) is an international standard developed by the buildingSMART organization with the aim of promoting openBIM. It serves as a neutral file format that enables data exchange between BIM and CAD systems without any loss of information. Due to these characteristics, IFC is widely utilized in architecture, engineering, and construction (AEC). Therefore, this paper proposed a novel BIM-based method for the intelligent and efficient generation of fabrication drawings for PC components. This framework encompasses four key modules: design information extraction, automatic generation of graphical element, automated annotation, and block layout. In this paper, the design information of the components is extracted from the BIM model to create its topology model based on the Open CASCADE [4] framework. The plane data are generated using computer graphic algorithms (projection and cutting operations), and the graphical element and potential annotation are automatically generated. The automated transformation of 3D

models into 2D view blocks has been successfully achieved. Subsequently, a graph-based representation method is utilized to describe the relationships between blocks, including adjacency, relative position, and alignment. The position and internal spacing of functional category block (FCB) are regarded as different agents. According to the drawing readability criteria, a series of reward mechanisms are created to guide the agent to find the optimal layout solution. Based on these characteristics, an efficient and intelligent framework for the fabrication drawing of PC components is formed to produce a lot of readable drawing layouts while providing high-quality and high-accuracy manufacturing information.

This paper is organized as follows: In Section 2, the related works are reviewed. Section 3 proposes a BIM-based framework for the fabrication drawing generation of PC components, including four major parts: (1) information extraction, (2) automatic generation of drawing views, (3) automatic annotation layout, and (4) intelligent block layout. The proposed multi-agent reinforcement learning algorithm for solving the block layout is discussed in Section 4. Two illustrative examples are given in Section 5 to verify the practicability and feasibility of the framework. Conclusions are included in Section 6.

2. Literature Review

Section 2.1 introduces the research background about the fabrication drawing generation, and Section 2.2 describes the introduction of reinforcement learning.

2.1. Fabrication Drawing Generation

To improve the drafting efficiency, various researchers in the past have tried to solve the problem of the drawing generation for mechanical products with the aid of CAD software. A semi-automated method for the fabrication drawing of mechanical elements was designed using Solid Edge, which provided a modification interface of model parameters for users. The user can quickly generate the manufacturing drawing by only inputting the parameters [5]. Chen et al. [6] utilized computer graphics and expert knowledge to automatically transform a 3D model of mechanical products into a 2D assembly drawing. Shah [7] designed a method combining Excel spreadsheets with Autodesk Inventor to produce the drawing. Users need to input specific parameters into the Excel spreadsheet, and the relevant 3D model can be updated immediately. In addition to the previous research, some existing CAD software, such as FreeCAD [8] and SolidWorks [9], can also generate the fabrication drawing by selecting specific mechanical elements.

In recent years, BIM has been widely used in the automatic generation of fabrication drawings [10–13]. The 3D models of buildings were created and shared for collaborative design, construction, and operations. Compared with the traditional CAD method, a number of advantages in the fabrication drawings generation were provided, such as accuracy, efficiency, and collaboration. Gankhuyag and Han [14] designed an automatic approach to generate the 2D floorplan drawing using 3D point clouds. The planes of structural component were extracted from point clouds data, and the Manhattan-World assumption was utilized to classify the point clouds data according to the character of component. Industry Foundation Classes (IFC) standard is utilized to create a 3D BIM model, and the line detection algorithm is used to produce a 2D floorplan drawing. Kong et al. [15] introduced an efficient method to rapidly generate the engineering drawings for large-scale old buildings using 3D point cloud reconstruction. The Lidar laser scanner was utilized to obtain the point cloud data of building. Some computer vision and computer graph algorithms are proposed to complete the extraction of wall lines and doors. However, these studies have a limitation on the simple geometric features, lacking generality.

The drawing generation for PC components in a manual way is labor-intensive and time-consuming. Since the design information required for component manufacturing

must be integrated into a single drawing, the various elements that need to be composed, dimensioned, and annotated should be clearly identified [16]. More importantly, the dimensions of drawing view significantly affect both the readability of drawing and the accuracy of information extraction in the automated generation process [17,18]. Some scholars begin to focus on the automatic dimensioning for drawing views based on the design constraints and the drawing template of the company [19,20]. Automated or intelligent dimensions require us to solve two problems: the annotation object must be determined based on the geometric feature [21], and the dimensioned position of the annotation object must be identified using an effective method. Yuen et al. [22] adapted an automated processing of the linear dimensions in the view from boundary representation of the solid models. Deng et al. [23] developed an automatic framework for façade building components to generate the fabrication drawing, and special characteristics, including holes and notches, are considered. A rule-based strategy is designed to determine the drawing dimension, and an improved K-means algorithm is used to optimize the zooming factor of different drawings for maintaining a consistent format. Then, a rule-based method is introduced to automatically find the drawing layout. Deng et al. [24] proposed an intelligent framework based on BIM to automatically generate fabrication drawings for facade panels, and the design information coming from BIM models and other external data sources, both graphical and non-graphical, is integrated by the framework. The appropriate views were acquired, and the drawing layouts were improved. However, these methods only focused on the steel components without considering the characteristics of the PC components, which consist of multiple elements (such as solid, rebar, embedded part, and reserved hole). In addition, numerous studies have focused on the automated layout of annotations (dimension and text) within fabrication drawings; however, the optimization of view block layout, which significantly impacts the readability of drawing, remains largely unexplored.

Although several studies have made great progress in drawing generation, the following characteristics of the fabrication drawing generation method can be further enhanced and used in complex PC components: (1) developing an independent and efficient framework to automatically generate the fabrication drawing; (2) generating readable drawing layouts for multiple blocks of different functions using a more efficient and intelligent method.

2.2. Reinforcement Learning

Reinforcement Learning (RL), a prominent branch of artificial intelligence, has gained recognition by distinguishing itself from supervised and unsupervised learning through its unique methodology of making optimal decisions in specific contexts by interacting with the environment [25]. The inherent adaptability and flexibility of reinforcement learning enable agents to find effective solutions in complex and dynamic environments through experimentation and learning [26]. Recently, RL has emerged as a pivotal artificial intelligence methodology widely adopted to address various engineering challenges. Soman and Molina-Solana [27] used deep RL techniques to help construction professionals efficiently plan construction activities and generate a conflict-free look-ahead schedule. Yao et al. [28] proposed an RL model with a valid action sampling mechanism to minimize duration and runtime for large construction projects. Jeong and Jo [29] proposed an RL framework to facilitate automated reinforced concrete (RC) beam design in a cost-effective manner while taking both flexural and shear reinforcement arrangements into consideration. Fu et al. [30] combined the finite element method with deep reinforcement learning (DRL) to establish a physics-informed framework for automated steel frame structure design, thereby identifying the optimal section areas for both beams and columns. The DRL approach can be used to solve the multi-objective optimization of green building design, enhancing

performance in terms of energy efficiency, environmental sustainability, and occupant comfort [31]. In some scenarios with more than one agent, multi-agent versions of the DRL approach are proposed to address large continuous action space problems [32]. Zhang et al. [33] used the multi-agent deep deterministic policy gradient (MADDPG) algorithm to achieve automated architectural space composition under established built environment conditions, addressing the extensive demands of old building renovation. Inspired by the aforementioned research, this study employs a multi-agent deep reinforcement learning (MADRL) approach to provide a more efficient and intelligent solution for generating the fabrication drawing of PC components.

3. BIM-Based Framework for the Fabrication Drawing Generation of PC Components

To efficiently generate the fabrication drawing, a BIM-based framework for the PC component is proposed. As shown in Figure 1, the proposed framework consists of four modules: (1) information extraction of digital model, (2) automatic generation of graphical element, (3) automatic annotation of graphical element, and (4) intelligent layout of view block. To improve the readability of the drawing view, a reinforcement learning method is utilized to find the optimal layout within the drawing space.

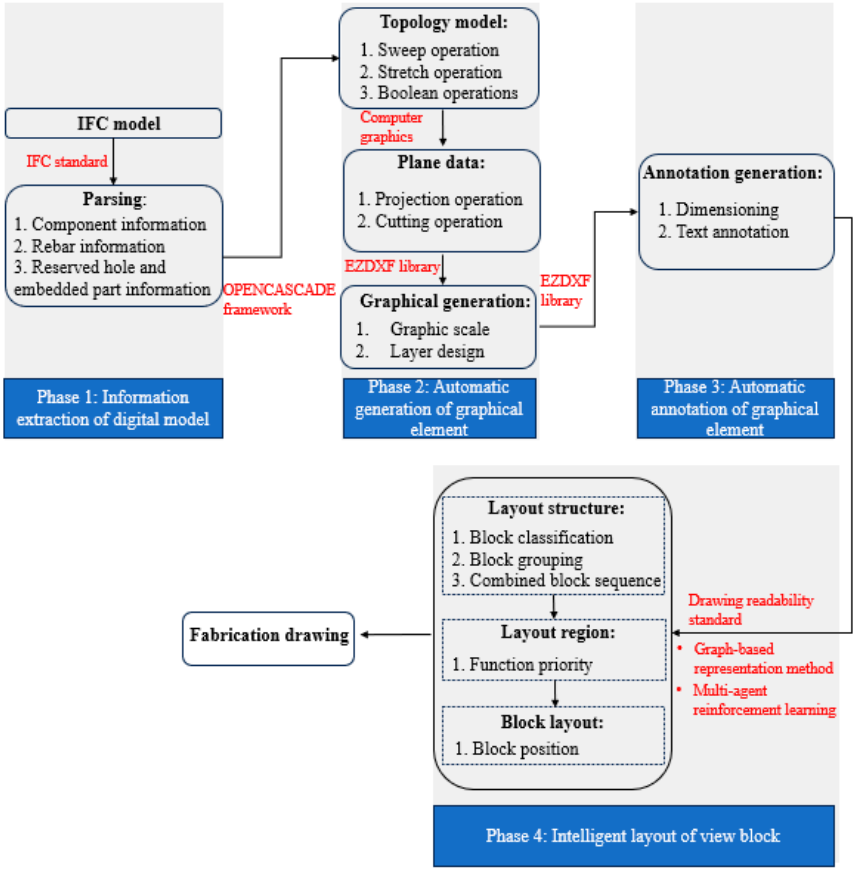


Figure 1. Flowchart of fabrication drawing generation for PC component.

3.1. Information Extraction of Digital Model

BIM as a digital management system is widely used in AEC industry. BIM integrates all the geometric, spatial, physical, and attribute information of the building components. IFC is a standardized and digital description of the buildings and infrastructure. It is an open and neutral standard commonly used for data sharing and exchange in different BIM software. In the PC component, the IFC model stores rich design information, including

the component's solid geometry, reinforcement bar details, embedded part specifications, and locations of reserved holes. To generate the fabrication drawing, corresponding design information is extracted from the IFC model by querying specific entity types. For instance, the IfcElement entity is used to obtain the solid component information, including its geometry, spatial position, and material. The IfcElementAssembly entity is utilized to retrieve the embedded parts of the PC component, encompassing its hoisting and railing embedded parts. Reinforcement bar details, such as bar diameter, length, spacing, and location, are extracted using the IfcReinforcingBar entity. Reserved holes and openings within the PC component are identified and extracted using the IfcOpeningElement entity, capturing their size, shape, and location within the component geometry. This granular extraction process, based on querying and filtering relevant IFC entities, enables the automated extraction of accurate and detailed design information, directly influencing the detail level of the topological model in the next stage. This approach ensures consistency between the digital model and the topological model, minimizing errors and facilitating efficient data exchange throughout the information extraction process.

3.2. Automatic Generation of Drawing View

To accurately represent the characteristics of the component and the relative positions of embedded parts, reinforcement bars, and reserved holes, a large number of projection and section drawings are required. To obtain the view, the topological models of the specific elements are constructed using the Open CASCADE 7.5.0 (OCC) framework. Subsequently, plan view data is generated by applying graphics algorithms.

3.2.1. Automatic Generation of OCC Model

OCC is an open-source software development platform for 3D CAD, CAM and CAE, including parameterized modeling, data structure, and geometric algorithm. Pythonocc is Python library which is based on the OCC modeling kernel. It provides 3D modeling and data exchange features, aiming at CAD/PDM/PLM- and BIM-related development. OCC provides extensive 2D and 3D geometry libraries to create the solid model, such as BRepPrimAPI_MakeBox, BRepOffsetAPI_MakePipe, BRepPrimAPI_MakePrism, and so on. Additionally, some boolean operations, including cut and other operations, are used to create the complicated solid. As shown in Figure 2, two simple solids are modified by Boolean operations to generate three solids.

