Optimal Design of new Hospitals

A Computational Workflow for Stacking, Zoning, and Routing

Cubukcuoglu, Cemre; Nourian, Pirouz; Sariyildiz, I. Sevil; Tasgetiren, M. Fatih

DOI
10.1016/j.autcon.2021.104102

Publication date
2022

Document Version
Final published version

Published in
Automation in Construction

Citation (APA)
Cubukcuoglu, C., Nourian, P., Sariyildiz, I. S., & Tasgetiren, M. F. (2022). Optimal Design of new Hospitals: A Computational Workflow for Stacking, Zoning, and Routing. *Automation in Construction*, *134*, Article 104102. https://doi.org/10.1016/j.autcon.2021.104102

Automation in Construction 134 (2022) 104102

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Optimal Design of new Hospitals: A Computational Workflow for Stacking, Zoning, and Routing

Cemre Cubukcuoglu [a,b,*], Pirouz Nourian [a], I. Sevil Sariyildiz [a], M. Fatih Tasgetiren [c]

[a] Department of Architectural Engineering and Technology, Chair of Design Informatics, Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, the Netherlands
[b] Department of Interior Architecture and Environmental Design, Faculty of Architecture, Yasar University, Izmir, Turkey
[c] Industrial and Systems Engineering Department, Auburn University, USA

ARTICLE INFO

Keywords:
Computational design
Hospital design optimization
Layout planning
Corridor generation
Mixed integer programming

ABSTRACT

The paper proposes a generative design workflow for three major hospital layout planning steps to satisfy multiplex configurational requirements. The initial step is stacking through clustering functional spaces into floor plans, for which a spectral method is presented. Subsequently, a novel simultaneous process of zoning and routing is proposed as a Mixed-Integer Programming problem-solving task; performed on a quadrilateral mesh whose faces and edges are allocated respectively to the rooms and the corridors. The paper situates the workflow in the context of an Activity-Relations-Chart for a general hospital while demonstrating, explaining, and justifying the generated optimal floor plans. The conversion of the hospital layout problem to a Mixed-Integer Programming problem enables the use of existing Operations Research solvers, allowing for the generation of optimal solutions in a digital design environment. The comprehensive problem formulation for a real-world scenario opens a new avenue for utilization of mathematical programming/optimization in healthcare design.

1. Introduction

The healthcare sector is one of the most challenging and fastest-growing industries around the world. Hospital buildings are the most complex buildings of architectural design field [1,2]. Thus, these facilities are requiring huge building programs for many different users while satisfying standards, architectural requirements, and engineering aspects [3]. These buildings are having a large impact on their environment [4]. In addition, their design requires expertise knowledge and experience. Due to these facts, healthcare facilities are accepted as an important architectural public design type in the built environment towards its design complexity. Especially in the early phase of design process, their functionality should not be disregarded and their spatial configuration has a great impact on their functional performance. Otherwise, unsuitable layout of the working spaces affected medical staff performance and can cause difficulties such as medical error, stress levels or work concentration [5]. Hospital designs typically require systematic design methodologies and computational design tools in order to satisfy sophisticated adjacency/closeness requirements in response to programmatic necessities, project site specifications, surrounding buildings, in addition to environmental requirements such as daylight.

Computational design techniques can alleviate the complexity of dealing with many such performance criteria practically. More specifically, by shifting the attention from the geometry of a layout to its topology and its network structure, spatial configurations can be directly analysed, synthesised, and evaluated in computational design workflows. The importance of directly dealing with a topological abstraction of spatial configuration lies in the direct link between the network of walkable spaces (hereinafter referred to as spatial configuration) and the adjacency/closeness requirements that must be fulfilled in hospital design processes. This link exists because of the direct link between network structure and movement potentials and probabilities [6]. More specifically, computational design of spatial configurations can be practiced in procedural CAD environments and support decision-making in design processes with several advantages [6] as follows:

* Corresponding author at: Department of Architectural Engineering and Technology, Chair of Design Informatics, Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, the Netherlands.
E-mail addresses: c.cubukcuoglu@tudelft.nl, cemre.cubukcuoglu@yasar.edu.tr (C. Cubukcuoglu), p.nourian@tudelft.nl (P. Nourian), i.s.sariyildiz@tudelft.nl (I.S. Sariyildiz), mzt0029@auburn.edu (M.F. Tasgetiren).

https://doi.org/10.1016/j.autcon.2021.104102
Received 6 April 2021; Received in revised form 18 November 2021; Accepted 9 December 2021
Available online 23 December 2021

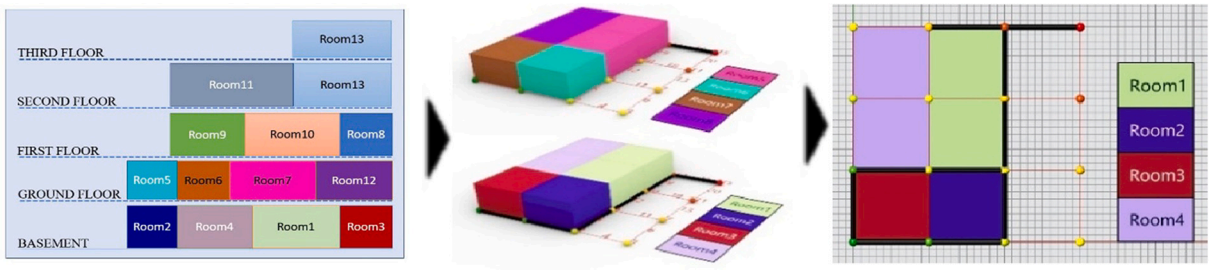

Fig. 1. (a) stacking, (b) zoning, (c) routing.

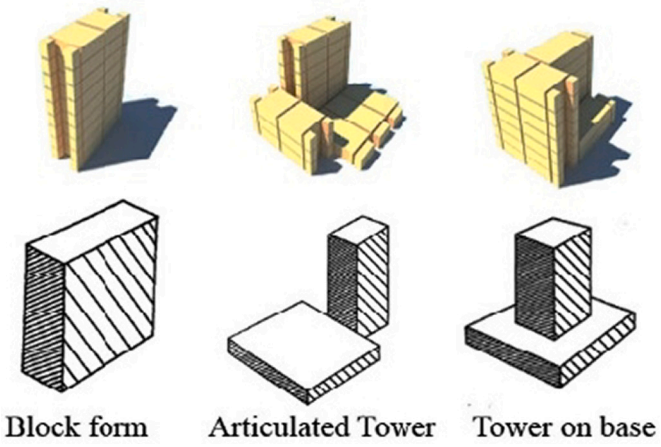

Fig. 2. Vertical Hospital Typology Examples [44,45].

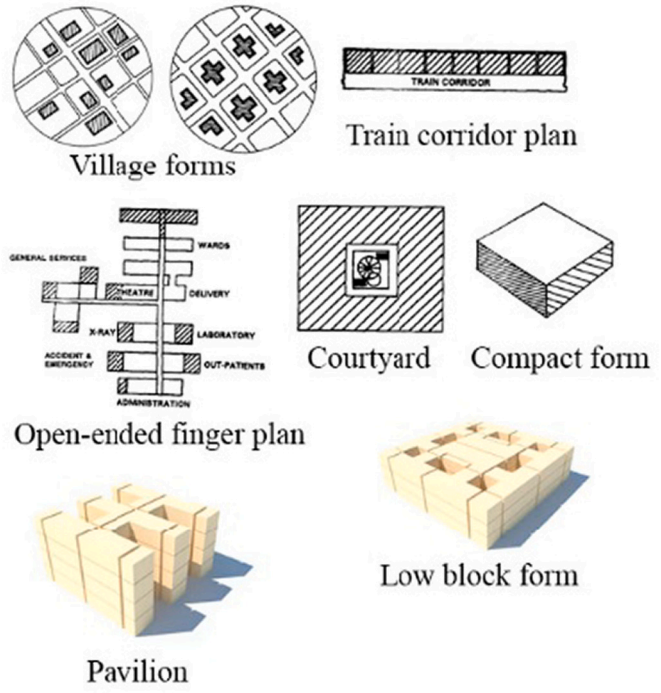

Fig. 3. Horizontal Hospital Typology Examples [44,45].

- Providing for objectivity in decision making by making decision bases transparent
- Providing for systematization of design processes;
- Creating new design tools by scripting;
- Customizing existing design tools;
- Allowing (semi) automation in the design process;
- Speeding up decision-making processes in conceptual design; and, most importantly,
- Facilitating evaluation of designs by enabling mathematical operations or computational simulations, and allowing for visual inspections (as enabling 2D and 3D models)

In this paper, we present a formulation of configuration design problems as a task combining room placement and corridor generation and present a configurational design methodology for hospital design for solving such problems. Briefly, the framework encompassing this methodology consists of stacking (Fig. 1a), zoning (Fig. 1b), and routing (Fig. 1c) steps; respectively deciding on how to divide the floors between functional areas, placing rooms, and embedding corridors at the same time. The methodology is intended to be modular, extendable, and easy to integrate with other computational design workflows and spatial decision support systems. All of the steps are implemented with a customized computational design tool that we developed especially for hospital designs. At the first step, the stacking problem is solved by graph-theoretical approaches. In zoning and routing parts, multi-level hospital layout problems and corridor design problems are addressed using Mixed Integer Programming Techniques (MIP) (See Figs. 2–5).

2. Overview of layout planning as MIP

Hospital layout problems can be positioned into a sub-category of Facility Layout Problems (FLP). According to our literature review, hospital layout problems as FLP have been solved by using Quadratic Assignment Problem (QAP) and MIP methods [7,8]. However, we see that the [ordinary formulation of] QAP is better suited to layout design in renovation scenarios for the placement of the centres of departments [9]. This is because QAP is essentially about minimizing the sum of travelled distances between points (a.k.a. the departments), whereas here we have the problem to first allocate locations/faces to the room surfaces; thus before having the rooms the QAP formulation is not straightforward to apply in a new design; this is because even after finding the locations of the room centres we still need to grow the rooms to certain sizes and deal with several constraints that are difficult to address in QAP. Therefore, in this paper, we refer to MIP methods that allow for the simultaneous layout of departments and corridors while dealing with such validity constraints as adjacency, cohesion, and alike. From an architectural point of view, this can only be possible if we can implement the MIP method on a discretised mesh model of the building envelope. We used quadrilateral meshes in the model because most of the real-world layout problems are either quadrilateral or can be tessellated as a quadrilateral mesh [10]. However, the same methods can be applied without loss of generality to other kinds of mesh tessellations (e.g. triangular, mixed triangular/quadrilateral, or polygonal) if needed in a design problem.

An optimization problem is called an Integer Program (IP) if any of its decision variables are restricted to be discrete (an integer as opposed to a more common float approximating a real number); if all variables

Proposed Hospital at Shillong (all areas are in sq. ft. and are carpet areas) **STACK PLAN**

Department	Basement	Lower Ground	Upper Ground	First	Second	Third	Fourth	Total
Public Areas			1152					1152
Casualty Department			1118					1118
Out-Patient Department				1313				1313
Radiology / Imaging & Diagnostics			1352					1352
Operation Theater Complex							2133	2133
Intensive Care Units							2444	2444
In-Patient Wing					6000	6000		12000
Central Sterilization & Processing Department							538	538
Clinical Laboratory				1287				1287
Kitchen & Dining			1269					1269
Administration				1356				1356
Building Services	6400		1100					7500
Parking		Parking						0
Sub-Total	6400	0	5991	3956	6000	6000	5115	33462
Add Vertical Circulation and BUA @25%	1600	0	1498	989	1500	1500	1279	8366
TOTAL	8000	0	7489	4945	7500	7500	6394	41828

Fig. 4. Stack plan example of a vertical hospital in Shillong, India [32].

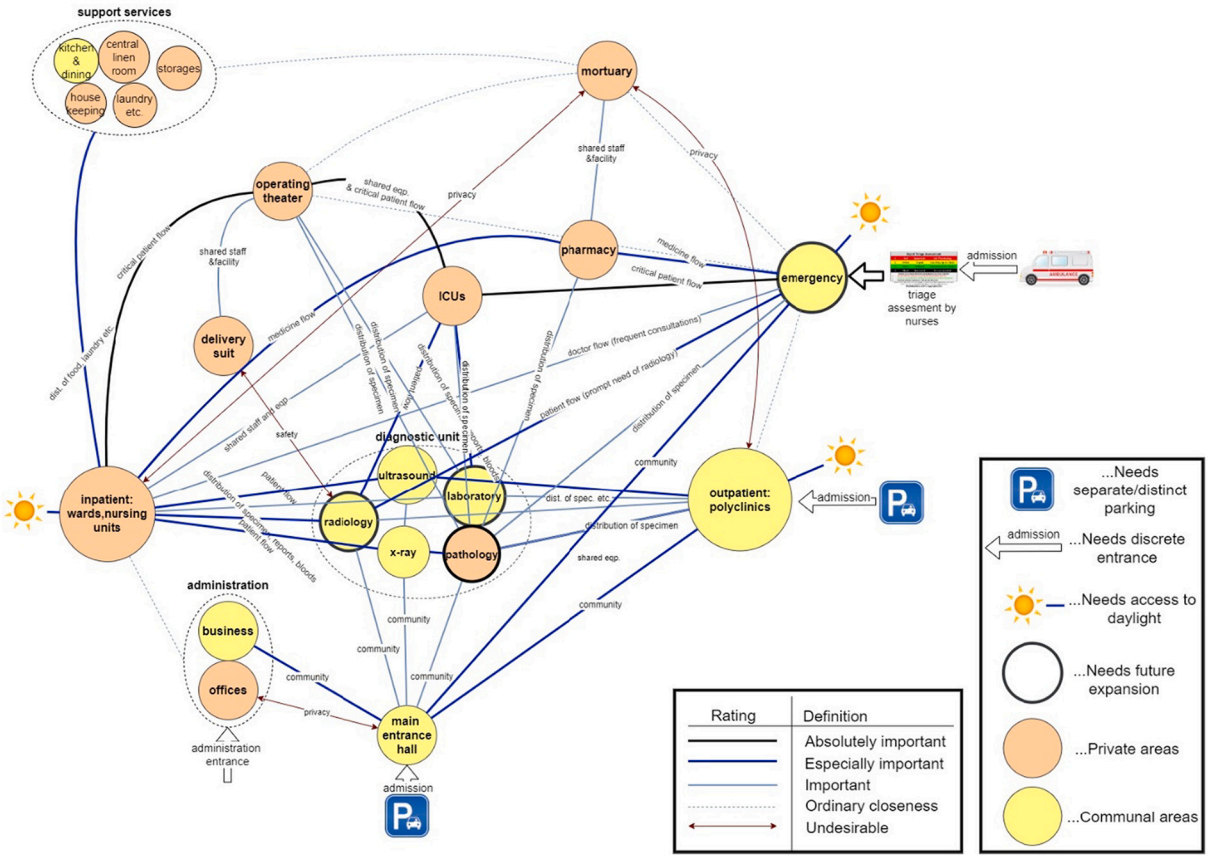

Fig. 5. A closeness diagram (graph) of main and sub-units of a typical middle-sized hospital (drawn by author after preliminary research using site visits, expert interviews, and literature review).

are discrete, the model is a pure IP; otherwise, the model is called a mixed-integer program (MIP). It must be noted that the terms programming and optimization are used almost interchangeably in computational contexts, albeit the term programming is colloquially associated with the most well-known types of optimization problems in Operations Research. MIP optimization problems with a quadratic objective are called Mixed-Integer Quadratic Programming (MIQP) problems. Whereas problems without any quadratic or higher-dimensional terms are often referred to as Mixed Integer Linear Programming (MILP) problems; if there are nonlinear functions in the objective function and/or the constraints, it is called Mixed-Integer Non-Linear Programming (MINLP).

