**Structural optimisation that architects don’t hate**

Matt Harrison  
Atkins

---

**Learning Objectives**

- Learn how to apply constraints to a problem domain to discretize complexity into manageable chunks.
- Discover how multi-disciplinary constraints can be applied to optimization algorithms.
- Learn about the positive and negative influences that structural design can have on the building occupants.
- Investigate the quantification of design development through structural engineering layout.

---

**Description**

Modern structural optimization, based on reducing materials, has facilitated marked improvements for some structures like canopies and bridges. However, when applied to buildings, these approaches to optimization are less appropriate, as they can impact internal building layouts, frustrating our architectural colleagues. Procedural generation—a method for using algorithms to generate geometry—when combined with more familiar structural optimization like Generative Design in Refinery, offers us a new methodology that can drastically reduce the complexity of the problem. This is achieved by mimicking human processes and applying them to the option generation process, such as aligning the building’s structural frame with key features in the building layout, like cores and atria. This technique also offers the opportunity to accommodate multidisciplinary constraints, such as internal walls and corridors, by adjusting the building structural frame to match the position of internal features.

**Speaker(s)**

Matt Harrison is a Chartered Structural Engineer from the UK. He works for Atkins in the Building Design Research and Innovation team, and here he has the role of Digital Developer. His role covers the whole of the Building Design practice, but he still has particular focus on the structural engineering practice where he designs, develops and implements digital tools for the practice.
Introduction

Current computational optimisation methods and algorithms are fantastic at reducing weight for structures, however they can lack the ability to effectively optimize building structures. I believe that these methods and algorithms can be adjusted by some simple additions, to incorporate multi-disciplinary design constraints to allow for effective and useful building optimisation in a real world design scenario. This presentation / handout will detail current methods, my proposed alterations and how this can be achieved using technology.

Conventional Optimisation

Statement structures

Conventional brute-force or physics based optimization methods are most utilized when dealing with what I’ve defined as a ‘statement’ structure. I define this as a structure which is free to respond in form to the load applied to it, such as roofs, canopies and bridges. These structures tend to have limited constraints, and using the example in the image below, there are only seven constraints. There are the six supports and the area that the canopy has to cover, and other than this the structure is free to take a form that supports the load the best. This is visible in the arching between the supports. A physics based optimization algorithm was used to reduce the frame weight of the structure to great effect, with a £140,000 reduction in steel cost per structure.

Building structures

This type of optimisation is much less effective when dealing with conventional building structures because it doesn’t respect the space that the structure occupies. For example, with buildings, this means that a column may fall into the middle of the room, ruining the space. Buildings tend to be laid out by architects, considering the human user, with the structural engineer fitting the structure into walls. However, if this was driven the other way round, we would end up with buildings that were unpleasant to be in, lack flexibility in spaces and consequently would be demolished sooner, working against our goals of Net Zero Carbon. Therefore, we need to build both respecting the human, whilst considering the impact of our decisions on the environment, rather than reducing for the sake of reduction.
Alternative Methods

In order to incorporate some respect of the building layout into our optimisation algorithms, I have proposed an alteration to a popular method called the genetic algorithm. This modification allows for the insertion of a set of rules into the option generation section, to only allow for ‘good’ options to be generated and therefore a much more efficient optimisation process. This is down to the massive reduction in problem domain by the application of these rules. In this example, I utilised a procedural method to generate a floorplate arrangement based on a small amount of parameters, allowing for repeatable generation of geometry.

Procedural method

The procedural method proposed is split into three distinct groups:

- The edge direction;
- The key elements; and finally
- The intermediate elements.

**Edge direction**

Firstly, the grid of almost every building is aligned with either the direction of one of the edges of the building or the core of the building. By capturing these, it is possible to create grid direction ‘pairs’ by looking at those directions and their perpendicular directions.

![GRID DIRECTION FROM EDGES](image)
Key Elements
Once the grid direction has been determined, the procedure then needs to generate what I'm calling 'key elements', i.e. those beams that will always exist within a floorplate. These include beams which will radiate from the core and beams which will radiate from the corners of the building.

This stage can create too many elements, and therefore it requires good consideration and adaptability. In practice, I implemented this stage using a scoring system to rank elements and remove those that I deemed too close to another feature in the building.

Intermediate Elements
Once this layout of key elements is created, the only remaining task is to iteratively break down the rest of the floorplate to achieve spans less than the maximum span permissible.