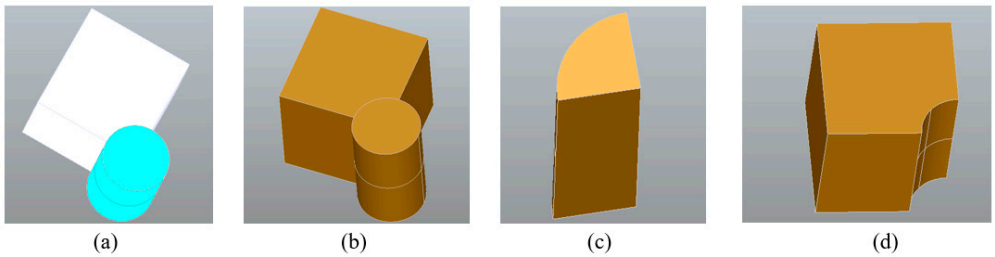


Figure 2. OCC model: (**a**) initial model; (**b**) Boolean union model; (**c**) Boolean intersection model; (**d**) Boolean difference model.

3.2.2. Automatic Generation of View

Projection and section transformation are utilized to generate the plane data of specific objects from OCC models. In the engineering drawing, the projection direction and coordinate system are established to realize the projection transformation of the model. Subsequently, an orthogonal projection transformation of the model is conducted to produce the plane data of three views. Due to the loss of depth information during the projection transformation, this leads to ambiguity in the graphical representation. Therefore, all

hidden lines or surfaces should be eliminated. In addition, the section transformation requires the determination of the cutting position and plane, thereby obtaining the plane data of the model. When employing the OCC framework for projection transformations, the origin, *x*-axis, and projection direction of the projection coordinate system are initially defined. The *y*-axis of the projection coordinate system is subsequently determined by calculating the cross product of the projection direction and the *x*-axis. Following this, the hidden line removal algorithms are employed to obtain the plane data. For section transformations using the OCC framework, it is essential to establish both the cutting position and the normal direction of the cutting plane, thereby defining the section plane. The topological model is then subjected to a Boolean intersection operation with the cutting plane, resulting in the plane data of the model. Algorithm 1 outlines the pseudocode for the projection algorithm, while Algorithm 2 details the pseudocode for the cutting algorithm. The functions within these algorithms are invoked from the OCC framework to obtain the model's plane data. Figure 3 illustrates the projection and section of the model.

Algorithm 1: Pseudocode of projection algorithm

Input: Topological shape A, Projection origin O, Projection direction Dir, X-axis of projection coordinate system X
Output: plane_data
hlr = HLRBRep_Algo() // Construct an empty framework for the calculation of visible and hidden lines of a shape in a projection
hlr.Add(A) // Add topological shape to the framework
ax2 = gp_Ax2(O, Dir, X) // Construct projection coordinate system
projector = HLRAlgo_ProjectorProjector(ax2) // Create a projector object
hlr.Projector(projector) // Set the projection direction
hlr.Update() // Update the outline of shape to be computed
hlr.Hide() // Hide the visible and hidden line of shape to be computed
hlr_shapes = HLRBRep_HLRToShape(hlr) // Construct a framework for filtering the results
visible_edge = [] // visible edge
sharp_edges = hlr_shapes.VCompound() // Extract the visible sharp edges from the results
contour_edges = hlr_shapes.OutLineVCompound() // Extract the visible contour edges from the results
visible_edge.add(sharp_edges)
visible_edge.add(contour_edges)
plane_data = visible_edge

Algorithm 2: Pseudocode of cutting algorithm

Input: Topological shape A, Cutting position P, Normal direction Dir
Output: plane_data
plane = gp_Pln(P, Dir) // Create the cutting plane
topology_contour = BRepAlgoAPI_Section(A, plane) // Find the contour of the intersection of plane and topological model A and obtain the cut trajectory on A
contour_edge = TopologyExplorer(topology_contour).edges() // Transform topological edge to geometric edge
plane_data = contour_edge

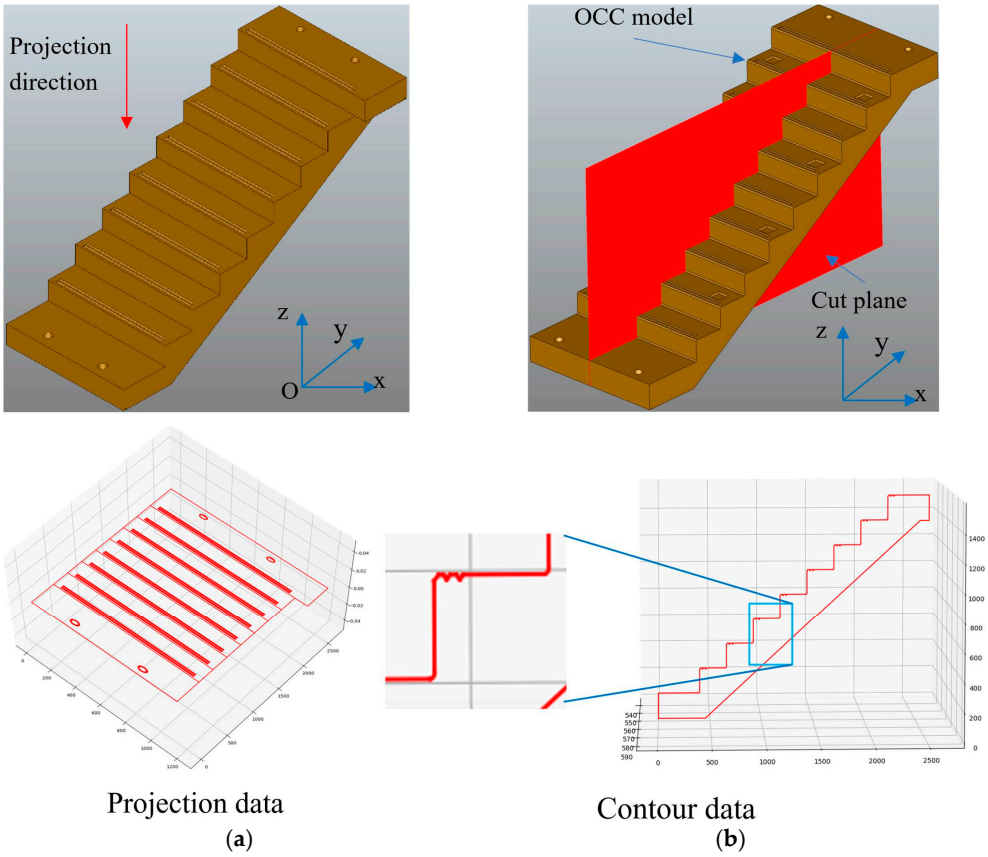


Figure 3. Model transformation: (**a**) projection; (**b**) section.

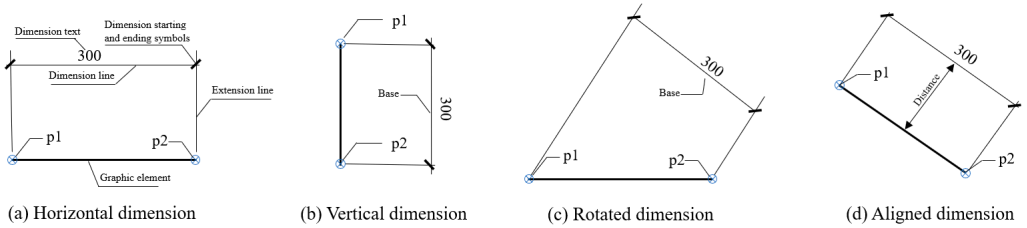
3.3. Automatic Annotation Layout in Each View

To ensure efficient and error-free fabrication, the production drawings for the PC component must include comprehensive annotations detailing key physical attributes such as dimensions, text, material specifications, and connection details. In the fabrication drawing, dimension and text annotation mainly involve annotating the size and characteristics of the graphical elements. Text annotation primarily describes the design and manufacturing details of the element, such as part number, material, technique requirements, and special process. Dimension annotations describe the size of an element and are categorized as linear, diameter, radius, angle, and arc length dimensions based on the element's geometric characteristics.

The details of dimensioning style are discussed as follows:

(1) Linear dimension

As shown in Figure 4a, linear dimension consists of dimension text, dimension line, extension line, and dimension starting and ending symbol. Figure 4a–d describe different types according to the relation of dimension line and graphic element, including horizontal, vertical, aligned, and rotated dimensioning.

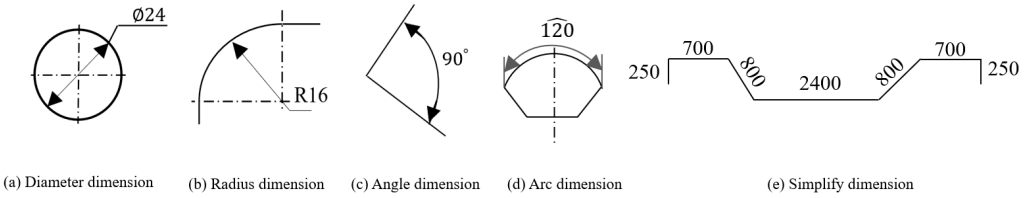


(a) Horizontal dimension (b) Vertical dimension (c) Rotated dimension (d) Aligned dimension

Figure 4. Type of linear dimension.

(2) Diameter dimension

As shown in Figure 5a, the diameter dimension should be placed inside the circle, and the dimension line should pass through the center of the circle, with arrows drawn at both ends pointing to the arc. The diameter symbol "ϕ" should be added before the diameter text.



(a) Diameter dimension (b) Radius dimension (c) Angle dimension (d) Arc dimension (e) Simplify dimension

Figure 5. Dimensioning type of different graphical element.

(3) Radius dimension

As shown in Figure 5b, the dimension line for the radius starts from the center of the circle, with an arrow pointing to the arc at the other end. The symbol "R" should be added before the radius number. For smaller arc radii, leader lines can be used for dimensioning.

(4) Angle dimension

As shown in Figure 5c, the dimension line for an angle should be represented by an arc, with the center of the arc being the vertex of the angle, and the two sides of the angle serving as the dimension limits.

(5) Arc dimension

As shown in Figure 5d, when annotating the arc length of a circle, the dimension line should be represented by an arc line concentric with the circle, and the dimension limits should point towards the center of the circle.

(6) Simplify dimension

As shown in Figure 5e, when annotating a simplified drawing of reinforcement bars, the simplified drawing describes the shape and content of the reinforcement bars, and the dimension text (dimensions or angles) can be directly placed along one side of the reinforcement bars.

In addition, the text annotation consists of multiple straight lines and text, generally with two straight lines. The first line connects to the graphical element that needs to be annotated, and its end connects to the beginning of the second line. The text content is placed at the top of the second line.

3.4. Intelligent Generation of Block Layout

After completing the annotation of graphical elements in the drawing view, all views are converted into blocks and placed within a 2D drawing space. The main task of block layout is to determine the position of given block within the drawing space in order to ensure the readability. To improve the readability of design drawings, the following principles should be followed: (1) Function consistency: blocks with the same function should be placed within the same region; (2) Function priority: more important functional blocks should be placed in the primary region of the drawing space, while less important blocks should be placed in the secondary region; (3) Relationship restraint: certain specialized blocks, such as three views and section views, should be arranged in adjacent positions, either horizontally or vertically; (4) Size constraint: blocks should be arranged in a sequential order based on their sizes, from largest to smallest; (5) Position constraint: there should be no collisions between blocks; (6) Uniform spacing: the spacing between adjacent blocks

within each FCB should be uniform; (7) Fullness: the outer contour of each block should occupy a significant proportion of the drawing space. Due to the large variation in the size and number of blocks, a graph-based representation method is employed to determine the layout structure of each FCB in accordance with principles 1–4.

3.4.1. Automatic Generation of Layout Region and Structure for Each Functional Category

In graph theory, a graph is made up of vertices that are connected by edges. The vertices represent single objects, while the edges represent the specific relationship between these objects [34]. A graph is a set of vertices and edges, denoted as $G = (V, E)$, where V represents a set of vertices and E represents a set of edges. A single edge is represented as $e_{ij} = \langle v_i, v_j \rangle$, where v_i is the start point of the edge, and v_j is the end point. If the edge e_{ij} is equal to e_{ji}, the graph is undirected. Conversely, the graph is directed. In the computer science domain, an adjacency matrix is commonly utilized to store the relationships between the vertices of the graph.

