Hindawi
Advances in Civil Engineering
Volume 2020, Article ID 8782740, 15 pages
https://doi.org/10.1155/2020/8782740

Research Article

A Graph-Based Method for IFC Data Merging

Qin Zhao,[1] **Yuchao Li**,[1] **Xinhong Hei**,[2] **and Mingsong Yang**[2]

[1]*School of Civil Engineering and Architecture, Xi'an University of Technology, Xi'an 710048, China*
[2]*School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China*

Correspondence should be addressed to Xinhong Hei; heixinhong@xaut.edu.cn

Received 7 August 2019; Revised 7 July 2020; Accepted 8 July 2020; Published 26 July 2020

Academic Editor: Venu G. M. Annamdas

Collaborative work in the construction industry has always been one of the problems solved by BIM (Building Information Modeling) technology. The integration of IFC (Industry Foundation Classes) data as a general building information standard is one of the indispensable functions in collaborative work. The most practical approach of merging IFC data depends on GUID (Global Universal Identifier) comparison at present. However, GUID is not stable in current applications and often changes when exported. The intact representation of relationships between IFC entities is an essential prerequisite for proper association of IFC entities in IFC mergence. This paper proposes a graph-based method for IFC data merging. The IFC data are represented as a graphical data structure, which completely preserves the relationship between IFC entities. IFC mergence is accomplished by associating other data with an isomorphic graph that is obtained by mining the IFC graph. The feasibility of the method is proven by a program, and the method can ignore the impacts of GUID and other factors.

1. Introduction

Modern construction engineering often involves many specialties, and the construction process is complex [1, 2]. Different participants need to exchange information during the construction period, and a large amount of data collaborative work is required. Information exchange in collaborative work depends on heterogeneous and domain-specific data formats [3], and different forms of drawings and documents are used in the conventional approach. The collaborative process requires manual data extraction and transformation, which causes poor efficiency.

With the development of BIM (Building Information Modeling), many researchers in the field of construction engineering utilize BIM to solve the problems in construction collaboration. It is a common method by which construction engineering participants acquire and store information through a BIM server or management system [4, 5]. A BIM server manages building information through uniform data format, and different participants share information through an integrated database of information of different specialties. After IFC (Industry Foundation Classes) are published by buildingSMART in order to provide a collaborative environment for different participants [6, 7], the BIM server usually uses IFC as data standard. IFC enable representation and exchange of a wide variety of project relevant information in a single data format, and MVD, as a subset of IFC, facilitates obtaining demand information for participants by defining templates for exchange information [8]. When a BIM server receives information from different specialties, it needs to integrate information to the complete data, and so integrating IFC data has become one of the indispensable functions of BIM servers [9]. A common way to integrate IFC data is to distinguish the same data by the attributes of GUID (Global Universal Identifier) in IFC entities and to merge the entities with different GUID attributes [10]. GUID is a globally unique ID that is specially used to identify a component or IFC entity. However, the current IFC files are not created directly but are exported by other BIM software, which results in different GUIDs of the same component being exported from different tools [11]. It is easy to make mistakes when integrating the IFC entity by GUID and entities in the IFC model which are related to each other and their relationships are used to determine the same data in IFC mergence, which are not considered in existing tools and collaborative servers. In addition, there are

some tools that only merge the building elements on the view [12], and the data are not merged.

Therefore, a graph-based IFC data merging method is proposed in this paper. The IFC data are represented by a graph, and the graph is mined to obtain the largest common graph to merge the IFC graphs, which avoids the influence of GUID and is suitable for different merge scenarios. Section 2 reviews the related work and summarizes the existing problems. Section 3 introduces the method of graph-based mergence. Section 4 demonstrates and analyzes the experimental results. The final section draws the conclusions, summarizes the contributions, and discusses some future works.

2. Related Work

IFC mergence, as an indispensable function of IFC-based collaborative work, has been mentioned in many tools and research endeavors. However, there are some problems in mergence, which are analyzed from the two aspects of tools and studies. In the perspective of tool application, many IFC open-source parsing tools or commercial applications lose the semantics of IFC in IFC mergence, in which different IFC entities are combined by comparing GUID. BIMserver [13] is an open-source IFC server developed using JAVA, which supports the parsing of IFC and provides an integration function. However, since the integration is a textual mergence of two IFC files, two *IfcProject* entities will appear in IFC files, which violates the IFC rule that only one *IfcProject* entity is allowed in an IFC file. xBIM [14] is an open-source IFC parser developed using C#, which supports parsing and model display of the IFC2X3 and IFC4. An IFC mergence function is also provided, which uses *IfcDocumentInformation* to represent different IFC files, but the actual file has not changed. BIMVision is a free model browser [15], which provides plug-ins for merging IFC files. IFC data can be merged in three ways: GUID, elevation, and name. However, they are only saved as a private format of BVF, which affects the sharing of information. FZKViewer [16] developed by Karlsruhe Institute of Technology is an IFC browser, which can display the model after merging, but the relationship entities will be lost after export. Although ArchiCAD [17] supports different IFC files merging by GUID, the entities with the same GUID are still duplicated after merging. There are also free or commercial modeling tools or IFC browsers, for example, DDS-CAD Viewer [18], Areddo [19], and Tekla BIMsight [20], which only merge IFC files in model view. Therefore, most of the tools are not good enough from the perspective of IFC mergence.

The methods of IFC mergence are proposed in some studies of IFC collaboration. The concept of "deferred reference" is proposed to solve the mergence problem of different IFC models [21]. A new entity *IfcDeferredReference* is created to replace the spatial structure entity or component entity to be merged in the submodel. It needs to compare the GUID in mergence and replace *IfcDeferredReference* with the corresponding entity, which can modify the submodel independently and preserve integrity. However, the spatial structures or construction entities in other submodels need to be modified to *IfcDeferredReference* in advance, and the structure of IFC has been changed, which makes the parser require further changes to understand the model.

EDMmodelServer [22] is used to test collaboration issues of BIM servers in Singh's research [23], which merges IFC file through the functions of check-in and check-out. These functions mark building components with their GUID and replace existing components according to GUID when the model is reimported. A special model server property is generated through check-out, which causes mergence errors if it is not preserved in modeling software. Only ArchiCAD can preserve it at present.

Berlo et al. [24] and Nour [25] have proposed a mergence method based on GUID. In Nour's research, the geometric data are returned to the user for determination through model view and the rest are compared by GUID. The IFC data are converted into tree data structures, which are compared by the positions and values in the tree. GUID will change in different software, and thus this approach can only solve the situation in which GUID does not change. Shi et al. [26] also mentioned the conversion of IFC data into a tree structure for comparison and deletion of the same entity, which can improve efficiency in calculating the differences of IFC files. However, the same entity to be removed may be associated with different information in the actual project information, and the deletion of this part of the entity will cause other information changes [9]. In addition, the tree structure only retains the referenced relationship between the data, which loses the semantics of the relationship between the data. There are data with different relationships located in the same location in the tree, which will cause distinguishing errors.

Most of the above studies merge IFC data depending on GUID, which cannot support all mergence scenarios and easily causes model duplication when GUID changes. The extension of IFC entities requires additional modifications to the IFC parser, which indirectly increases the complexity of information sharing. The representation of an IFC model by tree structure destroys the relationship meaning of data. To solve these problems, IFC data require a reasonable method of representation which conforms to the structure of IFC and keeps IFC information intact during the merging process, but the relationship between IFC entities can also be correlated correctly without impact of GUID. The graph structure representation of the IFC model is proposed in this paper, and IFC graphs are mined for comparing the IFC model that is able to ignore the impact of GUID. At last, the mergence of IFC data is completed by rebuilding the relationship between entities.

3. Graph-Based IFC Merging Method

The merging method is divided into three parts in this section: (1) The IFC model is transformed into graph structure. (2) The maximum common graph of IFC graph is mined to obtain the same data in IFC model. (3) Different IFC graphs are combined to complete mergence of IFC data.

3.1. Graphic Structure of IFC Data. An instance is an attribute of other instances according to the IFC model description, which is shown by citing the instance number in the IFC file. For example, in Figure 1, #42 and #118 are cited by #278, respectively, and #42 is also cited by #118.

