COWI

Research Memo: Artificial intelligence in building design

A project sponsored by COWIfonden

COWI fonden

By
Jørgen Grønvold Roven
Hauk-Morten H. Lykke

March, 2025

Attribution

This project was funded by COWI Fonden with grant number F-167.04.

Contents

Acronyms

AI Artificial Intelligence.

BIM Building Information Model.

FEM Finite Element Method.

GenAI Generative Artificial Intelligence.

IFC Industry Foundation Classes. An open BIM file format defined by BuildingSMART.

ML Machine Learning.

TO Topology optimization.

List of Figures

1 Introduction

The last two years have shown beyond doubt that Artificial Intelligence (AI) is here to stay. The disruptive development in this field has led to an increased confidence that problems previously believed too complex and unpredictable to be solved by machines might now be solvable. This study aims to investigate the potential of artificial intelligence in performing cross-disciplinary engineering of buildings.

Artificial Intelligence (AI) is a branch of computer science that focuses on creating systems capable of performing tasks that require a human level intelligence.

In building projects, engineers from various disciplines often work in isolation, focusing on their specific areas of expertise. Although each engineer optimizes their solutions based on physical constraints, they rarely consider the broader implications across all disciplines. This siloed approach is a result of human specialization, which limits the ability to see the project holistically. However, by leveraging Generative Artificial Intelligence (GenAI), there is potential to change this, ensuring designs are optimal for the entire project, not just individual components.

When designing buildings, the most common exchange format today is probably the open Industry Foundation Classes. An open BIM file format defined by BuildingSMART format (IFC). The format is widely adopted in BIM-applications, is human readable (to a certain extent), and the fact that it is open has enabled open programs and libraries to evolve to process the information more efficiently and directly than the older proprietary formats.

The aim of this project has been to create a small working prototype of a tool that given an architectural floor plan can create a first draft of a suitable air supply layout and a structural layout. Optimally these two layouts should be optimized in relation to each other. To make something that is practically useful is outside the scope of this project, but we intend to develop a proof of concept to determine the feasibility of a multidisciplinary generative engineering tool for buildings.

2 Theory

2.1 Ventilation

When designing ventilation layouts, there are several considerations the ventilation engineer must adhere to. An mechanically optimal ventilation layout causes a minimal loss of energy due to air friction, which is measured as loss of pressure from the air handling unit to any room. The economic and ecologic costs must be balanced between a duct layout that is easy and inexpensive to build versus one which is easy and inexpensive to run and maintain. In addition to the engineering considerations for ventilation, their needs must be balanced with the needs for space by other disciplines. It is preferable to avoid crossing walls unnecessarily, but when crossing is necessary, doing it perpendicular to the wall is most practical. Crossing at a different angle is possible, but very unfavorable. Mounting a duct close to other installations, in particular parallel objects such as walls is physically difficult, and also makes it hard to maintain or replace duct accessories. It is therefore favorable with a certain space clearance around ventilation installations in general. Further, air ducts and equipment generate noise. Thus, it is generally favorable to place large main ducts in rooms not intended for long term occupancy like i.e. storage rooms or washrooms, and to avoid putting any more ducts than necessary in rooms like classrooms or offices. For this reason, a layout topology similar to a tree, with a few large main branches with progressively smaller sub-branches is common.

Ducts also conduct sound between rooms, and in case of fire they will also transfer smoke and heat between rooms. Thus, crossing walls with a high sound rating or fire rating has a certain cost. It's possible to mitigate the effects of heat and smoke transfer by installing fire dampers. For sound however, sound attenuators can only do so much.

To adhere to all these needs, it is necessary with frequent and effective coordination between disciplines, which is highly dependent on good communication. When human relations between disciplines in the project are good, this is realistic to achieve, but is by no means guaranteed. Bad communication commonly lead to suboptimal solutions and longer engineering time.

2.2 Structural design

Sketching different options for the layout of a structural system and comparing the best options is common in early phases of structural design. The architectural layout of the building forms the structure's domain. The design process involve choosing the topology, the shape and size of the each members (Ørsted and Baker, 2015). The topology of the structure is determined by the location of structural members (columns, beams, braces etc.) which are often placed systematically. The shape of the system is closely related to the topology. While the topology determine what members are placed, how they are connected and roughly how they are placed, the shape describes the finer details of their orientation. Finally sizing of each member must be done to ensure sufficient capacity. The lecture by Ørsted and Baker (2015) illustrate this well.

The economy of the structural system is mostly determined by the topology and shape. Of course, over-sized members cost more, but proper size is almost always ensured. However, there are not always a targeted effort to find the best topology. This can be explained by a number of project constraints, like project timelines and available software.

The common approach is to rely on experienced engineers to sketch out the topologies and then estimate their efficiencies using a combination of intuition and hand calculations. The accurate method of choice for most engineers is the finite element method (FEM), which takes time to implement. For this reason FEM is deployed in later stages, especially for common buildings design. An alternative to this method is to use topology optimization (TO). This is a mathematical method that optimize the topology based on a set of constrains and performance measures.