As shown in Figure 6, the process of block layout are described as follows: (1) Information extraction: the rectangular bounding boxes and semantic information of the blocks are extracted from the drawing; (2) Function classification: multiple functional categories are identified based on their purposes, each comprising several blocks; (3) Block grouping: within each functional category, all blocks are divided into multiple groups based on principle 3; (4) Size sorting: the groups are organized in accordance with the requirements of principle 4; (5) Layout structure: considering the compactness, the layout structure can be determined, and adjacent blocks have a position and alignment relationships; (6) Layout region: multiple functional categories are placed in the drawing space according to the function priority.

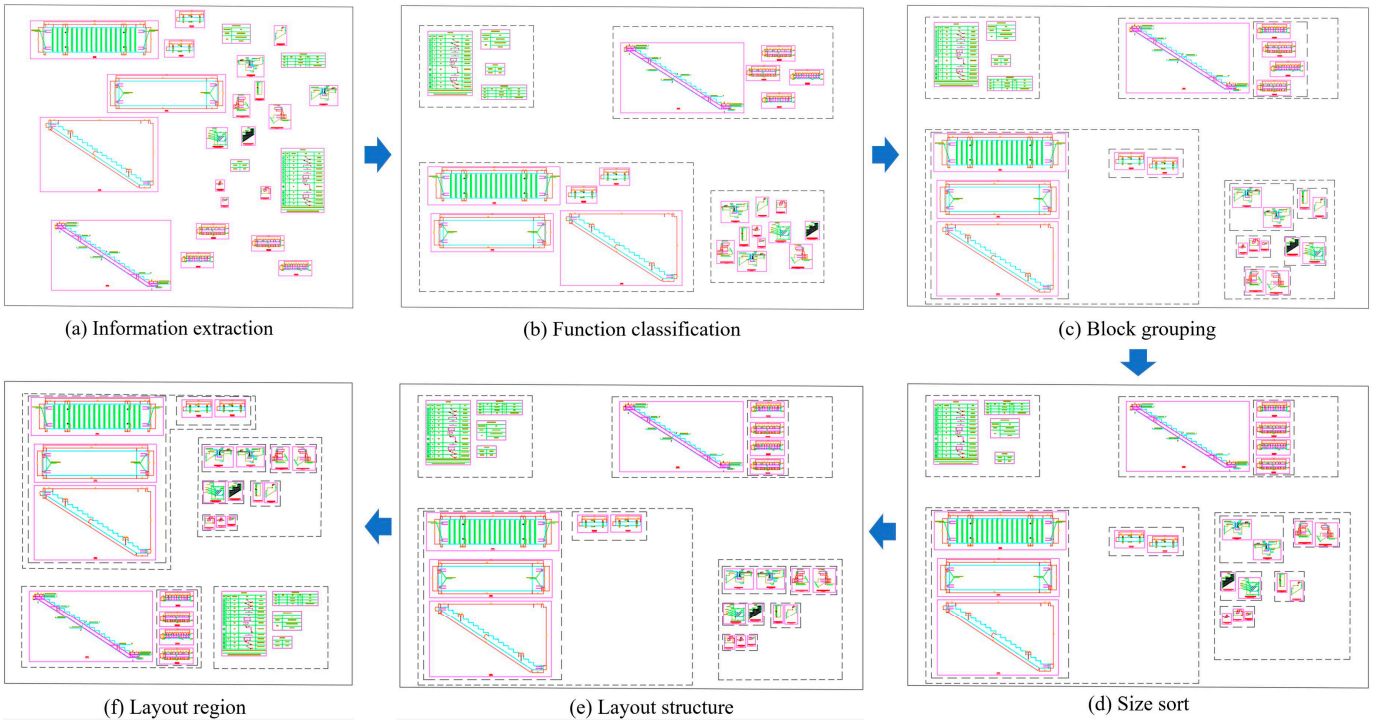


Figure 6. Process of block layout.

As shown in Figure 7a–d, there are four alignment relationships between block one and block two, namely top-align (value set to 1), bottom-align (set to 2), left-align (set to 3), and right-align (set to 4). In addition, two adjacent blocks have four relative positions, such as right-side (value set to 1), left-side (set to −1), bottom-side (set to −2), and top-side

(set to 2) position. An illustrative example is depicted in Figure 8 to represent the layout structure of a block in a single functional category. They can be automatically updated according to given blocks and easily generalized to present other fabrication drawings of PC components. To determine the position and internal spacing of each FCB, Section 4 introduces a multi-agent deep reinforcement learning method based on principles 5–7.

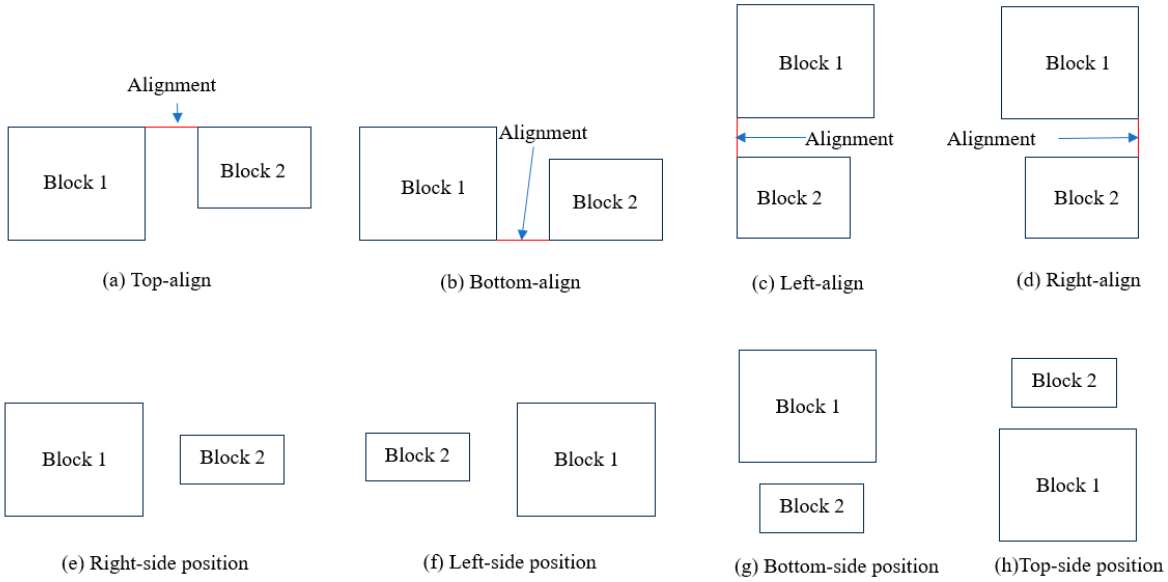
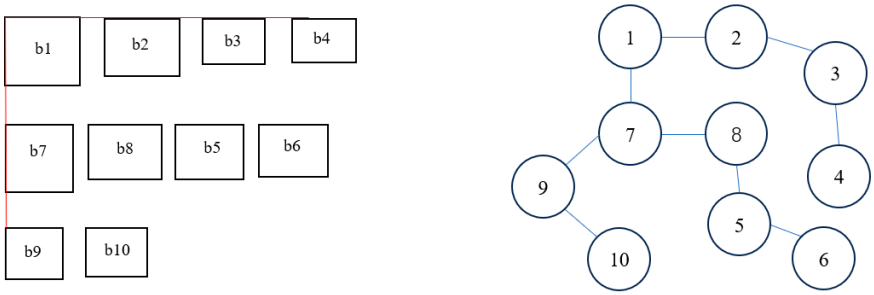


Figure 7. Alignment and position relationship of block.



(a) Example for block layout (b) Connectivity graph

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10
b1	0	1	0	0	0	0	1	0	0	0
b2	1	0	1	0	0	0	0	0	0	0
b3	0	1	0	1	0	0	0	0	0	0
b4	0	0	1	0	0	0	0	0	0	0
b5	0	0	0	0	0	1	0	1	0	0
b6	0	0	0	0	1	0	0	0	0	0
b7	1	0	0	0	0	0	0	1	1	0
b8	0	0	0	0	1	0	1	0	0	0
b9	0	0	0	0	0	0	1	0	0	1
b10	0	0	0	0	0	0	0	0	1	0

(c) Block connection matrix

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10
b1	0	1	0	0	0	0	−2	0	0	0
b2	−1	0	1	0	0	0	0	0	0	0
b3	0	−1	0	1	0	0	0	0	0	0
b4	0	0	−1	0	0	0	0	0	0	0
b5	0	0	0	0	0	1	0	−1	0	0
b6	0	0	0	0	−1	0	0	0	0	0
b7	2	0	0	0	0	0	0	1	−2	0
b8	0	0	0	0	1	0	−1	0	0	0
b9	0	0	0	0	0	0	2	0	0	1
b10	0	0	0	0	0	0	0	0	−1	0

(d) Block position matrix

	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10
b1	0	1	0	0	0	0	3	0	0	0
b2	1	0	1	0	0	0	0	0	0	0
b3	0	1	0	1	0	0	0	0	0	0
b4	0	0	1	0	0	0	0	0	0	0
b5	0	0	0	0	0	1	0	1	0	0
b6	0	0	0	0	1	0	0	0	0	0
b7	3	0	0	0	0	0	0	1	3	0
b8	0	0	0	0	1	0	1	0	0	0
b9	0	0	0	0	0	0	3	0	0	1
b10	0	0	0	0	0	0	0	0	1	0

(e) Block alignment matrix

Figure 8. Example for block matrix representation.

4. Proposed Multi-Agent Reinforcement Learning Algorithm for Block Layout

Multi-agent reinforcement learning (MARL) integrates multi-agent system (MAS) with reinforcement learning (RL), whereby agents learn optimal decision-making strategies through continuous trial and error (select action) and the accumulation of maximum

rewards. A MAS consists of an environment and multiple decision-making agents, each of which can observe information about the environment and select actions to achieve their objectives. As illustrated in Figure 9, a MAS includes two or more agents, with each agent possessing its own objectives, action sets, and knowledge bases. During the interaction process, the agents perceive changes in the environment and continuously adjust their actions to achieve specific goals [35]. RL is a machine learning method designed to assist agents in selecting actions based on the current state of an environment, with the objective of maximizing the cumulative reward after a series of actions. RL problems are typically modeled as Markov Decision Processes (MDP) [36]. MDP serves as the foundation of RL theory, characterizing the states of the environment in which an agent selects actions. Each action taken by the agent leads to a transition to a subsequent state, with the environment providing corresponding rewards. An MDP primarily consists of four elements $M = \langle S,\ A,\ P,\ \mathcal{R} \rangle$, where S represents the set of available states perceived by the agent from the environment, A defines the set of actions available for the RL agent, P denotes the state transition probabilities of the agent, mapping state-action pairs to a probability distribution over the subsequent states, and $\mathcal{R}$ specifies the reward function for the agent, which indicates the rewards received from the environment.

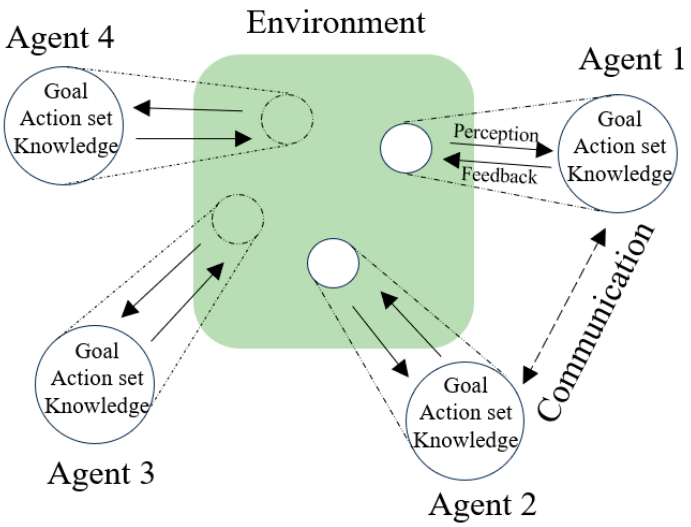


Figure 9. Multi-agent system.

In the block layout, the environment for the agent is structured as a 2D grid space. State S represents the current layout, including the positions and internal spacings of the FCBs. Action A contains adjustment in the position and internal spacing of the FCBs, and state transition P signifies the probability of layout changes resulting from each operation. Reward $\mathcal{R}$ is evaluated based on the readability of the drawing in the current state. The number of agents is related to the types and functions of the FCBs. The position of each FCB is regarded as an agent, while the internal spacing between blocks in the FCB is also considered as an agent. The environment evaluates the drawing's readability based on the current state. The action represents the changes in the positions and internal spacings along the x-axis and y-axis directions. Each decision aims to optimize the reward through the agents' actions, thereby guiding the overall layout towards improved readability.