Therefore, there are instances with characteristic of multilevel references and cross-references in an IFC file. A graph is a data structure consisting of a set of vertices and edges, which is conformed to the characteristics of an IFC entity. The graphical representation of the IFC model has been studied before. The RDF semantic graph by Pauwels et al. [27] is used to store and represent IFC data. The application of RDF is to infer domain-specific knowledge from semantic graphs to solve problems, and a large number of vertices and edges are added in graph to describe ontology, which increases the complexity of graphs. However, domain knowledge is a huge system so it is unnecessary to be applied in merging of IFC data. The directed graph is used to represent the relationship between IFC entities in the study of Arthaud and Lombardo [28]. The relationship between inverse attribute and attribute is represented by bidirectional edge, and the nodes that are cited by other nodes will be duplicated in the IFC graph. The relationship entities are represented by nodes that are connected with other entities by two edges of opposite direction, which is proposed by Tauscher et al. [29]. It is obvious that the graph in the above methods needs more storage space. The inverse attribute does not need to be required in merging, which is not necessary to represent in the graph.

To simplify the structure of the graph and facilitate the mergence of IFC files, a directed node graph that is suitable for IFC mergence is proposed to represent IFC data. Nodes and edges both have labels and values in the graph.

The types of nodes are classified into three categories by labels, namely, *SingleValue*, *ListValue*, and *Entity*. The value types of IFC entities are marked as *SingleValue* and *ListValue*, which consist of *IfcSimpleValue*, *IfcMeasureValue*, and *IfcDerivedMeasureValue* in IFC specification. *IfcSimpleValue* is similar to a numerical data type in programming languages, such as *Interger* and *Logical*. *IfcMeasureValue* is a value in the basic unit described in ISO31-0:1992, such as length and area. *IfcDerivedMeasureValue* is derived from different basic units, such as voltage and density. The entities of value type in IFC files do not cite other entities, which are the terminal nodes in the graph. The difference between *SingleValue* and *ListValue* is whether values are single or aggregative in IFC. For example, the description of *IfcCartesianPoint* in the EXPRESS specification is shown in Figure 2. The value of attribute marked by red is a list of *IfcLengthMeasure*. The nodes labeled *Entity* are representations of entities in an IFC file, which have attributes cited by other entities or values.

If the node is labeled *Entity*, the value of the node is the entity name. If the node is labeled *SingleValue* or *ListValue*, the value of the node is the actual value in the IFC file.

There are two categories of edge labels: *Relationship* and *Attribute*. The relationship entity in IFC is labeled by *Relationship*. The relationship entity defines different relationships between entities, which include spatial logic relationship, attribute connection, component relationship, and work resource assignment. All relationship entities are inherited from *IfcRelationship*. The EXPRESS description of *IfcRelationship* is shown in Figure 3. *IfcRelationship* inherits *IfcRoot*, whose attribute is about relationship creation information. These attributes are useless in IFC mergence. Although some relationship entities involve special attributes of relationships, this does not affect the mergence of IFC data because the IFC files are regenerated by the entities in the original IFC file corresponding to these relationships in graphs. The beginning and end of the edge are connected with two entity nodes that are correlated entities described in the relationship entity. If one of the entities is a form of set, every entity in the set will be connected to another entity described in the relationship entity by the same labeled edges. The direction of the edge is not specified., which does not affect graph mining.

IFC entity nodes are connected to their attribute nodes by edges labeled *Attribute*. The direction of the edge is that in which the entity node points to its attribute node. Every IFC entity is similar to the class described by EXPRESS language, whose attributes could be another class. For example, the value type of attribute, namely, OwnerHistory marked by red in Figure 3, is *IfcOwnerHistory*.

The value of the edge labeled *Relationship* is the name of the relationship entity, and the value of the edge labeled *Attribute* is the attribute name. The IFC graph representation method is illustrated by the IFC graph shown in Figure 4, which is derived from the IFC fragment shown in Figure 5. The entity type of IFC is described as orange nodes and the value type of IFC is described as blue nodes. The relationship entity is described as red edges and the relationship between an entity and its attribute is described as black edges, whose labels are not displayed.

3.2. Mining the Maximum Common Subgraph of the IFC Graph. The goal of maximum common graph mining is to find the isomorphic IFC graph with the largest number of nodes from different IFC graphs. The nodes are compared in the process of traversal in the graph, but not every node needs to be compared. Different traversing methods are adopted due to the peculiarity of nodes in the graph.

It is known from the structure of the IFC graph that the entity nodes connect attribute nodes by edges labeled *Attribute*. Some entities are not referenced by other entities in an IFC file; that is, they are not attributes of other entities (except the relationship entity). These entities are not indicated by edges labeled *Attribute* in an IFC graph, such as the *IfcBuilding* node in Figure 4.

A transformation process is performed on the IFC graph according to these features before mining, and the nodes labeled *Entity* which are not indicated by the edge labeled *Attribute* and its attribute nodes are regarded as an integrated part, which is a node set with this entity node as the root node, such as *IfcBuilding* and its attributes nodes in Figure 6. The integrated part is represented as an entity node, and Figure 4 is transformed into Figure 7. It is noted that the

```
#278=IFCRELAGGREGATES('27PCKGLxT4mxtV9cw6mgBW',#42,$,$,#118,(#131));
#42=IFCOWNERHISTORY(#39,#5,$,.NOCHANGE.,$,$,$,0);
#118= IFCBUILDING('19vqXmFM5DFRZ_VtisI90B',#42,'',$,$,#33,$,'',.ELEMENT.,$,$,#114);
#131=IFCBUILDINGSTOREY('19vqXmFM5DFRZ_Vtl9jsxD',#42,'L1',$,$,#129,$,'L1',.ELEMENT.,0.);
```

Figure 1: IFC instance cross-reference fragment.

```
ENTITY IfcCartesianPoint
        SUBTYPE OF IfcPoint;
                Coordinates :   LIST [1:3] OF IfcLengthMeasure;
        DERIVE
                Dim:            IfcDimensionCount := HIINDEX(Coordinates)
        WHERE
                CP2Dor3D:       HIINDEX(Coordinates) >= 2;
END_ENTITY;
```

Figure 2: EXPRESS document description of *IfcCartesianPoint*.

```
ENTITY IfcRelationship
        ENTITY IfcRoot
                GlobalId:           IfcGloballyUniqueId;
                OwnerHistory:       OPTIONAL IfcOwnerHistory;
                Name:               OPTIONAL IfcLabel;
                Description:        OPTIONAL IfcText;
        ENTITY IfcRelationship
END_ENTITY;
```

Figure 3: EXPRESS document description of *IfcRelationship*.

nodes in this node set are not exclusive, because different entities may include the same attribute.

The graph is traversed in two ways in the process of mining the IFC graph. One is traversal of the transformed IFC graph, in which the entity node and its attribute nodes are regarded as an integrated part, and the transformed IFC graph only has an edge labeled *Relationship*. Another is the traversal of the node set, which is composed of the entity node and its attribute nodes in the transformation process. The mining methods are illustrated by example in detail.

The mining method of the node set is that the node is traversed depth-first by an edge labeled *Attribute* with the entity node as the starting point. It is determined whether each node is identical to nodes in other graphs, which means that the label and value of the node are both identical. If a node is not matched, the traversal stops. In contrast, the node sets are included in the maximum common subgraph if all nodes are matched. The mining method of the node set is illustrated in Figure 8.

A node set with the entity node as the root node (the node is tagged "R" in the figure) is shown in Figures 8(a) and 8(b), where the letters indicate the values of edges and nodes. It is assumed that the "R" nodes are identical in graphs, and the traversing steps are as follows:

(1) The traversal starts with R, next to RE1a, and the same is found in Figure 8(b).

(2) Next to aE3d, and the same is found in Figure 8(b).

(3) Next to dE5g, but dE5g is not found in Figure 8(b).

(4) Exit the traversal, and the result is that the entity node as root node in Figure 8(a) is not identical to the compared node in Figure 8(b).

