p

THE PROTOTYPE WORKFLOW FOR AUTO-ROUTING IN MEP DESIGN

by

BARTŁOMIEJ TOKARZEWSKI

16TH SEPTEMBER 2020

Word count: 6030

Dissertation submitted in part fulfilment of the

Degree of Master of Science Built Environment:

Smart Buildings and Digital Engineering

Institute for Environmental Design and Engineering

The Bartlett School of Energy, Environment and Resources

University College London

TABLE OF CONTENTS

ABSTRACT **3**

ACKNOWLEDGMENTS **3**

NOMENCLATURE **3**

1 INTRODUCTION **4**

2 LITERATURE REVIEW **6**

3 METHODOLOGY **7**

3.1 OVERVIEW 7
3.2 CONTEXT 8
3.3 PLACEHOLDERS 9
3.4 KNOWLEDGE AND REASONING 11
3.5 STRUCTURE 13
3.6 ALGORITHM 14

4 RESULTS & ANALYSIS **15**

4.1 CONTEXT 16
4.2 PLACEHOLDERS 17
4.3 KNOWLEDGE AND REASONING 18
4.4 STRUCTURE 20
4.5 ALGORITHM 21

5 DISCUSSION **22**

6 CONCLUSIONS **24**

7 REFERENCES **25**

FIGURES

Figure 1: Simple example of a BIM model for pathfinding tests modelled in Revit.....7
Figure 2: Technical room with MEP systems [15].....8
Figure 3: Revit elements and its geometrical representation.....9
Figure 4: Uniform 3D grid.....9
Figure 5: Example of the 2D approximate cell decomposition method.....10
Figure 6: Three iterations of cells from the non-uniform 3D method.....10
Figure 7: Non-uniform 3D grid of edges.....10
Figure 8: Knowledge items identified in previous research [18].....11
Figure 9: From left to right: BREP cylinder, object converted to mesh, bounding box around cylinder.....12
Figure 10: Grid of edges with parameters.....13
Figure 11: Example of a single shortest path found on the graph.....14
Figure 12: Concept of a flow diagram.....15
Figure 13: Prototype workflow created in Grasshopper environment.....15
Figure 14: Simplified structure of the workflow with the run times.....15
Figure 15: Simple example of the Revit model with single path found.....16
Figure 16: Close-up view on loaded placeholders inside Rhino.....16
Figure 17: Loading time for BIM elements.....16
Figure 18: Ceiling void section – non-uniform 2D method with an error.....17
Figure 19: Cell decomposition approximation 3D method with an error.....17
Figure 20: Placeholders with and without alignment.....17
Figure 21: Calculation time for generating placeholders.....18
Figure 22 Result with PiercingWeight equal 50.....18
Figure 23 Result with PiercingWeight equal 1.....19
Figure 24 Graph with a thin partition wall and a thick structural wall.....19
Figure 25 Calculation time for evaluating parameters.....19
Figure 26 Edges after maxD filtering.....20
Figure 27 Run time for graph operations.....20
Figure 28 Chart with calculation time for shortest path algorithms.....21
Figure 29: Revit error for wrong connection – faulty bending radius.....22
Figure 30: Close-up view on path generated inside Rhino.....23
Figure 31: Two paths with equal lengths.....23
Figure 32: Node splits [7].....23

ABSTRACT

Drafting and coordination in MEP design is a multi-objective, manual, time consuming and complex process. The MEP design is being improved by integrating and streamlining processes with automatic or semi-automatic scripts. Drafting and coordination are one of the last standing processes that are not automated yet. A workflow for automatic pathfinding in MEP BIM environment is proposed and implemented. Graph theory and shortest-path algorithms were used to find a path that obeys three rules: shortest, collision-free and minimal wall penetration count. MEP knowledge items were stored in the graph as dictionaries. Edges that did not meet constraints were filtering out from the graph and objectives were met by estimating the cost of edges so that the algorithm can find a path with the smallest cost. Multiple methods were implemented to create an effective and efficient workflow. The workflow created can serve as a good basis for other engineers to experiment and add more functionalities by codifying knowledge and reasoning that they use in their projects.

Keywords: shortest, path, pathfinding, routing, graph, workflow, BIM, MEP

ACKNOWLEDGMENTS

I would like to acknowledge lecturers from Smart Building and Digital Engineer course. Dimitrios Rovas for asking the most thought-provoking questions and sharing insights on how the future, technology and industry might look. Dr Ivan Korolija for explaining in detail existing engineering elements and their interaction between each other. Dr Yi Zhang for teaching me how to create useful models, ways to test and optimise them to design better HVAC systems. Additionally, I would like to thank Dr Yi Zhang for guidance during writing my thesis.

NOMENCLATURE

AEC – Architecture, engineering and construction

BIM – Building information modelling (model)

BREP – Boundary representation

HVAC – Heating, ventilation and air conditioning

MEP – Mechanical, electrical and plumbing

VDFP – Visual data flow programming

1 INTRODUCTION

Context

Designing MEP systems for a building with the whole lifecycle in mind which is governed by hundreds of objectives set by multiple stakeholders is a complex and integrated process. “MEP systems must meet multiple objectives and criteria for design, installation, commissioning, operation and maintenance.” [1] Additionally, we want to create sustainable buildings with more functionalities. MEP engineers know what possible good solutions are, but due to limited time, budget or tools capabilities it is not feasible to discuss every possible objective and evaluate it. It forces engineers to choose only the most important objectives and use hundreds of simplifications which may lead to the performance gap.

BIM process gives a possibility to work on a single model in an integrated way so all stakeholders can communicate its design and objectives more efficiently. In this integrated process, many engineers are creating BIM models that have the structure of a database with the 3D representation of objects. However, calculations and analysis are very often decoupled from the BIM model as available software tools very often lack interoperability by design. Software manufacturers very often want to keep an engineer inside their tool ecosystem creating disconnected or centralized tools. [2]

Not only tools but also knowledge and reasoning are contained within minds of experienced specialists as in separate silos. Nowadays integrated design involves charrette meetings for rapid information exchange and efficient communication. But the process is still a manual, time-consuming multidiscipline effort that is very often repeated multiple times due to the design changes and its nature. Even with the newest tools and with experienced MEP engineers most of the time is spent on transferring data from one tool to another, drafting and coordination.

Challenges in the industry

MEP knowledge is contained within silos. Design and coordination are manual and time-consuming processes. Although today’s building design is more and more digital and many workflows are automated, drafting and coordination is one of the last standing processes that are not automated yet.

Research question

How MEP criteria could be codified?

Research aim

Create a prototype for the auto-routing workflow of mechanical, electrical and plumbing design in building information modelling process with efficiency in mind. The prototype will be built in a way so that it can be open-sourced and shared with engineers to enable codification of MEP design and coordination criteria.

Objectives

Algorithms must connect to existing workflows and be general enough to support multiple possible rules that govern different MEP systems and find paths based on specific criteria for each system individually. The workflow must be efficient enough to be considered useful in practice. Continuous efficiency comparisons are needed to optimise for both accuracy and speed of the final workflow. To narrow research question, this dissertation focuses on finding a single path from one source to one target point and is limited to only few MEP criteria.

To achieve that, we need to understand and link together five elements:

- **context** that has geometry and properties, which we can analyse and traverse in with our systems,
- **placeholders** that could represent possible locations for the MEP elements,
- **knowledge** and reasoning of the MEP engineer and a method to codify it,
- **structure** that could contain both possibilities and constraints,
- **algorithm** that could traverse in this space and find an optimal path for the given criteria.

Overview of the dissertation

Context sets the stage and presents the current situation in the industry. Challenges were synthesised and main research question was presented followed by research aim. Objectives were the milestones developed in the dissertation and are described in further sections.

The literature review presents current trends, an example of good digital practice, current manual practice and efforts of other researchers in the auto-routing in design and coordination domain.

The methodology describes in detail five steps that were taken to create the whole workflow using known concepts such as BIM, approximate cell decomposition and graph theory with shortest path algorithms.

Results and analysis present visual data flow programming concept which was used to develop the prototype workflow. Results and issues that came from the chosen methods are discussed and measured runtimes are presented for each step from the methodology.

Discussion section comments on issues and proposes possible solutions that were developed by other researchers.

Conclusions section encapsulates what was created, what can be done with the prototype and presents the finding of the whole research.

2 LITERATURE REVIEW

Digitalisation in AEC provides greater efficiency, speed and the possibility for innovation by providing streamlined operations through integrated workflows, collaboration between organizations, real-time response and up-to-date information [3]. Good examples come from “Algorithms for MEP Design” created by Sigma-AEC [4] where they show streamlined semi-automatic workflow with data mining, air balance calculations, zone creation, placing airside equipment and diffusers, system creation, pressure drop calculations and setting zone equipment for a building calculated within one minute. The next step would be to create the layouts with connections between MEP components.

The complexity of creating MEP layout depends on many criteria such as design stage, the advancement of the model and multitude of system design standards, rules etc. Currently drafting and coordination are performed manually by teams of engineers. There are examples of current practice in MEP coordination where projects have millions of clashes. Engineers use Navisworks from Autodesk for reporting and managing collisions, but it does not have capabilities to resolve even simplest collision inside Navisworks. Similar services for coordination and clash detection are available in the cloud such as BIM 360 or Colibri Model Checker. [5] Even if Navisworks could resolve collisions, models that are loaded into it are very often static exports. Additional plugins are used to connect those reports and resolve collisions collaboratively in BIM tools.

Built-in Revit functionality called “generate layout” is too simple to be considered useful in practice. It does not avoid collisions and the user must specify manually offsets. Finding collision-free path, that meets multiple objectives from plant room or roof through shafts, ceiling voids to spaces or terminals is not a trivial task.

There are researchers that studied the subject of finding a path in buildings or ships using graph theory and shortest-path algorithms. [6,7,8] However, the developed methods are not widely used in the industry. Shin-Hyung Kim and his team concluded that “pipe routing algorithms generally do not consider the knowledge and the preference of the designer suitably as required in the actual design work”. [8] This type of limitation is not related only to the algorithm itself. It is rather a matter of knowledge representation [9].