In the literature, MIP is widely considered a central formulation in facility layout planning [11]. In general, Peters [12] introduced an MIP model for facility layout design in flexible manufacturing systems by proposing a genetic algorithm (GA). Liu and Meller [13] proposed an MIP formulation based on a sequence-pair representation approach for unequal area facility layout design by presenting a GA-based heuristic algorithm. Bozer and Wang [14] formulated an unequal area facility layout problem as an MIP model in which binary (0/1) variables are used to prevent departments from overlapping with one another by proposing a heuristic procedure based on the graph-pair representation technique and a simulated annealing (SA) algorithm. Kosucuoglu and Bilge [15] proposed a mathematical programming approach including

Table 1
Customary codes for closeness ratings.

Absolutely Important		A	1.00
Especially Important		E	0.75
Important		I	0.50
Ordinary		O	0.25
Unimportant		U	0.00

one MINLP and two MIP models for facility layout of flexible manufacturing systems by using a GA-LP heuristic algorithm. Murray et al. [16] considered a double row layout as a facility layout problem where rectangular machines of unequal size must be placed in two rows separated by a straight aisle of predetermined width using a tabu search heuristic. Kulturel-Konak and Konak [17] formulated an MIP to minimize the total material handling cost as a function of the distance and the amount of material flow between departments by proposing a hybrid GA/LP approach for unequal area facility layout design. Xiao and Seo [18] formulated an MIP for unequal area layout design to minimize the total material-handling in a manufacturing system by using two-step heuristics comprised of both construction and improvement using simulated annealing. Hong et al. [19] developed a MIP model for solving the design problem of facility layout and automated material handling system (AMHS) for semiconductor fabrication facility by using a pairwise interchange method with a tabu search metaheuristic. Hammad et al. [20] proposed a general MIP formulation for the site layout planning problem, which is a well-studied layout problem that requires finding an appropriate physical arrangement of temporary facilities operating on construction sites using a cutting plane algorithm and exact location-decomposition algorithm. Lacksonen [21] used an MILP model to find the block diagram layouts with varying department areas using a branch and bound algorithm. Leno et al. [22] solved an unequal area facility layout problem with an MIP formulation by using elitist strategy hybrid GA–SA algorithm. Recently, Wan et al. [23] dealt with a multi-row facility layout problem by using an improved multi-objective greedy randomised adaptive search procedure. Liu et al. [24] addressed a single-row facility layout problem by using CPLEX software [25] and a constrained improved fireworks algorithm. A recent literature review on facility layout planning can be found in [7].

There are, however, only a few examples that make use of MIP techniques for architectural layout planning (ALP) despite their common use in FLPs. In conjunction to find optimal solutions, Keatruangkamala and Sinapiromsaran [26] firstly proposed MIP to solve ALPs by adapting more objectives than classical FLPs have. They formulated an architectural layout design problem as a multi-objective MIP using an MIP solver with a weighted sum approach. Problem objectives are the minimization of the absolute distance among rooms and the maximization of room spaces. Problem constraints are the connectivity, the unused grid cells, the fixed room location, the boundary and the fixed border location and the non-intersecting, the overlapping, the

Table 2
An exemplary REL-Chart for the main units of a typical middle-size hospital considered in this paper.

	Outpatient	Inpatient	Intensive Care Unit	Operation Theatre	Emergency	Diagnostics	Administration	Entrance Hall	Support
Outpatient									
Inpatient									
Intensive Care Unit									
Operation Theatre									
Emergency									
Diagnostics Unit									
Administration									
Entrance Hall									
Support									

length, and the ratio constraints. Keatruangkamala and Nilkaew [27] presented a practical formulation of MIP based on the strong valid inequality constraints, which reduced the feasible region using LP relaxations to solve the medium size architectural layout design. In a later work, Keatruangkamala and Sinapiromsaran [28] used an MIP formulation for the architectural layout to optimize room positioning, room sizes, and distances according to the architect's preferences subject to functional, dimensional, non-circular, and guided constraints using the CPLEX solver [25].

Recent hospital layout design reviews can be found in [9] and [29]. More specifically, regarding the MIP methods in hospital planning, we provide a short review here: Amaral [30] focused on the corridor-allocation problem for double-row hospital layout to minimize daily traffic and communication cost among facilities. It was formulated as an MIP with a quadratic objective (multiplication of flows and distances) using heuristic approaches (2-opt and 3-opt algorithm). Helber et al. [31] proposed a hierarchical modelling approach that divides the whole problem into sub-problems, where the first stage was formulated as a QAP for assigning elements to locations using fix-optimize heuristic, and the second stage was formulated as an MIP for detailed positioning within a site, considering space requirements, in a large hospital facility. They concentrated on transportation processes, fixing some units in specific locations, ensuring the direct neighbourhood of some pairs of units. Chraibi et al. [32] minimized total traveling cost and rearrangement cost in a dynamic facility layout problem (deciding the locations of the departments in a facility over multiple planning periods) for the operation theatre department of a hospital. They used both MIP and QAP formulations using a CPLEX solver. Acar and Butt [33] dealt with the nurse-patient assignment problem in a 29-bed oncology unit and formulated it as MILP and solving it with a simplex solver. Butler et al. [34] proposed a multi-level approach consists of MIP formulation and system simulation for capacity allocation problem of facility layout in a general-purpose hospital by considering distances between services. Safarzadeh and Koosha [35] formulated a non-linear MIP model with fuzzy constraints and converted it to a linear MIP model for a multi-row hospital facility layout. They minimized handling cost and lost opportunity cost related to waste spaces using GA. Recently, Wang et al. [36] proposed a MINLP model for beds allocation of a hospital in Shanghai by using an Adaptive Hyperbox Algorithm. Huo et al. [8] addressed a multi-floor hospital facility layout problem in a hospital in Shanghai, China, based on a double-row model in which all departments are arranged into two rows on each floor for minimizing the total movement distance of patients and maximizing the total closeness rating score by using a Non-Dominated Sorting GA-II algorithm.

To sum up, the works that utilize MIP in hospital layout design mostly consider transportation cost as an objective and often see the problem as a facility layout problem. They typically do not consider some specific architectural features required by hospital layout designs such as way-finding [37], daylighting [38], or privacy [39]. As stated in [9], the adaptation of FLP techniques to the hospital is not commonly known or used by architects. However, in practice, hospital layouts are designed by architects in collaboration with space programmers and doctors. Thus, we propose to utilize MIP techniques in hospital layout design by additionally considering architectural aspects of hospital design. In this paper, we present a hierarchical layout design methodology for hospital layout planning and consider MILP integrated with required architectural design features of hospital layouts. Unlike QAP, this methodology is also convenient for designing new buildings.

2.1. Related works

In Peng et al. [10], the floor planning problem is solved by a simultaneous approach to creating rooms and corridors in one optimization problem formulated as an MIP model with linear objectives and constraints or a Mixed Integer Linear Programming (MILP) problem.

In [10], plan layouts are generated with a set of pre-defined room templates that have fixed areas and shapes as combinations of squares. However, in a hospital, each room can require different square meters generally due to restrictive hospital design standards. Therefore, our plan layout approach enables each room (hospital department) to have various areas and shapes.

While the corridor design procedure of our approach is inspired by the work of Peng et al. [10], we have followed a step-by-step procedure combining room layout and corridor layout in our approach. First, varying hospital-department shapes are assigned to mesh faces based on an MIP model, which considers area requirements and some architectural needs. Then, using the output of the room layout procedure, corridors are created in-between the hospital departments with another MIP model, which considers objectives and constraints related to the hospital logistics network.

In addition, unlike [10], we consider relationships between rooms in layout planning by using relationship charts (REL chart), which is a common tool in FLP in industrial engineering, and Graph Drawing algorithms.

Similar to our approach, some works utilize such a graph-theoretical coordinate system and use MIP models in layout planning [26,40]. However, they do not focus on corridor design problems. In other words, the flow of pedestrians or materials in a building layout is disregarded not only in these papers but also in the majority of computational layout approaches. Besides, they are not based on a mesh model nor do they interact with CAD software used in architectural practice.

There are also some studies [41,42] focusing on both packing the rooms and routing passageways with a hierarchical approach. Merrell et al. [41] has some additional similarities with our approach in terms of considering relationships between spaces using Bayesian networks trained based on a corpus of observed realistic floor plans (in order to form a plausible adjacency matrix). Even though their method seeks to comply with such machine-generated adjacency recommendations, it does not focus specifically on optimizing closeness criteria, because their focus is only on generating realistically looking buildings for computer graphics applications, whereas in our case, hospital design, optimization with respect to closeness criteria is central to the whole problem. However, their flexible multi-level approach allows rooms to be placed within a boundary area without having unused cells. For room layout, they have applied a stochastic optimization approach by sliding walls, swapping rooms, and created a 2D layout. When generalizing the model from 2D to 3D, passageways are created using some rules e.g. wide passageways are placed between public rooms. Nevertheless, as mentioned earlier, they do not consider any logistics cost optimization in their approach. Unlike this, Wu et al. [42] proposed a hierarchical framework for the generation of building interiors based on a mixed-integer quadratic programming (MIQP) formulation focused on the geometric layout of rooms in a 2D plan. Their considered objective function is in a quadratic form, which means that it is non-linear. In this work, each room is represented as an axis-aligned polygon defined by points on the bounding area as a polygon that consists of a set of small rectangles. During the room layout step, the polygonal regions are generated by considering adjacency constraints. Then, aisles are generated by expanding the gap between two regions with a fixed width if needed. However, none of them uses a mesh model with MIP approach for corridor generation. Their corridor design step is mostly based on intuition rather than a computational logic.

In general, there is a subtle point on why satisfying adjacency requirements is not good enough for a serious case like hospital layout. The reason is twofold:

Unless there are literal reasons pertaining to 'piping, noise, vibration and alike', producing adjacency requirements can be somewhat subjective, arbitrary and confusing tasks if meant as a replacement of closeness requirements, while closeness criteria can make an objective sense. In a general case other than those mentioned above, from a logistics point of view, it is not easy or even necessary to state why two rooms must be adjacent, however, they may easily need to be close by.

That could easily be achieved by placing them on the opposite sides of a corridor; whereas by requiring that the two be adjacent, we may not only make the problem unnecessarily complicated but also make it unrealistic due to the infeasibility of making planar layouts with many required adjacencies. Secondly, the emphasis on adjacency requirements may simply be procrastination from the corridor generation problem in small floor plans while in large and complex floor plans, there does always exist a part specifically about corridor generation. Access space in large and complex buildings is often so large that it may even take a significant part of the space.

2.2. Contribution

The main contribution of this paper is to combine and transform two major space-planning problems, i.e. room layout (zoning/packing) and corridor generation (routing) to the standard forms of MILP problems so that they can be solved using standard engineering optimization engines. In other words, the main contribution of the paper is the formulation of the hospital layout problem to multiple standard MILP problems, so that the standard solvers such as Gurobi, CPlex, and OR-Tools can be utilized to solve them efficiently. As such, the study of the time-complexity of these well-known algorithms falls out of the scope of the paper. Nevertheless, we can relate the time-complexity of the brute-force search problems in each major sub-problem. The two major sub-problems formulated as combinatorial optimization problems in the paper are the **zoning** and **routing** problems, discretized on a Mesh $\mathcal{M}(V, E, F)$:

- For the **zoning** problem (face to room assignment problem):
  - For n rooms, we shall have n^2integer variables to find, for each of which we can have $|F|$ possible choices of integers, where $|F|$ is the size (number of pixel-like faces in each direction) of a hypothetical square grid of pixels whose integer coordinates are tested for the room sizes x_i, y_i, and $i \in [0, n)$. This means that for a brute-force test, there will be $\mathcal{O}(n^{2|F|})$ choices to be made. This means that the problem quickly becomes intractable with brute force or naïve searches.
- For the **routing** problem (edge to corridor assignment problem):
  - The number of edges to be designated as corridors is unknown in advance, because one does not have a target total length for the corridors at the beginning of the search process. However, the problem in this case is formulated in such a way that a vector of decision variables with the size of the number of edges in the underlying mesh will be the main decision variable of a canonical MILP formulation, i.e. in the form o f $\min_x \boldsymbol{c}^T \boldsymbol{x}$. Therefore, the complexity of the problem is on a par with a canonical MILP problem. Similarly, if one was to make a naïve brute-force search for fining the best configuration of edges for the corridors, considering the binary variables indicating the assignment of an edge to a corridor, it would take $\mathcal{O}(2^{|E|})$. This also indicates that evern with binary variables this integer programming problem becomes intractable whence the mesh is of a considerable size.

The binary ILP (the case of routing) is listed in the Karp's list of 21 NP-Complete problems [43]. The MILP is also known to be an NP-Hard problem. However, the solvers, in practice, manage to approximate solutions in polynomial time.

3. Stacking

In this stage, we aim to find the number of levels (or parts) needed for a to-be-built hospital. Before this stage, the shape of the outer envelope should be determined, whether as systematically as exemplified in this paper or as conventionally proposed based on the so-called common 'typologies' formed according to construction-related constraints and preferences. The decision on how to dissect the entire configuration of a hospital into parts (vertically, horizontally, or a combination of the two) is taken 'intuitively', i.e. conventionally by considering area requirements, site-specific circumstances, construction-related considerations, and alike. As a systematic alternative, further, in the text, we shall introduce a mathematical method for taking stacking decisions with respect to 'accessibility' requirements that are typically given in the so-called Activity Relationship Charts (3.1)0.3.2 The latter method is based on the idea of forming clusters as to the closeness requirements between the departments (3.2).

3.1. Activity relationship charts

In the stage of stacking, the floor levels of each main unit of hospitals are defined. In this paper, the main typical units for a middle-size general hospital are considered and listed with some accompanying subunits as follows:

1. **Outpatient Department (OPD):** consists of polyclinics or specialty units and waiting areas.
2. **Inpatient Department (IPD):** consists of wards, nursing units, surgical and medical services, the delivery suite that consists of delivery rooms and nurseries.
3. **Intensive Care Units (ICU):** are specialty nursing units designed, equipped, and staffed with specially skilled personnel for treating very critical patients or those requiring specialized care and equipment [46].
4. **Operating Theatre (OPT):** consists of operation rooms, post-anesthesia recovery rooms, scrub-up, changing rooms, etcetera. This unit requires different entrances for materials, patients, and staff due to sterilization procedures.
5. **Emergency (EMG):** is one of the polyclinics. It welcomes ambulance or private cars with a distinct entrance. Triage and treatment areas are needed.
6. **Diagnostic units (DGU):** labs and medical imaging units, typically consisting of:

   - Radiology
   - X-ray
   - Ultrasound
   - Laboratory
   - Pathology

7. **Administration (ADM):**

   - Business, accounting, auditing, cashiers, records
   - Offices for hospital management etc.

