Volume xx (200y), Number z, pp. 1–13

Integrated Structural-Architectural Design for Interactive Planning

B. Steiner[†1] E. Mousavian[†2] F. Mehdizadeh Saradj[2] M. Wimmer[1] P. Musialski[1]

[1]TU Wien, Institut of Computergraphics and Algorithms, Austria
[2]Iran University of Science and Technology, School of Architecture and Environmental Design, Iran

Abstract

Traditionally, building floorplans are designed by architects with their usability, functionality, and architectural aesthetics in mind, however, the structural properties of the distribution of load-bearing walls and columns are usually not taken into account at this stage. In this paper we propose a novel approach for the design of architectural floorplans by integrating structural layout analysis directly into the planning process. In order to achieve this, we introduce a planning tool which interactively enforces checks for structural stability of the current design, and which on demand proposes how to stabilize it if necessary. Technically, our solution contains an interactive architectural modeling framework as well as a constrained optimization module where both are based on respective architectural rules. Using our tool, an architect can predict already in a very early planning stage which designs are structurally sound such that later changes due to stability reasons can be prevented. We compare manually computed solutions with optimal results of our proposed automated design process in order to show how much our proposed system can help architects to improve the process of laying out structural models optimally.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages J.6 [Computer Applications]: Computer-aided Engineerings—Computer-aided design

1. Introduction

In the workflow of architectural building planning, one of the critical early tasks is to find the best structural layout for a given architectural model. This involves deciding the size and placement of structural elements like beams, slabs, and columns in an underlying design. Currently, this task is usually performed in a rather ad-hoc fashion driven by the intuition and knowledge of architectural and structural rules by the architect, requiring a high degree of expert knowledge and experience, while the exact structural feasibility of the design is usually computed in a later stage.

In practice, an experienced architect can oversee most structurally critical decisions. However, if due to the growing complexity of the actual design, the number of structural elements that have to be added or modified increase, finding a solution that fulfills all necessary optimality criteria becomes a laborious and error-prone task. Therefore, it is desirable to determine feasible structural layouts computationally, and present them to the user already in the early design and shape-exploration stage. This allows exploring multiple feasible solutions and avoiding flaws in the structural design. We call such an approach *integrated structural-architectural design*.

The high-level goal of integrated structural-architectural design is to maximize the architectural potential while at the same time minimizing the number of "unwanted" elements, i.e., such elements that the architect is forced to introduce due to stability reasons. In fact, most structurally important elements, like beams and columns, are actually unwanted elements. For example, columns in a room influence the spatial properties of the room and thus limits the freedom of interior design. In addition, removing such unwanted elements lowers the costs of the building. Current work in computer graphics about finding structural feasible designs [WSW*12] focuses mainly on changing the placement and shape of the given elements, while the introduction of new or removal of existing stabilizing elements is an open research problem.

In this paper, we propose two methods to support architects in multi-level integrated structural-architectural design: (1) An interactive rule-based solver assisting the architect in finding the optimal vertical structural element placement by automatically adjusting floorplans based on architectural and structural design principles. (2) We provide the user with fast feedback about the structural stability of the building by introducing a approximative calculation model. Based on this model, our method provides suggestions on how to place structural elements like beams, slabs and columns such that loads are optimally distributed by utilizing existing walls and finding the minimum number of necessary columns. In contrast to previous work in structural design exploration [WSW*12], our method can introduce and remove load-bearing elements in order to obtain a feasible design. This is achieved by a novel non-linear optimization algorithm that tries to optimally distribute loads between

† B. Steiner and E. Mousavian are joint first authors

existing walls and a minimum number of needed columns.

In the following, we review the literature on architectural and structural design in Section 2. In Section 3 we describe our interactive modeling schema, while Section 4 describes the structural analysis model used. Our main contribution, the structural optimization algorithm is presented in Section 5. Finally, in Sections 6 and 7, we present our results and conclusions.

2. Related Work

In the literature, structural laying out as an effort to find optimal arrangement and placement of structural elements is a topic with long research history in structural engineering, architecture and CAD, but also in computer graphics and modeling. Generally, it can be categorized based on two main approaches of early design optimization.

Shape Optimization. These approaches use architectural elements that will be applied as structural elements as input defined by the user and try to find an optimal structural solution by resizing or relocating the elements. Usually, also form-finding methods that search for the optimal configuration of structure [PSB*08, Blo09, MIB15], or structural shape optimization of interior architectural-structural elements [WOD09, WSW*12] are categorized in this category. Compared to our work, this methods don't allow to add additional structural elements and just deal with modifying existing ones.

Topology Optimization. In topology optimization, which our method belongs to, the optimization is not performed on the input directly, but a structural model is derived from it. The whole optimization is then done on the structural model and results are transferred back to the initial one. In general, architectural elements are not considered as explicit structural elements a priori, but have the potential of becoming such elements, depending on their usage. For instance, in a multi-floor building, an inner wall on a single floor would be considered as non-structural. However, if the wall is placed at the same position in all floors, it becomes a load-bearing hence structural element.

Early efforts tried to employ heuristic methods such as knowledge based expert systems (KBES) [Mah84, Owe90, SWK00] which used architectural, structural, and financial knowledge to propose appropriate structural grid layouts as regularly spaced bay sizes [Mah84, Owe90, SW97], or regular spaced bay sizes with local irregularity of column placement [SWK00]. Other methods utilized case-based reasoning (CBR) [KR97, FRG00, SPnM00] to generate structural floor layouts by reusing solutions obtained from example designs. Genetic algorithms (GA) have also been used to find regular structural grid layouts according to different objectives including minimizing different types of building project costs [GK02, RMB03], maximizing the flexibility of architectural spaces [PG99], or both of them [SMM03]. All of this methods just deal with rectangular buildings and can only handle equally spaced grid cells defined by columns and beams, not taking into account load bearing walls.

Eventually, also buildings with more complex overall horizontal configurations and interior design have been approached with GA. For example, Shaw et al. [SMG08] applied GA for structural layout of rectilinear floor plan shapes but still only uses equally sized grid cells. Nimtawat [NN09, NN10] proposed a method to find optimal beam-slab layouts for floor plans with given interior walls and columns using GA. Again load-bearing walls were not taken into account and only beam locations, but not column locations were optimized. Our method, in addition, allows for different layouts in different floors, which was also not possible here.

One of the latest related efforts have been presented by Hofmeyer et al. [HD15] where a 3D architectural design is provided with structural elements via a grammar [DH13]. These elements were then automatically made conformal [HvRG11], stable [SH12], and optimized [HD13]. Subsequently, the architectural design is modified -as a function of its related structural performance- and updated to conform to the initial building plan. Afterwards, the architect can again modify the resulting floorplan. This co-evolutionary approach has been verified with a genetic algorithm, and the grammar can also be omitted by letting the optimization find complete structural topologies itself. This methods were suited mainly for structural engineers and not for early design since they tend to modify the overall configuration of the building.

Finally, the use of automated generation of *spatial layouts* without taking the structural aspect into account has also been investigated in the computer graphics community [MSK10, LYAM13, BYMW13]. Another approach to ensure stability of constructed objects is given by Schulz et al. [SSL*14], where a FEM-model is used to ensure that constructed furniture is stable. Also general shape editing, in particular free-form shape editing [BSK*13, ZTY*13], has been investigated in an architectural context by optimizing surfaces such that they fulfill architectural requirements.

3. Architectural Modeling

The architectural modeling framework proposed in this paper is an interactive tool that allows the architect to create floorplans for masonry buildings. The limitation to this type of buildings is not a general limitation of the proposed algorithmic solution, but the real architectural and structural principles for this type of buildings are well documented, for example, in Eurocodes [Eur05] or ACI [ACI14]. In this work, we focus mainly on regulations given by the Eurocodes, but the same rules are also used in other standards for structural engineering, although the values of some parameters like the distances between objects might differ.

3.1. Initial Modeling

Template Elements. In our design system, the architect generates the floorplan at each level using the *horizontal overall configuration*, which in other words is the 2d top view of the building. All members of the configuration are modeled by using basic 2d rectangular template elements, which are abstracted representations of the building's main components. In particular, there are templates for the main boundary, projections, rooms, stairs and connected voids, and also for openings which comprise doors and windows. Figure 1(left) shows an exemplary composition of such elementary members as well as an a description of the members of a stair (right). A *projection* in the architectural sense is defined as a component, member, or part of an architectural volume which juts out from the building.