4.1. Reward System

In an RL algorithm, rewards are critical for guiding agent learning and achieving a specific goal. The reward function evaluates the feedback provided by the environment following the agents' actions, guiding the agents to make a series of decisions within a specific environment to achieve the maximum cumulative reward. The quality of block

layout is primarily assessed through the drawing's readability, using metrics encompassing the number of collisions, target area attainment, alignment of edges, uniformity of edge distances, and overall fullness. Table 1 defines the reward functions and variables used in the reward calculation formula. The final reward value for each agent, denoted as R_{to} is defined as the sum of the rewards contributed by each evaluation metric. $R_c(n_c)$ represents the reward value for collisions between FCBs, where n_c is the number of collisions. $R_g(n_g)$ is the reward value for an agent reaching the target area, and n_g is the number of agents that reach the target region. For a single agent, n_g is typically 1 if the agent is not in the target area and 0 if it is. $R_f(s_a)$ denotes the reward value for the overall fullness of drawing, where s_a is the ratio of the total area occupied by the FCB to the total drawing area, s_l is the ratio of the actual area of all blocks to the total drawing area. $R_a(n_a)$ is the reward value for the alignment of edges between FCBs, and n_a is the number of alignments of edges between FCBs within the drawing space. $R_e(n_e)$ represents the reward corresponding to the uniformity of edge distances between FCB and the drawing edge, where n_e is the cumulative number of the alignment of edges. $R_v(n_v)$ is the reward value for the FCBs exceeding the drawing space, and n_v is the number of times beyond the drawing boundary. $R_l(n_{sl}, n_{vl})$ represents the reward designed to incentivize agent movement toward the target. Compared to the previous state, n_{sl} is the number of changes in a better direction, and n_{vl} is the number of changes in a worse direction. Figure 10 describes the basic reward functions.

Figure 10. Graphical representation of reward function.

Table 1. Reward computation formula.

Definition	Formula	Coefficient Value
Reward for block collision	$R_c(n_c) = c_1 \cdot n_c$ # (1)	$c_1 = -30$
Reward for reaching the goal region	$R_g(n_g) = c_2 \cdot n_g$ # (2)	$c_2 = -5$
Reward for the overall fullness	$R_f(s_a) = c_3 \cdot (s_a - s_l)$ # (3)	$c_3 = 80$
Reward for the alignment of edges	$R_a(n_a) = c_4 \cdot n_a$ # (4)	$c_4 = 20$
Reward for the uniformity of edge distances	$R_e(n_e) = c_5 \cdot n_e$ # (5)	$c_5 = 10$
Reward for exceeding drawing boundary	$R_v(n_v) = c_6 \cdot n_v$ # (6)	$c_6 = -30$
Reward for guiding learning	$R_l(n_{sl}, n_{vl}) = c_7 \cdot (n_{sl} - n_{vl})$ # (7)	$c_7 = 1$

To compute the final reward value of each agent, the corresponding reward computation formulas of each evaluation metric are presented in Table 1. The overall reward formula of each agent is described in Equation (8).

$$R_{to} = R_c(n_c) + R_g(n_g) + R_f(s_a) + R_a(n_a) + R_e(n_e) + R_v(n_v) + R_l(n_{sl}, n_{vl}) \quad (8)$$

4.2. Environment, States and Actions

As shown in Figure 11, the RL environment for the block layout problem is represented in a 2D drawing space, which is discretized into a planar grid. The functional requirement of the block layout problem is to find the optimal position and internal spacing of FCB within the drawing space. Since each FCB has a defined size, the point closest to the target area is selected as the reference point for the FCB's position. Due to the functional requirement of block layout, two types of agents are constructed: one agent is responsible for finding the optimal position of each FCB, while the other agent focuses on determining the optimal internal spacing among all FCBs. In the block layout, the action space A encompasses of the position and spacing adjustments of the agent within the two-dimensional grid space. The agent can perform actions $a \in \{s_l, s_r, s_u, s_d, s_0\}$, corresponding to movements of $(-1, 0)$, $(1, 0)$, $(0, 1)$, $(0, -1)$, and $(0, 0)$, respectively.

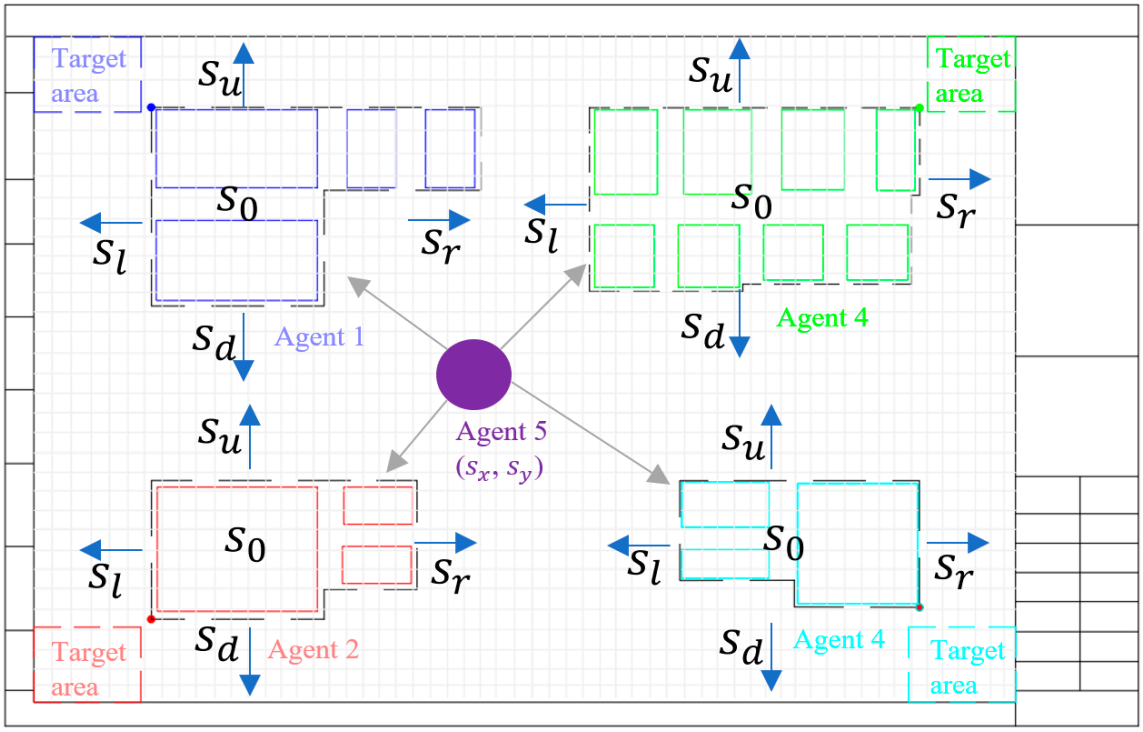


Figure 11. RL environment.

4.3. Multi-Agent Reinforcement Learning for Block

Currently, deep reinforcement learning (DRL) algorithms are utilized to solve the structure design problem, such as structure optimization [29], rebar layout [37]. While numerous DRL algorithms have been proposed, some classic algorithms like DQN [38], DDPG [39], and PPO [40] continue to be widely adopted in finding optimal strategies through interaction with environments. The deep Q-network (DQN) algorithm is a great advancement in reinforcement learning, integrating deep learning with Q-learning by employing a neural network as a function approximator for the Q-function. This approach is particularly effective in solving the discrete action space problem. The deep deterministic policy gradient (DDPG) algorithm is a deep reinforcement learning approach operating within the actor–critic framework. It utilizes two distinct neural networks: an actor network for policy representation and a critic network for action–value function approximation. These networks are updated independently according to their respective gradient-based update rules, with the overall objective of maximizing expected cumulative reward. The DDPG algorithm is more suitable for solving the continuous control and high-dimensional observation space problems. The proximal policy optimization (PPO) is a significant advancement in policy gradient methods, offering a more practical and efficient alternative to trust region policy optimization (TRPO). By constraining the policy updates within a defined trust region, the PPO algorithm mitigates the risk of performance collapse associated with large policy changes. This constraint is achieved through a novel surrogate objective function that effectively approximates the theoretically justified constraints of TRPO while maintaining computational tractability. The PPO algorithm exhibits strong stability and simplifies the process of hyperparameter adjustment. In summary, the DQN algorithm is a good choice for addressing the discrete action space problem due to its simplicity; the DDPG algorithm is more suitable for handling the continuous action space problem but can be challenging to adjust the hyperparameters and may require careful consideration of exploration strategies; the PPO algorithm is more appropriate for solving both continuous and discrete action space with good stability and sample efficiency, requiring the expensive computation cost per update. Therefore, the DQN algorithm is selected to solve the drawing block layout problem.

Figure 12 illustrates a block layout algorithm using the multi-agent deep Q-network (MADQN) algorithm. The first step is to obtain the layout structures and regions of each FCB from Section 3.4.1, and grid the two-dimensional drawing space. In the second step, the outer bounding boxes of FCB are computed, and one type of agent is utilized to find the optimal position of FCB (agent 1–4). In addition, the other type of agent (agent 5) is employed to dynamically adjust the spacing between blocks within each FCB. Therefore, the five agents work collaboratively to achieve the block layout. In the initial stage, the FCBs are randomly placed close to the target area and do not collide with each other. The state values of block spacing agents are set to 3 and 2. If the spacing value is set to 0, the blocks within each FCB will collide. Conversely, if the value is set to a larger value, the initial state of all FCBs in drawing space will result in collisions. Subsequently, each agent selects an action based on its strategy, and the environment provides a corresponding reward. The tuple $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{s_{t+1}}, \boldsymbol{r_t})$ in each history step is stored in the experience replay buffer for subsequent neural network training. The third step is in the training phase; a batch of experience is randomly sampled from the experience replay buffer to update the parameter ω of the Q-network. In the network update process, the gradient descent algorithm is used to reduce the loss value. After training N steps, the parameters of the Q-network are assigned to the target network to update its parameters. In the execution process, each agent follows an epsilon-greedy policy, selecting actions based on its current observation. With probability $1 - \varepsilon$, the trained network is used to determine the optimal action (exploitation);

otherwise, a random action is chosen (exploration). The agents collaborate throughout the process to complete the block layout task, receiving an identical reward.

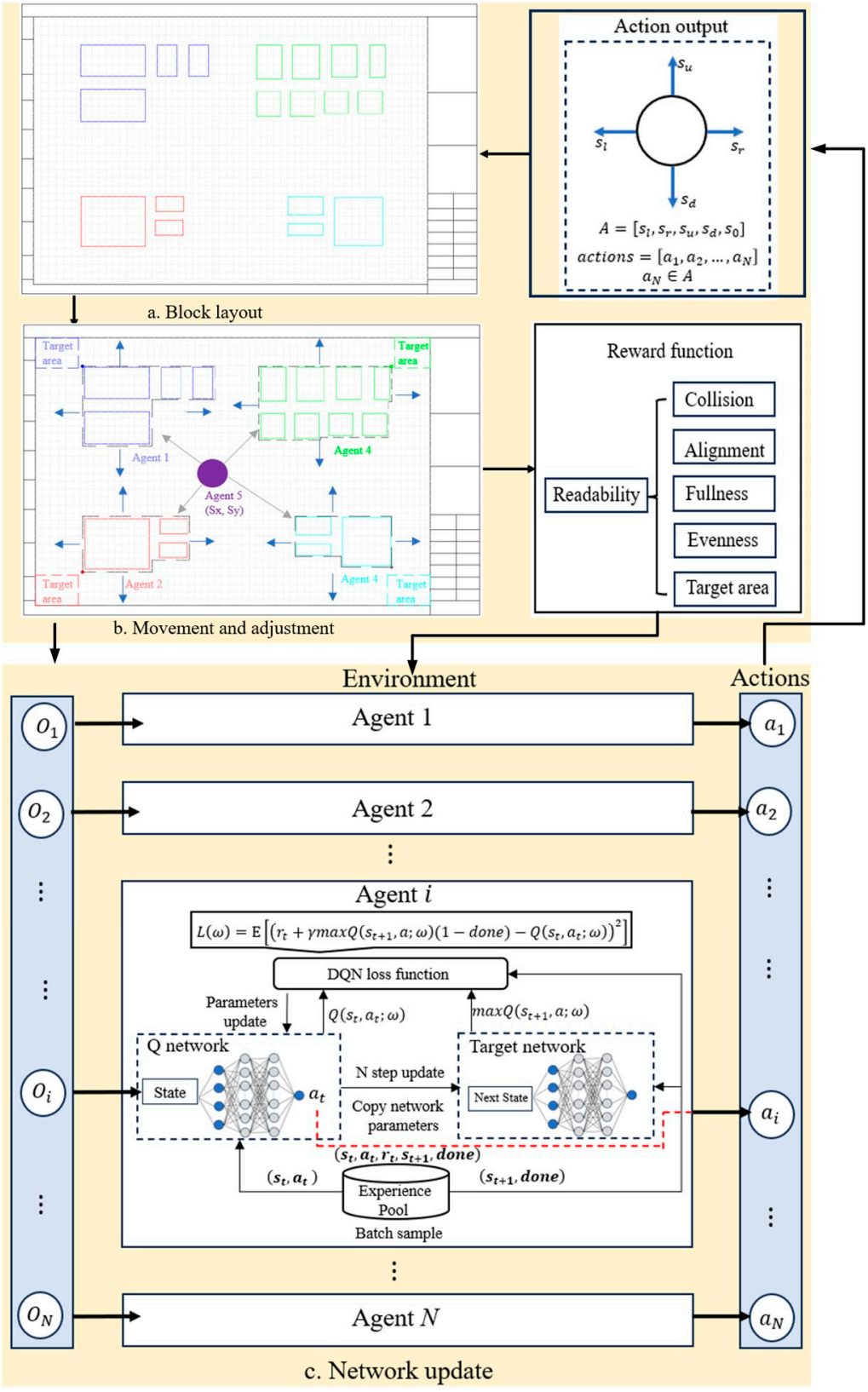


Figure 12. MADQN-based algorithm for block layout.