8. **Main Entrance Hall (ENT):** consists of a welcome desk, waiting areas, shopping, café, and alike.
9. **Support (SUP):**

   - Housekeeping and linen rooms
   - Storages
   - Kitchen and dining
   - Central Sterile Supply Department (CSSD)
   - Mortuary: where dead bodies are kept, requiring a separate entrance.
   - Pharmacy: supplies medicine for emergency and inpatient units.
   - Parking
   - Bunker

Each main unit is spatially related (meant to be close-by) to each other due to many technical and non-technical relationships (see Tables 1–2). Such closeness relations are typically expressed in terms of so-called closeness ratings in an Activity Relationship Charts (ARC in short, a.k.a. REL-Charts), see e.g. the Tables 1 & 2 for a hypothetical

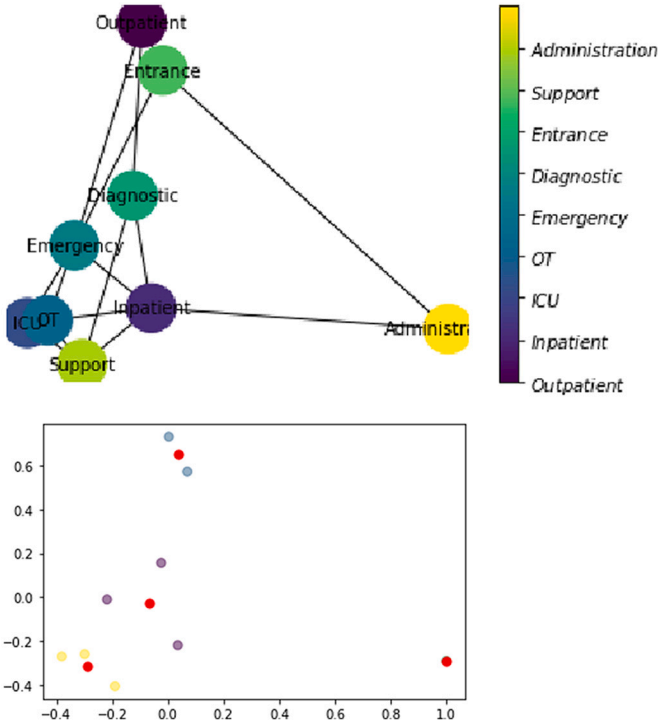

Fig. 6. A spectral embedding of the main units based on the REL-chart (weighted adjacency matrix, the same graph as shown in Table 2) and k-means clustering.

hospital configuration. If we consider the main units as a graph, then the ARC can effectively be considered as a weighted graph because of these relationships.

The typical reason for stacking in the horizontal floor is that horizontal access is often more efficient and thus preferred for critical or high-frequency connections in hospitals. Thus, we argue that in a systematic layout process, pairs of highly interrelated spaces within the same 'cluster' should be placed on the same level.

3.2. Stacking as clustering

To find the clusters according to proximity requirements, we propose to perform a spectral clustering on the weighted configuration graph of the hospital. The idea of spectral clustering [47] is to find clusters on a dimensionality-reduced embedding of the graph in a so-called spectral-domain (see Fig. 6), i.e. the space of the most dominant eigenvectors of the weighted adjacency matrix that, roughly speaking, represent the steady-state of random walks on the graph in question. The idea behind this process and its connection with closeness are explained in [48]. For clustering the main units within the spectral space (a.k.a. the frequency domain), we distinguish two different cases:

- If we know the required number of levels, then we perform spectral clustering based on the K-Means algorithm.
- If we are to find the number of required levels, then we perform spectral clustering based on the DB-Scan algorithm (not used here).

.)

By clustering the entire space into horizontal clusters, we implicitly decide to provide (fast) vertical connections between spaces on different levels instead of locating them on one single huge horizontal area (which would lead to long walking distances and it is not applicable when the site area is not enough). If some spaces need the same level requirements but they are in different clusters, then we can define them on the same level if the available space is sufficient. They can be also designed as two different building blocks if located at the ground level. For example, two blocks on the ground level (main block: ground level of the main building and the attached block: added to the main block) can be defined. There should be a short connection between those two blocks and the attached block should have access to the vertical circulation of the main block. In addition, during the stacking stage, we should consider some special conditions. Some of the specialized spaces need certain levels due to some specific features. For example, most hospital design guidelines suggest that an emergency needs a distinct and separate entrance and that it should be placed on the ground level for quick access. Therefore, all spaces in the cluster that includes the emergency unit should be placed on the ground level.

As seen in Fig. 6, upper picture is a spectral drawing of the weighted adjacency matrix given in Table 2 and lower picture is the output of k-means clustering with 4 by using the result of the spectral drawing. As result, outpatient and entrance are in the first cluster; diagnostic units, emergency and inpatient are the second cluster, intensive care unit and operation theatre are in the third cluster and administration is in the fourth cluster. The units that are in the same cluster can be located at the same level or these clusters can give an idea for a stacking task.

4. Zoning

In this stage, the zoning problem for each level is formulated based on MIP models introduced in [26,40]. At this step, the building plot (e.g. input problem domain) is discretized into a quad mesh to define the structural system [axes] of the building and to decide the positions of the rooms on the mesh faces. Therefore, the problem at hand can also be deemed as a room-to-face assignment problem, where multiple faces are assigned to a labelled room/department. As such, this can be considered a problem of graph colouring or vertex labelling. Each level has its own room-to-face problem to solve for the levels determined at the stage of stacking. Each level can have different goals and requirements so that general problem objectives and constraints can be specialized for each level. At this stage, we focus on the allocation of space to the list of sub-units within each cluster/floor. For example, during the stage of stacking, we would consider the cluster of all diagnostic units as the main unit, but here we consider sub-units of the diagnostic department as rooms to be laid out, such as radiology, x-ray, laboratories, etcetera. In this sub-problem, enough mesh faces are to be allocated to the rooms of the floor plan such that their area requirements are fulfilled. We consider the following aspects in our zoning model:

- Closeness objective for interrelated areas
- Space area requirements
- Cohesion (as in connectedness and contiguity)
- Contiguous areas (adjacent rooms)
- [optional] fixed locations/adjacencies for some spaces
- Non-overlapping spaces (To avoid different rooms assigned to same mesh faces)
- Adjacency to NEWS[1] borders because of lighting requirements, discrete entrance requirements.
- Flexibility requirements for the spaces that can need future expansion
- Privacy/Community requirements
- Boundary constraints

For various reasons we perform the zoning operation prior to corridor generation, namely that: We need to know in the routing problem the shape of the rooms so as to avoid cross-cutting the rooms with corridors and more importantly, to ensure that all the rooms are connected to the set of edges designated as corridors. Additionally, the so-called sink vertices (the main entrance) are defined based on the

[1] N: North, E: East, W: West, S: South

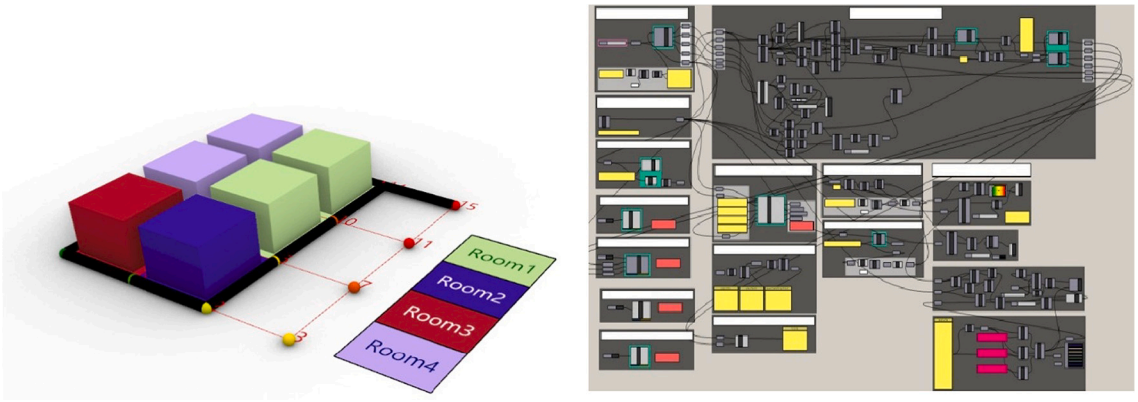

Fig. 7. Interface of the layout model developed in Rhino (illustrates CAD at the left) & Grasshopper3D (plug-in of Rhino & illustrates the computational model including python script components at the right)

known location of the zones. Moreover, in 6.2 (Route Separation). In the zoning problem formulation, however, we have already considered the closeness ratings by minimizing the Manhattan distance between the zones. This is because the Manhattan distance between pairs of rooms is the lower limit of their eventual geodesic distance through the corridors.

Algorithm 1. Zoning and Room Generation, based on assigned floors.

Input Notation	*[Data-Structure] Data Type*	*Input Name: Notes*
$M(V,E,F)$	*Quad Mesh*	*Space: A map consisting of vertices, edges, and faces in the form of pixels/quads to be assigned to rooms*
$[s_i]_{n\times1}$	*Array of Float*	*Surface Areas: To be converted to a list of integers and to be realized as constraints, where n is the number of rooms*
$[T_{i,\,j}]_{n\times n}$	*Matrix of Float*	*Closeness Matrix: indicating whether closeness (in the network space) is to be sought as an objective with respect to every two nodes*
$[A_{i,\,j}]_{n\times n}$	*Matrix of Boolean*	*Adjacency Matrix: indicating whether adjacency (as in sharing walls) is to be ensured as a constraint with respect to every two nodes*
C	*List/Set of (Integer, Integer, Integer)*	*Locations: integer tuple coordinates (i,x,y) of the index of room and the coordinates of the top left corners of spaces/faces which are to be constrained to a specific location*
B	*List/Set of (Integer, Integer)*	*Adjacency to Borders: integer tuples in the form of (i, which_border) indicating whether space is to be constrained to be adjacent to a border as to light or accessibility reasons.*
Output Notation	*[Data-Structure] Data Type*	*Output Name: Notes*
$[x_i,y_i]_{n\times2}$	*Array of Integer*	*Top-Left Coordinates:*
$[w_i,l_i]_{n\times2}$	*Array of Integer*	*Width-Length from Top-Left Corners*
$[R_{i,\,f}]_{n\times\lvert F\rvert}$	*Matrix of Boolean*	*Face to Room Assignment Matrix: indicating Face Regions in its rows*

Problem: given the input space as a map of vertices, edges, and faces, find an assignment of faces to rooms, such that the sum of Manhattan distances between close pairs is minimized, subject to several validity constraints, namely, cohesion and non-overlapping zones. The problem-solver method can take the problem in the [minimization] standard MILP form[a]*:*

$$\min_{x,y} \sum_{(i,j)\in E} d_{i,j} = \sum_{(i,j)\in E} |x_i - x_j| + |y_i - y_j| \forall (i,j) \in E$$

, where E is the edge list of the ARC/Adjacency graph, *subject to*:

$h_k(x,y,w,l) = 0, k \in \{0,1,2\}$ % *equality constraints for adjacent rooms, fixed locations, adjacency to NEWS borders*

$g_n(x,y,w,l) \le 0, \ \in \{0,1,2\}$ % *ensuring cohesion, non − overlapping spaces, boundary*

$0 < x[i] < f_1$

$0 < y[i] < f_2$

$w_{min}[i] < w[i] < w_{max}[i]$

$l_{min}[i] < l[i] < l_{max}[i]$

$W = f_1$

$L = f_2$

Where f_1 is number of faces in M along {x} direction and f_2 is in {y} direction.

Procedure Zoning & Room Generation:

1. For the m_{th} floor mesh M

(*continued on next column*)

(*continued*)

2. Define Mesh with several faces along {x} direction: f_1;and of faces along {y} direction: f_2
3. Enter w_{min}, w_{max}, l_{min}, l_{max} of each space
4. Enter $T_{i,j}$ matrix between each space
5. Enter the number of spaces (rooms): num_rooms
6. Define decision variables: $[x_i,y_i]_{n\times2}$, $[w_i,l_i]_{n\times2}$
7. Generate constraints (details are in sub-sections: h_k, $k = 0, 1, 2$ & g_n, $n = 0, 1, 2$)
8. Generate objective function (details are in sub-section: closeness objective $d_{i,\,j}$)
9. Run the model
10. Get x, y, w, l *and* $[R_{i,\,f}]_{n\times|F|}$ values and visualize the allocated rooms with nominal colors on the 3D model of the m_{th} floor
11. $m \leftarrow m - 1$ and go back to Step 2 and repeat until all floors are finished

[a]Please note that this formulation is showing our implementation in Google OR-Tools. As we will show below, this formulation is not exactly in the linear form that would be acceptable by a conventional LP solver. However, as explained in the implementation details, due to the possibility of inserting auxillary conditional statements, we can utilize the library for dealing with this non-linear objective function as if it is linear. Code 1 reveals this point. In the following we can show the closed form of the objective function solved in this way:

Consider vector at the length of the number of edges in the adjacency graph $G = (V,E)$, with two columns containing the coordinate differences of the edge-lines, we can define:

$$\mathbf{E} := \begin{bmatrix} \vdots & \vdots \\ (x_i - x_j) & (y_i - y_j) \\ \vdots & \vdots \end{bmatrix}_{|E|\times2}$$

Now, the L_1 norm of this vector will be:

$$\|\mathbf{E}\|_1 = \mathbf{1}^T|\mathbf{E}| = \sum_{\boldsymbol{e}[k,:]=V[j,:]-V[i,:]|(i,j)\in E} |\mathbf{E}[:,0]| + |\mathbf{E}[:,1]|$$

where $\boldsymbol{V} := \begin{bmatrix} \vdots & \vdots \\ x_i & y_i \\ \vdots & \vdots \end{bmatrix}_{|V|\times2}$ is a matrix containing the vertex coordinates of a graph embedding.

However, **e** in this equation is not written in terms of the decision variables yet. To write it in terms of the decision variables, we need to use a matrix called the Incidence Matrix of the graph in question defined as below:

$$\mathbf{M}_{|E|\times|V|} = [M_{e,v}]_{|E|\times|V|} \,|\, M_{e,v} = \begin{cases} M_{e,v} = -1, \textit{if } v = s \textit{ in } e = (s,t) \\ M_{e,v} = +1, \textit{if } v = t \textit{ in } e = (s,t) \\ M_{e,v} = \mathbf{0}, \textit{otherwise} \end{cases}$$

Then we can write:

$\mathbf{E} = \mathbf{MV}$

So, the closed form of the objective function can be written as follows:

$$\min_{V} \mathbf{1}^T|\mathbf{MV}|$$

where the decision variable is actually a matrix rather than a vector.

Which is not exactly in the canonical form of an LP problem, as a classical LP problem would be in the form of:

$$\min_{x} \boldsymbol{c}^T\boldsymbol{x}$$

So, in conclusion, we get to solve a problem that is not exactly linear with the help of the conditional statements and the possibility of incremental development of a problem instance in OR-Tools.

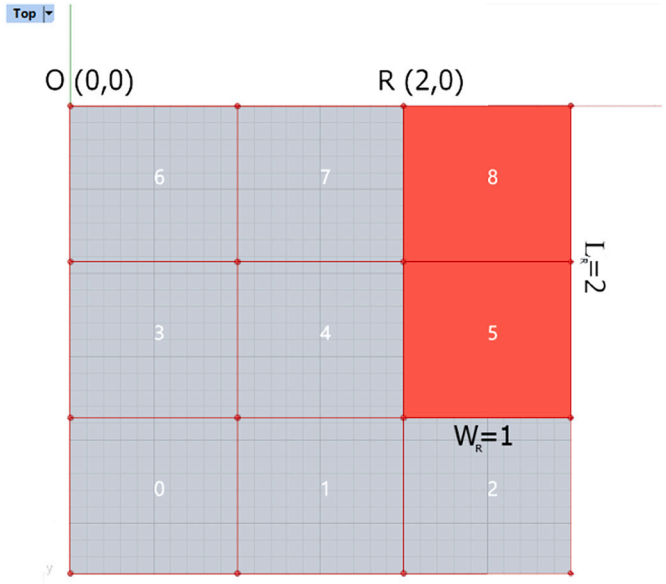

Fig. 8. Integer-coordinate of the layout model (Room R is placed on x = 2, y = 0 with length = 2 and width = 1 so that room R is assigned to mesh faces indexed by 5 and 8).

4.1. Design variables

For each room indexed with *i*, four decision variables are defined as integer coordinates of each space and their sizes on the mesh. The origin point is the top left corner vertex of the mesh (See Fig. 8).