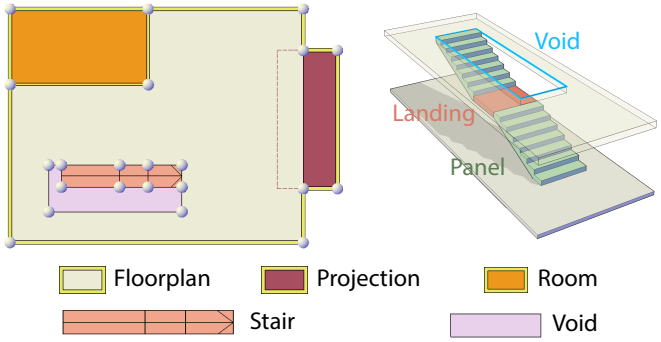

Figure 1: *Left: Floorplans are composed of multiple template elements. Right: Stairs consist of two stair panels (green) and a landing (red) between them. A void (blue) has to be present above each stair.*

Modeling Process. The spatial design always starts with the generation of the initial boundary rectangle and by defining the number of stories. The architect can then use further template elements, e.g. projections, to extend the main boundary and to model the horizontal overall configuration for all floors. After that, rooms and stairs can be modeled for each floor inside the generated building envelope. At each step of the process, the main envelope, including the boundary and projections, can be manipulated. Finally, after adding rooms in the interior, openings, including doors and windows, can be added to the walls of the envelope. Adding openings is not allowed in earlier stages since inserting rooms can split previously modeled openings on the wall of main envelope.

User Interactions. During the modeling process, three types of user interactions are possible: (1) Adding or removing objects, i.e., each template element can be added or deleted individually on the canvas. (2) Translate (dragging) whole objects, i.e., all template elements can be moved. (3) Translation of segments, in particular, a segment is a part of an object, like a wall, an opening, or stair boundary. A wall can either belong to the main boundary, to projections or to rooms. In this way, objects can be resized.

3.2. Design Principles

In architectural design, a floorplan model needs to conform to sets of (1) architectural and (2) structural constraints. In the traditional manual workflow, the architect is responsible for the appropriate placement of all elements in a way that respects all these constraints. In contrast, the goal of our work is to lift the burden of having to take care of all small details by providing automatic layout adjustments. This is done during the initial generation and spatial placement of elements, as well as during the refinement of the generated layout. Also, keeping the layout consistent with these constraints makes sure the layout forms a suitable input for the structural optimization stage later on.

Architectural Constraints. The initial spatial design used for the structural layout should conform to basic architectural requirements, which ensure that the building is usable for its intended purpose. For this, we enforce the following principles during the modeling phase:

- The distance between two parallel segments representing walls, boundaries of voids in the floor above stairs, or stair boundaries in a single floor shall be more than 0.8 m.
- The width of a stair should not be more than 1.5 m.
- The length of each stair panel (c.f. Figure 1, right) is related to the height of the floor and the position of the landing between the panels. Since the slope of the stair is constant, lowering the landing would shorten the lower panel and enlarge the upper panel..
- A void above a stair has to be generated automatically based on the minimum space needed to be open above. This is mainly based on the height needed for a human to walk up the stair.

Structural Constraints. The geometry of each template element as well as the geometrical and topological relationships between different templates are constrained by predefined structural requirements that are imposed from physics and from model limitations. We formulated a set of geometric rules which conform to principles according to building codes for masonry structures ([Eur05, Eur04]):

- The ratio of the longer to the shorter side of the initial boundary rectangle (exterior walls excluding projections) shall be equal to or less than 4.
- The longer edge of the initial boundary rectangle shall be equal or less than 40 m.
- The longer side of a room has to be shorter than 8 meters to prevent beams from bending (c.f. 4.2).

These rules limit the horizontal overall configuration and are enforce interactively during the modeling process. Additionally, the existence of projections reduces the regularity of the configuration. To avoid irregularity in an independent structural volume, we apply a set of regulations that limit the geometry of projections. We enforce them by introducing the following rules based on the Eurocodes [Eur05]:

- The minimum distance between two projections on the same wall is 0.2 m.
- The overall 2d boundary of a building in top view, including the main boundary and projections, is similar for all levels.
- The ratio of the area of projections to the total floor area shall be less than 0.15.
- A projection cannot intersect two edges of the boundary at same time.
- A projection cannot intersect another projection.

3.3. Consistency Check.

All discussed constraints are enforced by an iterative consistency-check procedure in real time. If one of the constraints is violated during user interaction, the system tries to automatically adjust elements such that all constraints are satisfied. This is done by using a set of iteratively applied checks, which first verify the validity of a constraint and in case of a violation adapt the floorplan. These constraints are applied in such an order that elements that limit other elements are always checked first, for example, boundary walls

are checked before projections, and projections are checked before rooms. Nevertheless, it is in some situations necessary to adjust parts of the floorplan that could violate previously evaluated rules. In this case, the consistency check is repeated from the last constraint that could be violated. For some of the constraints, namely the height of stair panels and the size of voids above stairs, it is not possible to come up with an automatic solution. In this case, the user is informed about the constraint violation by marking the violating template in red.

In addition to the iterative consistency check that is applied after each modification of the building, some constraints are already communicated to the user during a manipulation operation. Stairs, for example, are marked in red when the width during a resize operation becomes too large.

4. Structural Analysis

In this section, we provide the details of our structural analysis module for masonry structures, which forms the core of our integrated structural-architectural design framework. This stage is called whenever the user has modified the input floorplan and all requirements described in Section 3 are fulfilled, which is typically ensured by the consistency check. In contrast to commercial FEM models, our method is fast enough to provide the user with immediate feedback about the structural stability.

The structural analysis uses a simplified abstract representation of a building, the so-called structural grids, and is based on the strength-of-material approach [Wah07]. We first describe how the structural model is represented using structural grids, and how to convert an input floorplan into a structural grid. This is then used to calculate the loads and maximum capacities or compressional strenghts of structural elements.

4.1. Structural Grid

Levels. A building consists of several levels, which form the topmost structure of our model. Each level (as shown in Figure 3a) is defined by one structural grid. Formally, a structural grid for level L is a planar graph

$$L = (C, B)\,, \tag{1}$$

with vertices $C = \{c_1, \ldots, c_n\}$ and edges $B \subseteq C \times C$. We also define S as the set of faces s of the graph. We write edges as $b_{ij} = \{c_i, c_j\}$ and refer to faces by closed edge loops, i.e., $s = b_{ij}, b_{jk}, \ldots, b_{li}$ for some $s \in S$.

Structurally, the vertices C represent *columns*, edges B represent *beams*, and faces S represent *slabs* (cf. Figure 3), and will be further denoted as such.

Columns. Columns are vertical structural elements where concentrated loads are transferred to the ground and are shown in Figure 3a as circles. Columns are either *real* structural columns (purple circles), or "virtual" *wall* columns (orange circles) that are used to represent wall-wall or wall-beam intersections. These wall columns do not physically exist in the building but describe special points on walls where forces have to be transferred. Further, columns are either *fixed*, i.e., their position is fixed or *variable*, i.e., their position can be moved during the optimization (see Figure 3b, dashed orange lines). This results in a partition of C into four subsets:

$$C = \{C_{FW}, C_{VW}, C_{FR}, C_{VR}\}. \tag{2}$$

Fixed wall columns (C_{FW}) correspond to endpoints of walls and wall-wall intersections. Variable wall columns (C_{VW}) correspond to wall-beam intersections, and are moved together with the beam. Fixed real columns (C_{FR}) are those that are present in all levels of a floorplan and need to be fixed so that loads can be transferred to the next level. Variable real columns (C_{VR}) can be moved freely. Both of these correspond to beam-beam intersections.

The position of a column c_i is given by $p(c_i) = \{x, y\}$.

Beams. Beams are used to represent horizontal load-bearing elements that resist distributed or concentrated loads and are shown in Figure 3a as thick lines. There are two different types of beams: *Real beams* B_R connect two arbitrary columns in places where there is no load-bearing wall, while *wall beams* B_W connect two wall columns and abstract load-bearing walls (see Figure 3b). Thus,

$$B = \{B_W, B_R\}. \tag{3}$$

Real beams bend and transfer load to the connected columns, while wall beams transmit load down to the ground. In order for this to work, a load-bearing wall has to be continuous from the topmost level to the ground level, i.e., a corresponding wall beam has to exist in each level (similar to fixed real columns).

Slabs. Slabs are the faces of the structural grid L. In practice, they correspond to horizontal or inclined flat rectangular panels which resist flexure in two directions [ACI08]. This means that in the creation of the structural grid, we need to make sure all faces are rectangular.