In 2003 Thomas M. Korman published the paper “Knowledge and Reasoning for MEP Coordination” [10] where he pointed out that there is no “knowledge-based computer technology to assist in the multidiscipline MEP coordination effort” and captured “knowledge related to design criteria, construction, operations and maintenance of MEP systems” to apply this knowledge in future computer tools to assist designers in resolving coordination issues. Routing MEP and coordination is a sequential process that needs to meet multiple design criteria and follow rules, guidelines, knowledge and reasoning [10,11]. Writers encapsulated design criteria and intent with construction issues, operation and maintenance into knowledge framework and created a heuristic reasoning structure with which they present how to resolve actual, extended, functional, temporal and future interferences. Creators underlined that “routing knowledge was not represented sufficiently and should be studied and formalized in future research.” [10] Routing and geometry knowledge were further developed by Tabesh and Staub-French [12].

“Developing a knowledge framework from large domains is difficult; the larger the domain, the more difficult it becomes to create a reasoning structure.” [9]. Although creating a collision-free MEP layout that meets multiple criteria with procedural graph traversal algorithms may be too complicated and not scalable enough to be implemented in practice. First rules, guidelines, standards and other criteria need to be codified by a set of functions so that more complicated behaviour can emerge.

Hsieh-Chih Hsu and his team combined a knowledge-based system with machine learning and heuristic optimization to resolve design clashes in the BIM model. [13] They developed an effective system for resolving physical clashes using machine learning and heuristic optimising techniques.

3 METHODOLOGY

This section outlines the process of creating the workflow. Overview encapsulates the concept and then the workflow is described in detail as a continuous process that integrates five elements: context, placeholders, knowledge, structure and algorithm.

3.1 Overview

Understanding the problem

The workflow must find a route from source to target automatically. The route must be a collision-free path that tries to meet an engineer's criteria.

Input definition

- A context that contains structured data for spaces, walls, floors, ceiling, systems etc.
- Source point, target point and pipe/duct/conduit size.
- Engineers criteria for MEP system e.g. collision-free, shortest path, avoid wall penetration

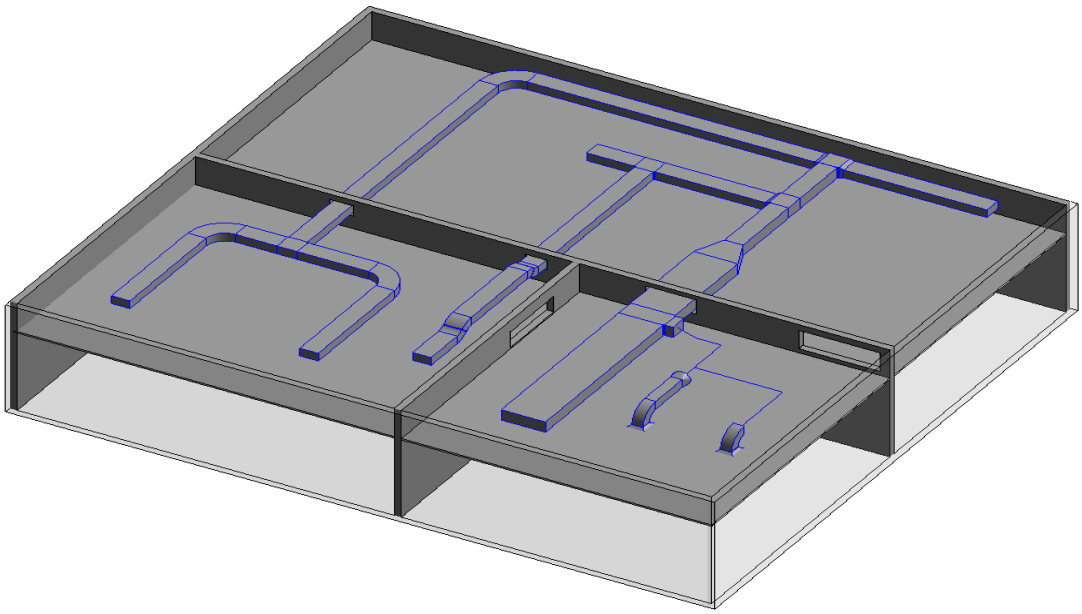
Output definition

Route from point A to point B returned as a MEP object that meets engineers' criteria.

Input example

Generic one storey building with three spaces, three ceiling voids, walls, floors, two free wall openings and ducts was generated to have a simple model for the study. Note: some ducts are modelled as duct placeholders.

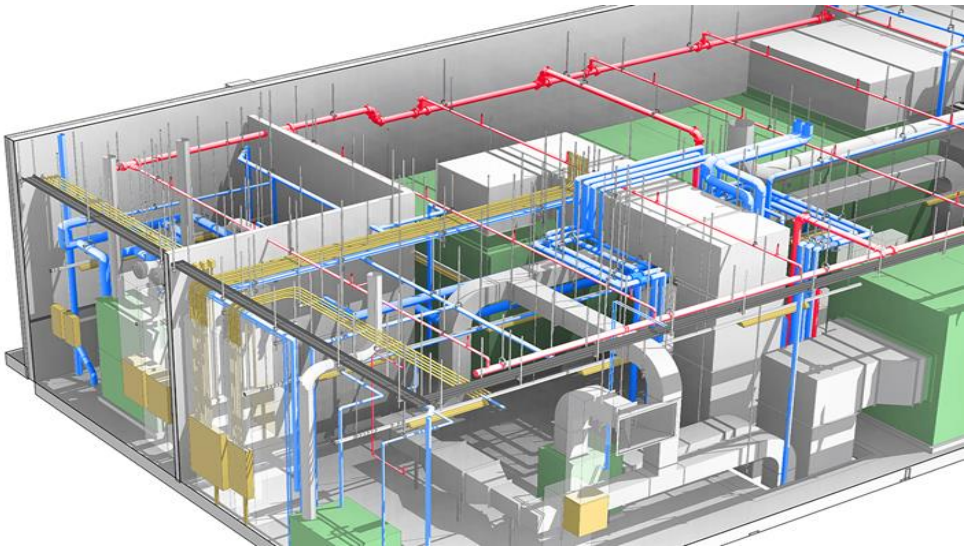
Figure 1: Simple example of a BIM model for pathfinding tests modelled in Revit.



3.2 Context

The context is our building and all elements it is composed of. It should have both geometry and properties which we can analyse and use in further steps to codify MEP criteria. Building information model (BIM) have well-structured data that we can use to retrieve our context. BIM is a digital 3D model of a building that stores information that is generated typically during the design stage and needed for the construction stage. The complexity of the model (context) follows the level of detail, building size, building type, design stage and diversity of stakeholders and specialists. BIM is composed of many elements that contain data, which can be grouped and classified according to known schemes, taxonomies, standards, ontologies and used to understand the context. Fig 2 represents a BIM model created in a level of detail (LOD) 350, which stands for coordination model that has MEP systems needed for coordination modelled in 3D with hangers and openings. [15]

Figure 2: Technical room with MEP systems [15]



There are many software tools for BIM models such as AECOSIM, ArchiCAD, OpenBuildings Designer, Revit, Tekla, Vectorworks etc. [16] There is also open format for BIM interoperability between multiple stakeholders called Industry Foundation Classes (IFC) that have a well-structured ontology. There are many software tools and file formats for BIM that would be useful for loading the context.

Revit BIM software tool from Autodesk was used to create simple BIM model as it is free for students and connects to visual data flow programming environments such as Dynamo and Grasshopper which are ideal environments for rapid prototyping and interoperability.

Grasshopper was chosen over Dynamo as it is more intuitive, has a bigger community and has a node structure that works better with nested lists. With Rhino.Inside.Revit and Grasshopper, it is possible to load data from the Revit model into Rhino 7 as BREP objects.

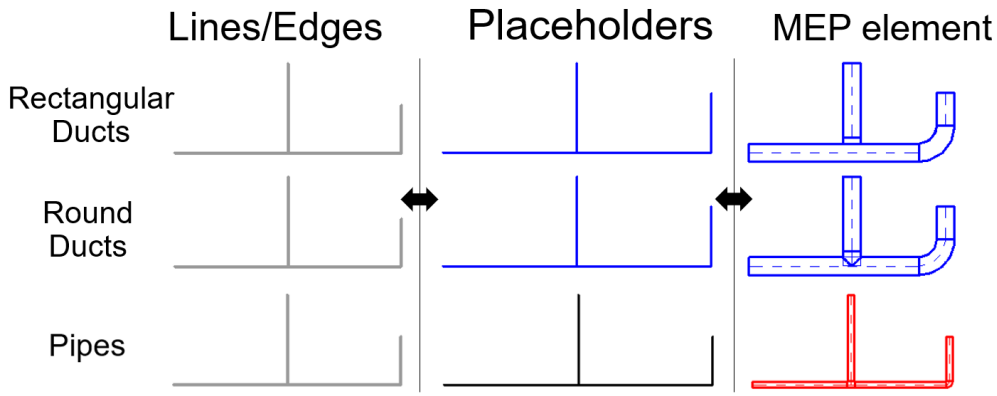
Each element from a BIM model that has 3D representation must be grouped into four categories:

- easily traversed though (e.g. ceiling voids, shafts, technical room, above the roof)
- objects that can be pierced perpendicularly to the plane (e.g. walls, floors, ceilings)
- cannot be pierced (e.g. external walls, windows, doors)
- cannot be pierced but can be moved (e.g. ducts, pipes, air terminals, VAV boxes)

3.3 Placeholders

From the context, we know where we can potentially place our systems. In this dissertation, we can traverse between 3 ceiling voids. These volumes than need to be populated with edges that will be our placeholders to represent future MEP elements. MEP elements can be converted into placeholders and edges or vice versa. In this process, we are losing parts of information or we need to supply information to do the transition.