Algorithm 3 presents the pseudocode for MADQN-based block layout algorithm, block information, FCB layout structure and region, and drawing space are used as input. At each time step, the agents select actions concurrently and share information to enhance cooperation. The environment aggregates the individual rewards of each agent and divides the total by the number of agents to calculate the average reward. In each episode, the current state and corresponding reward are recorded at every time step. Upon completion of all episodes, the state that yields the maximum reward is identified as the optimal solution. Finally, the position of the corresponding block is determined based on this optimal state.

Algorithm 3: Pseudocode for MADQN-based block layout algorithm

Input: Block information; FCB layout structure and region; Drawing space
Output: Block position
Initialize replay buffer D to capacity P
for agent i = 1 to N **do**
    Initialize action-value function Q_i with random weight ω_i
    Initialize target action-value function $\hat{Q}_i$ with weight ${\omega_i}^- = \omega_i$
end for
for episode = 1 to M **do**
    Initialize state $\boldsymbol{s_t}$ for each agent
    for t = 1 to T **do**
        for agent i = 1 to N **do**
            With probability ε select a random action ${a_t}^i$
            Otherwise select ${a_t}^i = argmax_a Q_i\left({a_t}^i, a; \omega\right)$
        end for
        Execute action $\boldsymbol{a_t}$, observe next state $\boldsymbol{s_{t+1}}$, and receive reward $\boldsymbol{r_t}$
        Record $\boldsymbol{s_t}$ and $\boldsymbol{r_t}$
        Store transition $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{s_{t+1}}, \boldsymbol{r_t})$ in D
        Set $\boldsymbol{s_{t+1}} = \boldsymbol{s_t}$
        Sample random batch of transitions $\left(\boldsymbol{s_j}, \boldsymbol{a_j}, \boldsymbol{s_{j+1}}, \boldsymbol{r_j}\right)$ from D
        for agent i = 1 to N **do**
            Set $y_j = \begin{cases} r_j, & \text{if episode terminates at step } j+1 \\ r_j + \gamma max_{a'} \hat{Q}\left(s_{j+1}, a'; \omega^-\right), & \text{otherwise} \end{cases}$
            Perform a gradient descent step on $\left(y_j - Q\left(s_j, a_j; \omega\right)\right)^2$ with respect to the network parameter ω
            After C steps, update $\hat{Q}_i = Q_i$ for agent i
        end for
    end for
end for
Compute the maximum reward, and determine the optimal state $\boldsymbol{s_t}^*$
Update the block position

5. Illustrative Examples

In this section, three different MARL algorithms are employed to address the proposed intelligent view block layout problem, thereby demonstrating the feasibility and efficiency of MADQN. Moreover, two different PC components are chosen to validate the generalization ability of the proposed drawing generation framework.

5.1. Evaluative Metric

In this experiment, the feasibility and efficiency of the proposed MADQN are evaluated through comparison with MADDPG and MAPPO. Multiple agents operate in a fully cooperative setting, where all agents share the same reward and pursue a common goal. The Gumbel-Softmax method is utilized in MADDPG to solve the discrete action space problem [41]. To compare the performance of three MARL algorithms in block layout, the computation time and maximum reward are adopted as the evaluative metrics.

5.2. Experimental Config

To ensure the reproducibility and performance, the experiment was conducted on a personal computer. The machine is equipped with a 12th Gen Intel® Core™ i5-12600KF 3.70 GHz processor and has 32 GB of RAM. In the three MADRL algorithms, each network is a multi-layer perceptron, in which the number of hidden layers is set as one and the number of neurons in a single layer is 128. The maximum episode is 10,000, and a maximum step per episode of 30. The capacity of the experience replay buffer $|D|$ is equal to 10,000, and a discount factor γ is 0.99. The target network update frequency N is determined by dividing the total training steps by an update interval of 200. The Adam optimizer with a learning rate of 0.0015 is employed for training the network. The other hyperparameter settings are listed in Table 2.

Table 2. Parameter setting.

Algorithm	Hyperparameter
MADQN	Exploration rate linearly reduces from 0.8 to 0.01; The batch size is 512.
MADDPG	Soft target update coefficient τ is 0.001, The batch size is 512; The Gaussian noise standard deviation σ is 0.1.
MAPPO	GAE parameter λ is 0.95; The minimum batch size is 128; The maximum Batch size is 2048; The clipping parameter ϵ is 0.2; The No. epochs is 10.

5.3. Example 1—PC Concrete Stairs

5.3.1. Case Description

Figure 13 illustrates a BIM model of PC stairs, the geometric dimensions of which are listed in Table 3. The proposed framework can complete the automatic generation of 24 graphical blocks. Each block contains graphical elements and annotation data; however, the position for block placement within the layout space is not determined. Table 4 provides a comprehensive list of the dimensions, function, and name associated with each individual block. According to design specifications and engineering requirements, all blocks are classified into four categories: formwork drawing (m), reinforcement drawing (r), material list (t), and detailed drawing (d). Formwork drawing provides a visual representation of PC stair's geometry, including the quantity and position of embedded parts and reserved holes. Engineers can utilize this information to complete the design and manufacturing of the mold. Reinforcement drawing provides a detailed representation of the interconnectivity and relative positioning of the reinforcing bars, thereby facilitating the accurate assembly of the reinforcement cage. Material lists provide comprehensive information on the design specifications of individual reinforcing bars, including diameter, quantity, shape, dimensions, and steel grade. Additionally, these material lists also include concrete grade, quantities of materials such as rebar and concrete volume, and details on the configuration of embedded parts, such as their shape, dimensions, and quantity. Engineers can use these lists to procure all necessary materials and complete the cutting and fabrication of the rebar. Detailed drawings illustrate the connection openings and joints within the stairs

assembly, as well as the details of construction joints and precise dimensional information for embedded parts, guiding the PC stairs installation. According to the drawing requirements of PC components, each block in a single layout space maintains a consistent scale. Consequently, the dimensions of all blocks remain constant in the layout process. In this paper, we select drawing space A1 (841 mm × 594 mm size, 1:20 scale) with margins of 36 mm (left), 113.45 mm (right), and 14.15 mm (top/bottom), yielding a usable area of 691.55 mm × 565.7 mm. The block layout environment consists of five agents: four agents are assigned to control the positions of blocks, while the fifth agent represents the internal spacing between these blocks within FCBs. The layout structures for each FCB are determined based on the design specifications and the methodology detailed in Section 3.4.1, as shown in Figure 14. In addition, the layout region for each FCB is determined according to principle 2. Three different MARL algorithms are employed to conduct the block layout optimization task; the experimental results obtained from these optimization procedures are presented and discussed in Section 5.3.2.

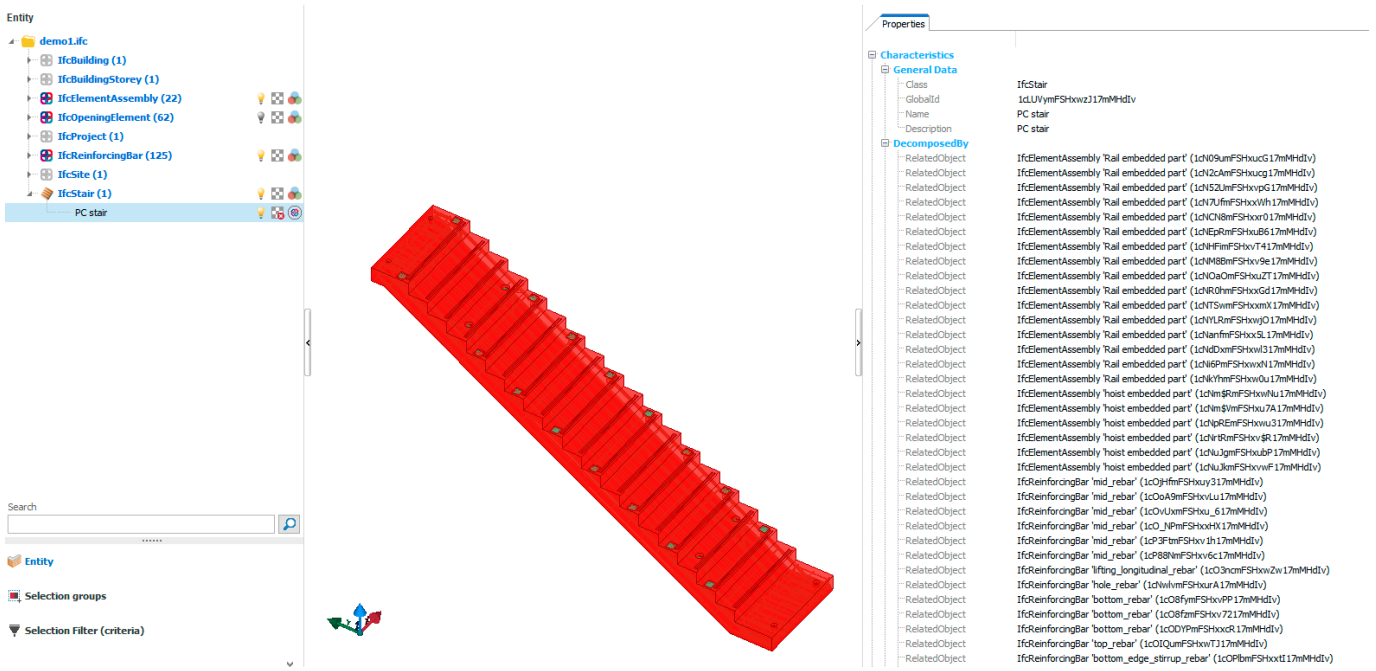


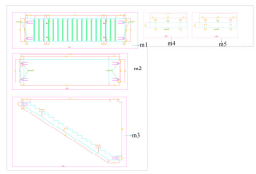
Figure 13. BIM model of PC stairs.

Table 3. Geometrical dimensions of PC stairs (unit: mm).

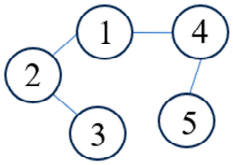
Length	Width	Height	lb	lt	hb	ht	t	n
5420	1280	3000	500	500	200	200	210	18

Table 4. Information of drawing blocks for PC stair (unit: mm).