The mining method of the graph in which the nodes and attribute nodes are regarded as an integrated part is that entity nodes are traversed depth-first by an edge labeled *Relationship*, with the *IfcProject* node as starting point, because *IfcProject*, as the only entity in the IFC file, does not have ambiguity.

When the entity node is not matched, which means that a node set that is composed of entity node and attribute nodes does not find isomorphic subgraphs in other IFC

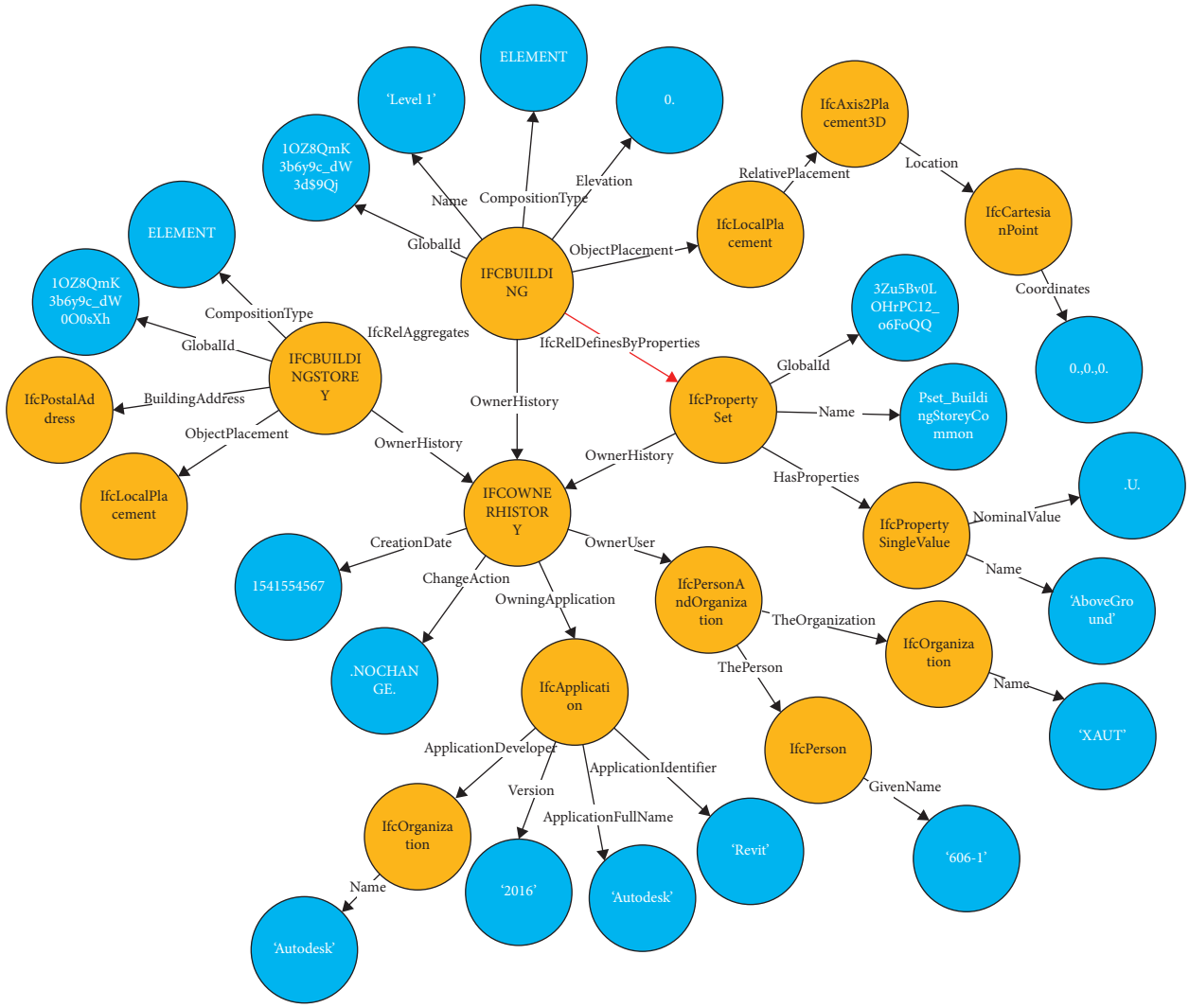

Figure 4: IFC graph.

```
#131= IFCBUILDINGSTOREY('1OZ8QmK3b6y9c_dW3d$9Qj',#42,'Level 1',$,$,#2615,$,$,.ELEMENT.,0.);
#42= IFCOWNERHISTORY(#39,#5,$,.NOCHANGE.,$,$,$,1541554567);
#39= IFCPERSONANDORGANIZATION(#36,#38,$);
#36= IFCPERSON($,$,'606-1',$,$,$,$,$);
#38= IFCORGANIZATION($,'XAUT',$,$,$);
#1= IFCORGANIZATION($,'Autodesk Revit 2016 (CHS)',$,$,$);
#5= IFCAPPLICATION(#1,'2016','Autodesk Revit 2016 (CHS)','Revit');
#6= IFCCARTESIANPOINT((0.,0.,0.));
#2614= IFCAXIS2PLACEMENT3D(#6,$,$);
#2615= IFCLOCALPLACEMENT($,#2614);
#2822= IFCRELAGGREGATES('27PCKGLxT4mxtV9cw6mgBW',#42,$,$,#118,(#131));
#118= IFCBUILDING('1OZ8QmK3b6y9c_dW0O0sXh',#42,$,$,$,#33,$,$,.ELEMENT.,$,$,#114);
#2658= IFCRELDEFINESBYPROPERTIES('24LRPYplv5tQT0zTk4ygdC',#42,$,$,(#131),#2655);
#2654= IFCPROPERTYSINGLEVALUE('AboveGround',$,.U.,$);
#2655= IFCPROPERTYSET('3Zu5Bv0LOHrPC12_o6FoQQ',#42,'Pset_BuildingStoreyCommon',$,(#2654));
```

Figure 5: Partial fragments of the IFC file.

graphs, the current traversing path terminates and restarts with the bifurcation points in the previous path. The mining method is illustrated in Figure 8.

The entity nodes and attribute nodes are represented by the single node in Figures 8(a) and 8(b), where the "R" node is the *IfcProject* node. It is assumed that the "R" nodes are identical in graphs. The graphs are regarded as undirected connected graphs because the edge labeled *Relationship* is undirected. The traversal steps are as follows:

(1) Starting with RE1a, an isomorphic part is found in Figure 9(b). Recording common subgraph is Figure 9(a)

(2) Next to aE3d, and isomorphic part is found in Figure 8(b). Recording common subgraph is Figure 9(b)

(3) Next to dE5g, but dE5g is not found in Figure 8(b). Return to the bifurcation point in the previous path to continue traversing

(4) Next to RE1b, and isomorphic part is found in Figure 8(b). Recording common subgraph is Figure 9(c).

(5) Next to bE3e, but bE3e is not found in Figure 8(b). Return to the bifurcation point in the previous path to continue traversing

(6) Next to RE2c, but RE2c is not found in Figure 8(b). All paths have been traversed.