$x_i = X - integer\ coordinate\ of\ the\ top\ left\ corner\ of\ the\ space\ i\ on\ mesh.$

$y_i = Y - integer\ coordinate\ of\ the\ top\ left\ corner\ of\ the\ space\ i\ on\ mesh$

$w_i = the\ integer\ width\ of\ the\ space\ i\ on\ mesh$

$l_i = the\ integer\ length\ of\ the\ space\ i\ on\ mesh$

There are also two basic parameters as width and length of the building plot, which are represented by *W and L*, respectively. As shown in the figure below, a rectangular red-coloured room R is placed at the (2,0) integer coordinates on mesh with a width = 1 and length = 2. In the end, this room is placed on the mesh faces indexed by 5 and 8. Even though it seems that we are dealing with the geometric coordinates of the spaces, it is a topological layout problem, since each coordinate pair is referenced by a vertex index in the mesh and width & length refer to the length of edges in the mesh.

4.2. Closeness objective

For this purpose, a binary closeness matrix is given between room *i* and *j*, denoted by $T_{i,j}$. This matrix consists of zeros and ones considering the closeness relationship between sub-units of each level. This objective is formulated as:

$$if\ T_{i,j} = 1; \min_{x,y} d_{i,j} = |x_i - x_j| + |y_i - y_j| \forall i,j \in [0,n)$$

where *num_rooms* is the number of rooms on each level. This is one of the other differences of our approach with the recent works in the literature using MILP such as [10]: they only consider the distance in corridor generation and we use it also in-room space allocation. Thus this is an additional objective in our zoning stage because rooms are defined with integer coordinates.

Code 1. Closeness objective

```
#closeness objective $min_{x,y} Σ_{(i,j)∈E} d_{i,j} = Σ_{(i,j)∈E} |x_i − x_j| + |y_i − y_j|$
def ClosenessObj(x,y,num_rooms, T):
        k=0
        d=0
        for i in range(num_rooms):
            for j in range(i+1,num_rooms):
                if T[i][j]==1:
                    if x[i]-x[j]>0:
                        a_1=x[i]-x[j]
                    else:
                        a_1=x[j]-x[i]
                    if y[i]-y[j]>0:
                        a_2=y[i]-y[j]
                    else:
                        a_2=y[j]-y[i]
                    d=d+a_1+a_2
        model.Minimize(d)
```

4.3. Space area requirements

There are two basic parameters as width and length of the building plot, which are represented by *W and L*, respectively. We control the width and the length of each room using the lower and upper limits, w_{min}, w_{max}, l_{min}, l_{max} where w_{min} *and* w_{max} are the minimal and the maximal width of space *i* while l_{min} *and* l_{max} are the minimal and maximal length of the i_{th} space, respectively. After having obtained the minimum area requirements of each room (verifiable by means of a Discrete Event Simulation process such as the one introduced in [49], we define the minimum area requirements to the integer-programming model by defining w_{min}, w_{max}, l_{min}, l_{max} values of each spaces. Since we are dealing with a discrete mesh model, these values correspond to the square area requirements. For example, if space needs 120 m^2 area and one mesh face is 20 m^2; then this space requires a minimum (120/20) six mesh-faces on the grid domain and bounding box constraints of *w* and *l* can be defined accordingly.

$$w_{min_i} < w_i < w_{max_i}\ for \forall room\ i = 1, .., \text{num\_rooms}$$

$$l_{min_i} < l_i < l_{max_i}\ for \forall room\ i = 1, .., \text{num\_rooms}$$

Since there is more than one space, $\boldsymbol{w}_{min}$, $\boldsymbol{w}_{max}$, $\boldsymbol{l}_{min}$, $\boldsymbol{l}_{max}$ values can be defined in an array structure.

Code 2. Space area requirements

```
#blackbox constraints, which also defines space area req.s
wmin=[5,3,3,2,2,2,2,2,2,3,3,3,2,3,1,1,1,1,1,1,3]
wmax=[5,3,3,3,3,3,3,3,3,3,3,3,6,2,3,1,1,1,1,1,1,3]
lmin=[1,1,1,1,1,1,4,3,6,1,3,8,2,1,1,1,1,1,1,1,2]
lmax=[2,2,2,2,2,2,4,4,6,2,3,8,2,2,1,1,1,1,1,1,2]
```

4.4. Cohesion between rooms

Ensuring cohesion between rooms is defined as a constraint to attach the rooms together [50]. As this constraint can be too strict for the problems with too many rooms, we have added a tolerance value *t* to this constraint to ensure room placements as connected as possible. For this purpose, we introduce two binary decision variables (a_{ij}, b_{ij}) encoding four positioning constraints together corresponding to left (00_b), right (10_b), above (01_b), and below (11_b), which are defined as follows:

$$if\ a_{ij} = 0 \& b_{ij} = 0; x_i + w_i + t \geq x_j - W(a_{ij} + b_{ij})\ left\ positioning$$

$$if\ a_{ij} = 1 \& b_{ij} = 0; x_j + w_j + t \geq x_i - W(1 - a_{ij} + b_{ij})\ right\ positioning$$

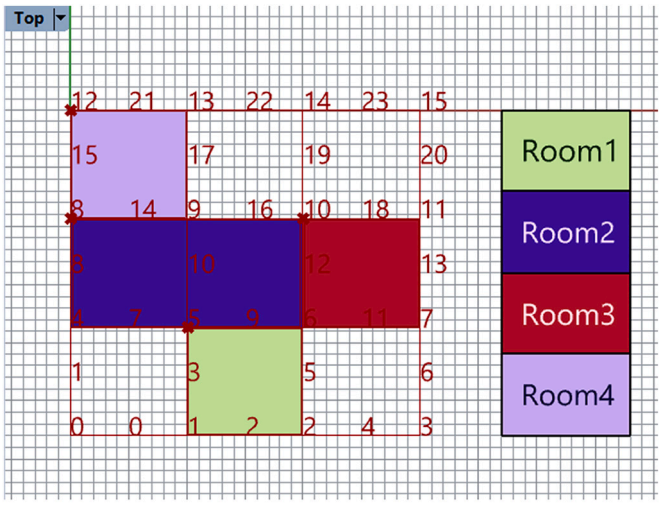

Fig. 9. Example layout representation as an output of our computational design tool (four rooms are placed to the faces of a boundary mesh with respect to their area requirements)

$$if\ a_{ij}=0 \& b_{ij}=1; y_j+l_j+t \geq y_i - L\left(1+a_{ij}-b_{ij}\right)\ above\ positioning$$

$$if\ a_{ij}=1 \& b_{ij}=1; y_i+l_i+t \geq y_j - L\left(2-a_{ij}-b_{ij}\right)\ below\ positioning$$

Note that these positioning constraints do not necessarily enforce adjacency but only relative positioning constraints. In other words, when some space is to be placed to the right of another one, it does not necessarily have to be immediately to the right of the other space but possibly with other spaces in between. Python script of the cohesion constraint is given below by taking the tolerance value equal to five (faces). The tolerance value controls how wide can a gap between two spaces be with respect to the pixel size.

Code 3. Ensuring cohesion constraint

4.5. Adjacent rooms

Especially in hospitals, some of the partition walls need to be attached. For example, the intensive care unit must be adjoined to the operating theatre recovery room due to the use of common equipment. Therefore, the adjacent room constraint is defined. It is used for two specific areas, which must be specifically adjoined. The following python script is written for two adjacent spaces indexed with $i = 1$ and $j = 2$. As can be seen in the example in Fig. 9., Rooms 2 & 3, which have indices 1 and 2 respectively, must be adjacent to each other.

Code 4. Contiguous areas constraint

4.6. Fixed locations

Some of the spaces require fixed or specific locations while also having a relationship with the other spaces. For example, vertical circulation areas or building core are considered at the fixed positions on the grid domain. Their certain coordinates are defined as constrained indices in the model. Python script example for the spaces with indices 14–20 are given below e.g. 14th space should be on (x = 5, y = 6) coordinates. Mathematically this means that some decision variables are predetermined, and so the problem gets somewhat smaller by changing them from variables to constants.

Code 5. Fixed location constraint

```
# $h_1(x,y,w,l) = 0$ fixed positions
def fixed(x,y,C,num_rooms):
#the fixed indices are read from the list C
      for k in range(num_rooms):
        if k in C:
            model.Add(x[k]=C[k][0])
            model.Add(y[k]=C[k][1])
##### examples:
        #defining fixed locations of elevators
        model.Add(x[14]==5)
```

4.7. Non-overlapping spaces

Non-overlapping constraint prevents two rooms from occupying the same space [50]. We use the same pair of binary decision variables (a_{ij}, b_{ij}) to define this constraint. These requirements can be illustrated as follows:

$$if\ a_{ij}=0 \& b_{ij}=0; x_i+w_i \leq x_j + W\left(a_{ij}+b_{ij}\right)\ left\ positioning$$

$$if\ a_{ij}=1 \& b_{ij}=0; x_j+w_j \leq x_i + W\left(1-a_{ij}+b_{ij}\right)\ right\ positioning$$

$$if\ a_{ij}=0 \& b_{ij}=1; y_j+l_j \leq y_i + L\left(1+a_{ij}-b_{ij}\right)\ above\ positioning$$

$$if\ a_{ij}=1 \& b_{ij}=1; y_i+l_i \leq y_j + L\left(2-a_{ij}-b_{ij}\right)\ below\ positioning$$

The example python script is given below:

Code 6. Non-overlapping spaces constraint

```
# $g_0(x,y,w,l) \leq 0$ Cohesion constraint with tolerance=5
def cohesion(x,y,w,l,num_rooms, W, L):
        tolerance=5
        for i in range(num_rooms):
            for j in range(i+1,num_rooms):
                if a[i,j]==0: # a and b are temporary parameters defined here
                    if b[i,j]==0:
                        model.Add(x[i]+w[i]+tolerance>=x[j]-W*(a[i,j]+b[i,j]))
                    if b[i,j]==1:
                        model.Add(y[j]+l[j]+tolerance>=y[i]-L*(1+a[i,j]-b[i,j]))
                if a[i,j]==1:
                    if b[i,j]==0:
                        model.Add(x[j]+w[j]+tolerance>=x[i]-W*(1-a[i,j]+b[i,j]))
                    if b[i,j]==1:
                        model.Add(y[i]+l[i]+tolerance>=y[j]-L*(2-a[i,j]-b[i,j]))
```

```
# $h_0(x,y,w,l) = 0$ contiguous areas, repeat it for each couples
def AdjacentRooms(x,y,w,l,A,num_rooms):
      for i in range(num_rooms):
        for j in range(i+1,num_rooms):
            if A[i,j]==1: #if i &j must be adjacent:
                    b_1 = model.NewBoolVar('') #horizontal or vertical
                    b_2 = model.NewBoolVar('') #left/top or right/bottom
                    b_3 = model.NewBoolVar('') #full or partial adjacency
                    if b_1: #horizontal adjoin
                        if b_3:
                            model.Add(y[i]==y[j])
                        else:
                            model.Add(y[i]==y[j]+(l[j]-l[i]))
                        model.Add(x[i]==x[j]+w[j]).OnlyEnforceIf(b_2) #to the
            left
                        model.Add(x[i]==x[j]-w[i]).OnlyEnforceIf(b_2.Not())
            #to the right
                    else: #vertical adjoin
                        if b_3:
                            model.Add(x[i]==x[j])
                        else:
                            model.Add(x[i]==x[j]+(w[j]-w[i]))
                        model.Add(y[i]==y[j]-l[i]).OnlyEnforceIf(b_2) #to the
            top
                        model.Add(y[i]==y[j]+l[j]).OnlyEnforceIf(b_2.Not())
            #to the bottom
```

```
# $g_1(x,y,w,l) \leq 0$ non-overlapping constraint
def overlap(x,y,w,l,W,L,num_rooms):
     for i in range(num_rooms):
         for j in range(i+1,num_rooms):
             if a[i,j]==0:  # a and b are temporary parameters
                 if b[i,j]==0: # room i to the left of room j
                     model.Add(x[i]+w[i]<=x[j]+W*(a[i,j]+b[i,j]))
                 if b[i,j]==1: # room i above room j
                     model.Add(y[j]+l[j]<=y[i]+L*(1+a[i,j]-b[i,j]))
             if a[i,j]==1:
                 if b[i,j]==0: # room i to the right of room j
                     model.Add(x[j]+w[j]<=x[i]+W*(1-a[i,j]+b[i,j]))
                 if b[i,j]==1: # room i below room j
                     model.Add(y[i]+l[i]<=y[j]+L*(2-a[i,j]-b[i,j]))
```

```
# $h_2(x,y,w,l) = 0$ adjacency to NEWS border 0: 'North', 1: 'South', 2:
'East', 3: 'West', the following is enforcing adjacency of the, space 1
to south, space 2 to east
Def NEWS(x,y,w,l,B, num_rooms):
      for k in range(num_rooms):
          if k in B:
            if B[k]=0:
              model.Add(y[k]==0) # space k to the North façade
            if B[k]=1:
               model.Add(y[k]+l[k]==L) #space k to the South façade
            if B[k]=2:
               model.Add(x[k]+w[k]==W) #space k to the East façade
            if B[k]=3:
               model.Add(x[k]==0) #space k to the West façade
```

```
Considering n-edges that generate a path between room 1 and room 2:
For all paths in k-shortest paths between room1 and room2:
        If a path in k-shortest paths does not include inner edges:
                If each path's vertices are different than other path's vertices, except the first vertex
        and the last vertex (because they are origin and destination vertices):
        These all paths' edges are set as active (1) in p .
```

Fig. 10. k-shortest path idea pseudo code.

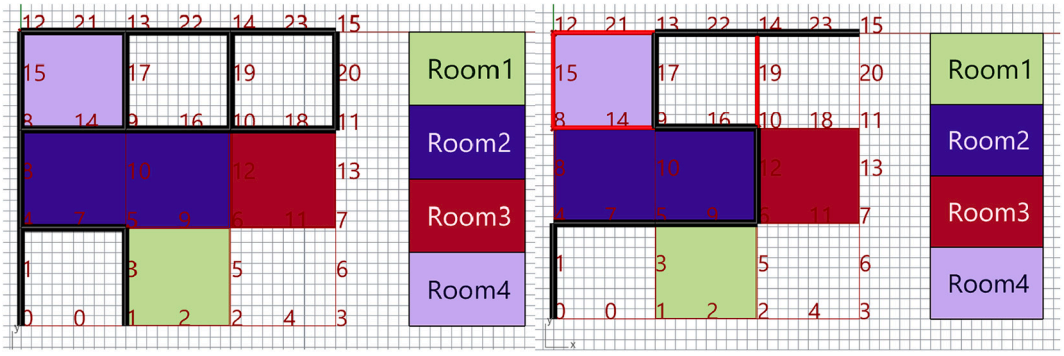

Fig. 11. Left: Example corridor representation as an output of our computational design tool; Right: Black lines refer to main circulation path, red lines refer to collector roads.

