Parameterized IFC-based Graph Generation for User-oriented Path Search

Barbara Strug, Grażyna Ślusarczyk, Jakub Jas
Jagiellonian University, Poland
barbara.strug@uj.edu.pl

Abstract. This paper deals with the problem of transforming the data obtained from the IFC file of a given building into a weighted graph, which is used for searching shortest routes accessible for different types of users including people and mobile objects. This graph contains information about the topology and accessibility between building spaces. It is created using the parameter specifying the permissible distance from the center of a moving object to a wall. Edge weights are calculated based on the Euclidean distance between nodes representing doors or internal points of rooms with concave shapes. On the basis of information encoded in the graph the application calculates the shortest path between designated rooms and creates its visualization. The presented approach is illustrated on examples of searching shortest routs between spaces of the building extracted from the IFC file belonging to the free IFC model database.

1. Introduction

Nowadays, BIM (Building Information Modelling) technology is intensively developed, i.e. modelling and management of building information throughout its entire life cycle. However, this information is difficult to access for people who are not professionally involved in design. Access to design data contained in the BIM database is possible using the IFC (Industry Foundation Classes) format (buildingSMART, 2013). However, although the IFC is an open standard, its complex nature makes finding useful information difficult. Therefore it is important to create algorithms that allow for an easy access to building elements essential for solving user-defined tasks, to relations between these elements and their attributes (Langenhan et al., 2013; Jin et al., 2018). There is also a lot of research related to shortest path finding problem inside buildings (Lee et al., 2010; Llanos et al., 2014), but most of them do not take into account the preferences of different types of users in respect to their mobility.

This paper introduces a method of transforming the data obtained from the IFC of a given building into a weighted graph, which is used for searching shortest routes accessible for different types of users including people and mobile objects. This graph contains building-related knowledge, like information about the topology of spatial layouts of buildings and characteristics of elements ensuring access between spaces.

The application extracts from the IFC format file objects representing rooms (IfcSpace), doors (IfcDoor) and stairs (IfcStair) along with related geometric data, as well as relationships between these objects using IfcOpenShell tool (IfcOpenShell Academy). In the next step the so called *space of movement* is calculated for each room. The area of each polygon representing a space is decreased by a value of a parameter *distance* specifying the permissible distance from the center of a moving object to a wall. This value is determined on the basis of the profile of persons or mobile objects, which specifies their sizes, preferences and ability to move. The *distance* parameter allows us to create a representation of the area on which a person or an object can move and to adapt searched routes to their capabilities, and therefore becomes a parameter for the created graph of a building.

During a graph creation process also shapes of spaces are taken into account. In case of spaces with shapes described by concave polygons, their vertexes lying inside convex envelopes, called concavity points, are determined (Preparata & Shamos, 1985). Then a weighted graph is created, whose nodes represent points corresponding to doors and concavity points computed for concave polygons (Jas, 2019). Edges are created between nodes representing different doors of a space, if the straight line connecting them does not go beyond the outline of this space, or between nodes representing doors and concavity points. Edge weights are calculated based on the Euclidean distance between nodes. On the basis of the information encoded in the created graph the application calculates the shortest path between specified rooms and creates its visualization. If these rooms are located on different floors, then the path found consists of two fragments: the path from the start room to the stairs on one floor, and the path from the stairs to the destination room on another floor.

The presented research is motivated by the desire to find shortest routes in a building, which would be accessible for different types of users including pedestrians, disabled people with possible restrictions in mobility and various types of mobile objects (i.e., stand-alone vacuum cleaners). Moreover, while searching for the optimal routes it is important to take into account spaces with shapes described by concave polygons, through which sometimes the shortest path can not lead directly (i.e., in a straight line) from one door to the other.

The presented approach is illustrated on examples of searching shortest routs between rooms located on the same floor and different floors. The floor plans of the building are created on the basis of the IFC file belonging to the free IFC model database (Open IFC model repository, 2010).