4.2. Computational Model

Based on the structural elements above, we developed a model to calculate the structural stability of a building that is on the one hand fast enough to give immediate feedback to the user and to allow an optimization to be run in reasonable time. In addition, this model has to be physically plausible and yield correct results. Our model is based on the strength-of-material approach [Wah07] which is also used by architects during design but makes some assumptions about the building that allows for faster calculations.

The first simplification is that our method constraints beams, and thus walls, to follow one of the horizontal principal axes. This allows for a much simpler algorithm to determine the force distribution of slabs since they are always rectangular.

According to Place [Pla07], the maximum length for two-way concrete slabs supported by boundary beams or load-bearing walls is 9 meters, while the ratio of the thickness to the longer side length is $1/30$. When assuming 25cm for the ceiling thickness, the maximum length of a beam is 8 meter in order to prevent violation of the maximum beam and slab shear and bending stresses. Using this assumption, it is not necessary to include the full calculation of the maximum bending moment in the optimization, since beams with a length below this value are guaranteed not to bend too much.

Another phenomenon that should be taken into account is buckling, which causes bending of columns due to compressional forces

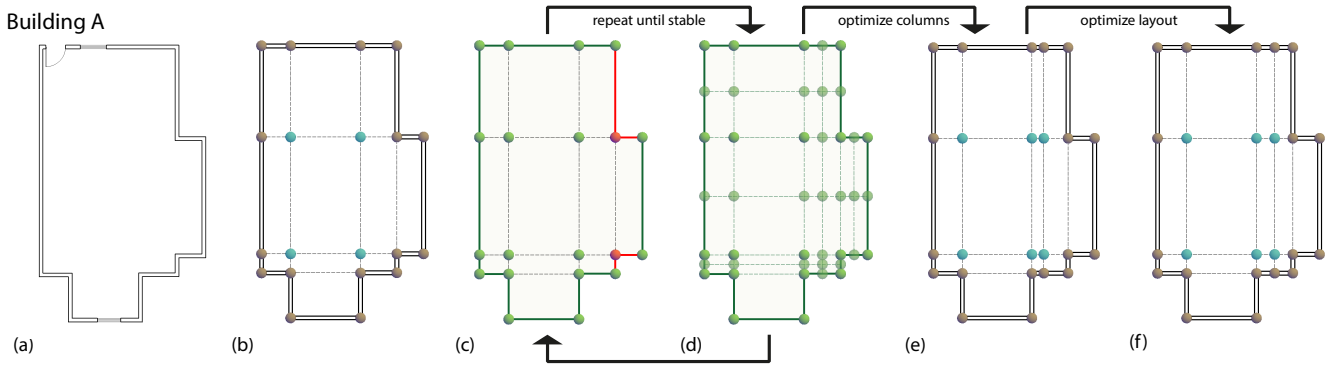

Figure 2: *Structural layout analysis and optimization consists of four stages. The input floor plan (a) is used to create an initial structural grid (b). This grid is analyzed and the feasibility of all elements is calculated (c, unstable elements are shown in red). In all slabs adjacent to infeasible elements additional columns are added (d). The steps (c) and (d) are repeated until there are no more infeasible elements and a stabilized grid is found. On this grid the column optimizer is run which merges columns and returns a minimal stable configuration (e). Afterwards the arrangement of columns is optimized (f).*

applied to them. In general, this is more of a problem when working with slender structural elements as used in truss structures.

According to ACI 318 [ACI08], buckling can be neglected as long as the Equation 10.6 of ACI 318 [ACI08] is met, resulting in non slender concrete columns. For columns with square shaped cross section this is the case when the ratio between the height of the column and the width of the column is less than 6.4. When assuming an effective column height of 2.6 m, buckling in concrete columns with four (six) steel bars and a compressional strength (C_{Col}) of at least 2100 kN (2500 kN) is negligible according to ACI 318, Equation 10.2 (Compressive strength of concrete = 16000 kN/m^2, compressive strength of steel = 400000 kN/m^2, diameter of steel bars = 20 *mm*, results are rounded). The goal of the model presented here is to have a structural calculation that can be done as fast as possible to allow for an iterative solver to evaluate it. In order to ensure that the simplifications made do not compromise the structural stability of a building, a comparison with a commercial finite-element system is presented in Section 6.3.2.

4.3. Structural Grid Generation

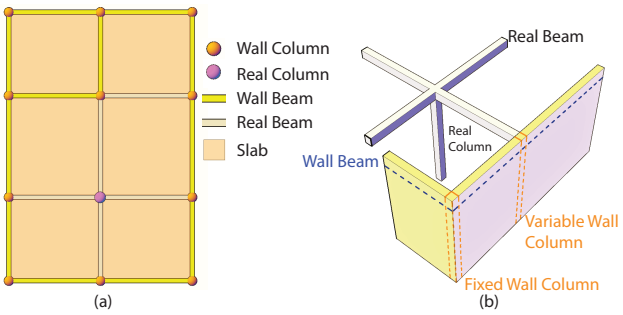

Figure 3: *Structural elements used in the optimization stage*

Taking the assumptions described above into account, the first goal is to create a structural grid as defined in Section 4.1 from an input floorplan. This grid serves as the basis for structural analysis and further optimization (cf. Figure 2a,b). It is created after each user interaction from the input modeled by the architect (cf. Figure 4a).

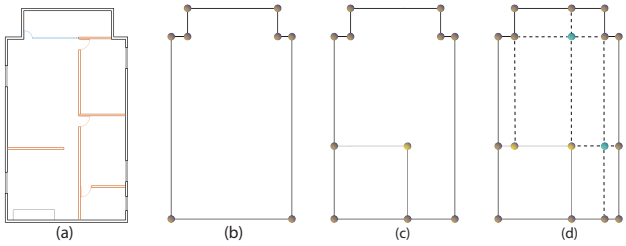

Figure 4: *Steps for generating a structural grid. (a) initial floorplan, (b) columns and beams are generated from boundary walls, (c) inner load bearing walls are added, (d) all slabs are split to rectangular areas by inserting additional beams.*

First, we generate a set of columns and beams (cf. Figure 4b) from the outer boundary of the floorplan. This is the envelope of the outer wall rectangle and all projections. Since all of these walls are load bearing, only fixed wall columns and wall beams are used. When the exterior wall beams are found, the interior is analyzed. For this, a grid containing all potential wall beams representing interior walls is constructed for each floor, which is done by inserting all beams and columns that are given by rooms. The grids of all floors are then intersected such that beams are split at beam-beam intersection points. Then all beams that do not exist in all floors are removed, just leaving such beams that represent interior wall segments that are present from top to bottom. The result of this operation is added to the exterior wall grid created in the first step and is depicted in Figure 4c. The so created grid is the base grid of the building, which is the same in all floors.

In order to analyze a single floor, the system starts with the base grid and adds all structural elements present in the floor above. Ad-

ditionally, such elements are added that are potentially only present in the particular floor, e.g., elements coming from stairs and voids which build connections between floors. For each void in the ceiling of a floor, four real columns connected by real beams are added below the boundary of the void. The columns are then connected to the next load-bearing wall segment in direction parallel to the shorter sides of the void. For stairs located in the floor above the current one, two columns are added below the lower ending of the stair. These columns are connected by real beams to the next load-bearing wall in all four directions.

Since several of the previously inserted structural elements can have intersections or overlaps, the grid is cleaned up by creating new columns at intersections and by splitting the beams there. In addition, columns and beams that share the same location are removed, just leaving the most restrictive element there.

The initial slabs are found as the faces of the planar graph defined by columns and beams. For each of these slabs, the system analyzes whether or not it is rectangular. If this is not the case, the slab is split by inserting additional beams going from columns with an inner angle larger than 180° to the next load bearing wall. This is repeated until all remaining slabs are rectangular. An example for a final structural grid determined by our algorithm is shown in Figure 4d.

4.4. Computation of Loads

At each level of a building, there are two main sources of load to be borne: Load coming from the ceiling itself (distributed loads) and loads that are transferred from levels above. The goal is to find the amount of load ω that is placed on columns and beams.