Figure 3: Revit elements and its geometrical representation

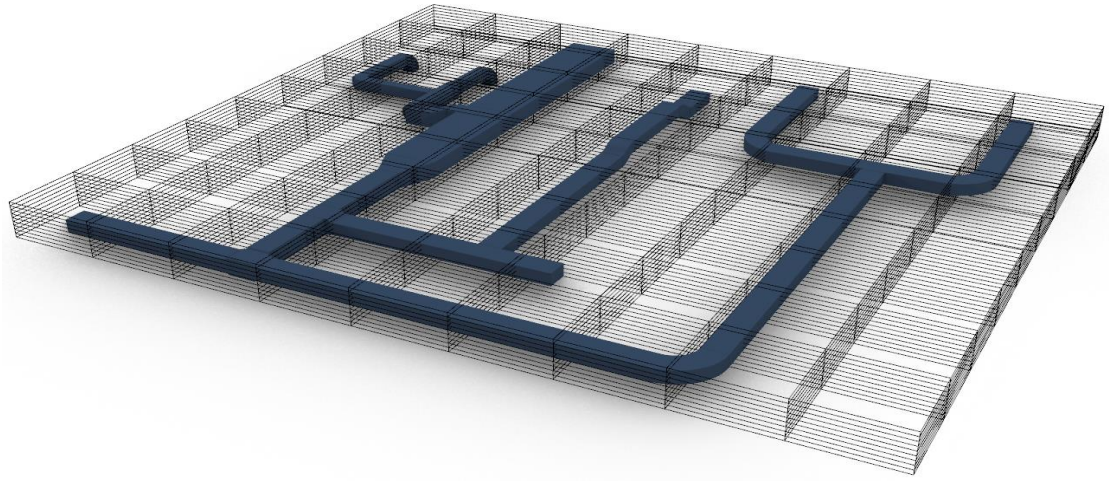


For populating volumes two methods were implemented, simple uniform method and non-uniform method. Many edges from the first method collided with the context and did not meet the first engineering criteria “collision-free”. It was important to run interface checks for objects that do collide and remove them. It was concluded that flexible distribution with a variable distance between edges would be useful, as it would yield more edges in tight areas and fewer edges in open areas. The second method was created as an answer to the above issue and represent available space better.

Uniform method

Edges were created by simply dividing bounding box uniformly around all three ceiling voids and deconstructing 3D cells to retrieve their edges. This method created edges that were evenly distributed in the volume and pierced MEP objects. If we were to use too big division for example 5 m, we may not find any path between two objects. If we would have a very small division for example 1 mm, we may end up with too many edges. It would be difficult to evaluate and calculate it in a short period. Additionally, we need to remove edges that are colliding with our existing objects in the model. This method is fast, easy to understand and easy to implement but it also has some limitations.

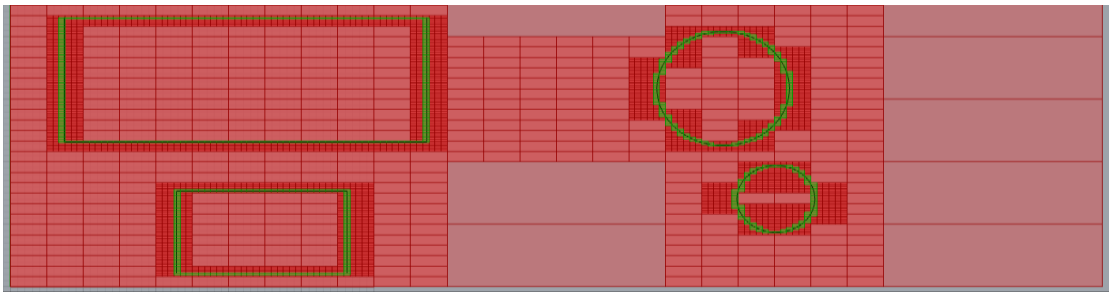
Figure 4: Uniform 3D grid.



Non-uniform method

The second method is proposed and resolves two issues from the previous method. A method similar to the approximate cell decomposition (ACD) was used. “The ACD method is also called “quadtree” decomposition because a cell is divided into four smaller cells of the same shape in each iteration.” [17] As intended this method creates variable distribution by creating smaller cells and edges around objects. Presented 2D method [Fig. 4] is based on the ACD method with a small adjustment that it is not fixed with four divisions for cell, but it gives a possibility to play with the number of divisions in each iteration. As a result, non-uniform method cuts down overall edge count and increases the density of edges in tight regions.

Figure 5: Example of the 2D approximate cell decomposition method.



This method was changed to be applicable in the 3D environment by using cubes generated with Voxeltools module for Grasshopper [Fig. 5]. Note: additional edges had to be added perpendicular to walls to create routes for MEP objects that would pierce walls as this method removes all objects that collide with the context. In this dissertation only 90° elbows and bends are possible. Settings used for 3D method in Fig. 6 were 6,6,2 divisions in x, y, z coordinates for 1st, 3,3,1 for 2nd and 2,2,2 for 3rd iteration.

Figure 6: Three iterations of cells from the non-uniform 3D method.

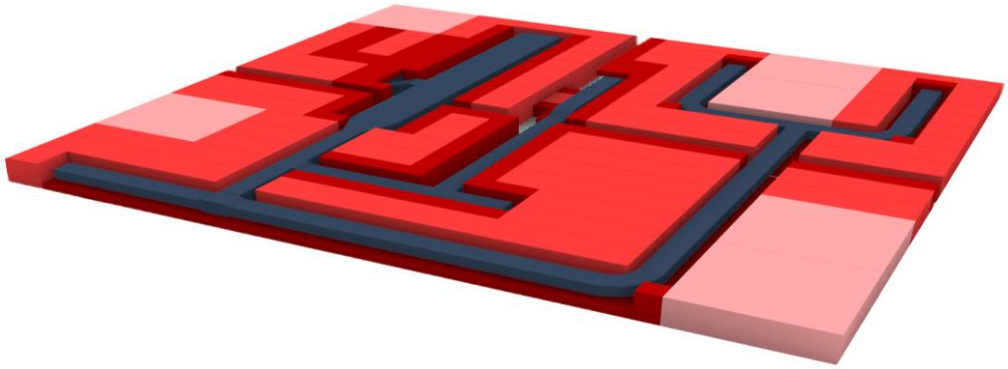
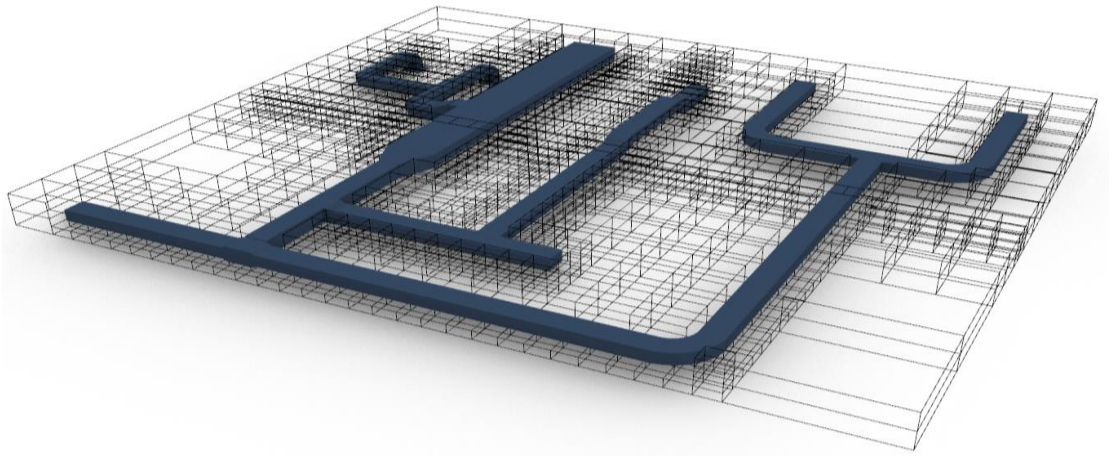


Figure 7: Non-uniform 3D grid of edges.



3.4 Knowledge and reasoning

MEP systems must obey rules, guidelines, standards, knowledge and reasoning of MEP engineers and other stakeholders. Each placeholder must be checked according to the given criteria. Sample of different types of criteria are described in Figure 8 but are not limited to. As MEP criteria must be codified. It was proposed to create functions to calculate values that would represent these criteria. A parameter with value was created for placeholder so that the algorithm can optimise path by the given values.

Figure 8: Knowledge items identified in previous research [18]

Phase	Attribute	Explanation	Korman et al. [10]	Tabesh & Staub-French [24]
Design	Function	Primary performance function of component	✓	✓
	System	System to which component belongs	✓	
	Material type	Material or choices of material used for specific component	✓	
	Material cost	Cost of component as per vendor data or estimating standards	✓	
	Supporting system	Typical system used to support component	✓	✓
	Insulation	Insulation type and thickness of particular component	✓	✓
	Clearance	Design clearance requirements of components	✓	✓
	Slope	Required slope for component	✓	
	Aesthetic	Aesthetic constraints		✓
	Performance	Performance-related constraints		✓
	Installation space	Space for installation of components	✓	✓
	Installation sequence	Typical installation of components	✓	✓
Construction	Lead time	Average lead time for fabrication of component	✓	
	Tolerance	Difference between design and as-built in architectural systems		✓
	Fabrication details	Fabrication constraints that reflect the practice of industry		✓
	Safety	Safety constraints		✓
	Variance	Difference between design and as-built in MEP systems		✓
	Productivity	Productivity constraints		✓
Operations and maintenance	Access space	Space required for operations and maintenance	✓	✓
	Access frequency	Access frequency required to maintain component	✓	✓
	Performance	Performance-related constraints		✓
	Safety	Safety constraints		✓
	Space	Space consideration imposed to ensure that systems are operational		✓

Only few knowledge items were chosen and grouped into objectives and constraints for the final path:

Objectives

- shortest
- minimum number of walls being pierced

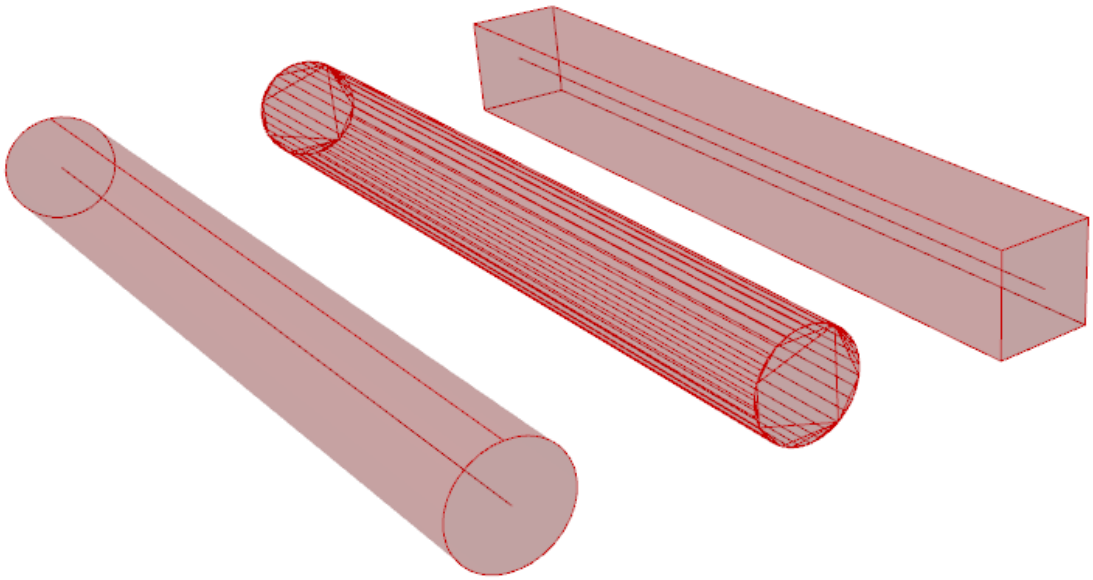
Constraints

- collision-free
- continuous path

In this dissertation, placeholders store two parameters: **max diameter size** (**maxD)** to indicate what is the biggest size of an object that would fit without colliding with other objects and edge **cost** to quantify objectives.