2.3 Artificial Intelligence

The subject of artificial intelligence can on a general level be divided in two categories, namely expert systems and machine learning. The different categories differ quite significantly in terms of characteristics. Expert systems use a set of pre-defined rules to provide solutions within a specific domain of knowledge, mimicking human decision-making. The rules require domain specific knowledge. Depending on the complexity of the problems the system is designed for, the system can become very complex with very many rules. Since these systems are defined by explicitly expressed rules, the outcome is normally to a certain extent deterministic and predictable (Radio, 2024). This makes expert systems well suited for well defined problems with known variables and a predictable context. If the number of variables becomes too large, or the relation between different parameters is not explicitly known, an expert system is generally less suitable. Making an expert system handle very complex or unpredictable tasks, or tasks where a human might solve it based on intuition or experience rather than definitive rules, is difficult and in some cases not possible.

Machine learning (ML) on the other hand is a subset of AI that enables systems to learn from data, identify patterns, and make decisions without explicit programming. The subcategory artificial neural networks are systems based on code mimicking biological brains and their neural patterns. These systems can improve over time by adapting to new data, offering more flexible and generalizable applications. As a result, neural networks can solve problems that are not possible to solve by other means. However, machine learning can be resource intensive and very quickly scales with complexity. Large systems are not generally possible to run on consumer hardware. Although some ML-models are well understood, many are not. Thus, it is generally much harder to follow the reasoning behind a result given by an ML model than by an expert system, and it is also much less predictable.

2.4 Pathfinding

When sketching out an HVAC route from one place to another, the process undergone by the HVAC engineer for each branch has certain similarities with the more general problem of finding the best route from one location to another. The problem in an abstract sense is applicable to many branches of science, and is a discipline within computer science known as route planning algorithms. The problem can be described with a network of nodes connected by edges, a graph. The goal of the algorithm is to find the shortest path between a node pair in the graph. Each edge has a "weight" or cost, so the graph can be called a weighted graph. The weight can describe a number of things, but in its simplest form it is the distance between to connected nodes. There are several well known algorithms available that compute the best path. Dijkstra's algorithm works by traversing all possible locations (called nodes) and computing a sum of the edge weights it needs to traverse to get to each node from the origin node. The fastest route is the route with the lowest total cost. This algorithm scales with $\mathcal{O}(|V|^2)$ where V is the number of nodes and $\mathcal{O}$ is the order of magnitude of the number of calculations performed. This makes the algorithm computationally expensive. The A*-algorithm was later designed to avoid computing the cost of node edges that are obviously a worse path. It does this by calculating a heuristic for each node, which can be thought of as an extra cost to consider. The heuristic in this case is the euclidean distance between the current node and the goal node. Each traversed node is placed in a priority queue sorted by the heuristic, where each node explored next is always the first node in the queue. That way, it's always the node found with the shortest distance left to the goal node. This makes the algorithm more efficient, $\mathcal{O}(|E| \log |V|)$ where E is the number of edges.

2.5 Multi-objective optimization

Multi-objective optimization, also called Pareto optimization, deals with optimizing something based on multiple criteria. This type of optimization involves trade-offs between different criteria (performance metrics). Cost vs performance is a simple example. When generating many possible solutions for a problem each solution balances the different performance metrics differently. A

common way of analyzing a set of solutions is finding the Pareto front, which means finding a subset of efficient solutions that limits the number of options to consider.

3 Literature Review

The potential for leveraging artificial intelligence in building engineering design has been investigated from a number of different angles. The research appears to fit quite clearly into a number of categories. Each study tends to focus either on considering an engineered product made by humans and generate one or multiple connected products based on this, or on engineering something completely on its own. Almost all studies in the covered literature focus exclusively on one discipline, either structural engineering, plumbing or HVAC.

Dang et al., 2024 automated the 3D-modeling of HVAC systems based on human-engineered 2D-drawings. They used image recognition and deep learning to extract features from drawings in order to create 3D BIM-models. Wang et al., 2024 used an architectural IFC model and a ventilation IFC-model as input to create a digital twin of a building by building a semantic model based on knowledge graphs of the system. They used IfcOpenShell to extract features from the IFC-models. This approach allowed them to build a system of multiple different kinds of relational quality checks of the system, and simplified the process of building simulation models of the building.Chen et al., 2022 focused on generating a ventilation layout for supply air diffusers. Their algorithm was based on a given set of rules, and did not use machine learning. Their scope was limited to connecting air diffusers in a single room to a supply air connection. It considered obstacle avoidance, duct pressure balance and pressure loss. Qi et al., 2023 automated the placement of CAV (Constant Air Volume) air supply diffusers and Fan Coil Units (FCU) in six multilevel buildings. The algorithm included assessing room geometry, calculating load per piece of equipment, avoiding obstacles(columns and walls) and sizing the equipment. The building layout input and the HVAC system output were both in files the Revit format, processing was handled in Python, with geometric calculations handled in the Shapely library.