In practice, this stage requires significant iteration in order to ensure elements aren’t duplicated, with a minimum number of cycles typically being around 50.
Benefits for layout
Whilst these steps don’t have any benefit for architectural layout within the optimisation, the key is to include these items as constraints within the floorplate. For example, the internal wall positions could be included as key elements, which would force the rest of the structure to be sympathetic to these positions, and minimise the structure outside of these wall positions.

In the process, it will be key to ensure these elements are included as early into the process as possible, to ensure that the rest of the structure is arranged appropriately around this. There are obviously other considerations you could take, for example specific column positions, but this will be more dependant on your requirements.

Summary & Benefits
Computation for the sake of it is fascinating, however we will only realise the true value when we start quantitively comparing options. The most obvious metric to compare is cost, particularly given that it is directly attributable to volume or weight of material. However, as we start to increase our consideration of carbon in design, we can use this as similar linear metric for comparison also. As we calculate more and more metrics associated to different designs, we can drastically improve the value that computational optioneering has on the design process of buildings, enabling scoring and comparison based on a number of different client-led drivers.

In the future, I imagine that design teams will work together on a model live, with constant feedback about the quantitive impacts of the decisions they are making. I.e. if this wall is moved, it reduces or increases the structural cost and reduces or increases the carbon impact of the total solution. Through this quantitative feedback, the design team as a whole can make a proactive and informed decision about what balance of human-centred design is valuable for specific projects.
Implementation of Technology

In order to implement this process into a technological solution, the use of custom components and methodical workflow was required. This is explained in the following five sections:

Understanding building geometry
The first step to implement this methodology in a real workflow is to understand the building geometry. Fortunately, the Autodesk suite, in particular Revit, does a good job with this. The easiest way to capture the geometry is to use Faces, which can then be translated into surfaces, loops and lines very easily.

Utilizing Dynamo and Zero-touch
Given the use of Revit, the obvious next step is to use Dynamo. This allowed for the simple extraction of geometry from Revit into easily parsable geometry for calculation and comparison. Given the iterative nature of most of the processes involved in this procedural design, Zero-touch nodes were essential.

I learned to use Zero-touch two years ago at Autodesk University, and it has since become invaluable to me, so I would recommend trying to pick up the skills if you are interested in exploiting Dynamo further. You can also use XML commenting to provide information on the node for Dynamo users.

Handling large-scale geometry calculations
There is a huge amount of geometry calculations that are required during the procedural design. These were initially prototyped using Dynamo before building these functions into Zero-touch nodes. However, I soon realised that the in-built objects and properties, called ProtoGeometry, were quite limited in their usability at scale. When approaching approximately a thousand elements, I found that Dynamo began to struggle and crash, a feature I know the Dynamo team are currently working on.

I ended up implementing a 3rd party .Net plugin to do geometry calculations to speed these calculations up. These 3rd party objects were fantastic, as they permitted much higher speed calculations, however there were not as many methods available and therefore I had to create a number of methods in order to conduct simple geometrical calculations.
Optimization using Refinery
Once I had a working script using Dynamo and Zero-touch nodes to complete iterative calculations, this allowed me to utilise Refinery to automate the generation and optimization of solutions. The plugin allows for really easy and simple generation, comparison and optimization of these tools, particularly given the limited number of input variables.

User Interface
Finally, after explaining all the technology behind the tool, I ended up hiding all of the details behind a really simple user interface. I found that my individual end users of the tool actually didn’t want to see the script or workings, they just wanted to type in the numbers and see what comes out of the other end. Given my work with Zero-touch, I implemented Windows Forms to produce simple user interface and handle all user inputs.

![Windows Forms to Facilitate User Input](image1.png)

We hope to integrate the tool with Revit directly in the future, to remove the need to use Dynamo. This should hopefully allow for more and more users to be able to access the tool. This, however, will require the removal of all ProtoGeometry from the scripts, as Revit cannot use ProtoGeometry without Dynamo being active.

Summary
Not all structures are alike, therefore we shouldn’t use the same methodology for optimizing them. The method proposed here, allows for a procedural method to be applied to option generation for buildings by allowing for the consideration of multi-disciplinary spatial constraints when setting the structural grid. This method, applied with the technological solutions provided by Autodesk, allows for rapid optioneering of building structural frames, in a way that is sympathetic to the building layout.