Max diameter size parameter was established by running multiple collision tests on a set of different sizes for each edge from step 3.3. MaxD value is equal to the biggest size that would fit without colliding. Collision check function converts BREPs to mesh geometries and then checks for the collision between meshes. Cylinder represented as a mesh has 88 surfaces and bounding box has 6. Bounding boxes were collided rather than cylinders with the context as it would be much faster to collide fewer surfaces and safer as conversion to mesh simplifies the geometry and make it smaller and the bounding box is bigger than the cylinder.

Figure 9: From left to right: BREP cylinder, object converted to mesh, bounding box around cylinder.



If we were to find paths for rectangular ducts or cable trays and use a similar approach by creating differently sized cuboids. We could store two parameters with maximum height and width and it still would not be precise enough to represent all standard sizes. We would probably need a matrix of rectangular surfaces that could be extruded along the edge.

For most placeholders cost value is equal to the edge length. Placeholders that pierce walls have cost updated and equal to length multiplied by PiercingWeight.

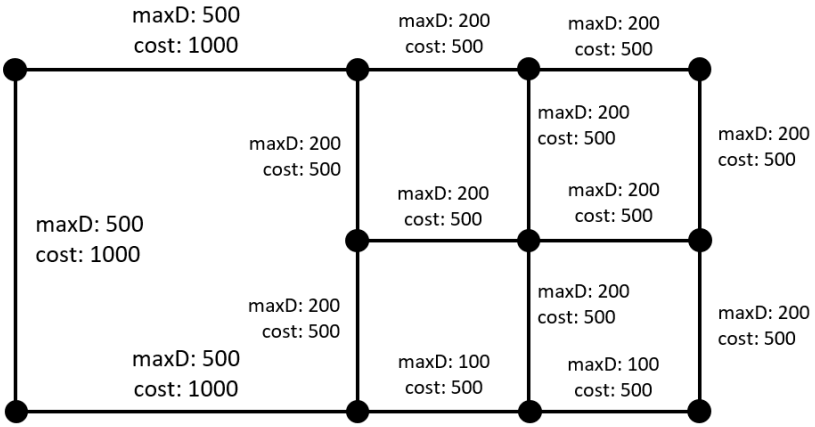
$$Cost := Length * PiercingWeight$$

PiercingWeight must be above 1 to create a penalty and multiple the cost. PiercingWeight was set to 50 and was found by pure experimentation just to mimic cost proportion between pipe per meter and wall penetration cost so yielded results would minimize wall penetration count or would go through openings.

3.5 Structure

In the previous sections, placeholders got the potential location for MEP elements by creating the set of edges and heuristics in section 3.4. These placeholders with parameters had to be connected into a single structure.

Figure 10: Grid of edges with parameters.



Structure visible in Fig 8 is exemplar fragment of our structure which can be represented as a graph. Graphs are mathematical structures used to represent relations between objects, the study of the graphs is called graph theory. The basic construct of a graph is made of two sets: nodes and edges, where the edge is a set of connected nodes. Each edge is a relationship between nodes which can additionally store information. The grid above can be modelled as an undirected graph constructed from 11 nodes, 15 edges and the dictionaries with two keys (maxD, cost) and values for each edge. In this workflow, the graph must be undirected as the directions should not be imposed on edges.

To create the graph in the workflow NetworkX python package was chosen for creation, manipulation and study of the structure as it is open-source, free python package with good documentation. Edges (E) and heuristics represented as a dictionary (D) with python method NetworkX.Graph() was used to create the graph (G). Python 3.8 interpreter and libraries were accessed through Grasshopper plugin GH_CPython.

G = NetworkX.Graph(E, D)

Nodes ID come from x, y, z coordinates so it would be easier to debug.

<**node id**=”(-3430.52608185801, -5000.27766724012, 3166.30000000056)”/>

Default unit of length in SI Revit is millimetre and the same units should be in Rhino. Values loaded had 11-digit precision. Precision during construction certainly is not that precise so node coordinates were rounded to a single millimetre.

<**node id**="(-3430, 5000, 3166)"/>

Edges are constructed from two nodes (vertices): source and target.

<**edge source**="(-15804, 3002, 3312)" **target**="(-15804, 3002, 3166)">

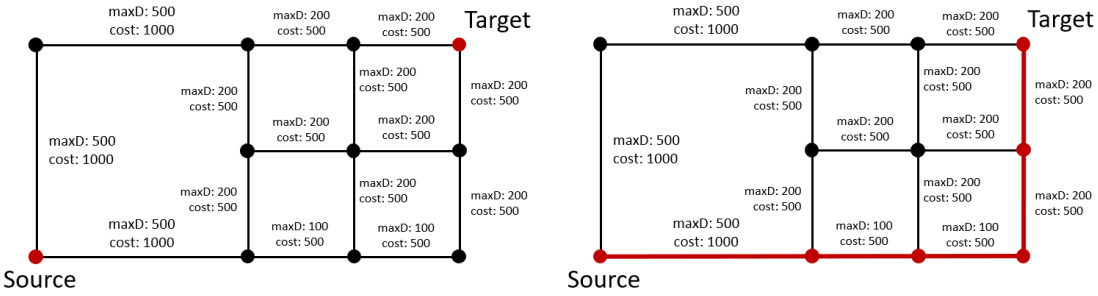
Each edge contains two parameters max diameter size and cost described in detail in step 3.4.

<data key="**maxD**">145</data>
<data key="**cost**">400</data>

3.6 Algorithm

To find a path between two points automatically known graph traversal algorithms were used. Graph traversal algorithms or shortest path algorithms are within graph theory and help with identifying the route that is the best for the given criteria (shortest, cheapest, fastest etc.) between two nodes (source and target) in a network. These algorithms start at the source node and keep searching until they find the optimal path to the target node. There are many shortest path algorithms, only a few were analysed and tested. They were used to find an optimal path for the weighted graph by minimising given objective cost.

Figure 11: Example of a single shortest path found on the graph.



Three known weighted shortest-path algorithms were tested: A-star, Dijkstra and Bellman-Ford. All these algorithms include weights, traverse on the graph to find an optimal solution by minimising whole path cost. A-star, Dijkstra algorithms have a “negative cycle” problem which means that if the edge cost value is negative than the algorithm falls into an infinite loop as it tries to minimize total cost. Bellman-Ford algorithm can detect and report the negative cycle. Edge cost value must be a positive number for the algorithms to find the solution without a stop.

Big O notation is “a theoretical measure of the execution of an algorithm, usually the time or memory needed, given the problem size n, which is usually the number of items.” [19]

Big O notation for the worst-case performance for the tested algorithms depending on the number of edges E and vertices V:

- A-star O(|E|)
- Dijkstra O(|E| + |V|log|V|)
- Bellman-Ford O(|V||E|)

Theoretically, an algorithm could generate a graph and find the optimal path as it would traverse, however, graph generated from that process would need heuristic checks on each new edge. For that reason, the shortest path algorithm is run on the graph with all heuristics already precalculated.

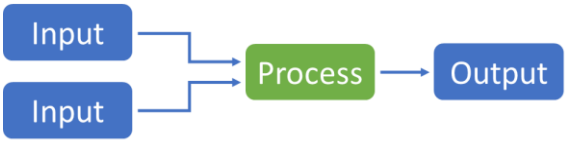
As described in 3.1 overview section, input definition has source point, target point and size. Graph from section 3.5 was filtered by max diameter size input to create subgraph that meets constraints - collision free edges for chosen diameter size. To find a route for DN 50 pipe we need edges that have max diameter size of at least DN50, otherwise, it wouldn't fit. Applying knowledge items for filtering the graph prevents the algorithm from traversing through edges that do not meet design constraints. Shortest-path algorithms traverse through the pre-calculated weighted subgraph from a source point to the target point to find the path with the lowest cost.

4 RESULTS & ANALYSIS

In the introduction, research question was asked: “How MEP criteria could be codified?”. The guess was that by creating a prototype workflow that would connect context, placeholders, knowledge, structure and algorithm it would be possible to create a workflow for codifying MEP criteria. Three knowledge items were codified as functions to constrain the possibilities or estimate the cost to help in evaluating the design objective. Functions were evaluated on a set of placeholders. Chosen MEP criteria were grouped into constraints and objectives. Placeholders that did not meet constraints were filtered out from the graph and objectives were met by estimating the cost of edges so that the algorithm can find the path with the smallest cost.

The prototype workflow for auto-routing of mechanical, electrical and plumbing design in building information modelling process was created in Grasshopper environment as a single streamlined script. Grasshopper is a node-based visual programming environment. Scripts created inside grasshopper are composed of nodes which are connected by wires to create flow diagrams where information flows from left to right. Concept of a flow diagram is presented below [Fig.12] and is composed of 4 nodes and 3 wires. Information flows from left to right. In this example process node uses two inputs to compute single output. There are hundreds of modules with thousands of nodes for Grasshopper with which it is possible to create streamlined processes, workflows and prototype rapidly.

Figure 12: Concept of a flow diagram.