2. Extraction of IFC Data

The proposed application extracts from the IFC format file, objects representing rooms (IfcSpace), doors (IfcDoor) and stairs (IfcStair) along with related geometric data, as well as relationships between these objects using IfcOpenShell tool (IfcOpenShell Academy). IfcOpenShell has a module enabling using the tool in Python. A method, which obtains all IfcStair elements on a given floor, written in IfcOpenShell-python is shown in the listing in Fig. 1. In Fig. 2 the listing of a method for obtaining all IfcSpace objects on a given floor is presented.

```
def getIFCStairsOnSpecifiedFloor(self, floor_name):
ifc_stairs = []
ifc_building_storeys = self.ifc_file.by_type('IfcBuildingStorey')
for ifc_building_storey in ifc_building_storeys:
if ifc_building_storey.Name == floor_name:
for related_element in ifc_building_storey.ContainsElements[0]
.RelatedElements:
if related_element.is_a('IfcStair'):
ifc_stairs.append(related_element)
return ifc_stairs
```

Figure 1: A method for obtaining all IfcStair elements on a given floor

```
def getIFCSpacesOnSpecifiedFloor(self, floor_name):
ifc_spaces = []
ifc_building_storeys = self.ifc_file.by_type('IfcBuildingStorey')
for ifc_building_storey in ifc_building_storeys:
if ifc_building_storey.Name == floor_name:
for ifc_object_definition in
ifc_building_storey.IsDecomposedBy[0].RelatedObjects:
ifc_spaces.append(ifc_object_definition)
return ifc_spaces
```

Figure 2: A method for obtaining all IfcSpace objects on a given floor

On the basis of the information from the IFC file the floor plans of the building are created. In Fig. 3 a model of the building, the IFC file of which is processed by our application, is shown. The layout of the building first floor is presented in Fig. 4. The spaces of the floor are labelled and the doors between them are denoted by circles.

Figure 3: An example of a building model

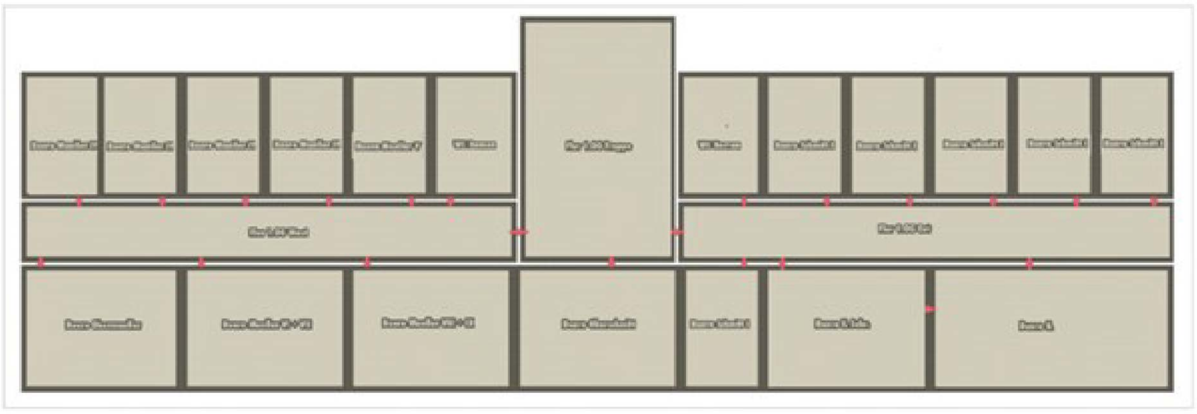
Figure 4: The layout of the building first floor

3. Graph Representation of the Extracted Data

In order to create a graph representing building topology and such that an algorithm calculating the shortest path between specified rooms will work efficiently on it, the position of graph nodes should be specified. We decided to calculate the path distance between points determining the position of doors instead of room centers. At first, so called *space of movement* is calculated for each room. The area of each polygon representing a space is decreased by a value of a parameter *distance* specifying the permissible distance from the center of a moving object to a wall (marked with a dotted line in Fig. 5a). This value is determined on the basis of profiles of different types of users, including pedestrians, people with disabilities (incorporating wheelchair users) and mobile objects. These profiles specify sizes, preferences and ability to move of persons and objects. The *distance* parameter allows us to create a representation of the area on which a person or an object can move and to adapt searched routes to their capabilities, and therefore the specified value becomes a parameter for the created graph of a building. During a graph creation process also shapes of spaces are taken into account. In case of spaces with shapes described by concave polygons, their vertexes lying inside convex envelopes, called concavity points, are determined (Preparata & Shamos, 1985) (Fig. 5b).