Beams. In a first step, the weight of the ceiling in each slab is distributed to the adjacent beams. In particular, three load areas are constructed for each side of the slab, two triangular ones and a rectangular one (cf. 5a). The force intervals along a side are given by $\{[0,a/2],[a/2,l-a/2],[l-a/2,l]\}$, where a is the length of the shorter side of the slab and l the current one. An example is given in Figure 5a, where load areas are marked by red lines and colored background. Note that the rectangular intervals have zero length on the smaller side of a slab, but for the model we assign a load of zero here to keep the calculation similar on all sides. For each beam incident to the slab, the overlap with these intervals is calculated, and the corresponding forces (shown by arrows) are computed. This procedure is performed for all slabs, such that each beam gets a number of distributed forces $F_D(b_{ij}) = \{f_0, f_1, ..., f_n\}$ attached to it. Each force has a position $p(f_i)$, which is in the center of the interval for rectangular areas and $2/3$ to the longer side for triangular ones. In addition, a load $w(f_i)$ is calculated, which corresponds to load area times a ceiling material constant C_{Ceil}.

The combined compressional load that is applied to a beam can then be calculated as the sum over all applied force:

$$\omega_D(b_{ij}) = \sum_{f \in F_D(b_{ij})} w(f), \quad b_{ij} \in B_W . \tag{4}$$

This load is only relevant in case of wall beams since real beams do not carry any loads themselves. It is nevertheless calculated for all beams in order to allow for merging of beams later on.

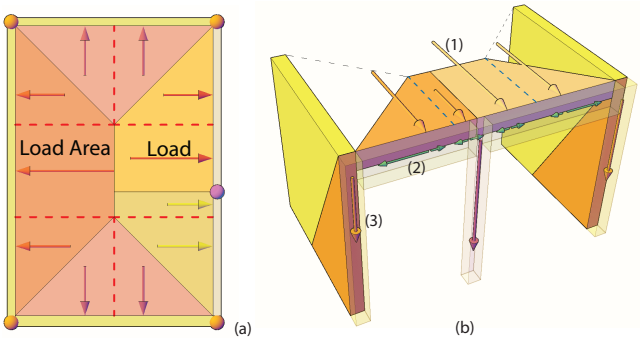

Figure 5: *(a) Each slab is split into several load areas (colored areas), with corresponding loads (arrows). (b) Loads coming from ceilings (1) are distributed via real beams (2) to columns, where they form concentrated loads (3).*

Columns. Since real beams cannot carry any load themselves, all loads applied to them needs to be transferred to the adjacent columns, resulting in equivalent concentrated loads on each column (see Figure 5b, where the loads of the ceiling (1) are distributed via beams (2) to the adjacent columns, where they result in concentrated loads (3)). To calculate these loads, the moment distribution method is used to find the reaction on supports based on the static equilibrium conditions:

$$\omega_D(c_i) = \sum_{b_{ij} \in B_R} \sum_{f \in F_D(b_{ij})} \frac{\|p(c_j) - p(f)\|}{l(b_{ij})} w(f), \tag{5}$$

where $l(b_{ij})$ is the length of the beam b_{ij}.

This gives a good estimation of the loads coming from the current level itself. In addition, each column and beam has an initial load ω_I, which is 0 for the topmost floor and for newly inserted structural elements. In all other floors, ω_I is given by the amount of load placed on the corresponding element in the floor above.

4.5. Computation of Compressional Strength and Capacities

In addition to the actual loads ω, also the maximum compressional strength Ω of a structural element has to be calculated. The method used depends on the type of structural element. Load-bearing wall beams have to resist distributed forces coming from the adjacent slabs. In contrast, real and wall columns have to resist concentrated loads transferred to them by real beams.

Beams. The capacity of each wall beam representing a load-bearing wall depends on the size of this wall and is computed as

$$\Omega(b_{ij}) = l(b_{ij}) \cdot T_{Wall} \cdot C_{Wall}, \; b_{ij} \in B_W, \tag{6}$$

where T_{Wall} is the thickness of the represented wall and C_{Wall} is a material constant describing the maximum load capacity of a wall. Since real beams transfer all their loads to the adjacent columns, their maximum capacity is set to zero.

Columns. For real columns, the capacity is given by a constant $\Omega_{c_i} = C_{Col}$, which depends on the material used and the size of the

column. Details on how this number can be determined for a given material are shown in ACI 318 [ACI08].

Wall columns are used to represent connections between real beams and walls, thus they represent transfer points where loads from beams are transferred onto walls. These loads are concentrated loads, since they are applied on a relatively small area of the wall. Concentrated loads spread out from the loaded area towards the bottom of a wall in a 60° trapezoid, but at maximum to the closest corner of the wall segment. The effective area of the bearing is determined as the size this trapezoidal shape has at the mid height of the wall. Since the same load is harder to bear on a smaller area, the enhancement factor β describes how much the capacity of the wall has to be enhanced for a given effective load area. According to the Eurocodes [Eur05], this maximum concentrated load is calculated as follows:

$$c_i \in C_{FW} \cup C_{VW} : \Omega c_i = \beta A_b C_{Concentrate}\,, \tag{7}$$

where

$$\beta = \left(1 + 0.3\frac{a_l}{h_c}\right)\left(1.5 - 1.1\frac{A_b}{A_{cf}}\right), \tag{8}$$

where

- $C_{Concentrate}$ is the compressional strength, which depends on the material used
- a_l is the distance from the nearest end of the beam to the loading point,
- h_c is the half room height,
- A_b is the loaded area,
- $A_{cf} = \min\left(l\left(b_{ij}\right)/2, h_c \sin\left(30^\circ\right)/\sin\left(60^\circ\right)\right) \cdot T_{Wall}$ is the effective area of the bearing,
- and $1.0 \leq \beta \leq \min(1.25a_l/2h_c, 1.5)$.

Note that here also combined structural elements have to be taken into account, especially for A_{cf}, since merging two columns can enlarge the number of walls beams adjacent to a wall column, which also increases the effective loaded area.

After calculating both, the actual load ω and the maximum capacity Ω, the system determines the stability of the level by checking the following relations:

$$\omega(c_i) \leq \Omega(c_i)\,,\ c_i \in C\,, \tag{9}$$
$$\omega(b_{ij}) \leq \Omega(b_{ij})\,,\ b_{ij} \in B\,. \tag{10}$$

Depending on the result of this comparison, columns and beams are marked as structurally feasible or not. Since wall columns represent the concentrated load of beams on walls, the adjacent wall beams of an unstable wall column are also marked as infeasible (cf. Figure 2c). Figure 7 also depicts the results of this stage for multiple buildings in column (2), with infeasible regions indicated in red.

5. Structural Optimization

When the result of the structural analysis module (Section 4) is presented to the user and infeasible structural elements are found, the user can request the system to suggest an optimal structural layout. This is done by first adding additional structural elements to get the grid in a stable configuration. Afterwards, an optimization is performed to remove as many columns as possible to find the best structural layout. Although we focus on masonry structures here, the algorithm can be applied to all types of structural systems where the load-bearing walls are made of homogeneous materials. The general method presented in this section can even be applied to load-bearing walls made of non-homogeneous material, e.g., steel reinforced concrete, when a proper structural analysis model is used.

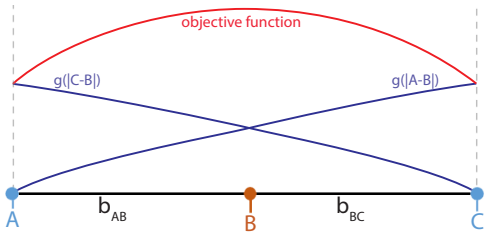

Figure 6: *The objective function (red) has its minima when the real column B is merged to either A or C.*

Our main idea here is to move as many columns to the same location, such that they can be merged. This means that only one representative column is required at the particular location, which reduced the total number of columns in the building.

The whole optimization takes several seconds up to a few minutes depending on the complexity of the floorplan and thus has to be requested by the user.

Stabilization. When a floor has elements that are not feasible, a stable superset of columns, beams and slabs is generated by adding additional columns in the centers of all slabs that contain unstable elements. Starting from these columns, additional beams are added along both principal directions ending at the first intersection with a wall beam. After insertion, the stability is calculated again, and the process is repeated until a stable grid is found (see Figure 2c,d).

Column Merging. During the optimization, the system will try to move as many columns as possible to the same location, thus the load calculation has to be extended to take into account that several structural elements at the same location act just as one structure. In case of columns, this means, that just one of the columns at a specific location will be used as representative column. This is done by calculating the sum of loads of all columns at the same location. This sum is then used as load for the representative column while the load on all other columns is set to zero. By always preferring wall columns over real columns when merging, it is ensured, that always the correct type is preserved (when a real column is integrated into a wall it becomes a wall column). Similarly, the system always prefers fixed columns over variable ones, which ensures that always the most restrictive one is used. This means that columns are always merged in the following order: $C_{VR} \rightarrow C_{FR} \rightarrow C_{VW} \rightarrow C_{FW}$.