The prototype was created by combining multiple nodes and processes to create a single streamlined workflow. [Fig. 13] Prototype workflow follows methodology structure and its five sections: context, placeholders, knowledge, structure and algorithm. [Fig. 14]

Figure 13: Prototype workflow created in Grasshopper environment.

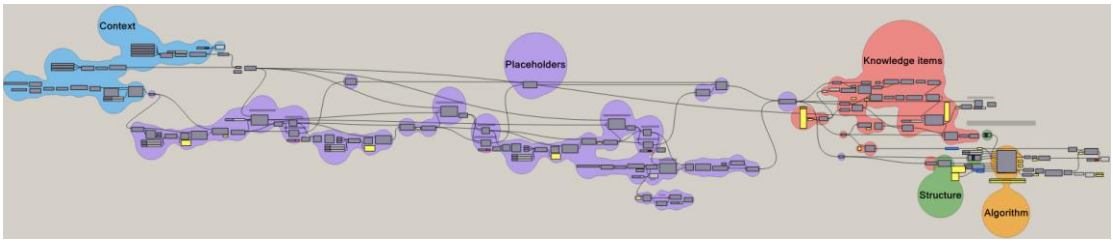
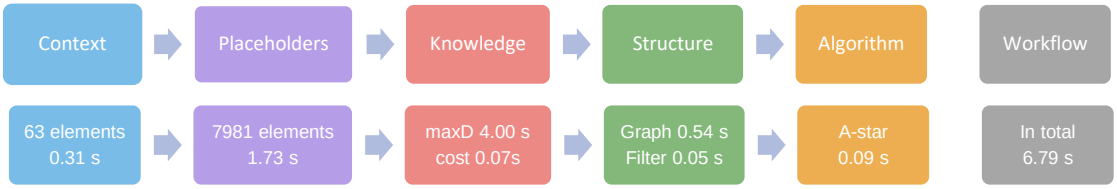


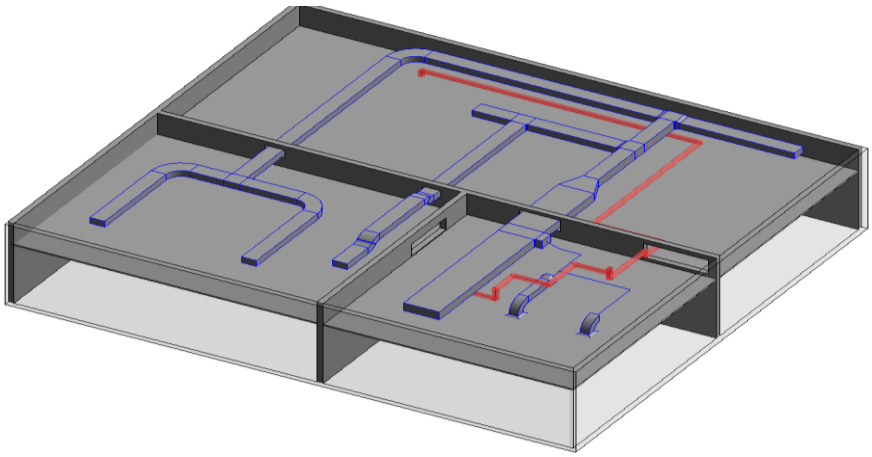
Figure 14: Simplified structure of the workflow with the run times.



Prototype workflow was optimised on multiple levels to generate results faster. Most of the time was spent on creating placeholders efficiently and optimising the creation of maxD parameter as it has the worst big O notation and takes the biggest percentage of the total run time. Shortest-path algorithm can analyse millions of placeholders and find a thousand paths in seconds automatically (once the graph is created) which is much faster compared to the engineer. The figure below presents the total calculation time for a single run of the workflow. Detailed results of calculation time are described in each section below.

Knowledge items of MEP engineer were codified and transferred to the graph. Although there are multiple design criteria in the workflow, used shortest-path algorithms perform only a single objective optimisation based on the cost value. The prototype finds viable paths for the given MEP criteria [Fig. 15], but it also has multiple issues which are explained in the following sections that correspond to the methodology structure. The tests were performed on the simple Revit model described in the methodology section and run on 3.8 GHz AMD Ryzen 3900X desktop computer with 16 GB dual-channel 3200 MHz RAM.

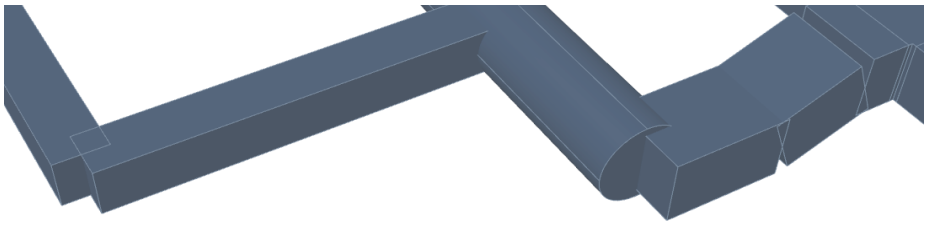
Figure 15: Simple example of the Revit model with single path found.



4.1 Context

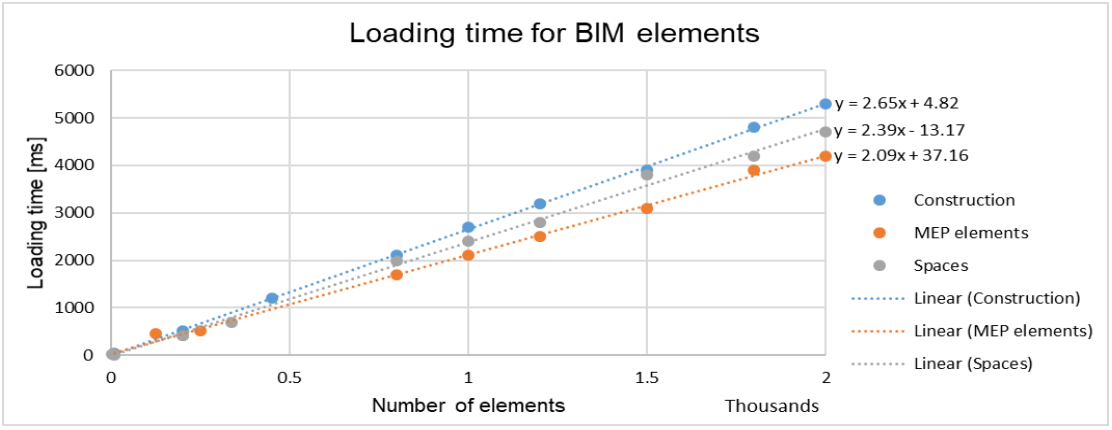
Geometry was loaded correctly from Revit to Rhino and duct placeholders were also represented as volumetric shapes. However, as Revit placeholders do not have connectors, geometry imported may cause issues as they leave empty volumes between elements. [Fig.16] One approach to resolve that would be to convert Revit placeholders into 3D elements.

Figure 16: Close-up view on loaded placeholders inside Rhino.



Big O notation of loading elements from the BIM model is O(n) and takes on average 2-3 ms for each element. [Fig. 17]

Figure 17: Loading time for BIM elements.



4.2 Placeholders

Below Fig. 18 shows some hypothetical section of ceiling void with MEP elements and 2D ACD method. If geometry is inside a cell and both boundaries are not crossing each other than collision is not found as the clash algorithm is looking for the intersection between boundary edges. This can also happen for the 3D method [Fig. 19] as collisions are checked for intersecting mesh surfaces and there was no volume check created. Additionally, when cell size is small enough that it can fit into MEP elements, the algorithm generates cells inside an object. MEP elements should not traverse inside other MEP elements and these cells and edges should be removed.

Figure 18: Ceiling void section – non-uniform 2D method with an error.

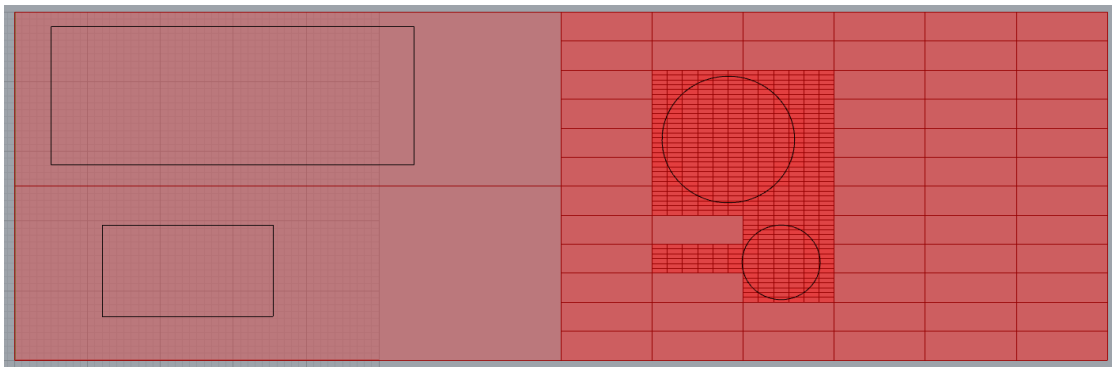
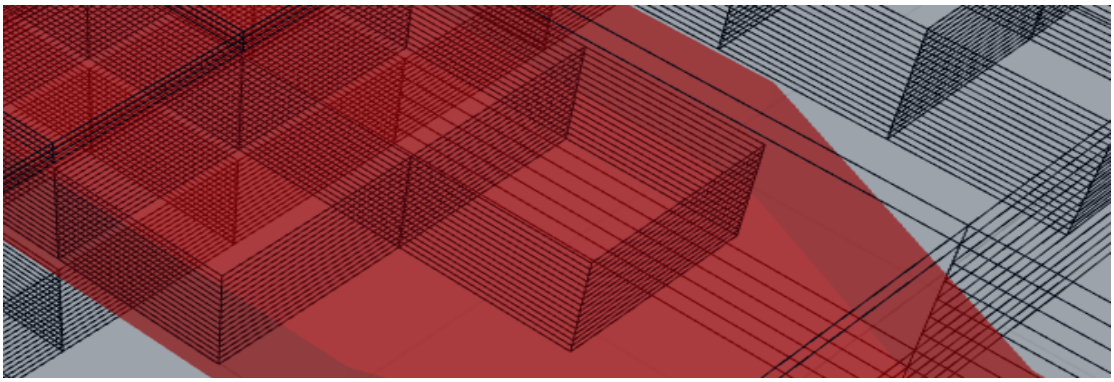
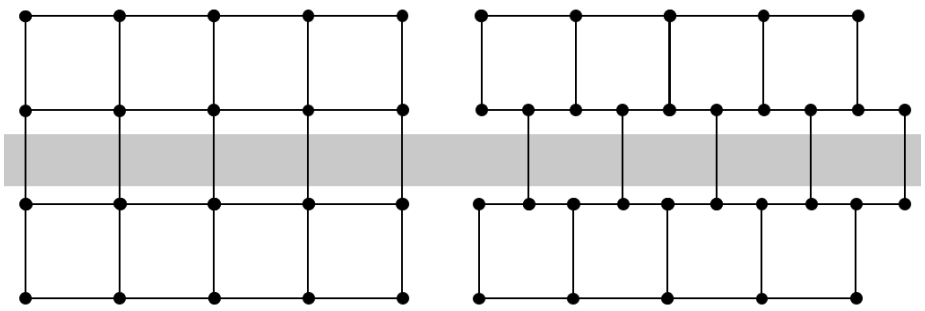


Figure 19: Cell decomposition approximation 3D method with an error.