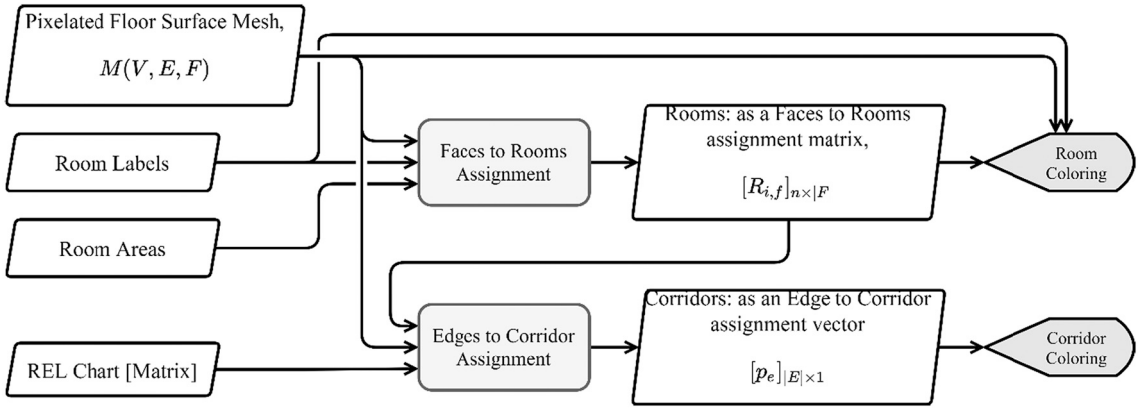

Fig. 12. General picture of whole GH model including both zoning and corridor generation steps.

4.8. Adjacency to NEWS borders

Due to natural lighting requirements, discrete entrance requirements, or the necessity of ensuring distinct entrances, we introduce fixed-border constraints to ensure adjacency to borders (boundaries) for some rooms. This constraint is divided into four types: north, south, east, and west. For example, a room is positioned to the "absolutely north" if it is to touch the top border. As an example to a hospital, a mortuary needs a discrete entrance, and so it should be placed on one of the borders of the grid domain. Similarly, patient wards need natural lighting and so they should be positioned adjacent to the south façade because of hospital design standards and due to positive effects of daylight on the patient recovery process.

Code 7. Adjacency to NEWS border constraint

4.9. Flexibility requirements

In hospitals, planning for future expansion is important. For example, space requirements for laboratories tend to double every 10 years according to the hospital design guidelines [46]. This type of space should be flexible.

Our design strategy for flexible spaces:

- if vertical expansion: place it on top
- if horizontal expansion (geometric and modular): place it on the perimeter and later add up some extra building modules when expansion needed
- place it close to temporary spaces where their functions can be later changed e.g. exhibition hall

The first point can be ensured during the stacking stage by placing the space that needs future expansion on the top level. The second point

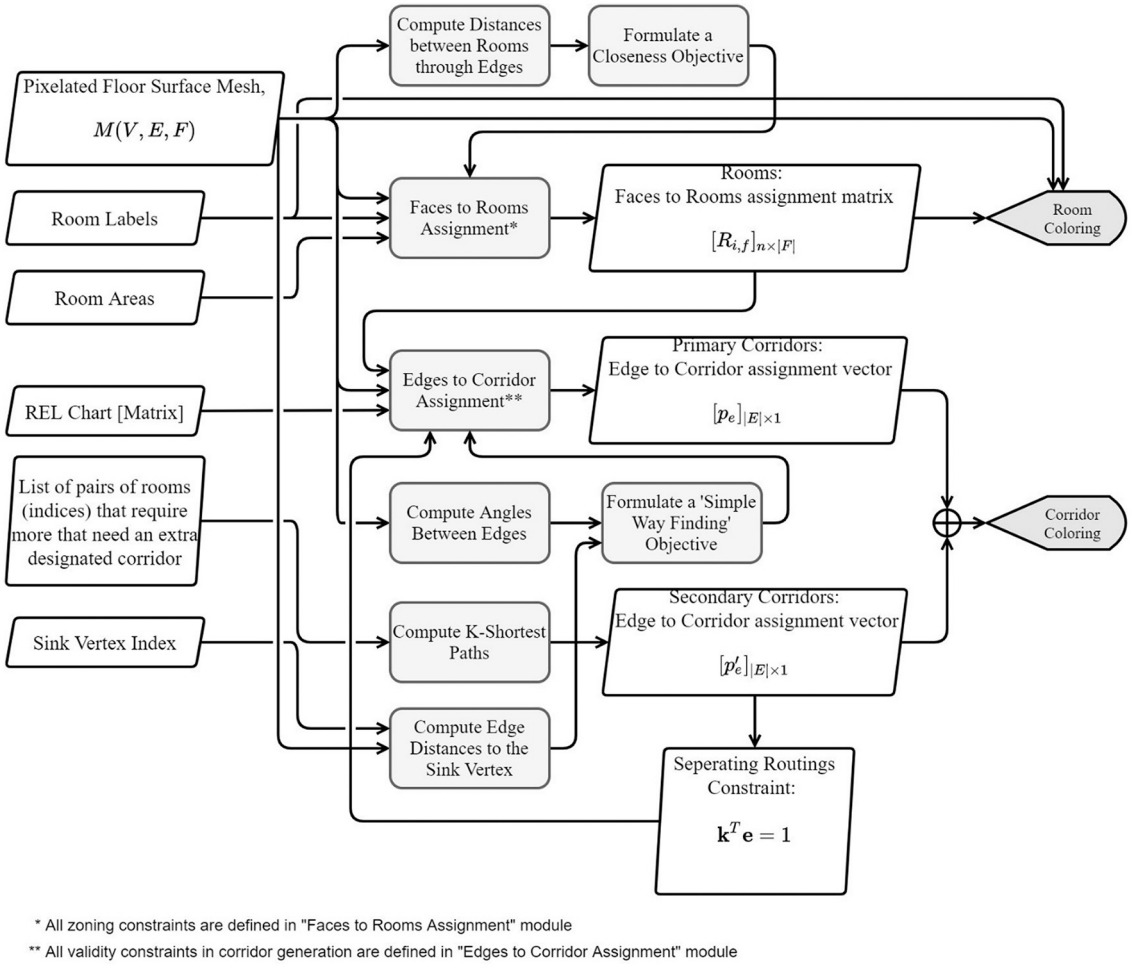

Fig. 13. Workflow of partial components of our computational design tool for new hospital layout designs.

can be ensured with adjacency to NEWS borders constraint (h_2) by putting the space that can need a future expansion at one of the borders. For the third point, we define an exhibition hall area in the hospital and place it adjoined to the space that needs flexibility using the adjoining areas (adjacent rooms h_0) constraint. If space needs an expansion then we can convert the exhibition hall's function to this space's function.

4.10. Privacy/community requirements

Our design strategy for private or communal spaces:

- communal spaces will be at the lower floors (close to the main entrance)
- private spaces should be carried away from the core or the main entrance

This requirement ensures with closeness objective (d_{ij}) as core and main entrance are considered as one of the functional units that are occupying areas on the grid domain.

4.11. Boundary constraint

Boundary constraint forces a room to be inside a boundary:

$x_i + w_i \leq W\ for \forall room\ i = 1, .., num\_rooms$

$y_i + l_i \leq L\ for \forall room\ i = 1, .., num\_rooms$

Code 8. Boundary constraint

```
# $g_2(x, y, w, l) \leq 0$ boundary constraint
def boundary(x, w, y, l, num_rooms,W, L):
        for i in range(num_rooms):
                model.Add(x[i]+w[i]<=W)
                model.Add(y[i]+l[i]<=L)
```

5. Generating corridors

After the rooms are positioned, the main circulation routes within the rooms are created in this stage utilizing Integer Programming. The decision variables of the problem are Boolean variables that indicate the active or inactive status of the mesh edges. The logic of selecting the active edges as decision variables for a network design is inspired by the [10]. Our corridor model is created based on typical hospital layout requirements. The details of the Integer Programming model are given in algorithm-1 and explained in the next sub-sections.

Algorithm 2. Routing and corridor generation based on assigned room

5.1. Way-finding

This objective is very important for minimizing distances and path complexity in the outpatient area of hospitals for first-time visitors and patients who are unfamiliar with the hospital. We define this objective

Input Notation	*[Data-Structure] Data Type*	*Input Name: Notes*
$M(V,E,F)$	*Quad Mesh*	*Space: A map consisting of vertices, edges, and faces, of which edges are to be assigned to corridors*
$[T_{i,j}]_{n\times n}$	*Matrix of Float*	*Closeness Matrix: indicating closeness in network space, if $T_{i,j} > 0$ then a corridor needs to be generated between i & j*
$[R_{i,f}]_{n\times\lvert F\rvert}$	*Matrix of Boolean*	*Face Assignment Matrix: indicating Face Regions in its rows*
σ	*Integer*	*Sink Index: a face index of the sink area, e.g. an elevator/stair core or the main entrance.*
$\mathscr{A}$	*Set of (Integer, Integer, Integer)*	*Alternative Corridors (Optional Input): a tuple of integers in the form of (o,d,k) where k is the number of shortest paths to be found as alternative paths, and o & d respectively denote origin and destination indices. Reasons for separating routes could be hygiene, privacy, and work efficiency for staff.*
Output Notation	*[Data-Structure] Data Type*	*Output Name: Notes*
$\boldsymbol{p} = [p_e]_{\lvert E\rvert\times 1}$	*Vector of Boolean*	*Edge to Corridor Assignment Vector: indicating which edges are to be designated as corridors in the whole floor plan.*
$\Gamma(\widehat{V},\widehat{E})$	*Graph*	*A graph connecting some of the mesh vertices $\widehat{V}\subset V$through some of the edges$\widehat{E}\subset E$marked as being parts of the corridors (True) in the above vector of Edge to Corridor assignment. This graph is used to compute the network distances below.*
$[D_{i,j}]_{n\times n}$	*Matrix of Integer*	*Graph Distance Matrix: containing the actual distances between rooms based on the found corridors*

Problem: given the input space as a map of vertices, edges, and faces, find an assignment of edges to corridors as paths, such that the sum of graph distances between close pairs and the angle between active graph edges is minimized (way-finding objective), subject to several validity constraints: namely ensuring all rooms to be accessible through a corridor, inner-edges (those passing through rooms) to be excluded from the corridor graph; the continuity of the paths to be ensured; and that every room is accessible from the sink through a path. The problem-solver method can take the problem in the [minimization] standard ILP form:

$$\min_{p} \alpha(\boldsymbol{\Theta 1})^T\boldsymbol{p} + (1-\alpha)\boldsymbol{d}^T\boldsymbol{p} \text{subject to}: h(\boldsymbol{p}) = 0\% \text{equality constraint for alternative corridors}$$

$$g_l(\boldsymbol{p}) \le 0, l = 0,1,2,3,4\% \text{validity constriants}$$

$$\boldsymbol{p} \in \{0,1\}^{\lvert E\rvert}$$

Where α is a weighting factor for enforcing the simple paths objective.

Procedure Routing & Corridor Generation:

1. For the m_{th} floor mesh M
2. Define Mesh with several faces along {x} direction: f_1;and of faces along {y} direction: f_2
3. Get outputs of the Zoning model ($[R_{i,f}]_{n\times\lvert F\rvert}$) of the m_{th} floor as an input of the Routing model
4. Enter REL Chart matrix T showing the predicted flow between spaces
5. Enter the number of spaces (rooms): num_rooms; the number of edges:$|E|$; the number of vertices: $|V|$
6. Define decision variables:

$$[p_e]_{\lvert E\rvert\times 1}\ s.t. p_e = \begin{cases} 1, \textit{if edge e is allocated to corridors} \\ 0, \textit{otherwise} \end{cases}, e \in [0,|E|)$$

8. Generate objective function (details are in sub-section: way-finding objective)
7. Generate constraints (details are in sub-sections related to separate routings and validity)
9. Run the model
10. Get *Edges*[e], Γ, $[D_{i,j}]_{n\times n}$ values and visualize the allocated edges as 3D corridors on the m_{th} floor
11. $m \leftarrow m-1$ and go back to Step 2 and repeat until all floors are finished

so as to minimize zig-zags and path lengths (distances from each active edge to the main entrance) that is one of the Wayfinding Cost Terms defined in [51]. The reason for minimizing path length (distance from the sink [main entrance]) is that pedestrians, in general, prefer to walk a short distance [52], [53], and [54]. On the other hand, this provides to decrease the number of nodes in the network. The nodes in our formulation correspond to decision points in the wayfinding literature [52]. Decision points are locations where pedestrians need to make a decision about which direction to go or where pedestrians need to confirm the identity of the current location. Directional or identification signs need

```
import rhinoscriptsyntax as rs
import math
import csv
def Theta(M, csvMeshEETopology):
      ang=[M.TopologyEdges.Count*[0] for e in range(M.TopologyEdges.Count)]

      adjacentedges=csv.reader(csvMeshEETopology) #reading adjacent edges
      from a csv file

      for ij in range(M.TopologyEdges.Count):
          IJPair = M.TopologyEdges.GetTopologyVertices(ij)
          i = IJPair.I #vertex index i
          j = IJPair.J #vertex index j
          EV1=M.TopologyVertices[j]-M.TopologyVertices[i]
          for ab in range(len(adjacentedges[ij])):
              ABPair =
      M.TopologyEdges.GetTopologyVertices(adjacentedges[ij][ab])
              a = ABPair.I #vertex index a
              b = ABPair.J #vertex index b
              EV2=M.TopologyVertices[b]-M.TopologyVertices[a]
              ang[ij][ab]=int(abs(math.degrees(math.atan2(v1.Y, v1.X) -
      math.atan2(v2.Y, v2.X))))
      return ang #a sparse matrix of angles between adjacent edges denoted
      as $Θ$
```

```
def distances(G, vlist, Sink): #Sink vertex is denoted as $σ$
      vlist=readfile('C:\\Users\\TOSHIBA\\Desktop\\files\\vertex.csv')
      vertexindex=vlist
      G.add_edges_from(vertexindex)
      #distance from all vertices to sink vertex
      Sink=0
      Ds=nx.single_source_dijkstra_path_length(G,Sink)
      n = M.TopologyEdges.Count          list
      edge_distance = [None] * n         #populate list, length n with n
      entries "None"
      for ij in range(M.TopologyEdges.Count):
            IJPair = M.TopologyEdges.GetTopologyVertices(ij)
            i = IJPair.I
            j = IJPair.J
            #distance from each edge to sink vertex
            edge_distance[ij] = Ds[j]
      return edge_distance#a vector of edge distances towards the sink
      vertex denoted as $d_{|E|×1}$
```

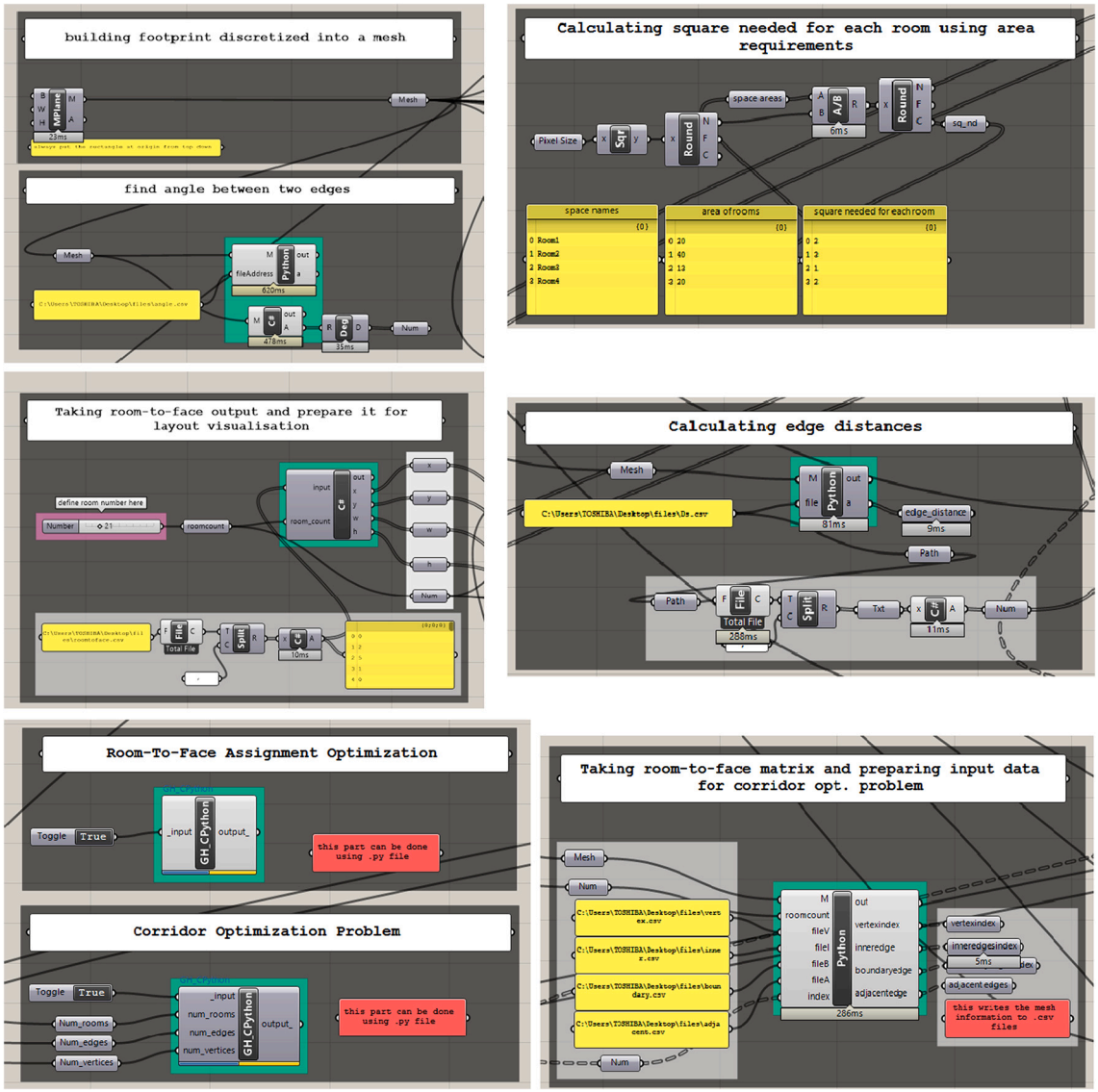

Fig. 14. Partial components of our computational design tool for new hospital layout designs in GH canvas.