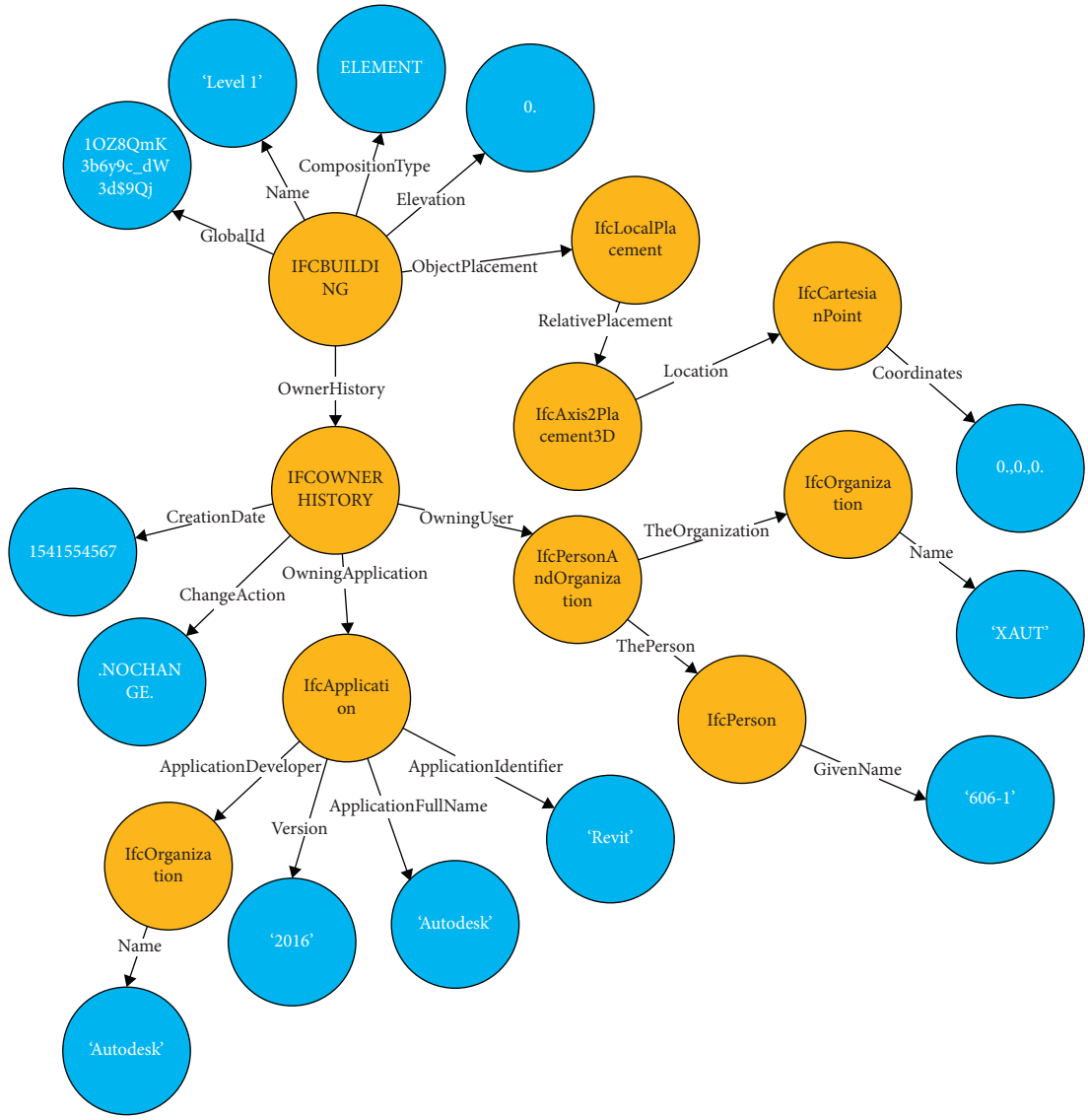

Figure 6: *IfcBuilding* and its attributes nodes.

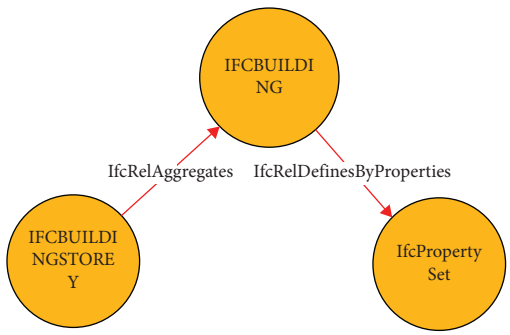

Figure 7: Graph after transformation.

The maximum common subgraph is shown in Figure 9(c). When traversing to the entity node, its attribute node set will be mined through the previous method. The isomorphic subgraph is recorded as a part of a common subgraph, which is composed of entity nodes and attribute nodes.

Each entity node that is not referenced by other entities is the representation of a specific concept in an actual project, such as a wall, column, and construction process. Therefore, if any attribute node of these entities is not matched, it is meaningless to compare other nodes in the node set that is composed of entity node and attribute nodes. The entire IFC graph is composed of these nodes and attribute nodes that connect to the *IfcProject* node by different relationships. Thus, the traversal of a graph is through the *Relationship* edges as a path. There are also special entities in IFC files which are not connected to *IfcProject* via *IfcRelationship* nor referenced by other entities. For example, *IfcStyledItem*, which represents the style information of geometry in IFC, usually shares some attributes with other entities. The mergence of these nodes is explained in the following section.

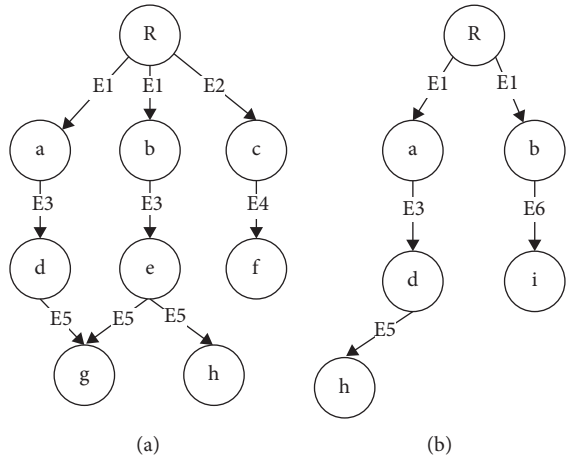

Figure 8: Node set mining graphs: (a) one of the graphs of the node set; (b) other graphs of the node set.

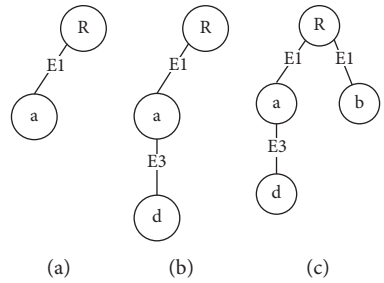

Figure 9: Maximum common graphs: (a) maximum common graph after step 1; (b) maximum common graph after step 2; (c) maximum common graph after step 4.

The whole process starts with an entity node that is not referenced by other entities. If the nodes in its attribute nodes set are not matched to nodes in another graph, it is considered that this entity node and its attribute nodes are excluded from the common graph. The current traversing branch is terminated, and the node in other branches continues to be traversed until the end. In contrast, if the nodes are matched, they are marked as the nodes in the common IFC graph. The maximum common IFC graph is composed of all marked nodes. The whole process is shown in Figure 10.

There are two special situations in the practical application, pertaining to nodes and edges. The first situation is the impact of GUID. GUID is the property of entities inherited from *IfcRoot*, which is the unique identity of entities. However, GUID may change depending on whether it is exported from the same or different software. Therefore, when it is ascertained that GUIDs in IFC files are unstable, all *Attribute* edges whose value is *GlobalID* are deleted in IFC graphs, which makes GUID unrelated to comparison. Conversely, when it is ascertained that GUIDs in the IFC file do not change, the *Attribute* edges whose value is *GlobalID* are retained and other *Attribute* edges that connect to the same node connected by the *GlobalID* edge are deleted, which makes GUID the only compared criterion. Another is the impact of *IfcOwnerHistory*. *IfcOwnerHistory*, which is a property of entities that inherit from *IfcRoot*, describes the history and identification of building components, such as the information of the creator and creation time. As long as the model is reopened in software, *IfcOwnerHistory* will be changed and saved. Therefore, this part of information cannot be used as the compared content during merging of IFC models. The preprocess procedure is to delete all edges connecting *IfcOwnerHistory* nodes in the IFC graph, and the *IfcOwnerHistory* entity is reconstructed after merging.