In the field of structural engineering, topology optimization has been a popular topic for researchers. A shallow investigation of the literature suggest that gradient-based methods has proven most effective, compared with non-gradient methods (Sigmund, 2011). Gradient-based topology optimization techniques involve minimizing (optimizing) a function using incremental steps over a gradient. Without supplementary methods the resulting structure is often highly optimized with regards to its performance, but not for buildability. This seems to be the strongest argument for Non-gradient methods. These methods involve random processes to change the layout at each iteration and can provide buildable designs for each iteration. Because of the randomness of the methods, the computational cost is much higher than for gradient-based methods. Sigmund (2011) and Woldseth et al. (2022) provide harsh and well founded judgments on topology optimization using non-gradient methods and neural networks respectively. These two sources are important in this context because this project involves using non-gradient topology optimization and will in due course suggest training a neural network to perform topology optimization. The key argument from Sigmund (2011) is that non-gradient methods require *"...immense computational resources for solving simple problems for which there already exists efficient solution techniques"*. Woldseth et al. (2022) provide a useful overview of research on topology optimization using artificial intelligence. It also provides a critical view of how the technology is being applied, arguing that *"it is the expense of the inter-iteration computations [of traditional topology optimization] that pose the real challenge."* and that *"...the focus should shift towards alleviating the computational load of the costly components within the iterative process"*.

Particularly relevant in this context, Rahbek (2023) developed a conceptual design tool capable of generating optimized structural layout using a genetic algorithm (a non gradient method) and a novel parametric modeling method. The tool was deployed as a grasshopper library. The results showed that the tool can conduct multi-objective optimization and provide the user with a Pareto front. This work is very interesting for practical structural optimization and deserves a more thorough evaluation than it got. It is highly relevant.

Thai (2022) provided a state-of-the-art review of machine learning in structural engineering covering important datasets and a statistical overview of the literature. The review found that performance of structural members and material behavior had received most attention. Another large part of the literature focused on damage assessments of existing buildings, like crack detection.

Most of the considered research on optimization focuses on single-disciplinary problems. One study, however, stood out. Flager et al. (2009) approached multidisciplinary design optimization. The author initially stated that project and tool limitations limit designers to only a few design cycles. The following citation summarizes the work.

> AEC practitioners today typically create very few design alternatives before choosing a final design. Design theory argues that this leads to underperforming designs. The aerospace and automotive industries have overcome similar limitations using MDO [Multi-Discipline Optimization] methods implemented in PIDO [Process Integration and Design Optimization] software, resulting in reduced design cycle time and improved product performance. For the AEC case study presented, we found that PIDO software enabled orders of magnitude improvements in the number of design cycles when compared to conventional methods. Instead of the usual two to three design cycles in a typical project, using PIDO we were able rapidly to analyze over 5,000 design alternatives and choose from a range of near-optimal solutions.

This approach aligns with the intent of this project and is highly relevant. In addition to generating a set of solutions, the software used provided a graph with a Pareto front. This allowed the authors to limit the number of solutions to evaluate. An important point that was made is that it is usually the client that needs to assess which trade-offs that make sense.

4 Methodology

This project has a quite limited time frame. As such, we have tried to limit the scope as much as possible while still being able to make a proof-of-concept that is relatable to discipline professionals.

4.1 Code development

In order to meet the demand of a limited scope, the design of a standalone code package was chosen. We have prioritized rapid development over execution speed, and have therefore chosen to build the first prototype in Python. As the user interface for the tools we have developed is still to be determined, flexibility has been important. Thus, we have attempted to adhere to the principles of object-oriented programming (OOP). For visualization the matplotlib library has been used, and visualization code has been kept separate from the core algorithms as much as possible to maintain modularity. In order to verify the code, the test framework pytest has been continuously used to test small portions of the code base separately during development. This has made locating problems within the more complex parts of the combined setup fairly straightforward. In order to perform geometric calculations, we have used the Shapely package where possible. Since the programs we wish to integrate with in the future have their own geometry handling, the dependencies on Shapely are hidden inside abstraction layers defined by our own classes and methods, ensuring flexibility and modularity. Execution speed might very possibly be a limiting factor in the future. Therefore, an effort has been made to ensure that the performance critical parts of the code are clearly bounded and have clearly defined interfaces. This can simplify the process of porting these portions of the code to a faster code language (e.g. C++) in the future, which can increase execution speed orders of magnitude. In order to increase the development speed, the tools Cline and Github Copilot have been used to enable support by the large language model Claude 3.5 Sonnet by Anthropic inside the integrated development environment.

4.2 Case setup

This project aims to generate a draft for engineering solutions for a building for given a room plan. The project has been limited to two engineering disciplines, namely ventilation- and structural engineering. The structural system, a slab supported by beams, simply supported by columns has been chosen. Floor plans has been generated in two ways, for HVAC optimization the plans has two fixed example layouts of 4 and 11 rooms. For each layout, a number of walls have been defined as load bearing while the others are dry walls. In each floor plan, one room is defined as the origin of a air supply branch(i.e. either a ventilation room or a ventilation shaft). All rooms are required to have fresh air supply. The fixed 11-room layout is shown in fig. 1.