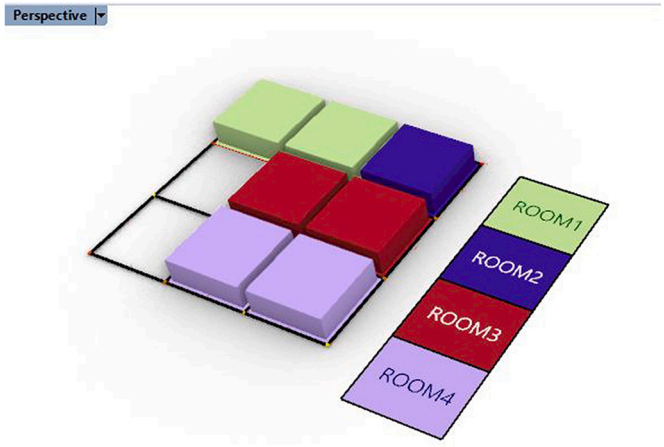

Fig. 15. A small example of optimized result of the zoning problem

to be placed at decision points to guide pedestrians to find their directions [52], [55], or identify their current locations. Paths with lots of decision points should be avoided [56] as making each navigation decision induces stresses to the pedestrians for the fear of making a wrong decision that may lead to a wrong place [56], [57]. Minimizing zigzags is equivalent to decrease network turns. Research in spatial orientation [58] suggests that paths with varying orientation tend to confuse pedestrians in wayfinding, causing disorientation, anxiety, and discomfort [59]. A wayfinding scheme composed of straight paths is more intuitive for navigation [60]. Another reason for minimizing path length (distance from the sink [vertical circulation core]) is to facilitate egress in case of emergencies such as fire.

For formulating this objective, firstly we need to find the angle between all pair of edges in the base mesh as shown in the python script below:

Code 9. Forming a lookup table of angles between all pair of adjacent edges

Then, we need to define the objective that minimizes the sum of the angle between each active edge and its active adjacent edges:

Code 10. Zig-zag objective (wayfinding)

```
#zig-zag objective $min_p(Θ1)^T p$
def SimpleWaysObjective(edges, angle):
      total_sum=0
      for e in all_edges:
          total_sum=total_sum+sum(edges[e]*angle[e])
      #angle is a list of angles of incidence between
      the e_th edge and its adjacent/active edges
      model.Minimize(total_sum)
```

Another objective that contributes the ease of wayfinding is minimizing distance to sink (main-entrance vertex point) from each edge as shown below.

Code 11. Calculating edge distances to the sink vertex

Code 12. Distance to sink objective (wayfinding)

```
#distance to sink objective $min_p d^T p$
def ShortWaysObjective(edges, edge_distance):
      model.Minimize(sum(edge_distance[e] * edges[e] for e in
      all_edges))
```

5.2. Route separation

This requirement is defined as an equality constraint in the corridor design model because of such things as cleanliness, privacy, and working efficiency. Technically, we set to turn on the indexes of all edges to be included in an alternative shortest path in the assignment of edges to corridors. Regarding the cleanliness issue, we aim to separate the routing of the dirty waste and the medical staff such as the critical materials routing between CSSD (central sterile services department) and OT (operation theatre). Regarding the privacy issue, we aim to separate public corridors from private spaces such as the mortuary and elevators area. Regarding the working efficiency, we aim to prevent unwanted

Table 3
Computational result of zoning problem for a small case

	RESULT:
	index 0
X	0
Y	0
W	2
H	1
	index 1
X	2
Y	0
W	1
H	1
	index 2
X	1
Y	1
W	2
H	1
	index 3
X	1
Y	2
W	2
H	1
$\mathbf{A_{LJ}}$	{(0,0): 0, (0,1): 0, (0,2): 1, (0,3): 1, (1,0): 0, (1,1): 0, (1,2): 1, (1,3): 1, (2,0): 0, (2,1): 0, (2,2): 0, (2,3): 1, (3,0): 0, (3,1): 0, (3,2): 0, (3,3): 0}
$\mathbf{B_{LJ}}$	{(0, 0): 0, (0, 1): 0, (0, 2): 1, (0, 3): 1, (1, 0): 0, (1, 1): 0, (1, 2): 1, (1, 3): 1, (2, 0): 0, (2, 1): 0, (2, 2): 0, (2, 3): 1, (3, 0): 0, (3, 1): 0, (3, 2): 0, (3, 3): 0}
	status: OPTIMAL
	conflicts: 0
	branches: 0
	wall time: 0.001015

Table 4
Sensitivity analysis

Global optimal solution found	
Objective value:	10.00000
Objective bound:	10.00000
Infeasibilities:	0.000000
Extended solver steps:	0
Total solver iterations:	4
Elapsed runtime seconds:	0.03
Model Class:	PINLP
Total variables:	13
Nonlinear variables:	5
Integer variables:	13
Total constraints:	21
Nonlinear constraints:	1
Total nonzeros:	48
Nonlinear nonzeros:	5

Variable	Value	Reduced Cost
X0	0.000000	0.000000
X1	2.000000	0.000000
Y0	0.000000	0.000000
Y1	0.000000	0.000000
X2	1.000000	0.000000
Y2	1.000000	0.000000
X3	1.000000	0.000000
Y3	2.000000	0.000000
W0	2.000000	2.000000
H0	1.000000	2.000000
H1	1.000000	1.000000
H2	1.000000	2.000000
W1	1.000000	0.000000
W2	2.000000	0.000000
W3	2.000000	0.000000
H3	1.000000	0.000000

Row	Slack or Surplus Price	Dual
1	10.00000	−1.000000
2	5.000000	0.000000
3	5.000000	0.000000
4	4.000000	0.000000
5	5.000000	0.000000
6	4.000000	0.000000
7	5.000000	0.000000
8	0.000000	1.000000
9	0.000000	−1.000000
10	1.000000	0.000000
11	0.000000	0.000000
12	0.000000	0.000000
13	0.000000	0.000000
14	2.000000	0.000000
15	2.000000	0.000000
16	1.000000	0.000000
17	0.000000	0.000000
18	0.000000	−1.000000
19	0.000000	2.000000
20	0.000000	2.000000
21	1.000000	0.000000
22	0.000000	1.000000
23	1.000000	0.000000
24	0.000000	2.000000

interruptions of medical staff by relatives such as defining two different routes for doctors and relatives in the inpatient area. Our preliminary research resulted that patients' relatives are likely to stop and ask an employee for directions or the situation of the patients very frequently. Therefore, separating the routing between waiting areas and doctors' rooms and nursing units (wards) is important for keeping medical staff focused, productive, and free from interruptions. For formulating this constraint, we define alternative paths between two rooms that are used by different types of people or material. In this way, different users will use different paths and their routes will not be conflicted. For this aim, we use the *k-shortest path idea* described in Fig. 10. (See Fig. 11.)

As an example, between room-1 and room-4, which are located at vertex 5 and 12 respectively, we will create two different paths that are not conflicting with each other. Providing that two alternative routes ($n = 2$) between room-1 and room-4 and inner edge's index is 10. Then, path-1 edges' indices can be 7, 8, 15; path-2 edges' indices can be 7, 8, 14, 17, and 21.

Code 13. Defining shortest-1 and shortest-2 routes between start and finish nodes

Table 5
PoR of Hospital-B [49].

Main-units	Sub-units	Area (M^2)	Number of faces
OUTPATIENT	Ear-Nose-Throat	56	2
	General Surgery	56	2
	Orthopedics	56	2
	Eye	28	1
	Obstetrics	56	2
	Tooth	56	2
	Urology	56	2
	Pulmonary	56	2
	Internal Medicine	56	2
	Dermatology	28	1
	Psychology	28	1
	Cardiology	56	2
	Child	56	2
	Infection	56	2
	Physiotherapy	56	2
	Neurology	56	2
INPATIENT	Medical Inpatient	555	16
	Surgical Inpatient	555	16
ICU	Wards + Waiting Areas	270	8
OT	Operation Rooms, Post Anesthesia Recovery, Scrub-Up Etc.	500	14
EMERGENCY	Beds + Diagnostic + Waiting + Etc.	900	25
DIAGNOSTIC	Radiology	56	2
	X-Ray & (Tomography)	56	2
	Ultrasound	56	2
	Laboratory	56	2
	Blood Centre	56	2
ADMINISTRATION	Business, Accounting Vs.	180	5
	Hospital Management Offices	110	4
ENTRANCE	Welcome Desk	50	2
	Information Desk	50	2
	Cafeteria	75	3
SUPPORT	Worship Places	50	2
	Pharmacy	75	3
	Housekeeping	150	5
	Storages	150	5
	Kitchen And Dining	150	5
	Parking Area	1312	37
	Bunker	120	4
	Mortuary	75	3
	TOTAL	6419	198

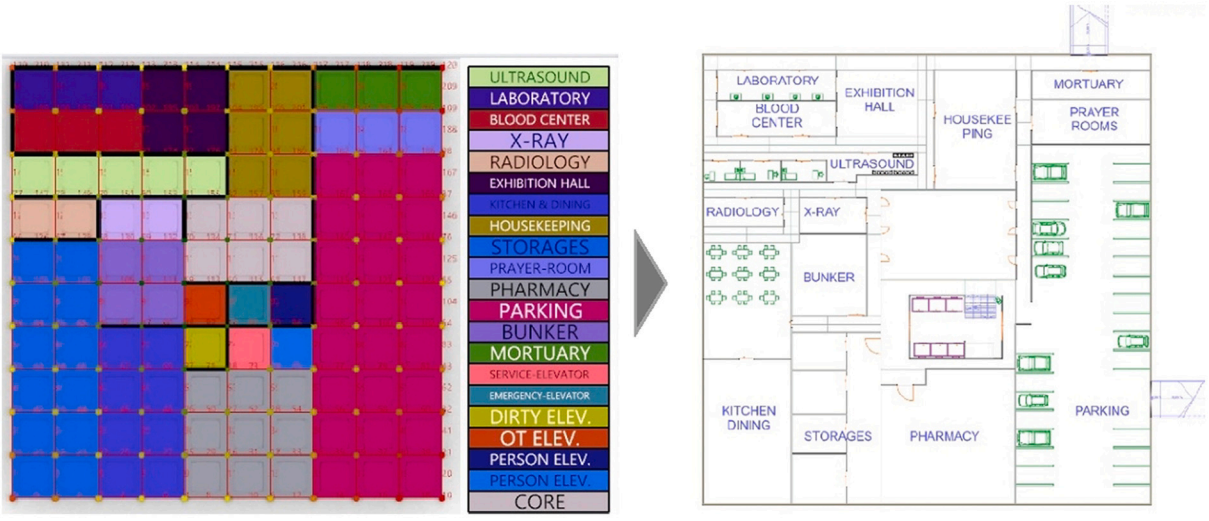

Fig. 16. Basement plans (left: output of the tool, right: technical plans).

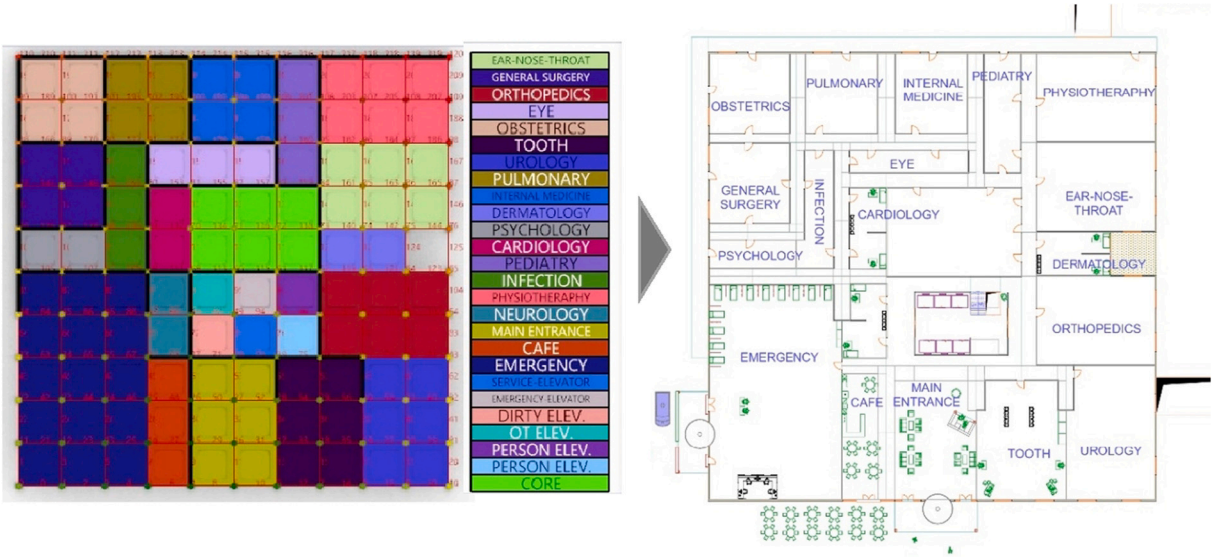

Fig. 17. Ground plans (left: output of the tool, right: technical plans).