3.3. IFC Graph Merging. The nodes and edges that are not in the maximum common graph are reconnected with the maximum common graph according to the connection mode of the original graph after mining the maximum common graph. Two IFC graphs that need to be merged are shown in Figures 11(a) and 12(b), respectively. The entity nodes that are not indicated by the edge labeled *Attribute* are notated with single letters, among which the gray nodes are marked to indicate that they are in the maximum common graph, and its attribute node set is represented by yellow nodes that connect to entity nodes through directed edges. The special entities mentioned in the previous section are also considered, which are represented as red nodes, and their relationships with the attribute node set are represented as directed edges in the graph. The *Relationship* edges

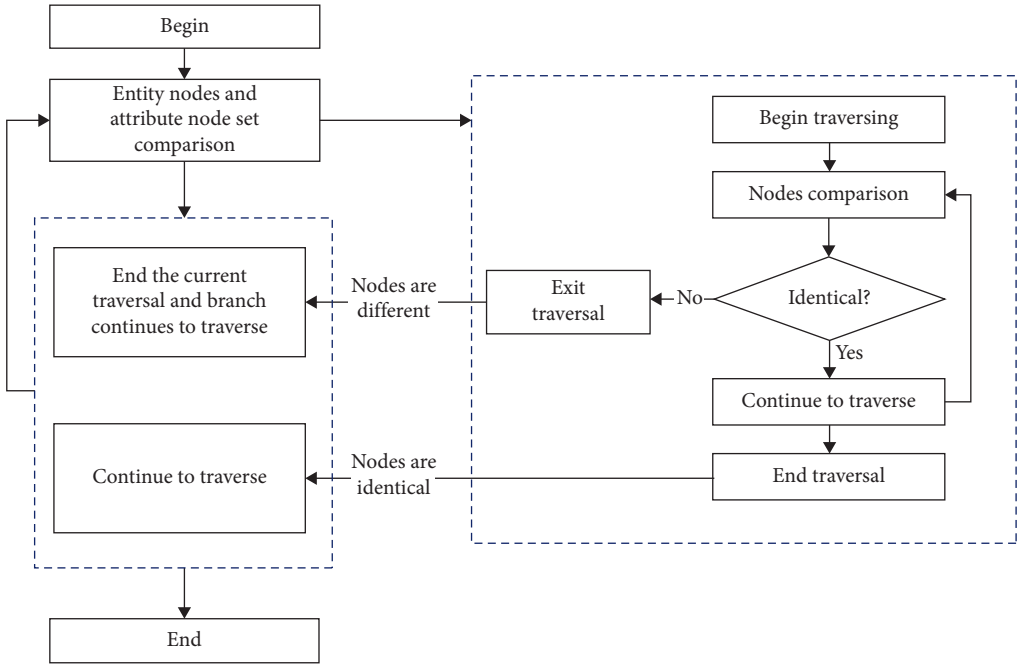

Figure 10: Mining flow chart.

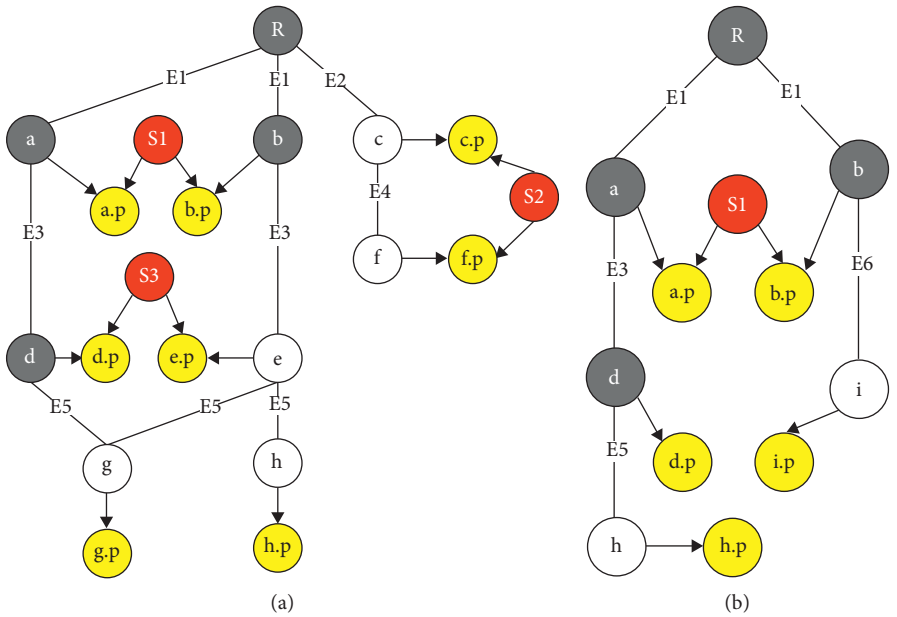

Figure 11: Two IFC graphs to be merged: (a) merging file graph and (b) merged file graph.

are undirected in the IFC graph. The merging method is illustrated in Figure 12.

(1) The nodes that are adjacent to the maximum common graph are copied to connect to the common graph. The condition for retrieving the adjacent nodes is to find unmarked entity nodes that connect to marked nodes through *Relationship* edges. This step is performed to establish connection with the maximum common graph, which is shown in Figure 12(a).

(2) The unmarked nodes that are connected to adjacent nodes are copied to the common graph, which includes a portion of the special entity nodes. It is shown in Figure 12(b).

(3) The remaining special entity nodes are added to the maximum common subgraph, as shown in Figure 12(c).

Some nodes have been replicated in the second step; thus it is necessary to determine the isomorphism of remaining

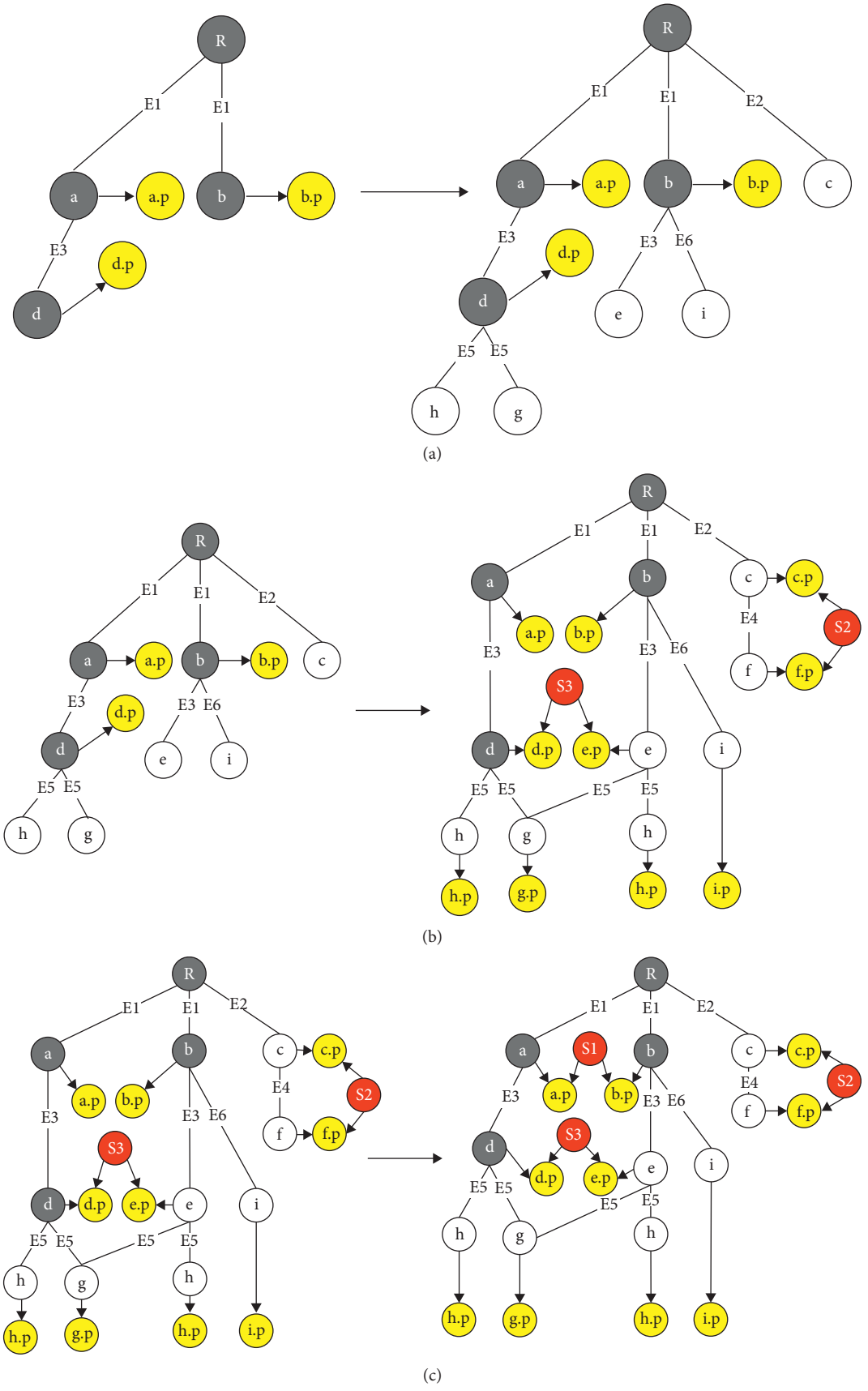

Figure 12: The steps of IFC merging: (a) first step of merging, (b) second step of merging, and (c) third step of merging.

special nodes during the third step. The determination is similar to the traversal of the attribute node set in the previous section. Starting from the special entity node, the nodes indicated by the special entity node are traversed depth-first: if different nodes or edges are found, the traversal is terminated and the special nodes are added. Otherwise, these nodes have already existed and do not need to be added.