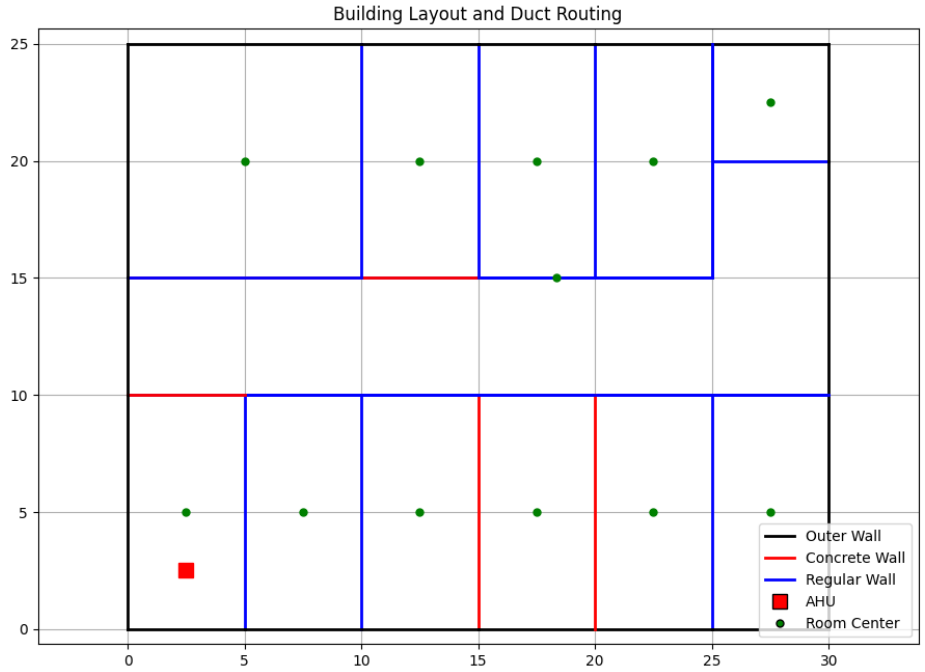

Figure 1: Test layout 2, 11-room floor plan.

For the structural engineering optimization a binary space partitioning algorithm was used to create generic floors with a undetermined number of rooms that conform to certain criteria fig. 2. The structure that populate the floor plan is subject to an evenly distributed load from above, making this a one-story building.

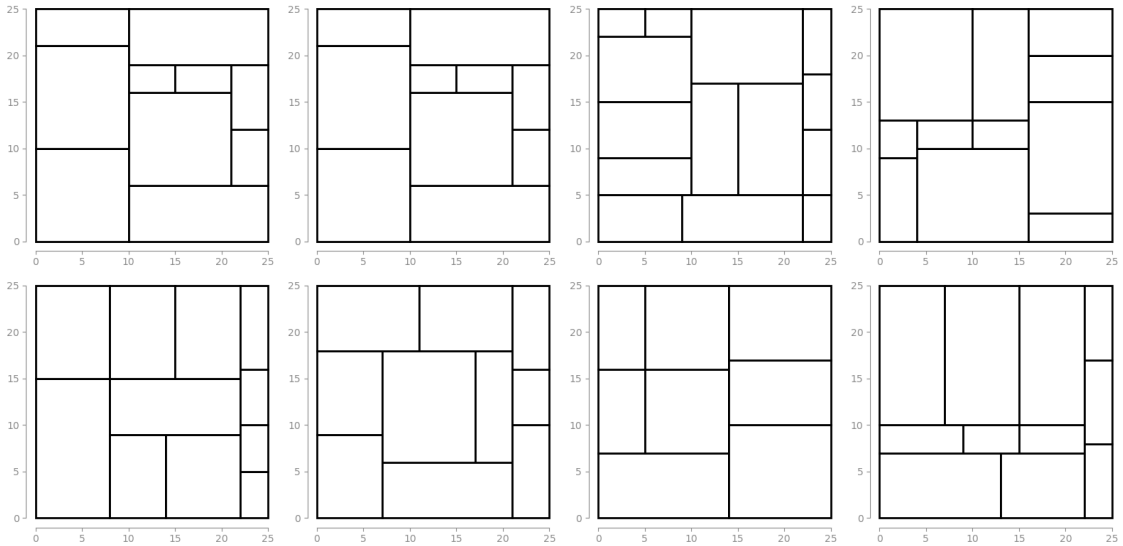

Figure 2: Eight examples of floor plans used for the structural optimization. The floor is 25 by 25 meter and each room is limited to minimum of 2% and a maximum of 20% of the total floor area as well as a limit on the aspect ratio of 3.