Name	Size (w×h)	Type	Name	Size (w×h)	Type	Name	Size (w×h)	Type
m1	6848 × 1979	m	m2	6334 × 1979	m	m3	6298 × 3779	m
m4	1564 × 907	m	m5	1588 × 893	m	r1	6389 × 3636	r
r2	1733 × 808	r	r3	1741 × 802	r	r4	1767 × 803	r
t1	2300 × 3355	t	t2	2350 × 750	t	t3	1500 × 1000	t
t4	1000 × 600	t	d1	1440 × 1110	d	d2	1500 × 1114	d
d3	906 × 1121	d	d4	1065 × 1130	d	d5	509 × 1000	d
d6	672 × 1002	d	d7	1130 × 1081	d	d8	737 × 1050	d
d9	481 × 582	d	d10	399 × 519	d	d11	532 × 589	d



(1) FCB layout structure of formwork drawing



(2) Connectivity graph

	m1	m2	m3	m4	m5
m1	0	1	0	1	0
m2	1	0	1	0	0
m3	0	1	0	0	0
m4	1	0	0	0	1
m5	0	0	0	1	0

(3) Block connection matrix

	m1	m2	m3	m4	m5
m1	0	−2	0	1	0
m2	2	0	−2	0	0
m3	0	2	0	0	0
m4	−1	0	0	0	1
m5	0	0	0	−1	0

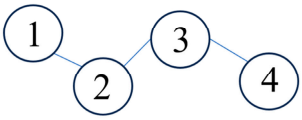
(4) Block position matrix

	m1	m2	m3	m4	m5
m1	0	3	0	1	0
m2	3	0	3	0	0
m3	0	3	0	0	0
m4	1	0	0	0	1
m5	0	0	0	1	0

(5) Block alignment matrix

(a)

(1) FCB layout structure of material list



(2) Connectivity graph

	t1	t2	t3	t4
t1	0	1	0	0
t2	1	0	1	0
t3	0	1	0	1
t4	0	0	1	0

(3) Block connection matrix

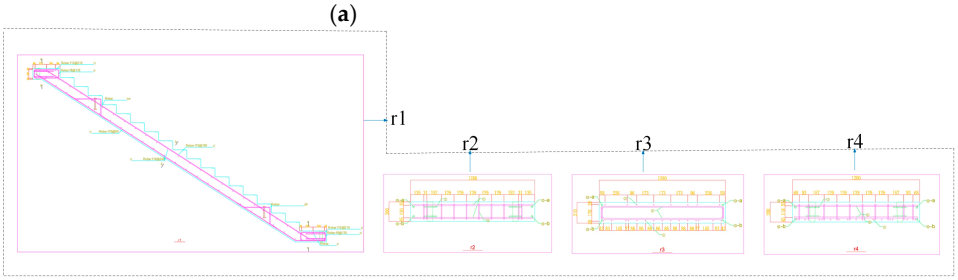
	t1	t2	t3	t4
t1	0	1	0	0
t2	−1	0	−2	0
t3	0	2	0	−2
t4	0	0	2	0

(4) Block position matrix

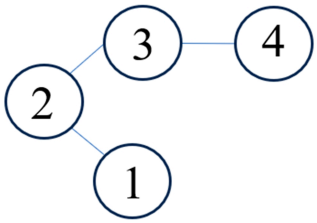
	t1	t2	t3	t4
t1	0	1	0	0
t2	1	0	3	0
t3	0	3	0	3
t4	0	0	3	0

(5) Block alignment matrix

(b)



(1) FCB layout structure of reinforcement drawing



(2) Connectivity graph

	r1	r2	r3	r4
r1	0	1	0	0
r2	1	0	1	0
r3	0	1	0	1
r4	0	0	1	0

(3) Block connection matrix

	r1	r2	r3	r4
r1	0	1	0	0
r2	−1	0	1	0
r3	0	−1	0	1
r4	0	0	−1	0

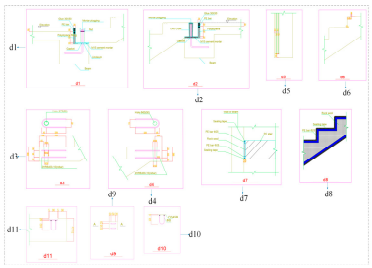
(4) Block position matrix

	r1	r2	r3	r4
r1	0	2	0	0
r2	2	0	2	0
r3	0	2	0	2
r4	0	0	2	0

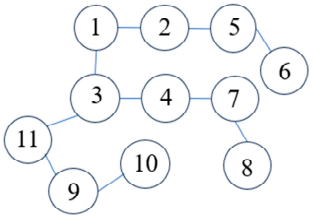
(5) Block alignment matrix

(c)

Figure 14. *Cont.*



(1) FCB layout structure of detailed drawing



(2) Connectivity graph

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
d1	0	1	1	0	0	0	0	0	0	0	0
d2	1	0	0	0	1	0	0	0	0	0	0
d3	1	0	0	1	0	0	0	0	0	0	1
d4	0	0	1	0	0	0	1	0	0	0	0
d5	0	1	0	0	0	1	0	0	0	0	0
d6	0	0	0	0	1	0	0	0	0	0	0
d7	0	0	0	1	0	0	0	1	0	0	0
d8	0	0	0	0	0	0	1	0	0	0	0
d9	0	0	0	0	0	0	0	0	0	1	1
d10	0	0	0	0	0	0	0	0	1	0	0
d11	0	0	1	0	0	0	0	0	1	0	0

(3) Block connection matrix

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
d1	0	1	−2	0	0	0	0	0	0	0	0
d2	−1	0	0	0	1	0	0	0	0	0	0
d3	2	0	0	1	0	0	0	0	0	0	0
d4	0	0	−1	0	0	0	1	0	0	0	0
d5	0	−1	0	0	0	1	0	0	0	0	0
d6	0	0	0	0	−1	0	0	0	0	0	0
d7	0	0	0	−1	0	0	0	1	0	0	0
d8	0	0	0	0	0	0	−1	0	0	0	0
d9	0	0	0	0	0	0	0	0	0	1	−1
d10	0	0	0	0	0	0	0	0	−1	0	0
d11	0	0	2	0	0	0	0	0	1	0	0

(4) Block position matrix

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11
d1	0	1	3	0	0	0	0	0	0	0	0
d2	1	0	0	0	1	0	0	0	0	0	0
d3	3	0	0	1	0	0	0	0	0	0	0
d4	0	0	1	0	0	0	1	0	0	0	0
d5	0	1	0	0	0	1	0	0	0	0	0
d6	0	0	0	0	1	0	0	0	0	0	0
d7	0	0	0	1	0	0	0	1	0	0	0
d8	0	0	0	0	0	0	1	0	0	0	0
d9	0	0	0	0	0	0	0	0	0	1	1
d10	0	0	0	0	0	0	0	0	1	0	0
d11	0	0	3	0	0	0	0	0	1	0	0

(5) Block alignment matrix

(**d**)

Figure 14. Layout structure of different FCBs for PC stairs: (**a**) formwork drawing; (**b**) material list; (**c**) reinforcement drawing; (**d**) detailed drawing.

5.3.2. Experimental Analysis

Under the fully cooperative paradigm adopted by the multi-agent system, all agents demonstrate congruent learning curves. Figure 15 depicts the reward curves of different algorithms. The x-axis represents the training episode, and the y-axis represents the average reward per step. The reward is calculated over a 20-episode sliding window and further averaged over 10 independent runs. Shaded regions indicate the 95% confidence interval. Figure 15 illustrates that the reward values for the three MARL algorithms gradually increase towards 0 as the number of training episodes increases, suggesting that the agents learn to find the layout strategy. In the MADQN algorithm, the reward increases rapidly with significant fluctuations during the initial training phase. As it approaches 8000 episodes, the reward increases slowly and gradually stabilizes. The MADDPG algorithm exhibits the fastest initial growth in the learning curve, followed by substantial fluctuations in the intermediate phase and eventual convergence in the later stages. For the MAPPO algorithm, the reward increases slowly in the initial training phase, then rapidly ascends around 3000 episodes and begins to converge after 8000 episodes, exhibiting small fluctuations in the later stages. Therefore, the MADQN and MADDPG algorithms demonstrate rapid early learning, and their training processes are unstable overall. Although the initial performance of the MAPPO algorithm is less impressive, it exhibits a rapid increase in the intermediate phase and maintains a more stable learning curve throughout the training process.

Figure 16a depicts the computation time of the three MARL algorithms, averaged over 10 independent runs. MADQN demonstrates the lowest time, followed by MAPPO, with the MADDPG algorithm incurring the highest time. Moreover, MADQN exhibits minimum variance in computation time, while both MAPPO and MADDPG demonstrate substantially greater variance. Figure 16b shows the maximum reward (optimal joint state) achieved by each algorithm over 10 independent runs, and the optimal reward is 54.7. MADQN finds this optimal solution in 90% of the runs, MAPPO in 50%, and MADDPG in 0%.

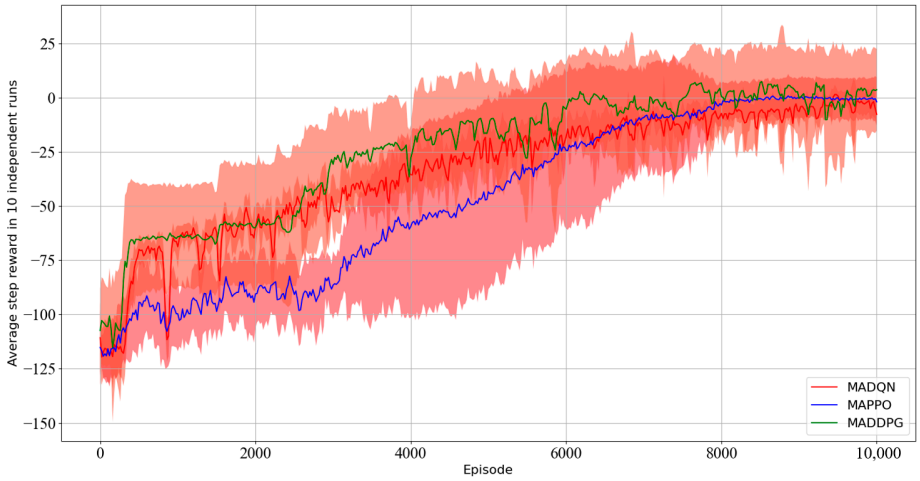
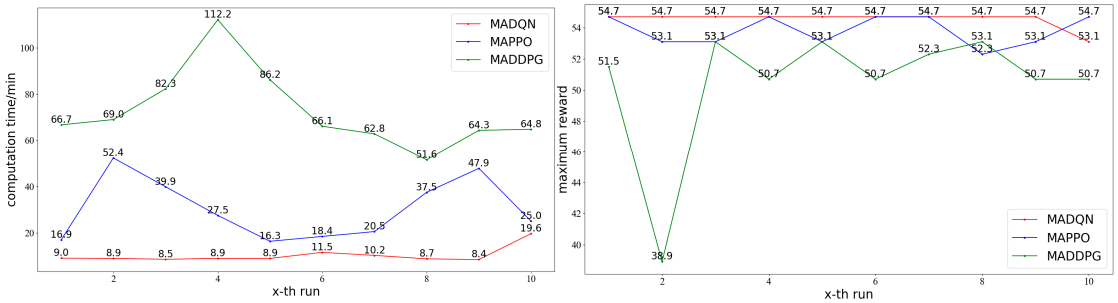


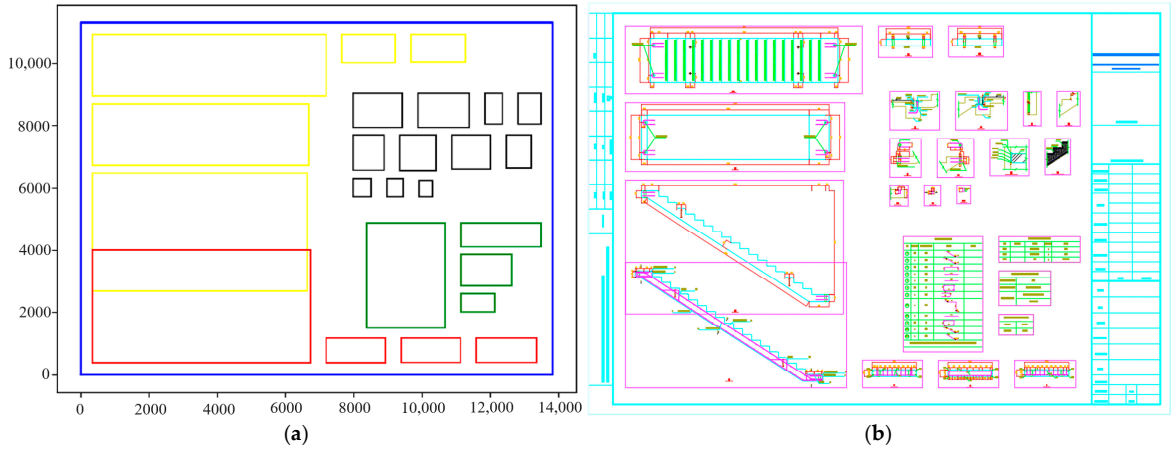
Figure 15. Reward curves of different MARL algorithms.