Then, a weighted graph is created, whose nodes represent points being orthogonal projections of the centers of doors (denoted by red points in Fig. 5a) onto the nearest edges of the spaces of movement (denoted by black points in Fig. 5a), and concavity points computed for concave polygons (Jas, 2019). Edges are created between nodes representing different doors of a space, if the straight line connecting them does not go beyond the outline of this space, or between nodes representing doors and concavity points. Edge weights are calculated based on the Euclidean distance between nodes.

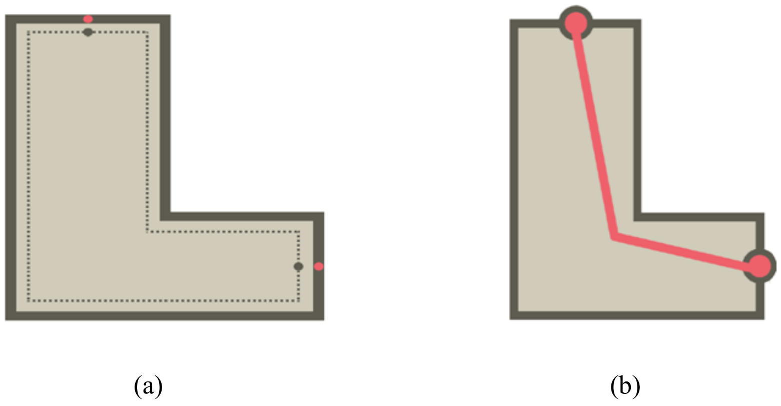

Figure 5: Example of a concave space in a building (Jas, 2019); a) a space of movement, b) a path going through a concavity point

A graph created according to the algorithm described above and representing topology of the first floor of the building shown in Fig. 4 is presented in Fig. 6. There are 49 nodes, denoted by green circles, where edges are drawn in red. The same graph marked on the first floor plan is shown in Fig. 7.

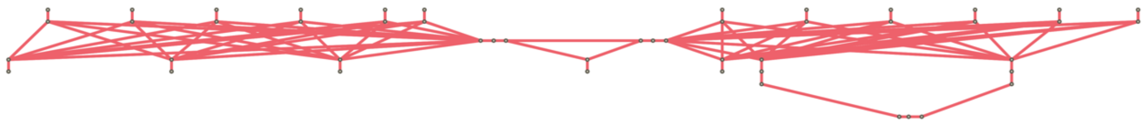

Figure 6: A graph representing topology of the first floor of the building shown in Fig. 4