4.3 Pathfinding

The engineering process of buildings is generally quite rule-driven, and so we have opted for trying to get as far as possible using conventional algorithms, or expert systems. The generation of the ventilation layout in the prototype is based on an adapted version of the A*-algorithm described in 2.4. There are more efficient pathfinding algorithms than A*. But to people not familiar with computer science beforehand, many of them are perhaps less accessible and less documented in commonly available sources. Also, several of these algorithms require preprocessing of the search space, which only really adds value if one intends to perform multiple searches between multiple node pairs in the same graph multiple times. This is often the case in e.g. consumer navigation software, but not as relevant here. In the context of this project, each search is only performed once per graph. This makes A* a suitable starting algorithm for a first prototype. In addition to the cost defined by the distance from start node to the current node, a number of costs have been added. For each cost added, verification tests have been conducted before adding another layer of complexity. The relative weight of each type of cost is determined by a user modifiable list of weights. The weighting is not an exact science, tuning is performed by trial and error.

Our setup is defined as follows:

Costs:

1. **Distance:** The distance from start node to the current node.
2. **Perpendicular wall crossing:** Cost of crossing a wall perpendicularly.
3. **Angled wall crossing:** Cost of crossing a wall when not perpendicular.
4. **Wall proximity:** Cost of mounting ducts unfavorably close to a wall.
5. **Crossing of structural walls:** Cost of crossing a load bearing wall.
6. **Sound rating cost:** Cost of movement inside a room with a sound rating.
7. **Other obstacles:** Intersecting with columns or other objects is not allowed.

For wall crossings, the costs are added on top of a base cost associated with the wall material. It is for instance less problematic to cross a drywall than any other kind of wall, and outside walls are to be avoided if possible.

The air supply layout generation algorithm is shown in the following python code excerpt. The A*-pathfinding is an integrated method into the `Branch2D` class.

```
# Import custom classes
from components import FloorPlan, Room
from core import Node, Point
from routing import Branch2D

# Setup
floorPlan = FloorPlan(rooms, ahuPos)
goal = floorPlan.findMostRemoteRoom().center
mainBranch = Branch2D(start,goal)

# Generate ventilation layout
mainBranch.generate()
for room in floorPlan.rooms:
    closestNode = mainBranch.getClosestNode(room.center)
    subBranch = Branch2D(closestNode,room.center)
    subBranch.generate()
    mainBranch.sub_branches.Add(subBranch)
```

4.4 Structural optimization

The optimization algorithm is a simple non-gradient method - simulated annealing. For each iteration it makes random changes and accept both better and worse solutions, as the iterations progress the probability of accepting a worse solution declines. In the end, the best solution is chosen.

For each iteration the load on the structure and its performance is assesed. The load distribution between beams are determined by using a Voronoi diagram based on points sampled along each beam at a fixed interval. This load distribution is shown in fig. 3. Shapely, which is a Python package for manipulation and analysis of planar geometric objects, was used to generate this diagram. Each Voronoi cell is assigned to one beam and merged with other cells assigned to that beam. This way of loading beams overestimates moments, because in reality the stiffness of the columns would make the slabs transfer more load to beam ends. For this case however, this is an acceptable approximation.

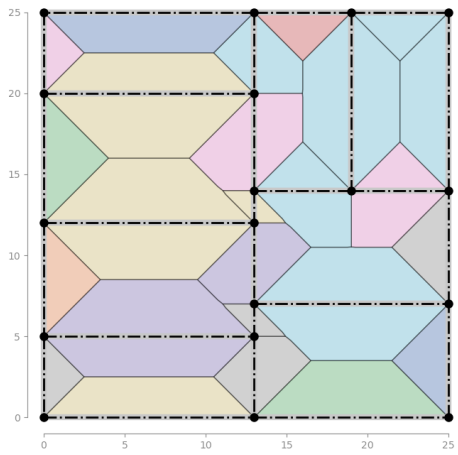

Figure 3: Merged Voronoi cells that represent the slab area that transfers loads to interior beams. Beams are shown with dashed-dotted lines.

The utilization of each beam is determined by formula 1, where the fractions show the ratio between the design load effect and the design capacity for moment (M) and shear (V) respectively.

$$u = \max\left(\frac{M_{Ed}}{M_{Rd}}, \frac{V_{Ed}}{V_{Rd}}\right) \tag{1}$$

The evaluation function is given by equation 2, where n is the number of columns and the utilization given by equation 1. This is the function the algorithm seeks to minimize.

$$f(n,u) = \begin{cases} 0.75n + u, & u \leq 1.0 \\ 0.75n + 1000, & u > 1.0 \end{cases} \tag{2}$$

This evaluation function ensures that a system with beams loaded to failure is considered really bad. Any solution that has a score larger than 1000 is considered unacceptable.