(**a**) (**b**)

Figure 16. Performance curves for each algorithm across 10 independent runs: (**a**) computation time; (**b**) maximum reward.

In conclusion, the MADQN algorithm is more suitable for solving block layout optimization problems in terms of computation efficiency and quality. The block layout results for the fabrication drawing of PC stairs are shown in Figure 17. The optimal positions of the four FCBs (m, r, d, and t) are respectively (345.77, 10,936.77), (345.77, 461.03), (13,485.03, 9051.12), and (13,485.03, 1508.52), and the optimal internal spacing between blocks is (461.03, 251.42). As depicted in Figure 17, the right-side edges of different FCBs (r, d, and t) in the drawing are aligned, and the contour area of FCBs occupy a larger drawing space, showing the good readability of layout result.



(**a**) (**b**)

Figure 17. Block layout result for the fabrication drawing of PC stairs: (**a**) bounding box layout; (**b**) block layout.

5.4. Example 2—PC Doubled-Sided Shear Wall

In this section, a PC double-sided shear wall is selected as an illustrative example to validate the generalizability of the proposed framework. A BIM model of a shear wall is shown in Figure 18, and its geometric dimensions are listed in Table 5. A total of 12 blocks are generated automatically through the implementation of the proposed framework. Details regarding the dimensions and categories of each block are summarized in Table 6. Considering the relatively low number of blocks necessary for the representation of the shear wall, an A3 drawing space extended by 1/4 along its width ((420 + 105) mm × 297 mm size; 1:20 scale) is selected. Margins of 10.5 mm are applied to the top-side and right-side edges, 40.4 mm to the bottom-side edge, and 25.5 mm to the left-side edge, resulting in a usable drawing space of 489 mm × 246.1 mm. The layout structure of different FCBs is illustrated in Figure 19. The specific regions allocated to each FCB are determined based on principle 2 of readability.

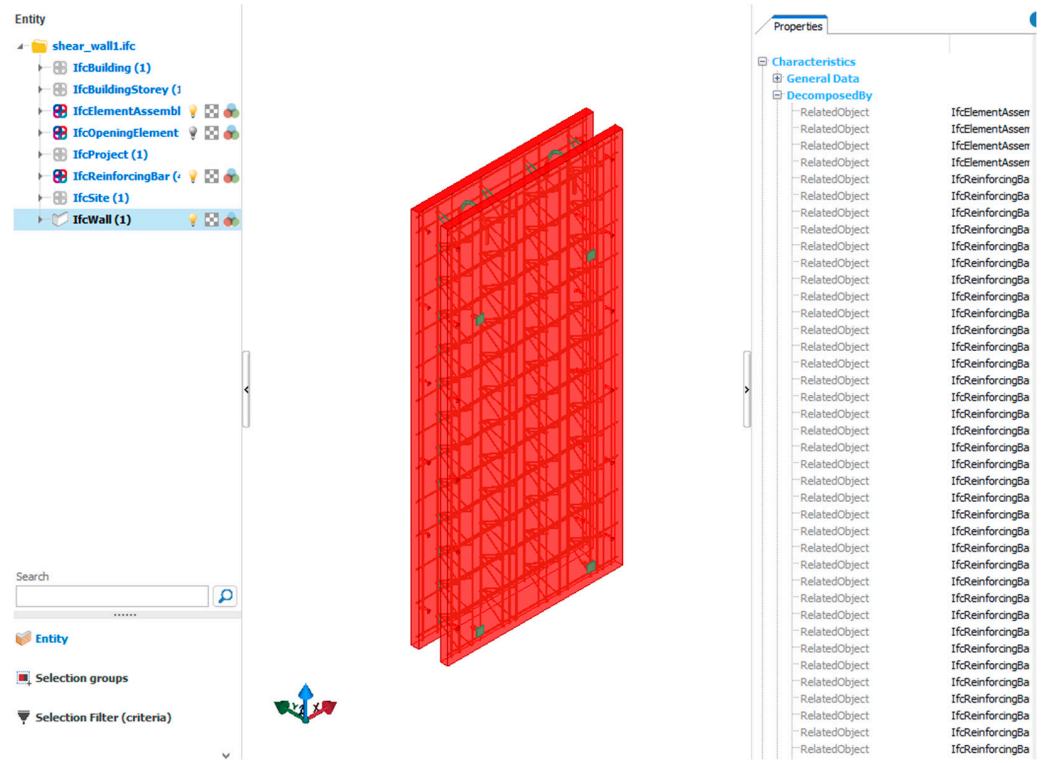


Figure 18. BIM model of PC double-sided shear wall.

Table 5. Geometrical dimensions of PC double-sided shear wall (unit: mm).

w_i	w_o	h_i	h_o	t_s	t_i	t_o
1200	1200	2590	2590	250	50	50

Table 6. Information of drawing blocks for PC double-sided shear wall (unit: mm).

Name	Size (w×h)	Type	Name	Size (w × h)	Type	Name	Size (w × h)	Type
m1	1720 × 3404	m	m2	1568 × 695	m	m3	535 × 3098	m
r1	1752 × 3404	r	r2	1466 × 862	r	r3	671 × 3253	r
d1	560 × 1033	d	d2	847 × 674	d	d3	817 × 636	d
t1	2537 × 1025	t	t2	2253 × 750	t	t3	1154 × 280	t

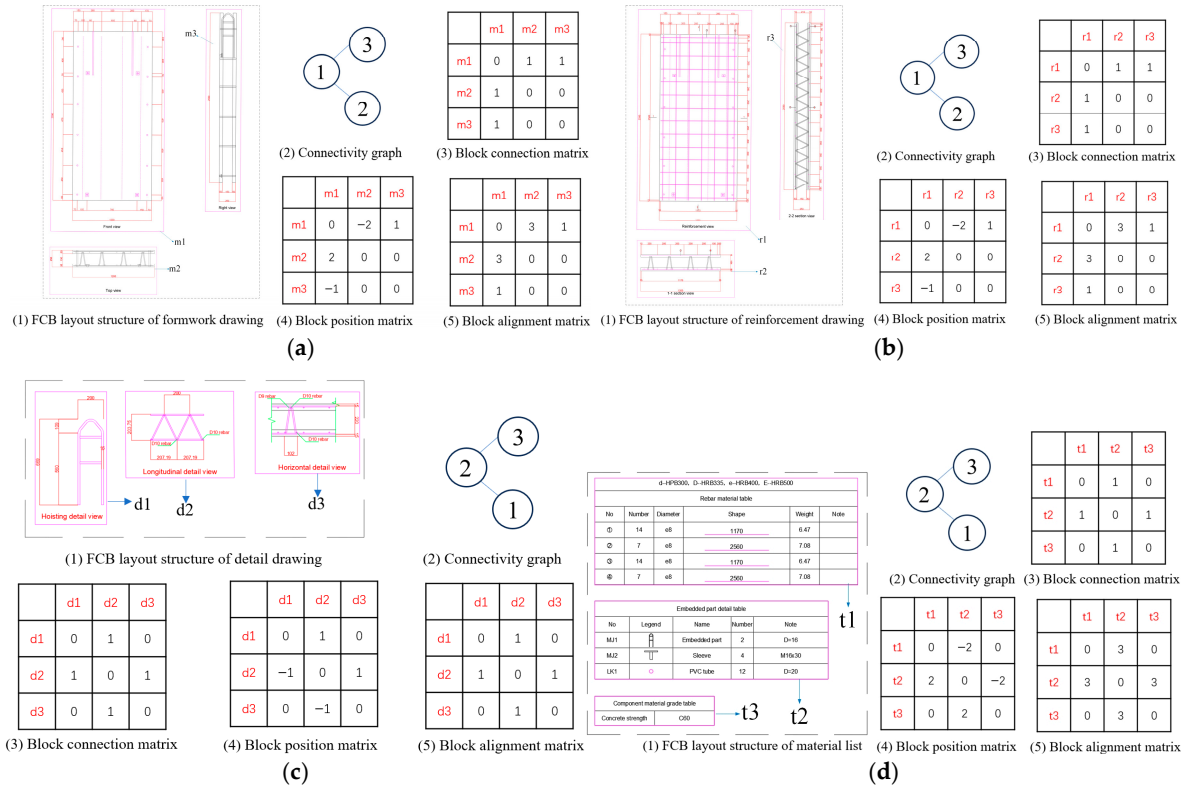


Figure 19. Layout structure of different FCBs for PC double-sided shear wall: (**a**) formwork drawing; (**b**) reinforcement drawing; (**c**) detailed drawing; (**d**) material list.

Figure 20 depicts the learning curve obtained after training the agent for 10,000 episodes. The reward exhibits a rapid initial increase, followed by convergence towards a maximum value of approximately 20 around episode 6000. This indicates that the agent successfully learns the block layout strategy within the environment. The average computation time of 10 independent runs is 12.1 min, and the block layout result for the fabrication drawing of the PC double-sided shear wall is shown in Figure 21. The optimal positions of four FCBs (m, r, d, and t) are, respectively, (244.5, 4758.03), (3423.0, 4758.03), (9535.5, 4758.03), and (9535.5, 164.07), and the optimal internal spacing between blocks is (407.5, 273.45). As shown in Figure 21, the top-side and right-side edges of the blocks in the drawing are aligned, and the area of FCBs occupies a larger drawing space, indicating the good readability of the layout result.

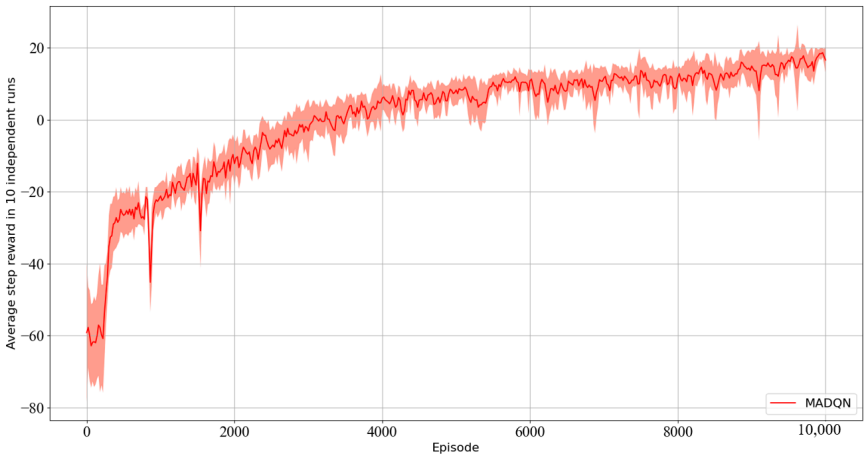


Figure 20. Learning curves of MADQN averaged in 10 independent runs.

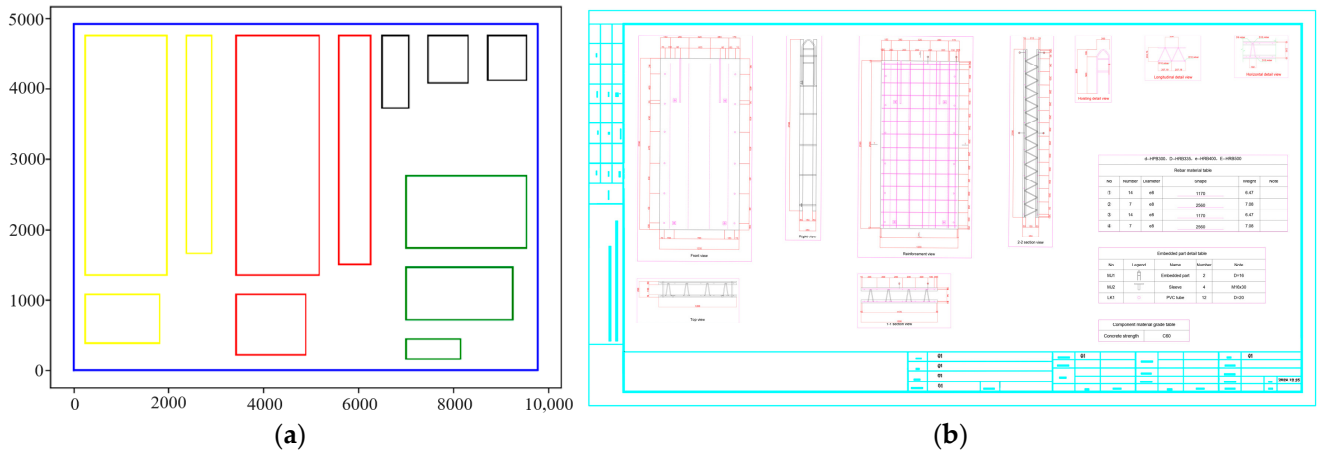


Figure 21. Block layout result for the fabrication drawing of PC double-sided shear wall: (**a**) bounding box layout; (**b**) block layout.