For beams, a similar method is used: The loads of all beams at the same location are transferred to a representative beam, where wall beams are always preferred over real beams ($B_R \rightarrow B_W$). This choice is important, since for wall columns the feasibility has to be evaluated. Note, that merging two columns together always means that also the columns of the beams are merged.

Column Optimization. The next step is to optimize the layout computed in the previous stage by finding a stable configuration that has as few columns as possible. In the following equations, columns in the resulting configurations are named c_i', while c_i describes the initial columns.

Moving columns to the same position can be reformulated as moving columns such that the length of as many beams as possible gets zero. This has to be reflected by the objective function that needs to assign lower values to situations where a beam has reached zero length. Since the beams and columns combination that lie on the same line start and end at a wall column, the first and the last column can never be moved in direction of the beams. Thus, the total length of such a group will always stay the same, and when decreasing the length of one beam, the length of another one will increase by the same amount. When trying to minimize the sum of the beam length directly, then this will result in a constant function that cannot be optimized since the first derivative will be zero. When looking at Figure 6, where only column B can be moved, an objective function is required that has minima when B is either at the location of A or C. In order to express this, we define the objective function (shown in red in Figure 6) as follows:

$$obj(C,B) = \sum_{b_{ij} \in B} g(|p(c_i') - p(c_j')|)\,, \tag{11}$$

where $g(x)$ has to be a strictly increasing, differentiable and concave function in the interval $[0, \text{inf}]$, which means that the slope of the gradients is strictly decreasing. In our implementation, we use $g(x) = x^{0.9}$, but other roots or $g(x) = log(x+1)$ will provide very similar results. Note that convergence speed is better when gradients are not too flat in the relevant domain.

Next, the problem has to be constrained such that the topology of the structural grid is preserved. For this, linear constraints are added such that the position of all columns in C_{FW} and C_{FR} are fixed:

$$\forall c_i \in C_{FR} \cup C_{FW} : p(c_i') = p(c_i)\,. \tag{12}$$

For beams, constraints are added such that columns have to have the same position on the axis perpendicular to the beam. Additionally, the order of parallel columns has to be the same in the initial and in the resulting configuration as long as the columns are not merged together:

$$\forall b_{ij} \in B | p_x(c_i) = p_x(c_j) :$$
$$p_x(c_i') = p_x(c_j') \tag{13}$$
$$(p_y(c_i') - p_y(c_j'))\,\text{sign}(p_y(c_j) - p_y(c_i)) \leq 0\,, \tag{14}$$

and

$$\forall b_{ij} \in B | p_y(c_i) = p_y(c_j) :$$
$$p_y(c_i') = p_y(c_j') \tag{15}$$
$$(p_x(c_i') - p_x(c_j'))\,\text{sign}(p_x(c_j) - p_x(c_i)) \leq 0\,. \tag{16}$$

To prevent too large bending moments, constraints are added for each real beam to ensure that they have a maximum length of 8m:

$$l(b_{ij}) <= 8\,\text{m}\,,\ b_{ij} \in B_R\,. \tag{17}$$

The final necessary constraints have to ensure that the generated solutions are always structurally plausible. This is achieved by keeping the actual load on all elements below their maximum capacity:

$$\forall c_i \in C : \omega(c_i') \leq \Omega(c_i')\,, \tag{18}$$
$$\forall b_{ij} \in B : \omega(b_{ij}) \leq \Omega(b_{ij})\,. \tag{19}$$

Layout Optimization. The result of the column optimization stage, as seen in Figure 2e, is a structural grid with a minimum number of columns used to keep the building in a stable configuration. Due to the optimization, which tries to move columns as close together as possible, structural elements are visually not well distributed over the building. From an architectural point of view, it is not appreciated when columns are placed very close to walls. Thus, an additional layout optimization is performed with the goal of distributing variable columns equally over the floor. The objective function here has to penalize unequal distributions, which means that for each column, the adjacent beams have to have a length close to the average length of beams around the column:

$$obj(C,B) = \sum_{\substack{c_i \in \\ C_{VR} \cup C_{VW}}} \sum_{b_{ij} \in B} \left| l_x(b_{ij}) - \bar{l}_x(c_i) \right| + \left| l_y(b_{ij}) - \bar{l}_y(c_i) \right|\,, \tag{20}$$

where $l_x(b_{ij})$ is the length of beam b_{ij} in x-direction and $\bar{l}_x(c_i)$ is the average length of all beams adjacent to c_i that are laid out in x-direction. Since beams are always axis-aligned in this work, either $l_x(b_{ij})$ or $l_y(b_{ij})$ is zero and the average length is calculated over either one or two beams. The linear and non-linear constraints (Equation 12-19) are the same as in the column optimization and ensure again that the topology of the grid is not changed and that all elements are structurally stable. In Figure 2f the result of the layout optimization can be seen where the variable columns are distributed equally over the building.

6. Results

6.1. Implementation

The user interface, together with grid generation and stabilization is implemented in a prototypical C# framework. The more complex calculations, load computation and optimization, are performed using MATLAB as backend. For the optimization itself, we use MATLAB's `fmincon` method with the `active-set` algorithm. Gradients are generated using central differencing. Since the objective function can be relatively flat, the function tolerance for the optimizer has to be set relatively low. To avoid numerical issues, we instead multiply the objective function by a factor of 10^2.

6.2. Results

Using the methods described in this paper, it is possible to generate plausible and structurally valid designs for a large number of buildings. In Figure 7, three different buildings are shown (B-D), where building D consists of four levels. Column (1) always shows the initial floor plan, column (2) shows the initial grid as described in Section 4, where the colors denote structural feasibility of elements. In column (3), the stabilized grid is shown, which serves as the input for the column optimization described in Section 5, followed by the final layout after layout optimization in column (4).