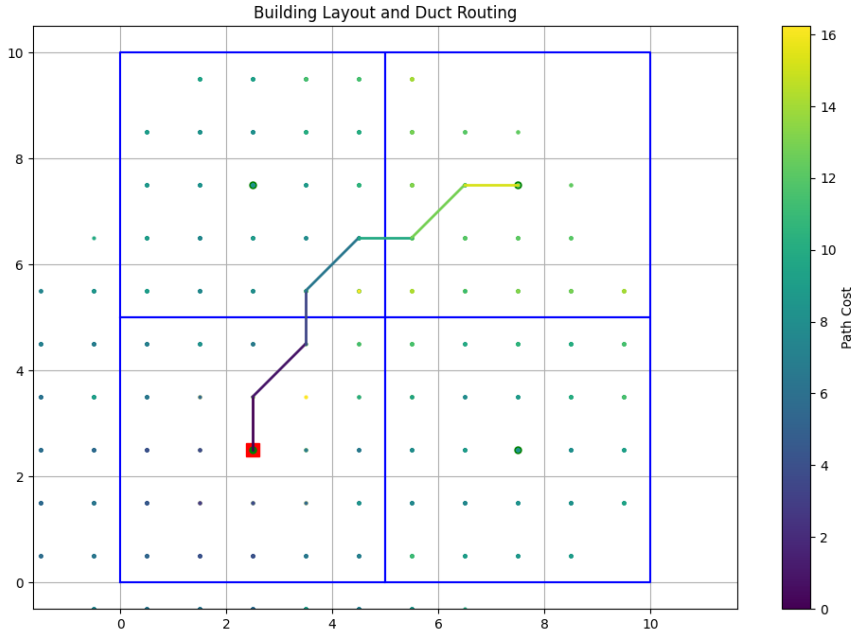

Figure 4: Test layout 1. First successful implementation of A*-algorithm with wall crossing cost for a basic floor plan with four rooms. Color of each point and path segment indicates the cost of reaching that point.

5 Results and Discussion

Experiments performed started with a simple floor plan with four equally sized quadratic rooms. The results from this were satisfactory, and shown in fig. 4. When the cost models of movement and wall crossing had been implemented and verified, each ventilation layout test was performed with first the four room floor plan and then a slightly more complex 11-room floor plan. There are still no diagonal walls present, but not all rooms are rectangular any more.

Figure 5 shows an early integration test with all components present and visualization applied. Only the main branch is generated to verify the path generation functionality.

Figure 6 shows an integration test with a simple random structural model and an optimized ventilation layout with all costs calculation variants in use. All rooms are here supplied with air, and each branch originates from the closest present branch. We clearly see that this approach gives unfavorable results, as this would lead to too high air flows being routed through occupied rooms.

Figure 7 shows a more successful result, where each room is supplied from a sub branch originating at the same main branch. Note the crossings of concrete walls, this is not optimal. It indicates that the scale of these weights relative to each other remains subject to further testing and tuning. The generated paths in fig. 7 are also not optimal in that they contain too many bends, which would cause a suboptimal pressure loss and a high financial cost. The current algorithm does not consider these costs, and we can see that future models will need to address this.

5.1 Optimization methods in structural engineering

The optimized structural systems shown in fig. 8 perform well given their performance measures. However, the systems clearly illustrate an issue with the representation of the slab without mitigating rules. We have assumed that the slab transfers loads in both directions, disregarded its

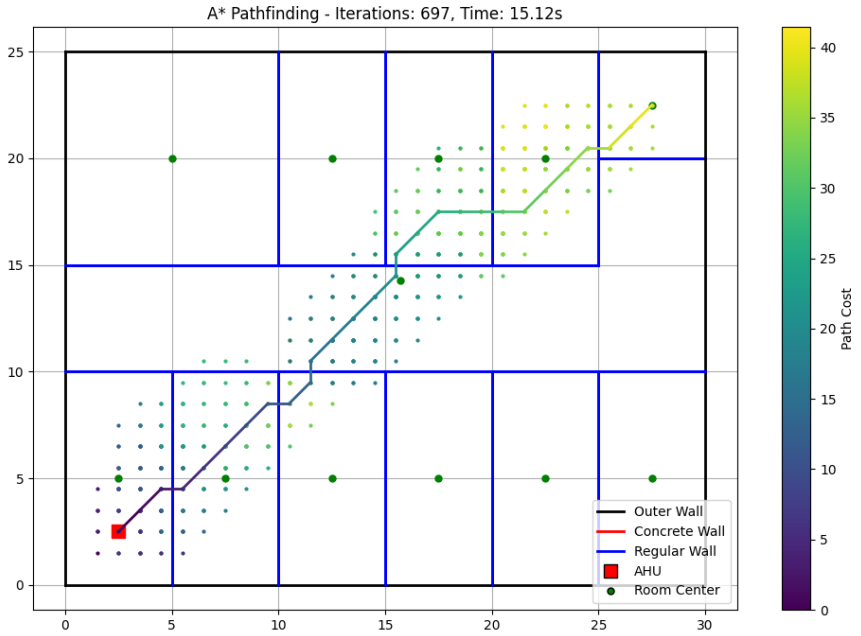

Figure 5: Path generation test with wall crossing costs, no sound ratings cost or concrete walls present. Only the main branch is generated.