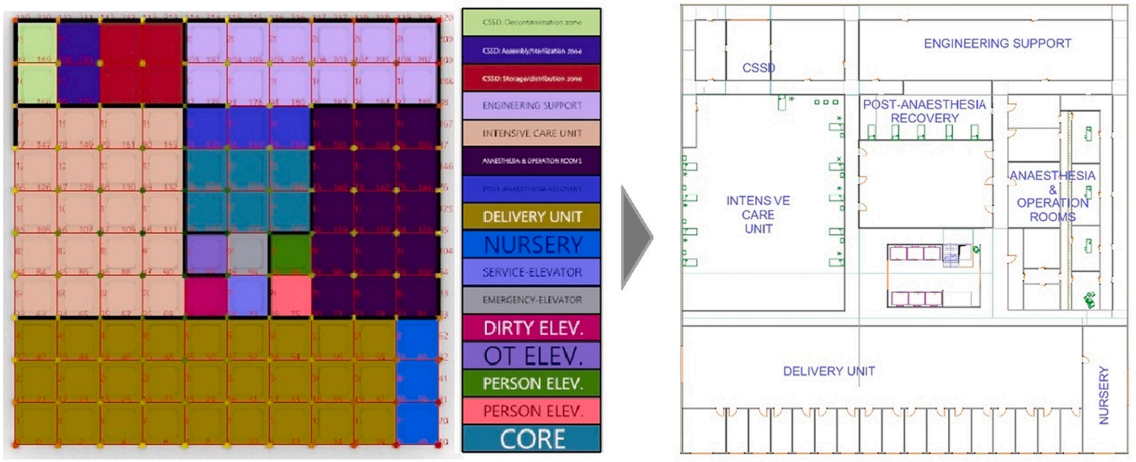

Fig. 18. First-floor plans (left: output of the tool, right: technical plans).

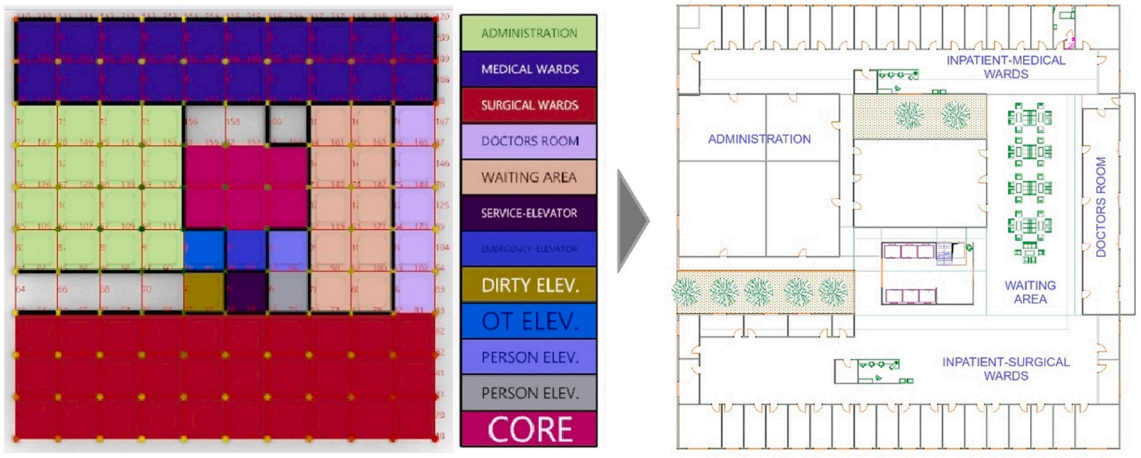

Fig. 19. Second-floor plans (left: output of the tool, right: technical plans).

Fig. 20. Perspective plans of hospital obtained by our computational design tool.

```
start=48
finish=110
def k_shortest_paths(G, source,
target, k, weight=None):
    return list(islice(nx.shortest_simple_paths
(G, source, target, weight=weight), k))

#find shortest1 and shortest2
R=[]
R=k_shortest_paths(G,start,finish,30)

#convert k_shortest vertices
to k_shortest edge indices
lt=[]
edg=[]
for t in range(0,len(R)):
    edg.append([])
    del lt[0:len(lt)]
    lt.extend(R[t])
    for i in range(len(vertexindex)):
        for k in range(len(lt)-1):
            if lt[k]==vertexindex[i][0]
               and lt[k+1]==vertexindex[i][1]:
                edg[t].append(i)
print(edg)

#find two shortest paths between start and
finish which don't include inner edges
for t in range(0,len(edg)):
    del list1[0:len(list1)]
    list1.extend(edg[t])
    for m in range(0,len(edg[t])):
        for v in range(0,len(inneredges)):
            if (edg[t][m] == inneredges[v]):
                del list1[0:len(list1)]
            #break
    if len(list1) > 0:
        indx=t #index of list1 in R[t][m]
        break
for t in range(indx+1,len(edg)):
    del list2[0:len(list2)]
    list2.extend(edg[t])
    for m in range(0,len(edg[t])):
        for v in range(0,len(inneredges)):
            if (edg[t][m] == inneredges[v]):
                del list2[0:len(list2)]
            #break
    if len(list2) > 0:
        indx=t #index of list2 in R[t][m]
        break
```

After reading list1 and list2 as shortest_1 and shortest_2, the constraint is defined in the model as following:

Code 14. Defining two different routes between pairs of rooms constraint

```
for i in range(len(shortest1)):
    model.Add(edges[shortest1[i]]==1)
for j in range(len(shortest2)):
    model.Add(edges[shortest2[j]]==1)
```

5.3. Validity constraints

Corridors cannot pass through the room and each room must connect to at least one corridor as defined in Code 15 and Code 16.

Code 15. Validity constraint #0, Inner Edges

```
# $g_0(p) \leq 0$ a room cannot have corridors in its interior
def g_0_Integrity(edges, all_rooms, inneredgesindex):
    for n in all_rooms:
        for j in range(len(inneredgesindex[n])):
            model.Add(edges[inneredgesindex[n][j]]==0)
```

Code 16. Validity constraint #1, Boundary Edges

Other validity constraint is related to connectivity, which does not allow loop corridors. It created by referring to TSP [61–63] as shown in Code 17.

Code 17. Validity constraint #2, Corridor Connectivity

```
# $g_2(p) \leq 0$ connectivity constraint
def g_2_Connectivity(edges, adjacentedges):
    for e in all_edges:
        model.Add((edges[e]-sum(edges[adjacentedges[e][v]]
        for v in range(len(adjacentedges[e]))))<=0)
```

Connection to sinks is another constraint that ensures the accessibility of the entire network from a sink point (a sink node could be the main entrance for the ground level or an elevator area for the other levels).

Code 18. Validity constraint#3, Reachability of Sink

In addition to the main circulation path, we defined some ‘collector roads’, which provide access to the main circulation path as well as short access between interrelated rooms using T_{ij}.

Code 19. an optional constraint to enforce when it is desired to ensure short access between adjacent rooms

6. Tool implementation

The workflow presented in this paper has been implemented within a popular computational design platform (Rhino3D + Grasshopper3D), as a laboratory and testbed environment; and developed our tools using Python within Anaconda Spyder as an Integrated Development Environment (IDE) (See Fig. 7). The codes are written in GHCPython and GHPython in Grasshopper 3D. Rhino is used for input processing and output visualization. All the algorithms presented in this paper are implemented in the GHPython nodes connected in the structured workflow diagram inside Grasshopper3D. A summary of this workflow is abstracted and illustrated in Fig. 12. The partial components are illustrated in Figs. 13 and 14. We have used NumPy, Matplotlib, and NetworkX python libraries for the implementation of spectral clustering methods in the stacking stage. Used libraries (all-free) in Python components at zoning and orienting stages are listed below:

- Google or-tools integer programming tool- > CP-SAT solver [64]
- Pandas
- Sci-kit learn
- NetworkX
- Csv (for data-stream between different components)
- Random
- Numpy

One important note about the Google OR-Tools (Operations Research Tools) is that the models built with this library can also accept

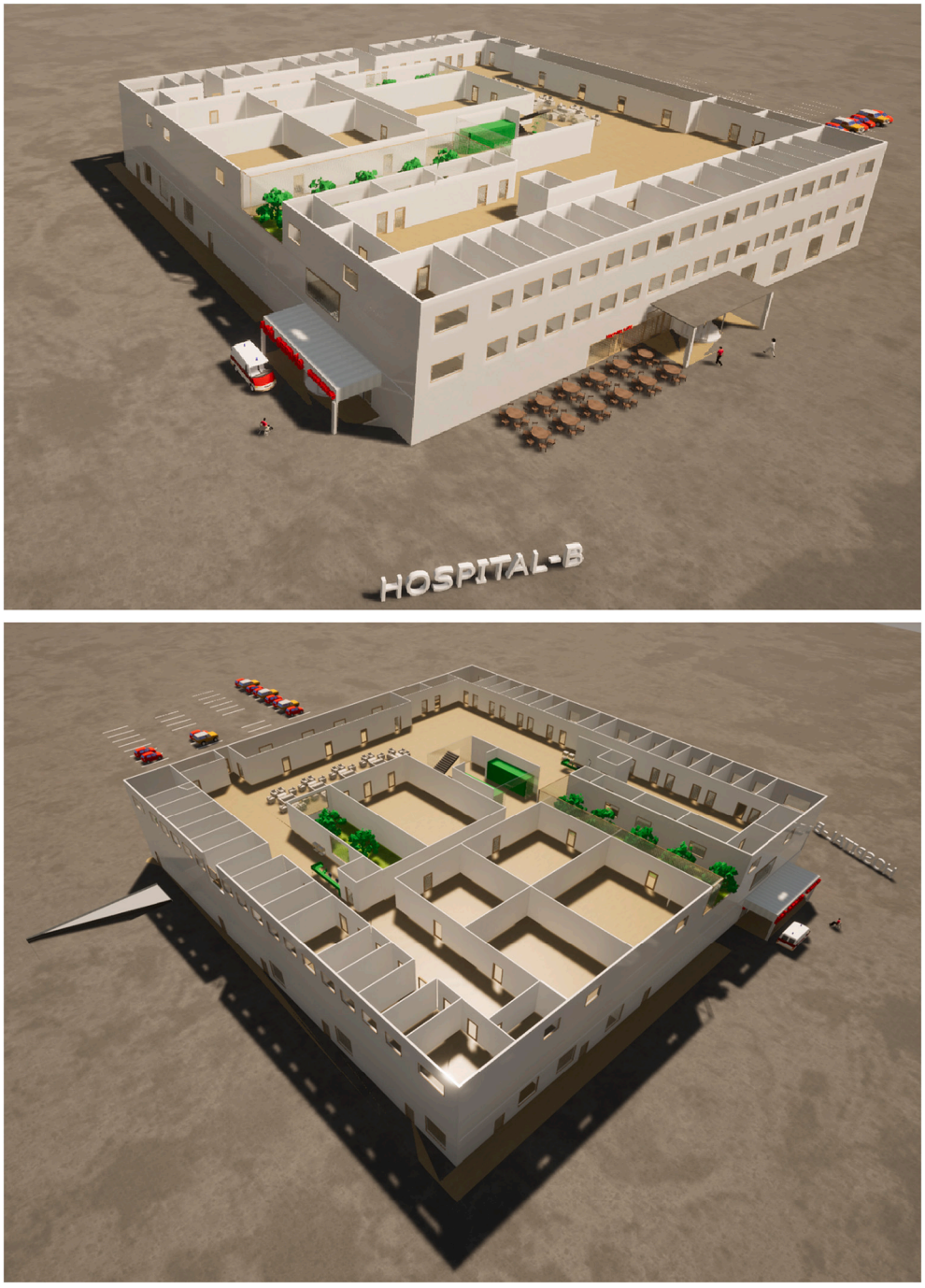

Fig. 21. Renders of the case study hospital.

Linear Programming problems that are only partially linear (such as the absolute function). We do this effectively by adding conditional statements to the model building algorithms (as shown in Code 1, Code 2, Code 3, Code 4, etc.) that would activate/deactivate partially some of the objectives or constraints that are all linear partially but not linear as a whole function. The possibility of adding these conditional statements means that this tool suite is more flexible than a tool suite that can only work with the canonical formulation of LP/MIP problems. The statements model.Add() in this tool suite are used for adding constraints and the statements in the form of model.Minimize() accept objective functions as input arguments. Additionally, statements in the form of model.NewBoolVar() and model.NewIntVar() are respectively used for adding Boolean and Integer decision variables. This tool consists of 6 components with component-0 for data preparation and series of modules for visualizations. Between each component, all data streams are successfully realized by reading & writing csv files using both csv and pandas libraries.

As a small case example to the tool implementation, for a building footprint discretized into a mesh with sizes of $W = 3$ and $H = 3$, the task is to locate 4 rooms with a T_{ij} = [[0,1,1,1], [1,0,0,1],[1,0,0,0], [1,1,0,0]], $w_{min} = \{2,1,2,2\}$, $w_{max} = \{2,1,2,2\}$, $h_{min} = \{1,1,1,1\}$, $h_{max} = \{1,1,1,1\}$. Regarding the defined constraints, Room 1 and Room2 are adjoined (using constraint explained in 4.5) and fixed location constraint is $x_0 = 0$; $y_0 = 0$; $x_2 = 1$. Fixed border constraint is defined as

Fig. 22. Modular construction in hospitals example [66].

$x_1 + w_1 \leq W$ where $x_i = [0,3]$, $y_i = [0,3]$ for all $i = 0,..,NA$ where NA is the number of rooms. As can be seen in the optimized result in Fig. 15, room 1 and 2 are adjoined and room 1 is located to the {0,0} coordinates and all of the size constraints are handled and the result is optimal. The computational result is given in Table 3 and the sensitivity analysis result is given in Table 4.

As presented in Table 4, fixed location constraints for x_0, y_0 and x_2 (Row 8,9,18), have an impact on the solution value when the right-hand side value of the constraints changed e.g. when increasing the RHS of the first fixed location constraint ($x_0 = 0$) to one, then objective value (distance minimization) will be decreased by dual price = 1. Likewise, the dual price of the non-intersecting constraints between Room 1&2 (Row 19), between Room 1&3 (Row 20), between Room 2&3 (Row22) and between Room 3&4 (Row 24) have non-zero values so that they have an effect on the objective value. Other than these constraints like connectivity constraints, boundary constraints and non-intersecting constraints between Room 1&4 and Room 2&4 have a zero dual price thus, changing the right-hand side a small amount will have no effect on the solution value. When decision variables result analyzing, it is observed that w_0, h_0, h_1 and h_2 values have non-zero reduced costs (2,2,1,2 respectively). Therefore, we can say that the sizes of the Room 1 have a great impact on the objective value as well as the height of the Room 2 & Room 3.

7. Case study: Hospital-B

Due to security reasons, we call the case study hospital as Hospital-B. Existing Hospital-B has 55 inpatient beds where 10 of them are in single-patient rooms. The side of the existing Hospital-B is not earthquake-resistant. Therefore, the existing hospital will be demolished and the new one will be built in a new place in the same district in Turkey. The new hospital is planned to have 75–100 inpatient beds and it will be classified in the 3rd level hospitals in Turkey. Levels of hospitals are changing in Turkey according to the property of rooms. The higher the number of rooms with single beds, the higher the level of the hospital. Polyclinics in the new hospital are listed as Emergency, Ear-Nose-Throat, General Surgery, Orthopedics, Pulmonary, Internal Medicine, Dermatology, Psychology, Cardiology, Eye, Child, Obstetrics, Infection, Tooth, Physiotherapy, Urology, Neurology. Services in the new hospital are medical units and surgical units. Diagnostic units in the new hospital are Radiology, Laboratory, Tomography, and Blood Center. The new hospital will have one ICU with 10 beds and 4 operation rooms while the existing one has 4 ICU beds and 4 operation room beds. In the hospital, frequent operations in the OT are eye operations 2 days in a week, ear-nose-throat operations and tooth operations 4 dentists (2 rooms: 1 is in one room and 3 are in one room). Other than these, the site area is a touristic place and very close to other touristic places. During summertime, Hospital-B takes many patient transfers from a University Hospital located in the city center because of its limited capacity and frequent traffic accidents around the place. Future expansion of the hospital is very important because the city is in development. The new hospital's place is very close to the historical sides. Before planning the layout of the new hospital, program of requirements of the Hospital-B is defined based on the methodology presented in [49]. Mainly, Turkey's hospital design standards are considered in this paper during this part. The minimum area requirements of each sub-unit are given in Table 5. In this table, the number of faces (in the form of the square) needed for each sub-unit is also calculated. Since each edge of the boundary mesh is 6 m, we divided the area requirement into 36 and get the minimum number of square faces for each unit in the Hospital-B. We have divided the desired surface area of each unit by the area of a [pixel-like] mesh-face and rounded up these values to end with integer surface values as shown in Table 5.