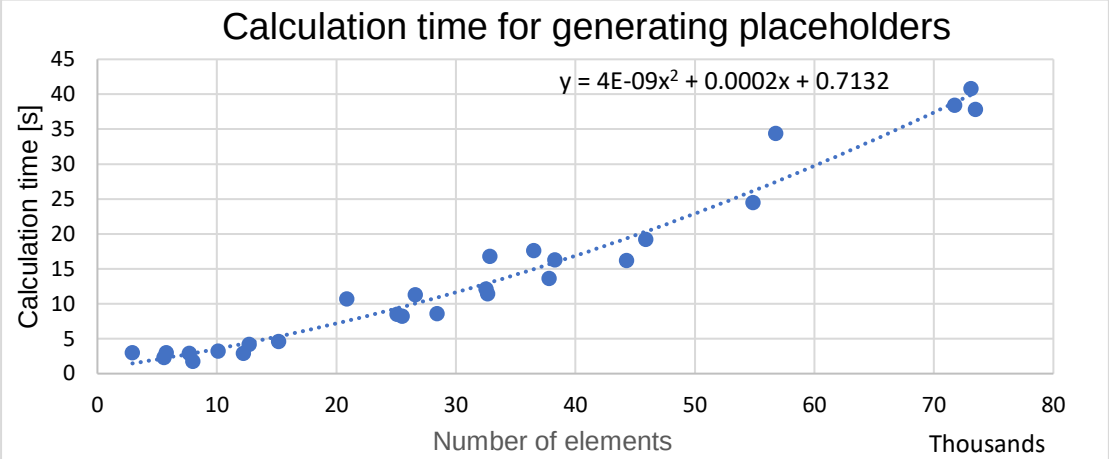
Edges between spaces should be aligned so that fewer bends would be created, and it would be easier to create a graph as each edge would have a set of nodes common to nearby edges. In other case, edges would have to be cut and split by other edges, which would increase calculation time. To do that a single bounding box was created around all spaces to create aligned cells and edges.

Figure 20: Placeholders with and without alignment.



Non-uniform method for placeholders is flexible and gives the possibility to control it with 9 parameters, 3 coordinates in 3 iterations. Calculation time for generating placeholders is not linear and rises exponentially. Although total calculation time depends heavily on the number of placeholders, it was not investigated which settings would be optimal in practice.

Figure 21: Calculation time for generating placeholders.



4.3 Knowledge and reasoning

Paths generated are viable solutions for the given objectives: shortest, collision-free and with minimal wall piercing count path. It was possible to codify MEP criteria and store it using dictionaries for each edge. Storing maxD and cost parameters in the graph by using dictionaries proves to be an effective method for storing MEP knowledge items. Edges (placeholders) that did not meet constraints were filtering out from the graph and objectives were met by estimating the cost of edges so that the algorithm can find the path with the smallest cost.

PiercingWeight is a simplification of the relation between two separate costs: pipe cost per meter and wall piercing cost. The more precise approach would be to calculate and store both values in the dictionaries. Two solutions are presented below: Fig 12 has PiercingWeight equal to 50 and path found is bypassing walls and goes through openings (light green volumes) and Fig 13 has PiercingWeight equal to 1 and the path found is piercing a wall. This simple cost multiplication changes the results of the algorithm so that we can receive paths with fewer walls being pierced. If we would like to favour certain edges, we also could set the weight below 1 to decrease cost. This way we can increase or decrease edge cost and control the result.

Figure 22 Result with PiercingWeight equal 50

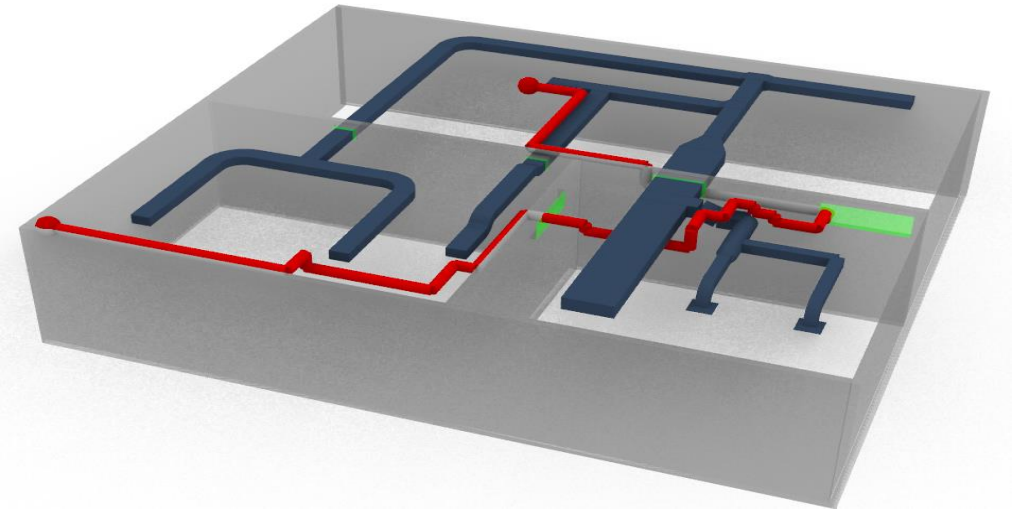
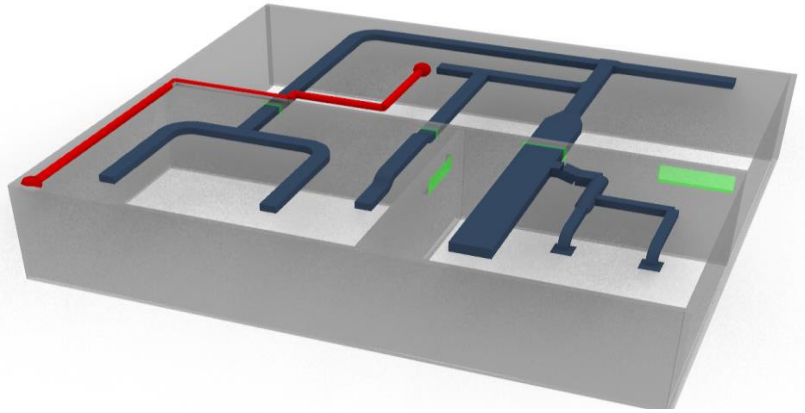
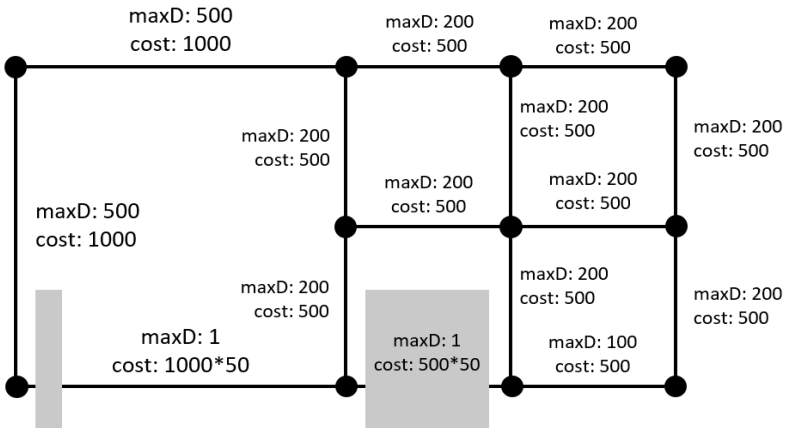


Figure 23 Result with PiercingWeight equal 1



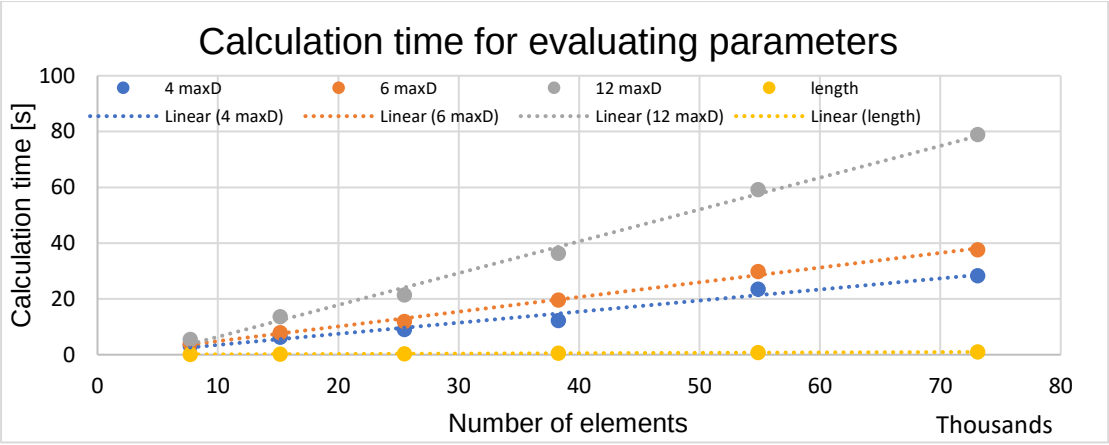
As stated in section 3.4 edges that pierce walls have cost multiplied by PiercingWeight. However, the length of the edge inside the wall is not equal to the wall thickness and neither there is a distinction between wall types. As a result, long edges that pierce thin partition walls may cost more than short edges that pierce thick structural walls. Figure 15 presents a simplified graph of this issue. This could be resolved by checking what is the wall thickness that is being pierced and multiplying only the length that is inside the wall.