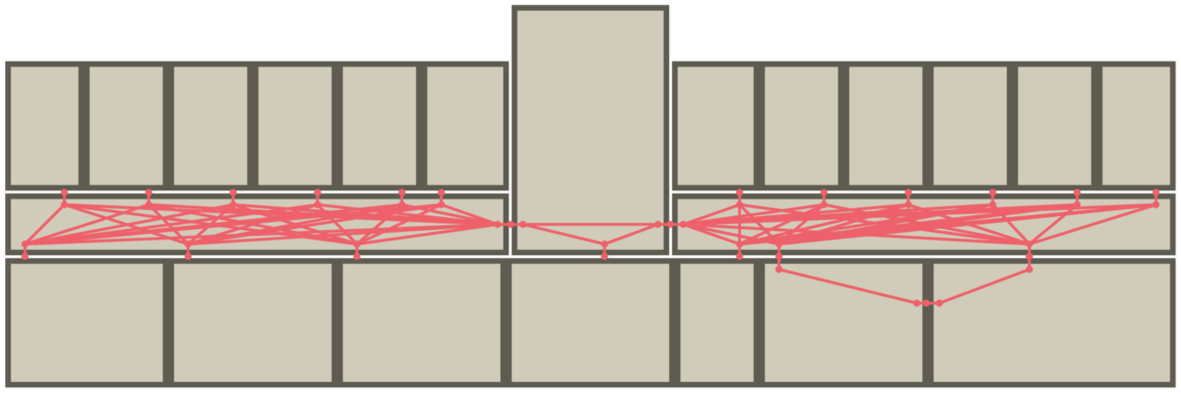

Figure 7: A graph from Fig. 6 marked on the first floor plan

4. Results of Path Finding

The method presented here has been implemented as a modular python application. The two main modules of the application are responsible for importing IFC file and generating a graph based on the obtained data. The third module calculates the shortest path on the derived graph and returns results and produces an image file with the path visualisation. Such a structure of the application makes it easy to change the way the shortest path is calculated as well as adding different graph-based algorithms that could perform other operations on the floor plan graph.

On the basis of information encoded in the created graph the application calculates the shortest path between specified rooms and creates its visualization. The shortest path is determined based on room names, for which thee corresponding doors are determined. If the connected rooms have more than one door, then it is necessary to calculate the shortest path between each pair in the form (door_room1, door_room2). The algorithm selects the pair of doors between which the distance is minimal.

If the selected rooms are located on different floors, then the found path consists of two fragments: the path from the start room to the stairs on one floor, and the path from the stairs to the destination room on another floor. The algorithm searches for the shortest path taking into account all possible staircases.

The shortest path from the west corridor on the building first floor to the office labelled *Buero_IL* on the same floor found by our application is shown in Fig. 8. The first part of the shortest path from the laboratory *K5* in the basement to the office *Buero Obermueller* on the first floor, leading from the laboratory to the stairs is presented in Fig. 9. The second part of this path leading through the first floor is shown in Fig. 10.

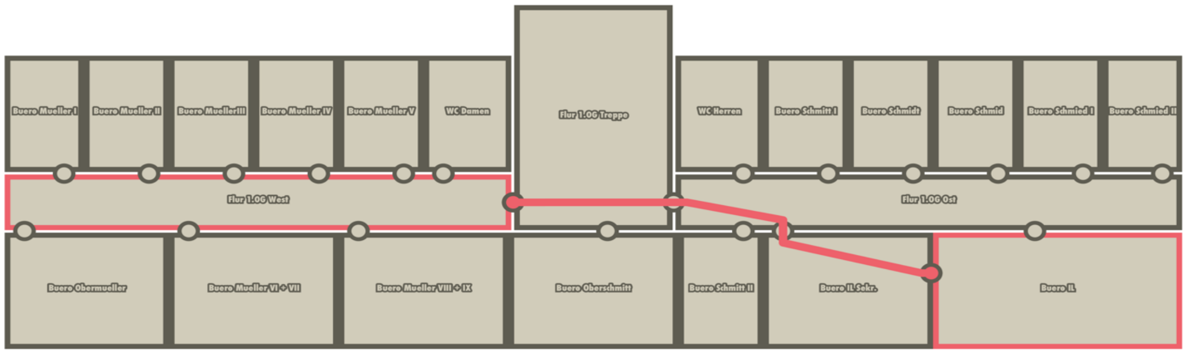

Figure 8: The shortest path between specified rooms on the same floor

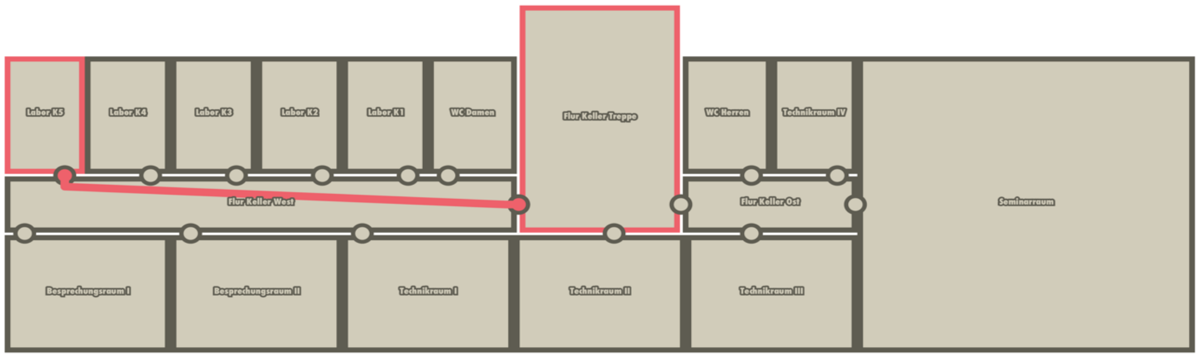

Figure 9: The first part of the shortest path from the laboratory *K5* to the office *Buero_IL*