After the mergence is completed, the IFC file is regenerated according to the merged graph of IFC, where the instances described in each line are regenerated by the nodes labeled *Entity* and the attribute node set. The relationship instances are regenerated by the edges labeled *Relationship* based on the corresponding instances in the original IFC file, where only two related attributes are modified. If the edge labeled *Relationship* exists in the maximum common graph, the IFC file referenced in the regeneration of this edge should be a basic file in collaborative work. This approach preserves other attributes of relationship entities in the file, and therefore the conversion of relationship entities into edges does not cause information loss in IFC mergence.

4. Experimental Results

An IFC mergence tool is developed using C# in order to verify the feasibility of the method. The experimental plan is divided into two parts: one is that IFC files are merged under the same modeling software conditions, and another involves different modeling software conditions. The difference between the two parts is that the models in the same modeling environment are derived from the same software, and the IFC models in different modeling environments are derived from different software. The feasibility of the merging method is verified by comparing the count of IFC entities in the IFC model, which are merged by GUID and graph. Revit and ArchiCAD, which are common modeling software packages for the general public, are used as IFC file export software. A simple two-story villa building is used as an experimental model, which is composed of building walls, structural columns, floors, windows, and doors. The IFC files are divided into six situations according to different scenarios, as shown in Figure 13. Figures 13(a) and 13(b) describe the first floor model and the second floor model of the building, respectively. Figures 13(c) and 13(d) describe the architecture model and the structural column model, respectively. Figure 13(e) describes the first-story structural column and the external wall, and Figure 13(f) describes the first-story external wall, the interior wall, windows, and doors. The IFC model display tool is IFC-Viewer [30]. There are three merging scenarios: scenario 1 is mergence between floors corresponding to Figures 13(a) and 13(b), scenario 2 is mergence of architectural and structural models corresponding to Figures 13(c) and 13(d), and scenario 3 is merging the model after adding and modifying components corresponding to Figures 13(e) and 13(f).

4.1. Same Modeling Software Environment. Verification in the same modeling software environment includes scenario 1, scenario 2, and scenario 3. The IFC models derived from Revit and ArchiCAD are merged, where *IfcOwnerHistory* is preprocessed. The counts of instances are determined after mergence, including spatial entities, walls, floors, windows, doors, column entities, and other entities such as relationships and attributes. Three scenarios are validated as shown in Tables 1–3, respectively. In the tables, "a," "b," "c," "d," "e," and "f" correspond to the IFC models in Figures 12(a)–12(f), respectively, and "a-b," "c-d," and "e-f" represent the model after the mergence of a model and b model, the model after the mergence of c model and d model, and the model after the mergence of e model and f model. "GUID" represents that the model is merged through comparing GUID. "Revit" and "ArchiCAD" represent IFC models exported from Revit and ArchiCAD.

It can be determined from Table 1 that the counts of spatial entities (*IfcSite*, *IfcBuilding*, and *IfcBuildingStorey*) are 1, 1, and 3 in the merged graph, which are not increased. By the use of GUID merging, the same space entity is merged as a different entity; however, the counts of the same spatial entities have increased in the mergence using GUID, corresponding to 2, 2, and 3. The counts of spatial entities are equal to the sum of two files in mergence by GUID because the same spatial entity is regarded as different entities. The same mistake also occurs in Table 2. The count of *IfcBuildingStorey* is changed to 5, although it is obvious that there are only 3 floors in the model. The count of walls in scenario 3 is 9, but the counts of *IfcWall* and *IfcWallStandardCase* in Table 3 increase to 15 in the mergence using GUID. The three-dimensional displays of three scenarios are shown in Figure 14. The displays of scenario 1 and scenario 2 are the same because the spatial entity has no geometric information and does not affect the display. There is an overlap of the external wall in scenario 3, where the same external wall is identified as different components due to comparison of GUID. It can be concluded that the graph-based merging method is more accurate in these scenarios, whether it involves the mergence of floors or the mergence of components.

4.2. Different Modeling Software Environments. The experiment is the same as that in the same modeling environment, where models from Revit and ArchiCAD are merged with each other. The counts of entities after merging are shown in Tables 4–6. "R" in the tables represents exporting from Revit, while "A" represents ArchiCAD, and a, b, c, d, e, and f represent the same content as above.

The counts of entities are contrary to those in the condition of the same modeling environment. There is no redundancy of spatial entities and component entities after merging by GUID, because the IFC export mechanisms of different modeling software packages are different, and the IFC file export is also affected by the MVD (Model View Definition), for which it is difficult to completely support modeling software. It is difficult for the same component to be consistent under different modeling software packages even through the same export configuration. If the new models are created based on IFC files derived from other

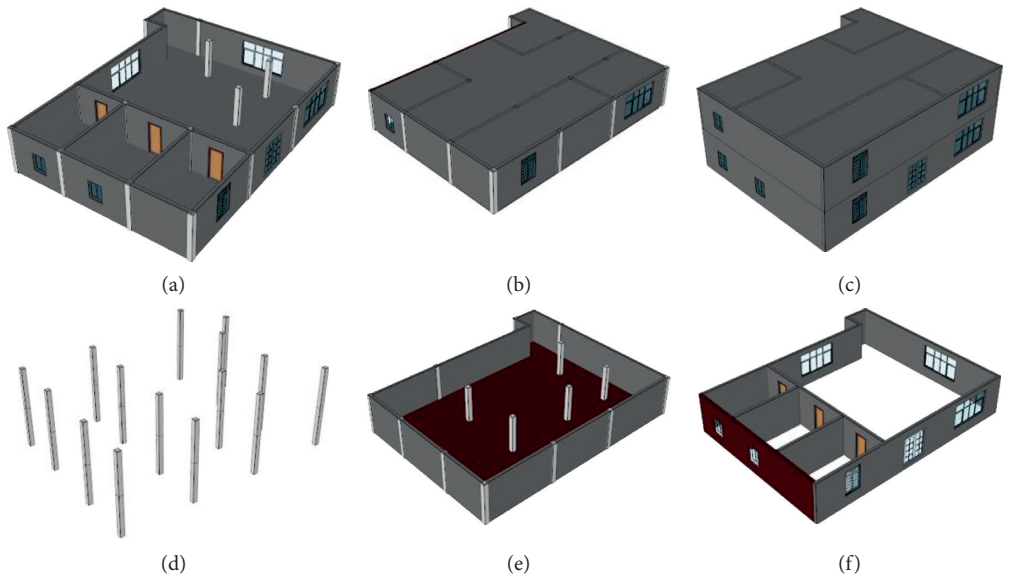

Figure 13: IFC model display of six scenes: (a) the first floor model of a building, (b) the second floor model of a building, (c) the architecture model, (d) the structural column model, (e) one-story structural column and the external wall, and (f) one-story external wall, the interior wall, windows, and doors.

Table 1: Entity statistics for scenarios a and b.