Figure 24 Graph with a thin partition wall and a thick structural wall.



Generating maxD parameter is the most compute-intensive task in the whole workflow. Calculation time depends on the number of elements (edges) and the number of diameters that we would like to test. The more diameters and edges the more it takes to calculate, big O notation for calculating maxD parameter is O(|E||D|), where D is number of diameters to test.

Figure 25 Calculation time for evaluating parameters

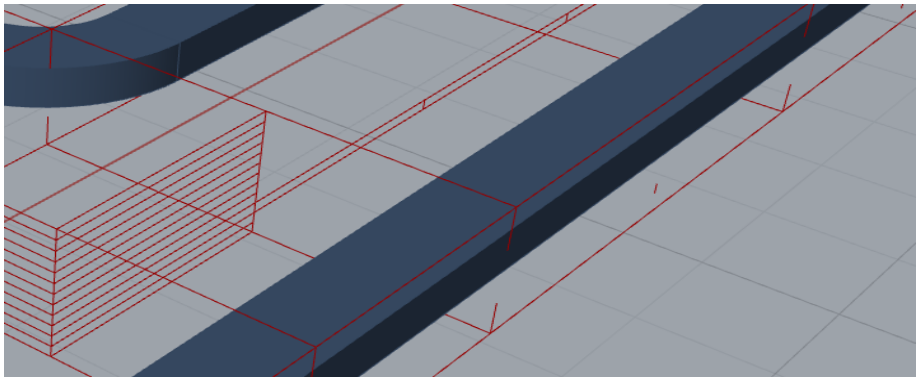


4.4 Structure

The single graph was created from dictionaries and edges. Afterwards, the graph was filtered with maxD parameter to create subgraph. The workflow created takes most of the calculation time to create placeholders, dictionaries and graph. Loading BIM data, filtering graph and running shortest path algorithm takes the least amount of time. [Fig. 26] Although the intent was to create workflow as efficient as possible, the one created with current structure may not be efficient enough to be scaled up on more complex context with more edges and used on a single computer as calculation time increases drastically.

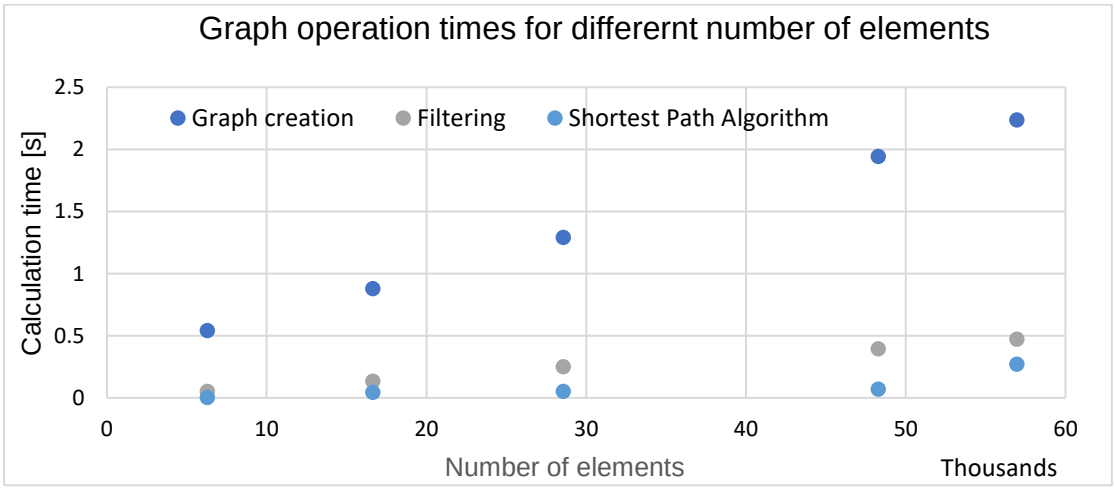
When edges are filtered by maxD parameter some edges are removed from the graph to create smaller subgraph. It may disconnect edges from the main grid. To make sure that the path is found between two nodes the graph should be connected. Note: Connected graph in graph theory means that there is a path from any node to any other node.

Figure 26 Edges after maxD filtering



The first library for creating dictionaries, graph and running shortest-path algorithm was based on Topologic module, “Topologic is a software development kit and plug-in that enables logical, hierarchical and topological representation of spaces and entities” [20] Later was changed to NetworkX module as Topologic wasn’t efficient enough in creating dictionaries. Creating a dictionary for 2706 edges in Topologic took 44.3 seconds, where for NetworkX creating the whole graph with dictionaries took 0.3 seconds. Dictionaries created in Topologic probably weren’t created with module authors best practice.

Figure 27 Run time for graph operations.

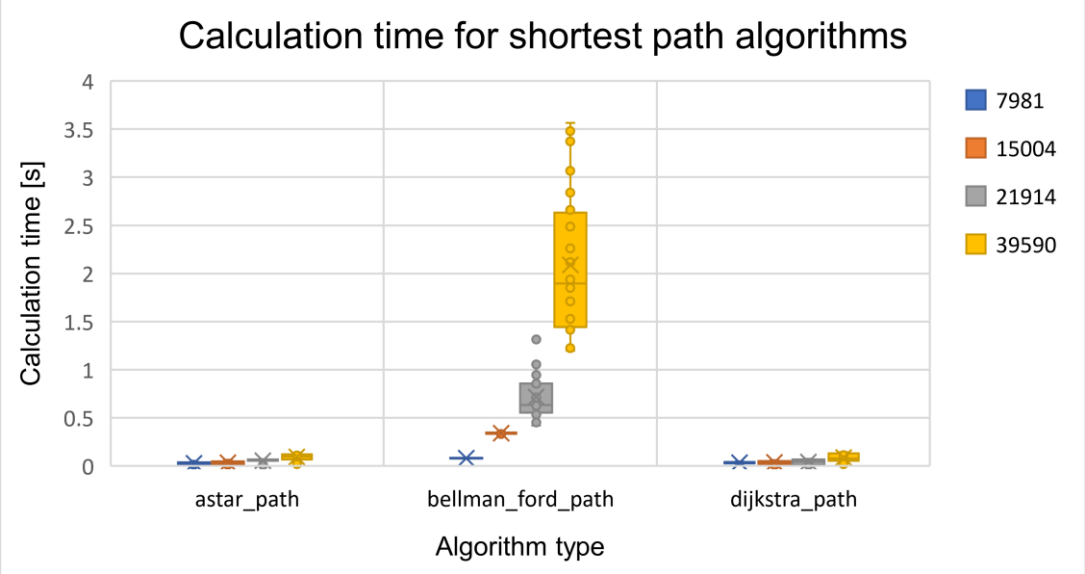


4.5 Algorithm

Shortest-path algorithms take the least amount of time compared to other tasks in the proposed workflow. It is possible to measure the time that was taken to calculate the path, but it depends on the graph given and the chosen source and target nodes. It is difficult to estimate the time necessary for the algorithm to find the “optimal” path. However, it is known that the worst-case scenario and the maximum time spent to calculate the path was expressed by the big O notation in section 3.6. Following the big O notation, A-Start and Dijkstra are faster algorithms than Bellman-Ford, this was also confirmed by running an experiment [Fig. 28].

Paths generated by A-star and Dijkstra are the same in both cost and geometry. Bellman-Ford algorithm generates paths that are slightly different in shape but are equally optimal as all three methods generate paths with the same total cost. This probably happens, because there is a set of multiple equally optimal paths and as Bellman-Ford algorithm calculates it differently, it finds a different path from that set.

Figure 28 Chart with calculation time for shortest path algorithms.



5 DISCUSSION

It was concluded in 4.4 that used structure and workflow is not efficient enough as a whole. However, there might be an approach that could resolve this inefficiency by dividing the graph into smaller chunks and merging only those that are needed. For example, each space and wall could have its placeholders with dictionaries as a single graph. We could run preliminary shortest path algorithm between space and walls to know through which volumes the algorithm must go through. Afterwards, multiple chosen spaces and walls with corresponding graphs would be merged to run the second shortest path algorithm on the stitched graph. This can be potentially done with Topologic. It may be also possible to run compute-intensive tasks on external servers, leaving only user interface on the local machine.

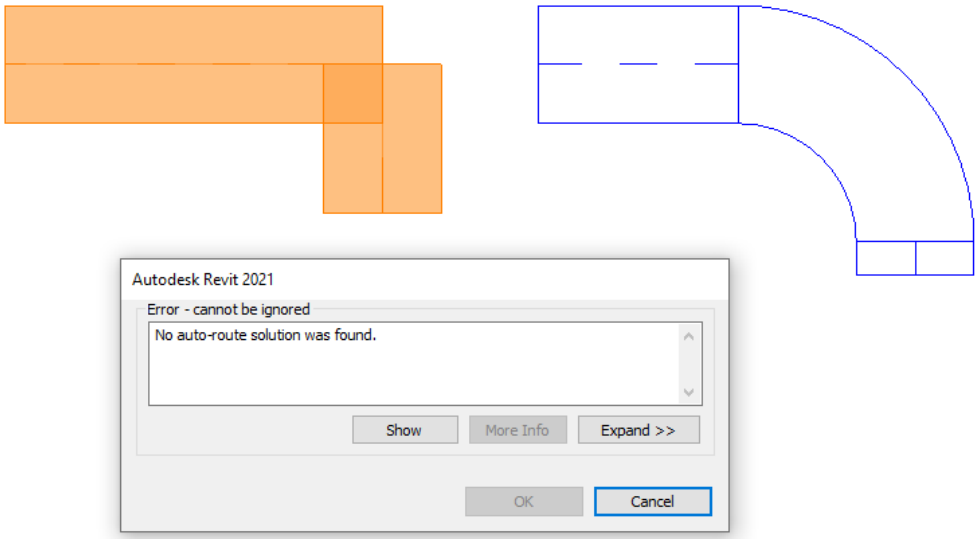
The workflow created focuses on streamlining all processes and codifying MEP criteria so that other researchers and engineers can recreate the described prototype.