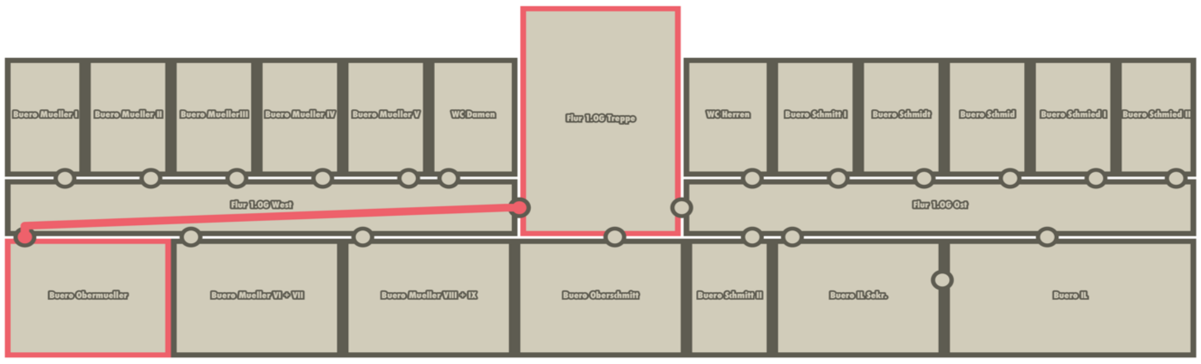

Figure 10: The second part of the shortest path from the laboratory *K5* to the office *Buero_IL*

5. Conclusions

In this paper a method of transforming the data obtained from the IFC file of a given building into a weighted graph, which is used for searching shortest routes accessible for different types of users including people and mobile objects is presented. The created graph contains information about the topology and accessibility between building spaces. It is created using the parameter specifying the permissible distance from the center of a moving object to a wall. This value of this parameter is determined on the basis of the profile of persons or mobile objects, which specifies their sizes, preferences and ability to move. Graph nodes represent points corresponding to doors and internal points of rooms with concave shapes. Edge weights are calculated based on the Euclidean distance between these points. On the basis of information encoded in the graph the application calculates the shortest path between designated rooms and creates its visualization.

In future work, while computing the space of movement for a given person or object, the additional obstacles, such as columns, furniture or unsafe places, which reduce this space, will be taken into account. Also adding other modules to the application, such as finding routes through several locations, is planned.

References

buildingSMART (2013). IFC2x3, http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm

IfcOpenShell Academy, http://academy.ifcopenshell.org

Jas, J. (2019). IFC-based path planning, Master thesis, Jagiellonian University, (in Polish).

Jin, C., Xu, M., Lin, L., Zhou, X. (2018). Exploring BIM Data by Graph-based Unsupervised Learning, Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods – ICPRAM, 2018.

Langenhan, C., Weber, M., Liwicki, M., Petzold, F., Dengel, A. (2013). Graph-based retrieval of building information models for supporting the early design stages, Advanced Engineering Informatics, 27, pp. 413–426.

Lee, J., Eastman, C.M., Lee, J., Kannala, M., Jeong, Y. (2010). Computing walking distances within building using the universal circulation network, Environment and Planning B: Planning and Design, 37, pp.628–645.

Open IFC model repository (2010). http://openifcmodel.cs.auckland.ac.nz/Model/Details/110

Llanos, D.R., Gonzalez-Escribano, A., Ortega-Arranz, H. (2014). The Shortest-Path Problem: Analysis and Comparison of Methods (Synthesis Lectures on Theoretical Computer Science), Morgan & Claypool.

Preparata, F.P., Shamos, M.I. (1985). Computational Geometry – An Introduction, Springer-Verlag.