	Revit				ArchiCAD			
	a	b	a-b	a-b GUID	a	b	a-b	a-b GUID
File size (kb)	167	171	319	321	352	292	649	650
IfcProject	1	1	1	1	1	1	1	1
IfcSite	1	1	1	2	1	1	1	2
IfcBuilding	1	1	1	2	1	1	1	2
IfcBuildingStorey	1	2	3	3	1	2	3	3
IfcWall								
IfcWallStandardCase	9	12	21	21	9	12	21	21
IfcSlab	1	2	3	3	1	2	3	3
IfcWindow	6	5	11	11	6	5	11	11
IfcDoor	4	4	8	8	4	4	8	8
IfcColumn	15	15	30	30	15	15	30	30
IfcTypeObject	7	8	15	15	8	9	17	17
IfcRelationship	94	112	204	206	185	204	387	389
IfcPropertyDefinition	44	49	93	93	135	141	276	276
IfcRepresentationItem	2683	2666	5253	5253	6169	4580	10667	10667

Table 2: Entity statistics for scenarios c and d.

	Revit				ArchiCAD			
	c	d	c-d	c-d GUID	c	d	c-d	c-d GUID
File size (kb)	202	30	220	223	542	93	638	640
IfcProject	1	1	1	1	1	1	1	1
IfcSite	1	1	1	2	1	1	1	2
IfcBuilding	1	1	1	2	1	1	1	2
IfcBuildingStorey	3	2	3	5	3	2	3	5
IfcWall IfcWallStandardCase	21	0	21	21	21	0	21	21
IfcSlab	3	0	3	3	3	0	3	3
IfcWindow	11	0	11	11	11	0	11	11
IfcDoor	8	0	8	8	8	0	8	8
IfcColumn	0	30	30	30	0	30	30	30
IfcTypeObject	8	2	10	10	9	2	11	11
IfcRelationship	160	41	196	201	222	161	378	383
IfcPropertyDefinition	57	33	88	90	123	153	274	276
IfcRepresentationItem	3018	155	3155	3155	10109	404	10508	10508

Table 3: Entity statistics for scenarios e and f.

	Revit				ArchiCAD			
	e	f	e-f	e-f GUID	e	f	e-f	e-f GUID
File size (kb)	30	153	161	169	72	302	353	370
IfcProject	1	1	1	1	1	1	1	1
IfcSite	1	1	1	2	1	1	1	2
IfcBuilding	1	1	1	2	1	1	1	2
IfcBuildingStorey	1	1	1	2	1	1	1	2
IfcWall IfcWallStandardCase	6	9	9	15	6	9	9	15
IfcSlab	1	0	1	1	1	0	1	1
IfcWindow	0	6	6	6	0	6	6	6
IfcDoor	0	4	4	4	0	4	4	4
IfcColumn	15	0	15	15	15	0	15	15
IfcTypeObject	2	6	8	8	3	6	9	9
IfcRelationship	45	75	97	120	112	104	187	216
IfcPropertySetDefinition	24	28	44	52	91	58	135	149
IfcRepresentationItem	155	2605	2627	2667	343	5958	6094	6186

IfcTypeObject represents the total of type entities of components. IfcRelationship represents the sum of all relational entities in an IFC file. IfcPropertyDefinition denotes the aggregation of attribute set entities in an IFC file. IfcRepresentationItem represents the sum of geometric representation entities in IFC files.

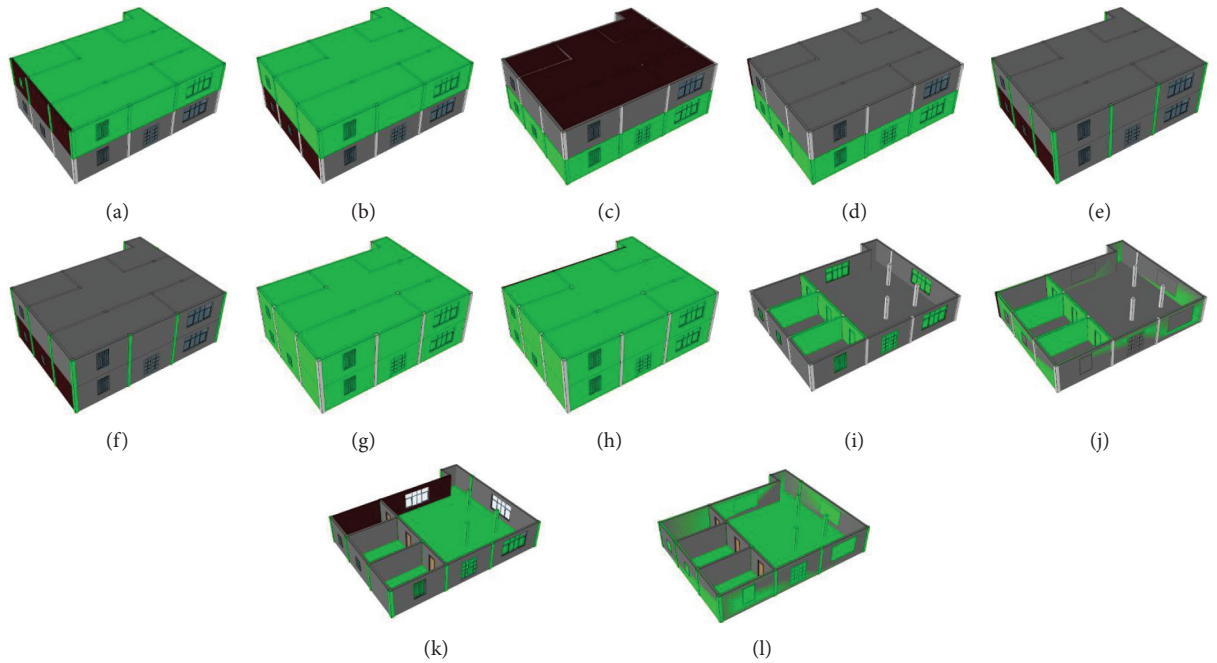

Figure 14: Display of the merged IFC model. (a) Revit-a-b. (b) Revit-a-b (GUID). (c) ArchiCAD-a-b. (d) ArchiCAD-a-b (GUID). (e) Revit-c-d. (f) Revit-c-d (GUID). (g) ArchiCAD-c-d. (h) ArchiCAD-c-d (GUID). (i) Revit-e-f. (j) Revit-e-f (GUID). (k) ArchiCAD-e-f. (l) ArchiCAD-e-f (GUID).

software, the GUID attribute will be retained. These experimental models are created in this way in order to enable comparison in different situations; however, this method causes loss of partial information of IFC, possibly because of incomplete support of IFC in software, which is not recommended in collaborative works.

We can draw the conclusion that GUID will change in the same modeling environment and that the same entities will be duplicated in the mergence relying on GUID, which will cause errors. Compared with the previous methods of GUID merging, it is obviously feasible to merge by graph. Since the IFC export mechanism of the modeling software is different, the accuracy of mergence based on GUID is higher in different modeling environments when the peculiar collaborative method is adopted. However, the method of using graph structure of IFC data can accurately complete the mergence of IFC models whether relying on GUID or not.

Table 4: Entity statistics for scenarios a and b in different modeling environments.

	a(R)-b(A)	a(R)-b(A) (GUID)	b(R)-a(A)	b(R)-a(A) (GUID)
File size (kb)	448	448	512	511
IfcProject	1	1	1	1
IfcSite	2	1	2	1
IfcBuilding	2	1	2	1
IfcBuildingStorey	3	3	3	3
IfcWall IfcWallStandardCase	21	21	21	21
IfcSlab	3	3	3	3
IfcWindow	11	11	11	11
IfcDoor	8	8	8	8
IfcColumn	30	30	30	30
IfcTypeObject	16	16	16	16
IfcRelationship	298	296	297	295
IfcPropertySetDefinition	185	185	184	184
IfcRepresentationItem	7181	7181	8715	8715

Table 5: Entity statistics for scenarios c and d in different modeling environments.

	c(R)-d(A)	c(R)-d(A) (GUID)	d(R)-c(A)	d(R)-c(A) (GUID)
File size (kb)	286	285	563	562
IfcProject	1	1	1	1
IfcSite	2	1	2	1
IfcBuilding	2	1	2	1
IfcBuildingStorey	5	3	5	3
IfcWall IfcWallStandardCase	21	21	21	21
IfcSlab	3	3	3	3
IfcWindow	11	11	11	11
IfcDoor	8	8	8	8
IfcColumn	30	30	30	30
IfcTypeObject	10	10	11	11
IfcRelationship	321	318	263	261
IfcPropertySetDefinition	210	210	156	156
IfcRepresentationItem	3417	3417	10078	10078