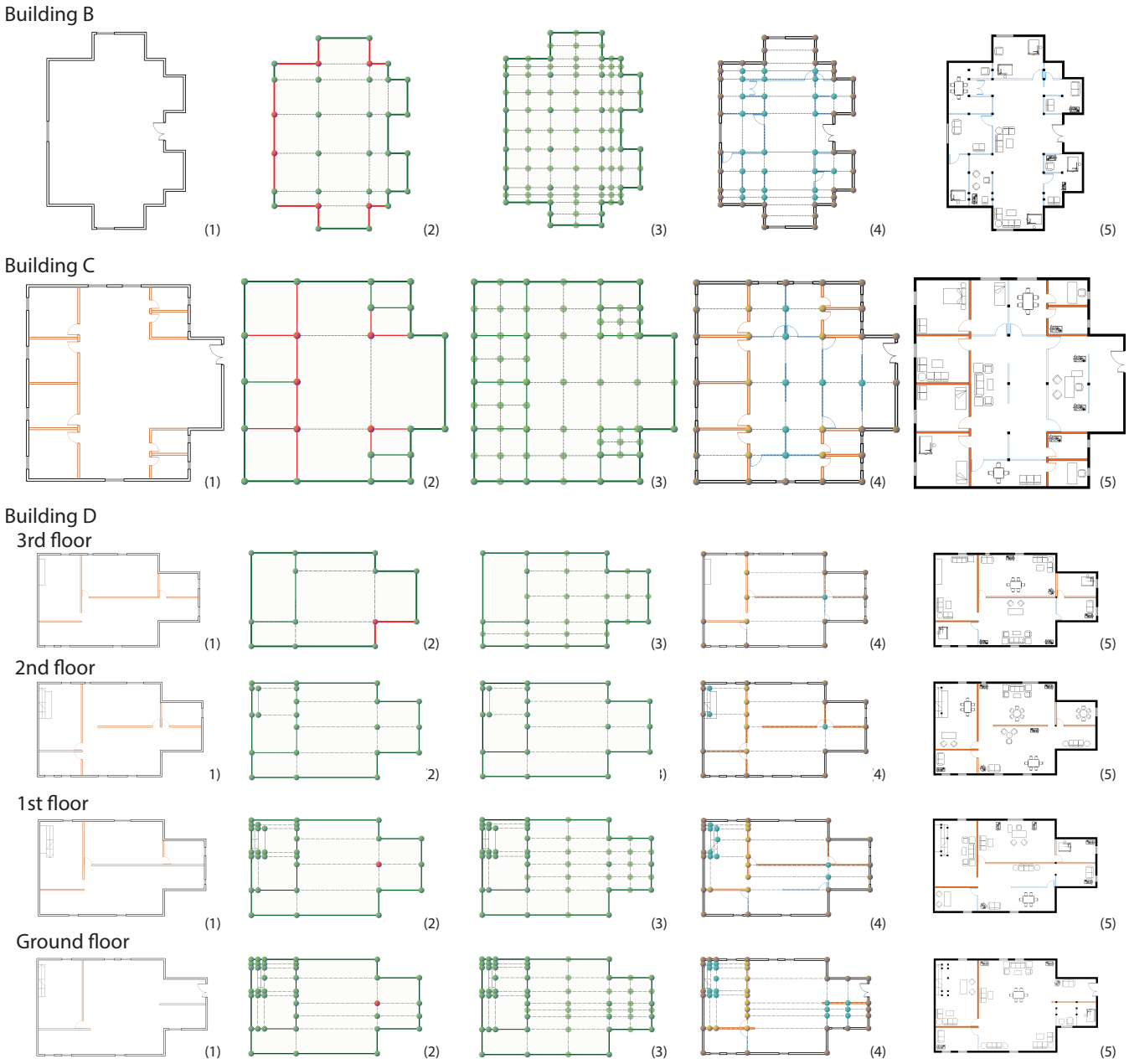

Figure 7: *Results for multiple buildings. Column (1) always show the initial floorplan, column (2) shows the initial layout derived from the floorplan, in column (3) the initial grid is stabilized. Column (4) shows the final result after column and layout optimization, while in column (5) an architect extended this results by non-load bearing elements. Load bearing walls are always given in orange, while partitions are painted in blue.*

To ensure that our results give architecturally plausible structural grids, we asked an architect to extend the generated results with additional non-load bearing elements like partitions and furniture. The results of this task can be seen in column (5). Figure 8a shows the input of building D rendered in 3D, while Figure 8b shows the final result of this building including added beams, columns and slabs. Similar results are shown in Figure 10, also showing how the final building could look like with added furniture.

As already mentioned in in the previous sections, there are several parameters that have to be specified in the optimization like the capacity of columns C_{Col}, the concentrate load capacity of walls $C_{Concentrate}$, the weight of the ceiling C_{Ceil} (per sqm) and the compressional strength of walls C_{Wall}. These values are given by the

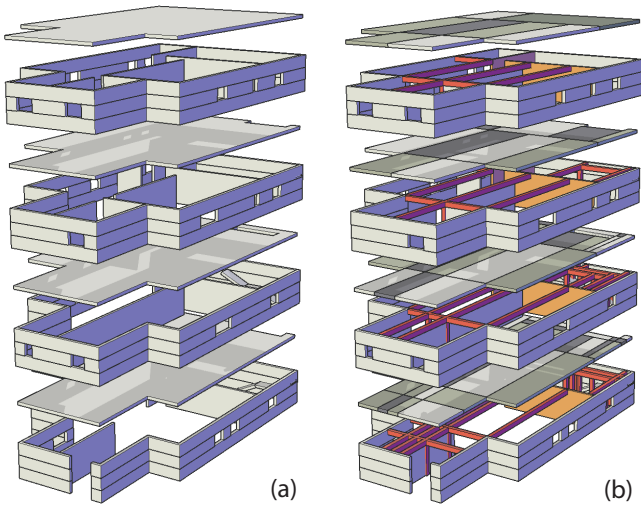

Figure 8: *Rendered 3D results of building D. Subfigure (a) shows the initial building while in (b) the added structural elements are overlayed.*

structural properties of the used building materials and the dimension of walls, beams and columns and can, for example, be found in the Eurocodes [Eur05], Section 3. Table 2 shows the used parameters for the four buildings presented in this paper.

According to the architects that worked with the implementation, the time required to design a structurally valid building could be decreased due to the integrated structural stability checking. Especially for adding windows, doors and partitions, the process gives valuable feedback about where they could be placed. The suggestions generated by this system are, in general, much faster available than with previous methods and do not only tell which elements are infeasible but can also propose a suggestion on where to place additional structural elements. For all the buildings shown here, the consistency check as well as the initial structural stability calculation finishes in several milliseconds. The FEM system in comparison took between 30 seconds (Building B) and 5 minutes (Building D).

The optimization takes, when requested by the user, between 45 seconds and several minutes for a final result, which is still an improvement compared to state-of-the-art work in this field (e.g.: Delgado et al. [Dav14] takes between 1.8 and 15.9 hours to complete).

6.3. Evaluation

To evaluate whether the proposed algorithm performs well, two major points have to be shown: That the proposed load calculation model produces correct results, which is done by comparing the initial load calculation and the result configuration of each building with a commercial FEM-package. In addition, it has to be shown that solutions produced by the algorithm are useful to architects and that they are similar or better than what an expert would produce.

6.3.1. Quality

In order to evaluate our method in terms of quality, a user study was performed with three users. (1) An expert, (2) an architecture student, and (3) an untrained person have been asked to find optimal solutions for the four example buildings shown in Figure 7. The goal here was to compare the results of our algorithm with manually designed solutions. When a user approaches this task and encounters a structurally infeasible configuration, they predict where new structural elements have to be added. The test persons started with the same inputs as shown in column (1) of Figure 7, and could use lines as beams, points as columns or any other ways to show how they would design a feasible model. When a floorplan was considered stable by the user, this was confirmed by a stability check using our load calculation method (Section 4) and the users were only allowed to continue with the lower storeys when their design was stable.

Since direct comparison of the user results with the solutions found by the optimization would not give meaningful results, a number of metrics is used to determine the quality:

- Number of inserted real columns for each floor (RC)
- Number of inserted wall columns for each floor (WC)
- Number of inserted real beams for each floor (RB) (intersected existing beams are not included)

An optimal solution would be a layout with as few real columns (RC) and beams (RB) as possible that is structurally stable. In addition, a lower number of wall columns (WC) means that beams are better aligned. When, for example, two adjacent rooms are split in the same direction, it is better to connect the two additional beams in a common point which makes them more robust against lateral forces. Comparing these three metrics from the manually created layouts with the automated solutions shows how well our system performs compared to users. In order to see whether our algorithm places additional elements on similar positions as users would do, four additional metrics are introduced:

$$M_a = \frac{N_a}{T_a}, \qquad M_b = \frac{N_b}{T_a}, \qquad M_c = \frac{N_c}{T_c}, \qquad M_d = \frac{N_d}{T_d},$$

where

- T_a := total number of split slabs in automatically generated model,
- T_c := total number of matching split slabs in both models,
- T_d := total number of matching slabs with same direction both models,
- N_a := number of matching split slabs in both models,
- N_b := number of different split slabs in manually created model,
- N_c :=number of matching split slabs with same directions of division in both models,
- N_d := number of matching split slabs with same direction of division and same divided parts in both models.

M_a: When inserting beams and columns to obtain feasible structural layouts, slabs are split. Therefore the placement users chose show whether they predict the correct slabs to be split, which is measured by this metric. This metric gives the ratio of similar split slabs compared to the total number of splits.

M_b: Although the majority of manually split slabs can be the same as in the automated optimal solutions, it is possible that users split additional, unnecessary slabs. To overcome this, the ratio of different split slabs to the total number of slabs is calculated here.

Building:	Building A				Building B				Building C				Building D															
Floor:	1st floor				1st floor				1st floor				4st floor				3st floor				2st floor				1st floor			
User:	Tool	1	2	3	Tool	1	2	3	Tool	1	2	3	Tool	1	2	3	Tool	1	2	3	Tool	1	2	3	Tool	1	2	3
RC	2	2	2	3	18	20	24	21	7	5	10	7	1	1	1	2	0	0	0	0	1	1	3	4	4	4	0	4
WC	2	2	2	2	14	8	10	12	7	11	8	10	2	2	2	2	0	0	0	0	2	1	2	3	4	2	0	2
RB	3	3	3	4	29	28	35	32	18	14	22	16	2	2	2	3	0	0	0	0	2	2	4	6	7	5	0	6
M_a	-	1	1	1	-	0.88	0.88	0.82	-	1	0.66	1	-	1	1	0.50	-	0	0	-	-	0	0.50	-	-	-	-	-
M_b	-	0	0	1	-	0.12	0.18	0.18	-	0.33	0	0.33	-	1	1	1	-	0	0	-	-	1	1.5	-	-	-	-	-
M_c	-	1	1	0	-	0.40	0.40	1	-	0.66	1	0.66	-	1	1	0	-	0	0	-	-	0	0	-	-	-	-	-
M_d	-	1	1	1	-	1	1	1	-	1	0.50	1	-	1	1	0	-	0	0	-	-	0	0	-	-	-	-	-