Parts of the workflow can be repurposed to generate multiple paths. If a new MEP element will be added to space, the graph would need an update to reflect the current context. Workflow gives a single optimal solution for a single path. However, if an engineer would want to find an alternative path, he might prefer to have a semi-automatic workflow where he could get multiple equally optimal solutions and he would then pick the one that would suit his criteria's the most.

Python is not the fastest programming language, but it enables rapid prototyping by giving access to a multitude of libraries and easy syntax. The whole workflow can be potentially optimised further to yield results faster if run with more efficient language such as C or C++. Grasshopper does not give access to create nodes or scripts in C or C++, but it is possible to call .dll libraries created in different languages.

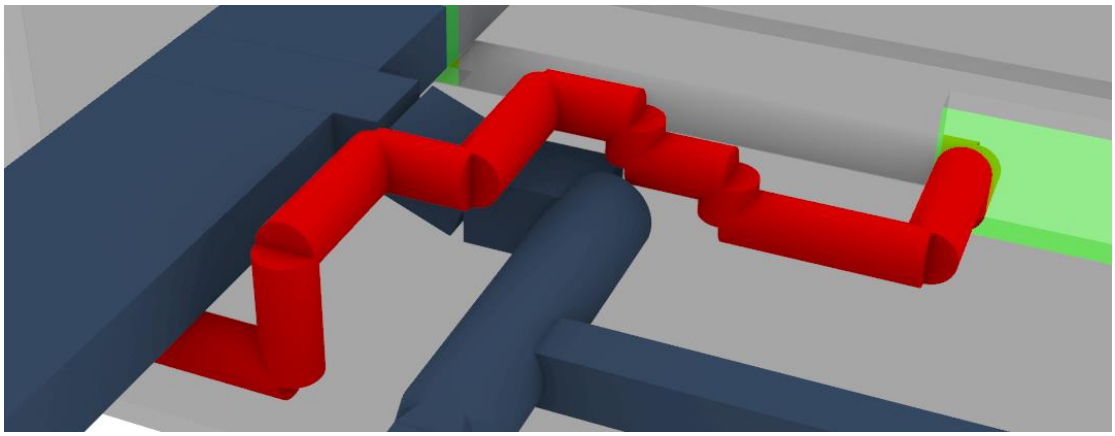
Pipes, ducts, cable trays have specific geometry rules that need to be considered. Pipes, ducts or cable trays have specific dimension rules for fittings such as minimum bending radius for elbows. For example, for duct elbows, there is a minimum bending radius. If the radius is smaller than specified by the manufacturer for that type than connection cannot be created as indicated by the Revit error message. [Fig. 29]

Figure 29: Revit error for wrong connection – faulty bending radius



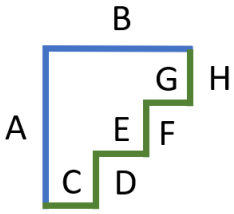
The path generated and graph created does not include knowledge item related to finding a path with minimal pressure drop. As a result, generated paths have too many bends.

Figure 30: Close-up view on path generated inside Rhino.



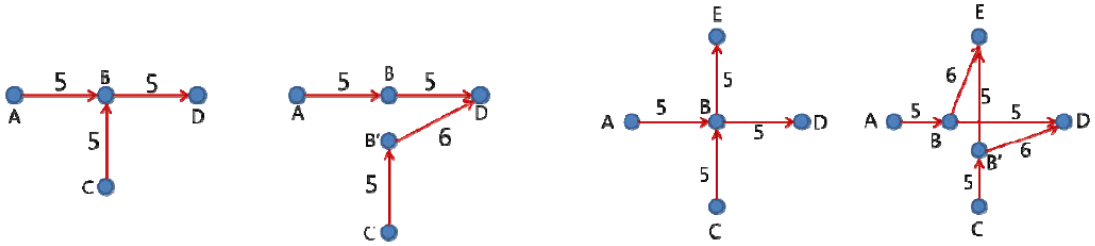
For the created prototype both paths A-B and C-D-E-F-G-H [Fig. 31] have equal costs as they are equal in length. If we were to choose a path with minimal pressure drop, we would choose path A-B as it has fewer bends.

Figure 31: Two paths with equal lengths.



Shin-Hyung Kim and his team [8] proposed a solution to this issue by splitting node and creating additional edges which removed the ambiguousness of node connections as they had different cost value associated to them. [Fig. 32]

Figure 32: Node splits [7]



Next limitation comes from simplification that there are only perpendicular edges, where in construction it is common to have bends with 15, 30, 45, 60, 75 degrees or other custom solutions.

The graph can be saved in multiple file formats: gexf, gpickle, graphml, sparse6, pajek. Possibility to store graphs in a separate file gives the ability to load them in a cloud environment where it is possible to run more compute-intensive tasks.

6 CONCLUSIONS

The research aim of this dissertation was to answer the question “How MEP criteria could be codified?” by creating prototype workflow for auto-routing in MEP design in BIM processes.

This dissertation describes developed prototype workflow that includes codified MEP criteria. Those are included in the created streamlined workflow composed of context, placeholders, knowledge, structure and algorithm. Final workflow loaded context from simple BIM model, generated placeholders with the non-uniform method, knowledge items were codified and run for each placeholder, placeholders were saved as edges into a graph with knowledge items as dictionaries, a graph was filtered to remove placeholders that did not meet constraints and shortest path algorithms was run on subgraph to find the path with the lowest cost.

The workflow created help in codifying MEP criteria, finding new paths, alternative paths or for evaluating existing MEP paths. Grasshopper environment enables easy access for editing and adding more criteria to further develop and test the workflow.

The current study had only three heuristics implemented: shortest path, collision-free and minimal wall piercing count. Other MEP criteria can be possibly codified using a similar approach (filtering and cost estimation), however, there are also knowledge items that may involve changing the shortest-path algorithm or structuring graph connections differently. Additional heuristics need to be codified so that these algorithms can yield better results. The cost function used was trivial, but the principle was correctly mimicked.

As cost estimation is not always based on simple linear functions, the more criteria will be codified the more complex estimation functions will have to be used. For complex auto-routing systems with multiple heuristic Hsieh-Chih Hsu optimisation [13] can be possibly applied and transferred from coordination to design.

MEP design is a complex and multidisciplinary process that is very specific as each system is governed by different criteria. Many tools are closed from sharing or it is hard to extract data and information from them. Nowadays with BIM processes and digitalisation tools with possible integration are very desirable. Tools and software engines that have application programming interfaces (API) with good documentation are of essence for speeding integration. It was possible to prototype rapidly in visual data flow programming environment Grasshopper using available Revit API and creating custom nodes that use external python module for graph operations.

7 REFERENCES

1. Tatum, C. B. and Korman, T. (1999) MEP Coordination in Building and Industrial Projects, Center for Integrated Facility Engineering.
2. Mackey, C. and Sadeghipour Roudsari, M. (2018) 'The Tool(s) Versus the Toolkit', in *Humanizing Digital Reality*.
3. Stoyanova, M. (2020). Good Practices and Recommendations for Success in Construction Digitalization. TEM Journal, 9(1), pp. 42-47.
4. Sigma-AEC (2019), *Algorithms for MEP design*, https://www.sigma-aec.com/post /algorithms-for-mep-design
5. Mehrbod, S. *et al.* (2019) Beyond the clash: Investigating BIM-based building design coordination issue representation and resolution, *Journal of Information Technology in Construction*.
6. Park, J. H. and Storch, R. L. (2002) 'Pipe-routing algorithm development: Case study of a ship engine room design', Expert Systems with Applications, 23(3), pp. 299–309.
7. Belov, G. et al. (2019) 'Position Paper: From Multi-Agent Pathfinding to Pipe Routing'. Available at: http://arxiv.org/abs/1905.08412.
8. Kim, S. H., Ruy, W. S. and Jang, B. S. (2013) 'The development of a practical pipe auto-routing system in a shipbuilding CAD environment using network optimization', *International Journal of Naval Architecture and Ocean Engineering*.
9. Carrico, M. A. (1989) Building knowledge systems, J. E. Girard, and J. P. Jones, eds., McGraw-Hill, New York.
10. Korman, T. M., Fischer, M. A. and Tatum, C. B. (2003) 'Knowledge and reasoning for MEP coordination', Journal of Construction Engineering and Management, 129(6), pp. 627–634.
11. Tatum, C. B. and Korman, T. (2000) Coordinating building systems: Process and knowledge', *Journal of Architectural Engineering*, 6(4), pp. 116–121.
12. Tabesh, A. R. and Staub-French, S. (2006) Modeling and coordinating building systems in three dimensions: A case study, *Canadian Journal of Civil Engineering*. Ottawa, Canada: NRC Research Press, 33(12), pp. 1490–1504.
13. Hsu, H. C. *et al.* (2020) 'Knowledge-based system for resolving design clashes in building information models', *Automation in Construction*. Elsevier B.V., 110.
14. Knowledge-based BIM System: Intelligent Combined Services Drawing (CSD) arrangement tool for automated MEP design, https://www.linkedin.com/pulse/knowledge-based-bim-system-intelligent-combined-services-albany-tam/.
15. Anne-Mieke Dekker (2018) How HVAC LOD Levels Impact Your 3D BIM Model, https://constructible.trimble.com/connstruction-industry/hvac-lod-levels-how-they-impact-your-3d-model
16. BIM Software, (2019) https://www.lodplanner.com/bim-software/
17. Katevas, N. I., Tzafestas, S. G. and Pnevmatikatos, C. G. (1998) 'The Approximate Cell Decomposition with Local Node Refinement Global Path Planning Method: Path Nodes Refinement and Curve Parametric Interpolation', Journal of Intelligent and Robotic Systems: Theory and Applications.
18. Wang, L. and Leite, F. (2016) Formalized knowledge representation for spatial conflict coordination of mechanical, electrical and plumbing (MEP) systems in new building projects, *Automation in Construction*. Elsevier B.V, 64, pp. 20–26.
19. Black, Paul E. (2005). Black, Paul E. (ed.). "big-O notation". Dictionary of Algorithms and Data Structures. U.S. National Institute of Standards and Technology.
20. Topologic (2020) https://topologic.app/