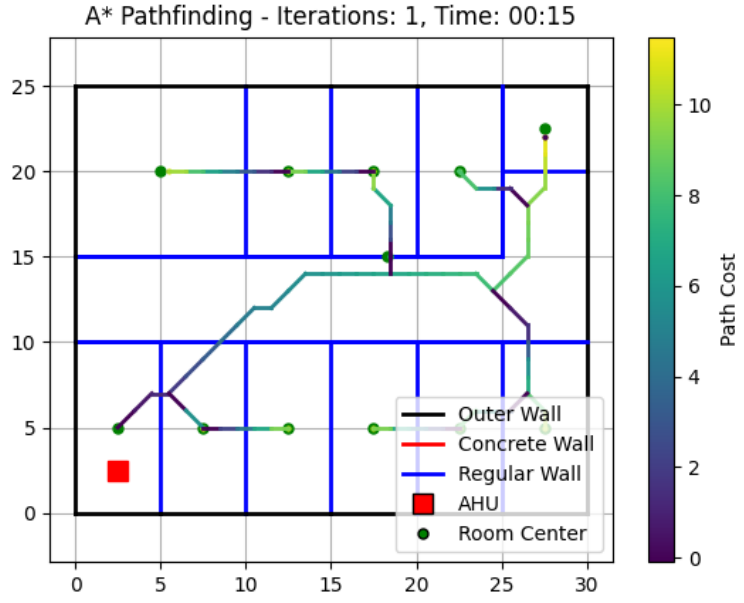

Figure 6: Integration test included concrete walls and sound ratings present. Sub-branches are generated and are allowed to originate from any existing node on any branch.

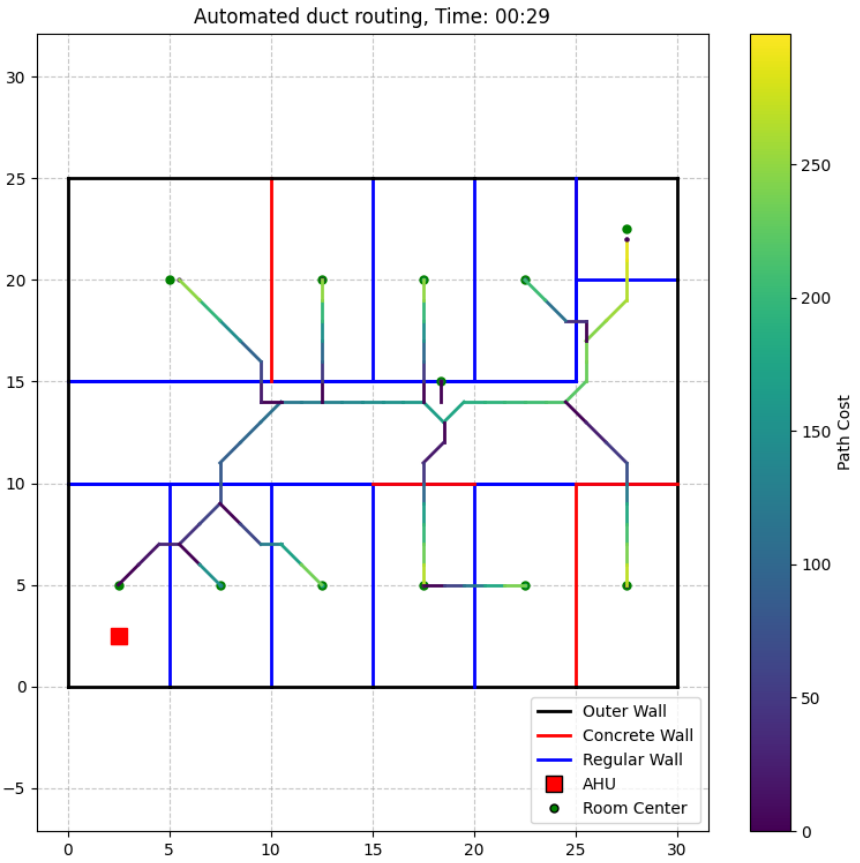

Figure 7: Integration test included concrete walls and sound ratings present. Sub-branches are generated.

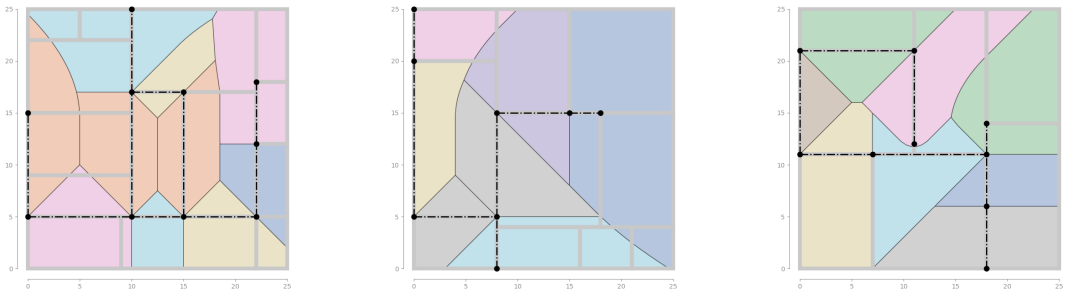

Figure 8: Three different optimization solutions. Gray lines represent architectural walls. Round dots represent columns and the dash-dotted line beams. The colored polygons show the area of the slab that transfer load to the intersecting are beams.

capacity and did not provide rules that keep its outer edges supported by beams. The result is unrealistic cantilever spans of the slab. However, what we seek to optimize is a multidisciplinary solution, and we have therefore prioritized optimization of objects that can collide with the HVAC system, like beams and columns. It should be noted that even though we used simple methods to evaluate at each iteration, the computational cost was high. This is partly due to known (and solvable) inefficiencies in our code, but also due to the number of iterations required.