Table 1: *Results of the userstudy: RC - Added real column, WC - Added wall columns, RB - Added real beam. M_a - M_b Metrics to compare the building (see Section 6.3.1)*

Building	Floor	$C_{Col}(kN)$	$C_{Wall}(kN/m^2)$	$C_{Concentrate}(kN/m^2)$	$C_{Ceil}(kN/m^2)$
A	0	20k	20k	10k	25
B	0	10k	10k	10k	25
C	0	20k	10k	10k	25
D	3	2100	20k	20k	25
D	2	4800	20k	20k	25
D	1	4800	20k	20k	25
D	0	5800	20k	20k	25

Table 2: *Individual values of parameters used for the computation of buildings A-D. The last two columns show the runtime of the optimization algorithm for each floor and the runtime of the FEM system*

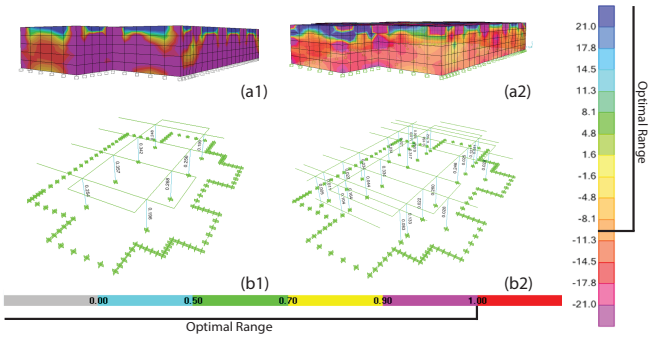

Figure 9: *Results of the FEM-model for building B: Vertical stress on walls are shown for the input model (a1) and for the final grid (a2). The vertical legend shows the optimal range for vertical stresses in kN. The stress ratio on beams and columns is shown in (b1) for the input model and in (b2) for the final grid. The horizontal legend describes these stress ratios for frames in percent (%).*

M_c: After comparing whether or not the same slabs are split, also the direction of this split is important. This metric gives the ratio of slabs split in the same direction to the total number of similar split slabs and is not evaluated when the result of M_a is already 0.

M_d: In contrast to the proposed algorithm, users could split one slab several times in the same direction. This metric measures the ratio of slabs that are split in the same direction and by the same number of splits to the total number slabs that are split in the same direction and is not evaluated when the result of M_c is already 0.

If the value of the results of M_a, M_c, and M_d is close to 1, and the result of M_b is close to 0, the generated solution is close to what the users have drawn. These additional metrics can only be used when the initial layout is the same for all users, which can not necessarily be assumed in multi-storey buildings.

Discussion. The results, given in Table 1, show that for simple models like Building A, whose load-bearing walls are just the boundary of a single floor, and which has a limited number of projections, our tool makes decisions which lead to a similar layout as drawn by educated architects (User 1 and 2 chose exactly the algorithmic solution, the untrained user placed too many additional columns). For more complicated models, where the number of projections is higher, the results show that decisions of the algorithm can be far from those made by an architect. In almost all cases, the users added more real columns and beams, which shows that our tool can specifically help architects in making decisions leading to more efficient designs. In multi-storey buildings, one user (User 2) was able to find a globally more optimal solution than our algorithm by choosing non-optimal solutions in upper levels that reduce the amount of columns in lower ones. Note that for this result, the user drawings had to be revised several times since the first tries were not stable. Finding such a solution is currently not possible with the proposed algorithm, since it optimizes each floor locally.

Regarding the location of splits, it can be seen that in simple buildings, our algorithm tends to split the same slabs in the same way as a user would do. Even in more complex buildings, the metrics M_a and M_b show that our tool splits mostly the same slabs. In particular, algorithm never splits more slabs then the users. Regarding the directions of splits, it can be seen that trained users split slabs mostly in the same direction. In some cases, the differences in choice happen when symmetric solutions give the same result, since such layouts have a similar quality in this case.

6.3.2. Correctness

In order to prove that our algorithm calculates the correct stability, we applied a Finite Element Model to our results by using the commercial software SAP2000. Beams and columns were modeled as frame elements, while slabs and load bearing masonry walls have been modeled as shell elements. Capacity parameters are assigned similar to our method. For each building, two grids were checked: The initial structural grid (c.f. Figure 7, Column 2) and the final optimized one (Column 5): For the initial grid, it was tested whether

our model marks exactly the same structural elements as infeasible that get a vertical ratio assigned by the FEM model which exceeds the allowed range. Walls are compressionally resistant when their vertical stress is greater than -10 kN, where negative stresses mean compression and positive numbers show tension stresses. Since walls have to be considered as a whole, they are unstable when more than 80% of a wall has a stress below -10 kN (marked in purple/red in Fig. 9(a1),(a2)). Stress ratios on frames describe the ratio between loads and capacities and should always stay below 1.0. Figure 9(a1) shows vertical stresses on walls while (b1) shows the stress ratio on frames for the initial model. For the final results of our algorithm, the FEM model was used to prove that all of the generated buildings are stable (Figure 9(a2),(b2)). In all the buildings tested, the comparison to the FEM model shows that exactly those structural elements are marked as infeasible that are also marked unstable by our method, while all our results are completely stable.

6.4. Limitations

Since this is a prototypical implementation, only one kind of each structural element is supported. For example, only straight one-way stairs are supported, but it would not be hard to add stairs with turns since this would only affect the initial grid generation. Another limitation is that the bending moment of real beams is not considered, which does not allow for beams longer than 8 meters. Since architects and structural engineers would also avoid such beams when working with conventional reinforced concrete slabs, this would only be needed in very rare cases.

Compared to previous work on structural optimization for masonry buildings [WSW*12], our method can modify the number of structural elements, but can handle only vertical structural elements and not arbitrary 3D designs. While the application of our method shows that it can improve structural layout decisions and reduce errors, the tool can also generate not exactly optimal solutions. This is since it currently does not cover additional possibilities of initial generation of structural members, like beam-to-beam connections, which could reduce the number of columns when the distances between columns are small, like with corners of voids and stairs. Our tool can also be improved by adding the possibility of changing the location of load-bearing walls. Finally, this tool can also be extended to find an optimal structural layout according to lateral forces, which has not been considered in this research.

7. Conclusions

We have presented an interactive system that helps architects to integrate structural knowledge into early architectural design. Our system interactively enforces structural and architectural rules and gives immediate feedback about the structural stability of a building, thus the architectural architect can explore a larger variety of actually feasible designs and reduce the risk of proposing designs that violate architectural or structural rules. The key contribution of our method is a novel optimization algorithm that makes suggestions on how to stabilize a building by varying not only the size and shape, but also the number of load-bearing elements. We proved by comparing our results with a traditional FEM-simulation that our method generates structural plausible results and showed that our system can be compared to manual results in a userstudy.

In the future, we would like to extend the approach to more general design problems not limited to vertical structural elements.

8. Acknowledgments

This research was funded by the Austrian Science Fund (FWF) projects Nr. FWF P27972-N31 and FWF P24600-N23.

References

[ACI08] ACI: COMMITTEE AND AMERICAN CONCRETE INSTITUTE AND INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: Building code requirements for structural concrete (aci 318-08) and commentary. American Concrete Institute. 4, 5, 7

[ACI14] ACI: COMMITTEE AND AMERICAN CONCRETE INSTITUTE AND INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: Building code requirements for structural concrete (aci 318-14) and commentary. American Concrete Institute. 2

[Blo09] BLOCK P.: *Thrust network analysis: A new methodology for three-dimensional equilibrium*. PhD thesis, 2009. 2

[BSK*13] BARTOŇ M., SHI L., KILIAN M., WALLNER J., POTTMANN H.: Circular arc snakes and kinematic surface generation. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 1–10. 2

[BYMW13] BAO F., YAN D.-M., MITRA N. J., WONKA P.: Generating and exploring good building layouts. *ACM Transactions on Graphics 32*, 4 (July 2013), 1. 2

[Dav14] DAVILA DELGADO J. M.: *Building structural design generation and optimisation including spatial modification*. PhD thesis, 2014. 10

[DH13] DAVILA DELGADO J. M., HOFMEYER H.: Automated generation of structural solutions based on spatial designs. *Automation in Construction 35* (Nov. 2013), 528–541. 2

[Eur04] EUROCODE: Eurocode 8: Design of structures for earthquake resistance-part 1: General rules, seismic actions and rules for buildings. *London: BSi* (2004), 10–12. 3

[Eur05] EUROCODE: Eurocode 6: Design of masonry structures, part 1–1: General rules for reinforced and unreinforced masonry structures. *European Committee for Standardization, Belgium* (2005). 2, 3, 7, 10