Table 6: Entity statistics for scenarios e and f in different modeling environments.

	e(R)-f(A)	e(R)-f(A) (GUID)	f(R)-e(A)	f(R)-e(A) (GUID)
File size (kb)	327	318	213	206
IfcProject	1	1	1	1
IfcSite	2	1	2	1
IfcBuilding	2	1	2	1
IfcBuildingStorey	2	1	2	1
IfcWall IfcWallStandardCase	15	9	15	9
IfcSlab	1	1	1	1
IfcWindow	6	6	6	6
IfcDoor	4	4	4	4
IfcColumn	15	15	15	15
IfcTypeObject	8	8	9	9
IfcRelationship	149	140	187	178
IfcPropertySetDefinition	82	82	119	119
IfcRepresentationItem	5998	5906	2855	2815

5. Conclusion and Discussion

IFC is a popular data format for building information models, which makes it easier for different participants to share information. Aiming at the problem of IFC mergence in information collaboration, this paper proposes a method for graph-based IFC data mergence. The graphs are established using the characteristics of IFC data, which are mined to obtain the maximum common subgraph. The mergence of the IFC graph is completed by this method of establishing the connection between IFC graph and common graph. Finally, an ordinary two-story residential building is used as the experimental model to verify the merging method, which proves the feasibility of the method. The main contributions of this paper are as follows:

(1) IFC data can be clearly and concisely represented by graphs, where IFC entities are represented by nodes and relationship entities are simplified as edges. This method retains the structure of IFC data and is more useful to manage IFC data in IFC data mergence compared with the previous graph representation.

(2) A mining method of IFC graphs is proposed. IFC graphs are traversed to obtain the maximum common subgraph using the characteristics of graphs, which do not need to traverse every IFC node. This method is more efficient when comparing IFC entities.

(3) The IFC graphs are merged to restore IFC files. The merging method that ignores the GUID and other impact factors can be applied in more scenarios than previous research methods. On the other hand, the integrity of IFC data is preserved because of the whole process without deletion of IFC data.

Information collaboration has always been a primary concern of the construction industry. The mergence of IFC data as a general building information standard is important for collaborative work. This paper only proposes a way to solve parts of problems, and a more understandable semantic representation of IFC is required to make it easier to apply IFC data in the future.

Data Availability

The IFC files used to support the findings of this study are available at https://github.com/Lyc-r/IFCmerge.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research is supported by the National Natural Science Foundation of China (no. 51878556) and Key Research and Development Program of Shaanxi Province (2017ZDXM-GY-098, 2019TD-014).

References

[1] R. Santos, A. A. Costa, and A. Grilo, "Bibliometric analysis and review of building information modelling literature published between 2005 and 2015," *Automation in Construction*, vol. 80, pp. 118–136, 2017.

[2] C. Boton and D. Forgues, "Practices and processes in BIM projects: an exploratory case study," *Advances in Civil Engineering*, vol. 2018, Article ID 7259659, 12 pages, 2018.

[3] S. Fuchs and R. J. Scherer, "Multimodels—instant nD-modeling using original data," *Automation in Construction*, vol. 75, pp. 22–32, 2017.

[4] D.-G. Lee, J.-Y. Park, and S.-H. Song, "BIM-based construction information management framework for site information management," *Advances in Civil Engineering*, vol. 2018, Article ID 5249548, 14 pages, 2018.

[5] M. Oh, J. Lee, S. W. Hong, and Y. Jeong, "Integrated system for BIM-based collaborative design," *Automation in Construction*, vol. 58, pp. 196–206, 2015.

[6] I. Faraj, M. Alshawi, G. Aouad, T. Child, and J. Underwood, "An industry foundation classes web-based collaborative construction computer environment: WISPER," *Automation in Construction*, vol. 10, no. 1, pp. 79–99, 2000.

[7] Z. Xu, T. Huang, B. Li, H. Li, and Q. Li, "Developing an IFC-based database for construction quality evaluation," *Advances in Civil Engineering*, vol. 2018, Article ID 3946051, 22 pages, 2018.

[8] Y.-C. Lee, C. M. Eastman, W. Solihin, and R. See, "Modularized rule-based validation of a BIM model pertaining to model views," *Automation in Construction*, vol. 63, pp. 1–11, 2016.

[9] J. Zhang, Q. Li, Z. Niu, J. Lin, and F. Yu, "A multi-server information-sharing environment for cross-party collaboration on a private cloud," *Automation in Construction*, vol. 81, pp. 180–195, 2017.

[10] K. A. Jørgensen, J. Skauge, P. Christiansson, K. Svidt, K. B. Sørensen, and J. Mitchell, *Use of IFC Model Servers-Modelling Collaboration Possibilities in Practice*, Department of Production, Aalborg University, Aalborg, Denmark, 2008, https://vbn.aau.dk/en/publications/use-of-ifc-model-servers-modelling-collaboration-possibilities-in.

[11] T. Pazlar and Ž. Turk, "Interoperability in practice: geometric data exchange using the IFC standard," *Electronic Journal of Information Technology in Construction*, vol. 13, pp. 362–380, 2008.

[12] J. Steel, R. Drogemuller, and B. Toth, "Model interoperability in building information modelling," *Software & Systems Modeling*, vol. 11, no. 1, pp. 99–109, 2012.

[13] BIMServer, 2018, http://bimserver.org/.

[14] xBIM Toolkit, 2018, http://docs.xbim.net/research/history.html.

[15] BIMVision, 2018, https://bimvision.eu/en/free-ifc-model-viewer/.

[16] FZKViewer, 2018, https://www.iai.kit.edu/english/1648.php.

[17] ArchiCAD, 2018, https://www.graphisoft.com/archicad/.

[18] DDS-CAD viewer, 2018, https://www.dds-cad.net/downloads/dds-cad-viewer/.

[19] Areddo, 2018, http://www.areddo.com/.

[20] Tekla BIMsight, 2018, https://www.tekla.com/products/tekla-bimsight.

[21] W. Solihin, C. Eastman, and Y. C. Lee, "A framework for fully integrated building information models in a federated environment," *Advanced Engineering Informatics*, vol. 30, no. 2, pp. 168–189, 2016.

[22] EDM Model Server, 2018, http://www.jotneit.no/products/edm-model-server-ifc.
[23] V. Singh, N. Gu, and X. Wang, "A theoretical framework of a BIM-based multi-disciplinary collaboration platform," *Automation in Construction*, vol. 20, no. 2, pp. 134–144, 2011.
[24] V. Berlo, J. Beetz, P. Bos, and V. Tongeren, "Collaborative engineering with IFC : new insights and technology," in *Proceedings of the European Conference on Product and Process Modelling*, Reykjavik, Iceland, July 2012.
[25] M. Nour, "Manipulating IFC sub-models in collaborative teamwork environments," in *Proceedings of the 24th CIB W-78 Conference on Information Technology in Construction*, Maribor, Slovenia, June 2007.
[26] X. Shi, Y.-S. Liu, G. Gao, M. Gu, and H. Li, "IFCdiff: a content-based automatic comparison approach for IFC files," *Automation in Construction*, vol. 86, pp. 53–68, 2018.
[27] P. Pauwels, D. Van Deursen, R. Verstraeten et al., "A semantic rule checking environment for building performance checking," *Automation in Construction*, vol. 20, no. 5, pp. 506–518, 2011.
[28] G. Arthaud and J. C. Lombardo, "Automatic semantic comparison of STEP product models," in *Innovations in Design & Decision Support Systems in Architecture and Urban Planning*, pp. 447–463, Springer, Dordrecht, Netherlands, 2006.
[29] E. Tauscher, H. J. Bargstädt, and K. Smarsly, "Generic BIM queries based on the IFC object model using graph theory," in *Proceedings of the 16th International Conference on Computing in Civil and Building Engineering*, Osaka, Japan, July 2016.
[30] IFC Viewer, 2018, http://rdf.bg/product-list/ifc-engine/ifc-viewer/.