6. Conclusions

This paper proposed a novel framework for automating the generation of PC component fabrication drawings using BIM and multi-agent reinforcement learning. The developed framework is particularly well suited for PC components characterized by many view blocks and significant variation in dimensions. In this research, the digital model is converted to a topological model based on the IFC standard and Open CASCADE geometric kernel. Then, plane data are automatically generated using the computer graphic methods, including projection, cutting, and geometric algorithms. Finally, a specific method is employed to create the graphical elements and annotations, thereby achieving the transformation from a specific 3D model to its corresponding 2D view blocks. To improve the readability of drawing, a reinforcement learning method is adopted to find the optimal layout of blocks in the drawing space. The relationships between blocks, including connectivity, relative positioning, and alignment, are formally represented using graph structures, which serve as the foundation for determining the relative position of blocks within each FCB. To efficiently perform the layout optimization task, two types of agents are designed. One agent learns to find the optimal position of FCB, and the other agent learns to find the optimal internal spacing of FCB. A reward mechanism related to the readability criterion is designed to guide the agents to cooperate and interact with the environment, ultimately finding the layout strategy.

Two different MARL algorithms (MADDPG and MAPPO) are employed to validate the feasibility and efficiency of the proposed block layout method. PC stairs and double-sided shear walls are selected as illustrative cases to demonstrate the practicability and generality of the proposed framework. Based on the results obtained from these experiments, the following conclusions can be drawn:

- Three algorithms (MADQN, MAPPO, and MADDPG) can solve the block layout optimization problem and find the layout solution. Compared with the MAPPO and MADDPG algorithms, the proposed method (MADQN) demonstrates superior performance in terms of computational efficiency and solution quality.
- A graph-based representation method is utilized to encode the relationship between blocks. This approach precisely captures inter-block connectivity, relative positioning, and alignment.
- The proposed BIM-based framework rapidly completes fabrication drawings of PC stairs and double-sided shear walls, requiring only approximately 60 s.

This study offers valuable insights into the fabrication drawing generation within the context of intelligent construction. However, a rule and experience-based method in this research is difficult to determine the number of necessary views when the PC component is extremely complicated, such as precast bay window and exterior wall panel. Furthermore, the annotated style, position, and other information of different graphical elements need to be automatically determined in the annotation process. Intelligent generation of annotations based on graph neural networks can be studied in the future. In addition, current research utilizes an engineer's experience to determine the FCB layout structure and region, ignoring the potential optimal layout solution. Future research should aim to expand the scope of investigation by combining graph neural networks with multi-agent reinforcement learning to complete the block layout optimization.

Author Contributions: C.Z.: conceptualization, methodology, visualization, validation, and writing—original draft; X.Z.: supervision and funding acquisition; C.X.: conceptualization, writing—review and editing, and funding acquisition; Z.W.: conceptualization, methodology, and supervision; J.L.: conceptualization, supervision, and resources; H.Q.: project administration and resources. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by the National Natural Science Foundation of China (Project Nos. 52130801 and 52208219), to which the authors are very grateful.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Acknowledgments: The authors would like to thank the reviewers for their constructive comments that improved the presentation as well as the content of the paper.

Conflicts of Interest: We declare that we have no financial or personal relationships with other people or organizations that can inappropriately influence our work, and there are no professional or other personal interests of any nature or kind in any product, service, and/or company that could be construed as influencing the position presented or the review of the manuscript.

References

1. Muñoz-La Rivera, F.; Mora-Serrano, J.; Valero, I.; Oñate, E. Methodological-Technological Framework for Construction 4.0. *Arch. Comput. Methods Eng.* **2020**, *28*, 689–711. [CrossRef]
2. Alwisy, A.; Al-Hussein, M.; Al-Jibouri, S.H. BIM Approach for Automated Drafting and Design for Modular Construction Manufacturing. In Proceedings of the Computing in Civil Engineering: American Society of Civil Engineers, Clearwater Beach, FL, USA, 11 June 2012; pp. 221–228. [CrossRef]
3. Wang, X.; Tang, Q. Research on Detailed Design Methods of Precast Components in Prefabricated Buildings Based on BIM Technology. *Heliyon* **2024**, *10*, e35922. [CrossRef] [PubMed]
4. OpenCascade. Available online: https://www.opencascade.com/ (accessed on 14 October 2024).
5. Nayak, H.B.; Trivedi, R.R.; Araniya, K.K. Drawing automation of reactor nozzle. *Int. J. Eng. Res. Appl.* **2012**, *2*, 281–286. Available online: https://api.semanticscholar.org/CorpusID:15714329 (accessed on 18 October 2024).
6. Chen, K.-Z.; Feng, X.-A.; Ding, L. Intelligent Approaches for Generating Assembly Drawings from 3-D Computer Models of Mechanical Products. *Comput. Aided Des.* **2002**, *34*, 347–355. [CrossRef]
7. Shah, D.B. Parametric modeling and drawing automation for flange coupling using excel spreadsheet. *Int. J. Res. Eng. Technol.* **2013**, *1*, 187–192. Available online: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1b1b9f04c52e0c91324ce58462b25fd55382d39d (accessed on 15 October 2024).
8. FreeCAD. Available online: https://www.freecad.org/ (accessed on 15 October 2024).
9. Solidworks 2019. Available online: https://www.solidworks.com/ (accessed on 14 October 2024).
10. Nath, T.; Attarzadeh, M.; Tiong, R.L.K.; Chidambaram, C.; Yu, Z. Productivity Improvement of Precast Shop Drawings Generation through BIM-Based Process Re-Engineering. *Autom. Constr.* **2015**, *54*, 54–68. [CrossRef]
11. Liu, H.; Singh, G.; Lu, M.; Bouferguene, A.; Al-Hussein, M. BIM-based automated design and planning for boarding of light-frame residential buildings. *Autom. Constr.* **2018**, *89*, 235–249. [CrossRef]
12. Alwisy, A.; Hamdan, S.B.; Barkokebas, B.; Bouferguene, A.; Al-Hussein, M. A BIM-based automation of design and drafting for manufacturing of wood panels for modular residential buildings. *Int. J. Constr. Manag.* **2019**, *19*, 187–205. [CrossRef]

13. Haghir, S.; Haghnazar, R.; Moghaddam, S.S.; Keramat, D.; Matini, M.R.; Taghizade, K. BIM based decision-support tool for automating design to fabrication process of freeform lattice space structure. *Int. J. Space Struct.* **2021**, *36*, 164–179. [CrossRef]
14. Gankhuyag, U.; Han, J.-H. Automatic 2D Floorplan CAD Generation from 3D Point Clouds. *Appl. Sci.* **2020**, *10*, 2817. [CrossRef]
15. Kong, Q.; Liao, L.; Yuan, C. Rapid Generation of Editable Engineering Drawings from 3D Point Cloud Reconstruction for Large-Scale Buildings. *J. Build. Eng.* **2023**, *63*, 105486. [CrossRef]
16. Sacks, R.; Eastman, C.M.; Lee, G.; Orndorff, D. A Target Benchmark of the Impact of Three-Dimensional Parametric Modeling in Precast Construction. *Pcij* **2005**, *50*, 126–139. [CrossRef]
17. Scheibel, B.; Mangler, J.; Rinderle-Ma, S. Extraction of Dimension Requirements from Engineering Drawings for Supporting Quality Control in Production Processes. *Comput. Ind.* **2021**, *129*, 103442. [CrossRef]
18. Cheng, Y.; Ni, Z.; Liu, T.; Liu, X. An Intelligent Approach for Dimensioning Completeness Inspection in 3D Based on Transient Geometric Elements. *Comput. Aided Des.* **2014**, *53*, 14–27. [CrossRef]
19. Serrano, D. Automatic dimensioning in design for manufacturing. In Proceedings of the First ACM Symposium on Solid Modeling Foundations and CAD/CAM Application, Austin, TX, USA, 5–7 June 1991; pp. 379–386. [CrossRef]
20. Yu, K.M.; Tan, S.T.; Yuen, M.F. A Review of Automatic Dimensioning and Tolerancing Schemes. *Eng. Comput.* **1994**, *10*, 63–80. [CrossRef]
21. Hua, F.; Kui, Z.Q. Study on Visual Expression of CAD Interior Design Drawing. In Proceedings of the 6th International Conference on Intelligent Systems Design and Engineering Applications (ISDEA), Guiyang, China, 18–19 August 2015; pp. 782–785. [CrossRef]
22. Yuen, M.M.F.; Tan, S.T.; Yu, K.M. Scheme for Automatic Dimensioning of CSG Defined Parts. *Comput. Aided Des.* **1988**, *20*, 151–159. [CrossRef]
23. Deng, M.; Gan, V.J.L.; Tan, Y.; Joneja, A.; Cheng, J.C.P. Automatic generation of fabrication drawings for façade mullions and transoms through BIM models. *Adv. Eng. Inf.* **2019**, *42*, 100964. [CrossRef]
24. Deng, M.; Tan, Y.; Singh, J.; Joneja, A.; Cheng, J.C.P. A BIM-Based Framework for Automated Generation of Fabrication Drawings for Façade Panels. *Comput. Ind.* **2021**, *126*, 103395. [CrossRef]
25. Littman, M.L. Reinforcement Learning Improves Behaviour from Evaluative Feedback. *Nature* **2015**, *521*, 445–451. [CrossRef] [PubMed]
26. Averbeck, B.B.; Costa, V.D. Motivational Neural Circuits Underlying Reinforcement Learning. *Nat. Neurosci.* **2017**, *20*, 505–512. [CrossRef]
27. Soman, R.K.; Molina-Solana, M. Automating look-ahead schedule generation for construction using linked-data based constraint checking and reinforcement learning. *Autom. Constr.* **2022**, *134*, 104069. [CrossRef]
28. Yao, Y.; Tam, V.W.Y.; Wang, J.; Le, K.N.; Butera, A. Automated construction scheduling using deep reinforcement learning with valid action sampling. *Autom. Constr.* **2024**, *166*, 105622. [CrossRef]
29. Jeong, J.; Jo, H. Deep Reinforcement Learning for Automated Design of Reinforced Concrete Structures. *Comput. Aided Civ. Infrastruct. Eng.* **2021**, *36*, 1508–1529. [CrossRef]
30. Fu, B.; Gao, Y.; Wang, W. A Physics-informed Deep Reinforcement Learning Framework for Autonomous Steel Frame Structure Design. *Comput. Aided Civ. Infrastruct. Eng.* **2024**, *39*, 3125–3144. [CrossRef]
31. Pan, Y.; Shen, Y.; Qin, J.; Zhang, L. Deep reinforcement learning for multi-objective optimization in BIM-based green building design. *Autom. Constr.* **2024**, *166*, 105598. [CrossRef]
32. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Curran Associates Inc., Long Beach, CA, USA, 4–9 December 2017; pp. 6382–6393. [CrossRef]
33. Zhang, Z.; Guo, Z.; Zheng, H.; Li, Z.; Yuan, P.F. Automated architectural spatial composition via multi-agent deep reinforcement learning for building renovation. *Autom. Constr.* **2024**, *167*, 105702. [CrossRef]
34. Fujita, T. Superhypergraph neural networks and plithogenic graph neural networks: Theoretical foundations. *arXiv* **2024**, arXiv:2412.01176. [CrossRef]
35. Albrecht, S.V.; Christianos, F.; Schäfer, L. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*; MIT Press: Cambridge, MA, USA, 2024; Available online: https://www.marl-book.com (accessed on 15 December 2024).
36. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
37. Liu, P.; Qi, H.; Liu, J.; Feng, L.; Li, D.; Guo, J. Automated clash resolution for reinforcement steel design in precast concrete wall panels via generative adversarial network and reinforcement learning. *Adv. Eng. Inf.* **2023**, *58*, 102131. [CrossRef]
38. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602. [CrossRef]
39. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2019**, arXiv:1509.02971. [CrossRef]

40. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
41. Tibury, C.R.; Christianos, F.; Albrecht, S.V. Revisiting the Gumbel-Softmax in MADDPG. *arXiv* **2023**, arXiv:2302.11793. [CrossRef]