[FRG00] FENVES S. J., RIVARD H., GOMEZ N.: SEED-Config: a tool for conceptual structural design in a collaborative building design environment. *Artificial Intelligence in Engineering 14*, 3 (July 2000), 233–247. 2

[GK02] GRIERSON D. E., KHAJEHPOUR S.: Method for Conceptual Design Applied to Office Buildings. *Journal of Computing in Civil Engineering 16*, 2 (Apr. 2002), 83–103. 2

[HD13] HOFMEYER H., DAVILA DELGADO J. M.: Automated design studies: Topology versus One-Step Evolutionary Structural Optimisation. *Advanced Engineering Informatics 27*, 4 (Oct. 2013), 427–443. 2

[HD15] HOFMEYER H., DAVILA DELGADO J. M.: Coevolutionary and genetic algorithm based building spatial and structural design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 29*, 04 (2015), 351–370. 2

[HvRG11] HOFMEYER H., VAN ROOSMALEN M., GELBAL F.: Preprocessing parallel and orthogonally positioned structural design elements to be used within the finite element method. *Advanced Engineering Informatics 25*, 2 (Apr. 2011), 245–258. 2

[KR97] KUMAR B., RAPHAEL B.: CADREM: A case-based system for conceptual structural design. *Engineering with Computers 13*, 3 (Sept. 1997), 153–164. 2

[LYAM13] LIU H., YANG Y.-L., ALHALAWANI S., MITRA N. J.: Constraint-aware interior layout exploration for pre-cast concrete-based buildings. *The Visual Computer 29*, 6-8 (May 2013), 663–673. 2

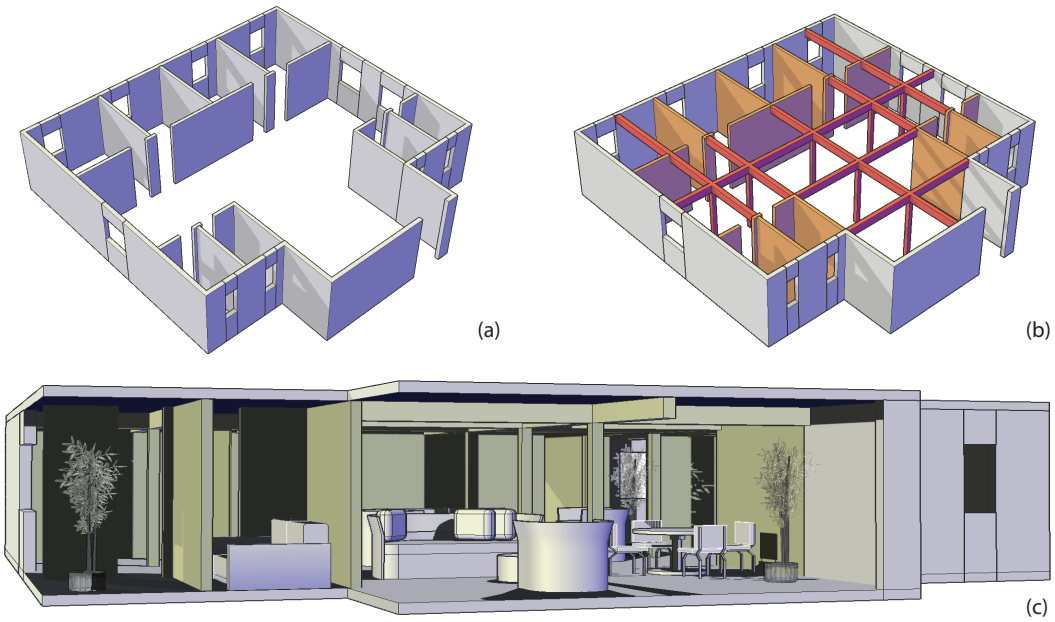

Figure 10: *Results of building C: The initial layout (a) is supported with additional structural members (b). Subfigure (c) shows how the building could look like with furniture.*

[Mah84] MAHER M. L.: *HI-RISE: a knowledge-based expert system for the preliminary structural design of high rise buildings*. PhD thesis, Carnegie-Mellon University, 1984. 2

[MIB15] MIKI M., IGARASHI T., BLOCK P.: Parametric self-supporting surfaces via direct computation of airy stress functions. *ACM Transactions on Graphics 34*, 4 (July 2015), 89:1–89:12. 2

[MSK10] MERRELL P., SCHKUFZA E., KOLTUN V.: Computer-generated residential building layouts. *ACM Transactions on Graphics 29*, 6 (Dec. 2010), 1. 2

[NN09] NIMTAWAT A., NANAKORN P.: Automated layout design of beam-slab floors using a genetic algorithm. *Computers & Structures 87*, 21-22 (Nov. 2009), 1308–1330. 2

[NN10] NIMTAWAT A., NANAKORN P.: A genetic algorithm for beam-slab layout design of rectilinear floors. *Engineering Structures 32*, 11 (Nov. 2010), 3488–3500. 2

[Owe90] OWEN T.: Knowledge-Based Approaches To Structural Design by D. Sriram Computational Mechanics Publications, (Topics in Engineering Volume 1) Southampton (UK), 164 pages, incl. index (Âč19.50). *Robotica 8*, 01 (Jan. 1990), 88. 2

[PG99] PARK K.-W., GRIERSON D. E.: Pareto-Optimal Conceptual Design of the Structural Layout of Buildings Using a Multicriteria Genetic Algorithm. *Computer-Aided Civil and Infrastructure Engineering 14*, 3 (May 1999), 163–170. 2

[Pla07] PLACE J. W.: *Architectural structures*. Wiley, 2007. 4

[PSB*08] POTTMANN H., SCHIFTNER A., BO P., SCHMIEDHOFER H., WANG W., BALDASSINI N., WALLNER J.: Freeform surfaces from single curved panels. *ACM Transactions on Graphics 27*, 3 (Aug. 2008), 1. 2

[RMB03] RAFIQ M. Y., MATHEWS J. D., BULLOCK G. N.: Conceptual Building Design - Evolutionary Approach. *Journal of Computing in Civil Engineering 17*, 3 (July 2003), 150–158. 2

[SH12] SMULDERS C. D. J., HOFMEYER H.: An automated stabilisation method for spatial to structural design transformations. *Advanced Engineering Informatics 26*, 4 (Oct. 2012), 691–704. 2

[SMG08] SHAW D., MILES J., GRAY A.: Determining the structural layout of orthogonal framed buildings. *Computers & Structures 86*, 19-20 (Oct. 2008), 1856–1864. 2

[SMM03] SISK G. M., MILES J. C., MOORE C. J.: Designer Centered Development of GA-Based DSS for Conceptual Design of Buildings. *Journal of Computing in Civil Engineering* (June 2003). 2

[SPnM00] SOIBELMAN L., PEÑA MORA F.: Distributed Multi-Reasoning Mechanism to Support Conceptual Structural Design. *Journal of Structural Engineering 126*, 6 (June 2000), 733–742. 2

[SSL*14] SCHULZ A., SHAMIR A., LEVIN D. I., SITTHI-AMORN P., MATUSIK W.: Design and fabrication by example. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 62. 2

[SW97] SACKS R., WARSZAWSKI A.: A project model for an automated building system: design and planning phases. *Automation in Construction 7*, 1 (Dec. 1997), 21–34. 2

[SWK00] SACKS R., WARSZAWSKI A., KIRSCH U.: Structural design in an automated building system. *Automation in Construction 10*, 1 (Nov. 2000), 181–197. 2

[Wah07] WAHL I.: *Building Anatomy (McGraw-Hill Construction Series): An Illustrated Guide to How Structures Work*. McGraw Hill Professional, 2007. 4

[WOD09] WHITING E., OCHSENDORF J., DURAND F.: Procedural modeling of structurally-sound masonry buildings. In *ACM SIGGRAPH Asia 2009 papers on - SIGGRAPH Asia '09* (New York, New York, USA, Dec. 2009), vol. 28, ACM Press, p. 1. 2

[WSW*12] WHITING E., SHIN H., WANG R., OCHSENDORF J., DURAND F.: Structural optimization of 3D masonry buildings. *ACM Transactions on Graphics 31*, 6 (Nov. 2012), 1. 1, 2, 12

[ZTY*13] ZHAO X., TANG C.-C., YANG Y.-L., POTTMANN H., MITRA N. J.: Intuitive design exploration of constrained meshes. In *Advances in Architectural Geometry 2012*. Springer, 2013, pp. 305–318. 2