The literature review showed that single-discipline oriented research mostly argued for gradient based methods. Sigmund (2011) and Woldseth et al. (2022) argued for more research aimed at improving the evaluation cost of each iteration. The cost issue is an even larger issue in non-gradient methods.

In this project, structural optimization was deployed without using finite element analysis in each iteration, but instead used estimates to speed up the process. Just like with current "manual" processes for early phase design it make sense to quickly search the design space for promising solutions and then use more accurate methods to optimize those. This makes a lot of sense at an abstract level. HVAC and structural systems are integral parts of a building, but they are separate systems. Optimizing two independent systems together, where space is the only common denominator, seems to mandate non-gradient methods.

5.2 Multidisciplinary optimization

Only the optimization of the HVAC system took another discipline into account. The structural optimization method should have rules that consider openings in beam flanges (and affect shear capacity), but this was unfortunately not covered by the project due to time constraints. We suggest this as a topic for further work.

When combining these two schemes, prioritization of a discipline in an optimization scheme is a difficult matter. To reduce the number of evaluations at each iteration one could consider to let one discipline 'get a head start' before all disciplines are optimized together. What discipline should take the lead should be determined by overall performance measures and might depend on the problem (e.g. building type).

5.3 Future work

Combining the optimization methods into one 'expert system' should be prioritized. Further work should assess this method qualitatively and visualize the solutions and their trade-offs.

For practical applications the evaluation criteria is central to get to preferred outcomes. Holistic design evaluation of buildings and their sub-systems for use in multidisciplinary design optimization appears to be an important and promising field of research.

Ideally future work will include integrations with the software used in practice today and in the future (e.g. Revit and Bonsai). This would make it more available for practitioners, with and without programming experience. In order for this tool to provide value for engineers in practical work, the tool also needs a modern graphical user interface.

Further, given the number of considerations that must be balanced that our model currently does not cover, it is relevant to perform experiments with machine learning in the future. We suggest training one or more neural networks in order to compare their performance with our suggested expert system.

References

Chen, Zhisen et al. (2022). 'Rule-based generation of HVAC duct routing'. In: *Automation in Construction* 139, p. 104264. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2022.104264. URL: https://www.sciencedirect.com/science/article/pii/S0926580522001376.

Dang, Long Hoang et al. (2024). *AHMsys: An Automated HVAC Modeling System for BIM Project*. arXiv: 2407.01987 [cs.CV]. URL: https://arxiv.org/abs/2407.01987.

Flager, Forest et al. (2009). 'Multidisciplinary process integration and design optimization of a classroom building'. In: *Journal of Information Technology in Construction (ITcon)* 14.38, pp. 595–612.

Ørsted, H C and William F. Baker (2015). *On the Harmony of Theory and Practice in the Design of Tall Buildings*. URL: https://www.youtube.com/watch?v=_XrvfBlcWcs&t=467s.

Qi, Zixuan et al. (2023). 'BIM-Based Automated Multi-Air Distribution Layout Generation for Office Buildings: A Case Study'. In: *Buildings*. URL: https://api.semanticscholar.org/CorpusID:260038745.

Radio, NRK (Mar. 2024). *Den store serien om KI (4:10) - Vaktskifte: Dr. KI stempler inn*. NRK. URL: https://radio.nrk.no/podkast/abels_taarn/sesong/2024/l_1a6d7e73-5568-411a-ad7e-735568911ada.

Rahbek, Lasse Weyergang (2023). 'Conceptual Design Tool for Structural Layout Optimization in the Early Design Phase'. PhD thesis. Aarhus University.

Sigmund, Ole (2011). 'On the usefulness of non-gradient approaches in topology optimization'. In: *Structural and Multidisciplinary Optimization* 43.5, pp. 589–596.

Thai, Huu-Tai (2022). 'Machine learning for structural engineering: A state-of-the-art review'. In: *Structures*. Vol. 38. Elsevier, pp. 448–491.

Wang, Meng et al. (July 2024). 'Automatic HVAC Topology Generation Using BIM Geometry Checking and Knowledge Graph Technologies'. en-GB. In: *Proceedings of the 2024 European Conference on Computing in Construction*. Vol. 5. Computing in Construction. Chania, Greece: European Council on Computing in Construction. ISBN: 978-9-083451-30-5. DOI: 10.35490/EC3.2024.257. URL: https://ec-3.org/publications/conference/paper/?id=EC32024_257.

Woldseth, Rebekka V et al. (2022). 'On the use of artificial neural networks in topology optimisation'. In: *Structural and Multidisciplinary Optimization* 65.10, p. 294.