In addition to the units given in this table, the hospital needs more areas for exhibition halls for enabling flexibility of the hospital. One service core is defined in the center of the building. This core constitutes service areas such as toilets, warehouses, stairs, and fire stairs. The area of a core is 68 m^2. Values of w_{min}, w_{max}, l_{min}, l_{max} are defined according to this PoR table for each level of the hospital. We consider hospital standards pertained to its spatial planning as follows [65]:

```
# $g_1(p) \leq 0$ a room has to be connected to a corridor,
i.e. at least one of the boundary edges of each room
must be active
def g_1_Accessibility(edges, all_rooms, boundaryedgesindex):
    for n in all_rooms:
        model.Add(sum(edges[boundaryedgesindex[n][t]] for t in
range(len(boundaryedgesindex[n])))>=1)
```

```
# $g_3(p) \leq 0$ Reachability of sink (one of the sink-incident
edges must be active)
def g_3_Connectivity(edges, sink, edgeToVertexIncidence):
    sinkEdges= edgeToVertexIncidence[sink]# sink-incident edges
    model.Add(sum([edges[sinkEdge] for sinkEdge in sinkEdges])>0)
```

```
# $g_4(p) \leq 0$ collector roads
V=[95,88,62,91,110,39,41,112,114,73,66,80,116,90,117,58,37,36,55,49,60,48,59,61
,50,81] #rooms top left vertices
def g_4_Collectivity(edges, V, num_rooms, G, T, vertexindex, inneredges):
    R=[]
    for a in range(num_rooms):
        for b in range(a+1,num_rooms):
            if T[a][b]==1: #T is adjacency matrix
                R=k_shortest_paths(G,V[a],V[b],30)
                list1=[]
                #convert k_shortest vertices to k_shortest edge indices
                lt=[]
                edg=[]
                for t in range(0,len(R)):
                    edg.append([])
                    del lt[0:len(lt)]
                    lt.extend(R[t])
                    for i in range(len(vertexindex)):
                        for k in range(len(lt)-1):
                            if lt[k]==vertexindex[i][0] and
lt[k+1]==vertexindex[i][1]:
                                edg[t].append(i)
#find shortest path between start and finish which don't include inner edge
                for t in range(0,len(edg)):
                    del list1[0:len(list1)]
                    list1.extend(edg[t])
                    for m in range(0,len(edg[t])):
                        for v in range(0,len(inneredges)):
                            if (edg[t][m] == inneredges[v]):
                                del list1[0:len(list1)]
                                #break
                    if len(list1) > 0:
                        indx=t #index of list1 in R[t][m]
                        break
                for m in range(len(list1)):
                    model.Add(edges[list1[m]]==1) #the shortest edges between
interrelated rooms will be active by this constraint
```

1. Outpatient rooms must be min 16 m^2.
2. Outpatient waiting areas must be
   a. min 12 m^2 for 1 doctor
   b. min 24 m^2 for 2 doctors
   c. additional 5 m^2 for each additional number of doctors.
3. There must be min 6 elevators in 60–200 bed hospitals.
4. There must be min 9 elevators in 201–350 bed hospitals.
5. One-bed patient rooms must be min 9 m^2.
6. Patient wards must be min 7 m^2 per bed.
7. One-bed delivery patient rooms must be min 12 m^2.
8. Delivery patient wards must be min 10 m^2 per bed.
9. ICU units must be min 12 m^2 per bed.
10. Neonatal ICU units must be min 6 m^2 per bed.
11. Administrative offices must be 8–12 m^2 for each personnel.
12. Bunker area = (number of beds) + (number of beds*20%)
13. Parking area for healthcare buildings: 125 m^2 closed area = min 1 parking area
14. Windows area must be between 1/5–1/7 of the floor area of each inpatient room.
15. Stair width 2 m., Step height 17 cm., Step depth 28 cm., Landing width 2 m.

We defined the 6-elevator system in the hospital considering the 3rd bullet of hospital design standards given above; one is a service elevator, one is a dirty elevator, one is for transporting to operating rooms and two are for person elevators. Discrete waiting areas are defined for each outpatient department on the ground level considering the 1st and 2nd bullets of hospital design standards presented above. Calculations are done by assuming 2 doctors in each outpatient department. According to the 13th bullet, the hospital needs 115 parking areas and we defined 31 parking areas in the basement and the rest are available in the outside area. Regarding the facade, windows are created based on the 14th bullet. The stairs areas are calculated according to the 15th bullet of the standards.

According to the method introduced in Chapter 4, we defined the stacking of the hospital. Ultrasound, laboratory, blood center, x-ray, radiology, exhibition hall, kitchen & dining, housekeeping, storages, prayer-room, pharmacy, parking area, bunker, mortuary are located on the basement level; all polyclinics with the accompanying waiting areas, emergency, main entrance, and cafeteria are located on the ground level; operation theatre, intensive care units, CSSD, engineering support, delivery units and nursery located on the first floor, administration, medical wards, surgical wards, doctors' rooms, waiting area are located on the second floor in this case study.

On each level, we selected different objectives and constraints with respect to the requirements of the sub-units and we used different T_{ij} matrices between sub-units for each level (given in Appendix A). For example, in the basement, we considered distance-to-sink (sink: elevator point) minimization of the way-finding objective and route separation constraint (for separating routing between mortuary and elevator area for privacy issue). On the ground floor, we considered the angle minimization of the way-finding objective in addition to the distance-to-sink objective because of the unfamiliar visitors who come to the hospital and don't know about their way in the hospital. On the first floor, we considered distance-to sink (sink: entrance point) minimization and route separation constraint (for separating routing between central sterile department's dirty materials stocking part and dirty elevator for cleanliness issue). On the second floor, we considered distance-to sink (sink: entrance point) minimization and route separation constraint (for separating routing between wards and doctors' room for work balance issue).

As can be seen in the basement plan, the parking area has a discrete entrance and thus it is located at one of the borders as expected. Blood center, laboratories, x-ray and radiology departments are adjoined and the exhibition area is located next to the laboratories for future expansion reasons. There are two routes passing through the housekeeping between the mortuary and elevator area as expected. The pharmacy is located next to the elevator area for the ease of distribution of medicines. In the ground floor plan, the emergency has a discrete entrance and located to one façade of the hospital building. Thus, the main entrance and emergency entrance are discretised as expected. The café is also located next to the main entrance. Related outpatient departments are close to other such as general surgery, paediatrics and obstetrics. In the first-floor plan, ICU is located next to the recovery room of OT (blue pixels), in addition, the delivery unit and nursery share a wall as expected. There are two different routes between the elevator area and CSSD units. In the second-floor plan, all wards have a possibility of having windows as they are located to the borders with "adjacency to NEWS" constraint. There are two different routes between doctors' room and medical wards for not passing through the visitors' waiting area as expected from the "route separation" constraint. All of the codes for each floor plan are run on an Intel Core-i7 computer, with 2 GB of RAM. Case-study application results are given in Figs. 16–19. (See also Figs. 20 and 21.)

8. Modular construction

The potential use of modular construction techniques is another advantage of the model developed in this paper. This part of the paper aims to show the outlook of our method for being applied to space planning for modular construction of hospitals. Our results show clearly that we can generate such modular configurations successfully. Many hospital and healthcare facility contractors are turning into modular, primarily for building components such as bathroom pods and head-walls, however, entire hospitals can be constructed utilizing modular construction techniques [66]. Modular construction offers quiet, safe, and clean applications for medical, surgical, clinical, and dental use and a fast and economical approach. The hospital layout design method based on the discrete mesh presented in this paper can provide a suitable infrastructure for modular construction in the next design stages of the hospital buildings (See Fig. 22).

9. Conclusion and discussion

We presented a formulation of the hospital layout problem as a Mixed-Integer Programming (optimization) problem, following a hierarchical framework that divides the main design stages into stacking, zoning, and routing (corridor-generation). The formulation has been fed into an industry-standard MIP solver from Google OR tools (Operations Research Tools), implemented in Python, and adapted to work within the Mc Neel's Rhino3D CAD environment. The toolkit has been tested in the context of a real-world hospital design case study considering the actual strict design codes and standards for good practice. This paper reports a part of a larger hospital design optimization framework, which has been partially reported in two other papers (removed references for the double-blind review). The results show the viability of this approach in dealing with an overwhelming amount of constraints imposed by strict design codes combined with realistic optimization objectives. Compared to several other approaches, the main advantage of this approach is the holistic consideration of the zoning and routing problems within the same framework. From a methodological point of view, this paper shows a direct application of Graph Theory combined with Operations Research in solving some of the most daunting complex design problems pertaining to layout configuration. Arguably, the OR problem classes are the most well-known optimization problems and their long history in engineering applications entails that their solvers are quite standard, robust (guaranteed to converge in a reasonable time), and accessible. For these reasons, bringing a problem to these standard forms makes it possible to benefit from these advantages. The challenging part of such an undertaking is of course bringing the problem into a standard form. What this paper offers is thus a practical way of bringing such complex problems into such a canonical problem formulation thus opening a gateway towards OR methodologies for solving architectural design problems in practice. The methodology as presented in this paper is specially designed to tackle the hospital design problem but it can potentially be generalized into other types of complex building layout problems, albeit contingent on dealing with challenges and limitations as follows.

9.1. Limitations

In general, the problem of layout is a 3D problem that is simplified into a series of flat 2D layout problems in this paper due to the focus on hospital layout problems and the necessity of stacking. The combination of zoning and routing problems as explained in the last paragraph of related works makes the layout problem a very daunting task. Instead of avoiding such complexity and focusing only on one, we have chosen to deal with both problems in one framework, albeit by assuming to avoid

the common chicken and egg problem: we deal with zoning first and later perform a routing task. In doing so, however, the consideration of the distance between rooms is based on Manhattan distance along the edges of a quadrilateral mesh, which is a supremum for network distances, i.e. a proxy indicator for the real network distance. While it is possible to generalize this part of the methodology to incorporate network distances by using a look-up table, in this paper, we have used Manhattan distance for simplicity. In this regard, it must be noted that the Manhattan distance is in fact not a linear function, but because of the flexibility of the OR tools implementation, it is possible to enforce it as an objective. Similarly, the OR tools' solver can accept multiple additional constraints thanks to the possibility of adding auxiliary Boolean variables.

In this approach, the condition of the rectilinear form of rooms may seem to be overly strict. However, this is a matter of scale. A hospital unit consists of several rooms and this hospital unit can have an irregular shape in the form of several rectangular rooms. Architects generally consider near-convex room shapes to be preferable to strongly concave ones (q.v. [67] and [68]). Our case study shows a building with a perfectly rectilinear outline; however, in general, this is obviously not always the case. While it is possible to generalize the presented methodology to deal with non-rectilinear boundaries, we have not tested such cases. The presence of large obstacles or unavailable areas in the middle of a floor plan (e.g. because of an atrium) is also not considered in this example.

The scalability of the proposed algorithms for large problem instances is dubious as they depend on LP solvers based on the solver's implementation (Simplex, Branch & Bound, etc.), which are not all linear in terms of computational time and known to have scale-up issues with large numbers of decision variables [69].

9.2. Future work

Generalizations and tests with regards to the boundary and interior of the floor plans are the top priorities for future work. The next order of priority for tests and generalizations will be to replace Manhattan distance with network distance in the zoning stage. Another generalization may seem to be possible if we consider quadrilateral meshes in general rather than meshes only consisting of pixel-like quads; however, we maintain that the pixel-like quads have higher potential because if the solver is fast we can work with virtually any shape that can be pixelated.

Declaration of Competing Interest

The authors declare that there is no conflict of interest in this paper.

Appendix A

T_{ij} of the basement floor:

Design Requirements		closeness matrix	ULTRASOUND	LABORATORY	BLOOD CENTER	X-RAY	RADIOLOGY	EXHIBITION HALL	KITCHEN & DINING	HOUSEKEEPING	STORAGES	PRAYER-ROOM	PHARMACY	PARKING	BUNKER	MORTUARY	SERVICE-ELEVATOR	EMERGENCY-ELEVATOR	DIRTY ELEV.	OT ELEV.	PERSON ELEV.	PERSON ELEV.	CORE
	0	ULTRASOUND	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
close to exhitibion	1	LABORATORY	1	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0
	2	BLOOD CENTER	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0
	3	X-RAY	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	RADIOLOGY	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
	5	EXHIBITION HALL	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
close to service elevator	6	KITCHEN & DINING	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
close to service elevator	7	HOUSEKEEPING	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	8	STORAGES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	9	PRAYER-ROOM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
close to emergency elevator	10	PHARMACY	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
discrete entrance	11	PARKING	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	12	BUNKER	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
discrete entrance	13	MORTUARY	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fixed space	14	SERVICE-ELEVATOR	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
fixed space	15	EMERGENCY-ELEVATOR	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
fixed space	16	DIRTY ELEV.	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
fixed space	17	OT ELEV.	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fixed space	18	PERSON ELEV.	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fixed space	19	PERSON ELEV.	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	CORE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1-5	ADJOIN
4-3	ADJOIN

T_{ij} of the ground floor:

Design Requirements		closeness matrix	EAR-NOSE-THROAT	GENERAL SURGERY	ORTHOPEDICS	EYE	OBSTETRICS	TOOTH	UROLOGY	PULMONARY	INTERNAL MEDICINE	DERMATOLOGY	PSYCHOLOGY	CARDIOLOGY	PEDIATRY	INFECTION	PHYSIOTHERAPHY	NEUROLOGY	MAIN ENTRANCE	CAFE	EMERGENCY	SERVICE-ELEVATOR	EMERGENCY-ELEVATOR	DIRTY ELEV.	OT ELEV.	PERSON ELEV.	PERSON ELEV.	CORE
	0	EAR-NOSE-THROAT	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Frequent consultation from inpatient: 2	1	GENERAL SURGERY	1	0	1	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
	2	ORTHOPEDICS	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	3	EYE	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Frequent consultation from inpatient: 5	4	OBSTETRICS	1	1	1	1	0	1	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
	5	TOOTH	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	6	UROLOGY	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Frequent consultation from inpatient: 6	7	PULMONARY	0	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0
Frequent consultation from inpatient: 1	8	INTERNAL MEDICINE	0	1	0	0	0	0	0	1	0	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0
	9	DERMATOLOGY	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
	10	PSYCHOLOGY	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
Frequent consultation from inpatient: 3	11	CARDIOLOGY	0	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0
Frequent consultation from inpatient: 7	12	PEDIATRY	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0
	13	INFECTION	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
façade and discrete entrance	14	PHYSIOTHERAPHY	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
Frequent consultation from inpatient: 4	15	NEUROLOGY	0	0	0	0	1	0	0	1	1	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0
close to sink	16	MAIN ENTRANCE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
façade and discrete entrance	17	CAFE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
façade and discrete entrance	18	EMERGENCY	0	1	0	0	1	0	0	1	1	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0
	19	SERVICE-ELEVATOR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	20	EMERGENCY-ELEVATOR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	21	DIRTY ELEV.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	22	OT ELEV.	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	23	PERSON ELEV.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	24	PERSON ELEV.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	CORE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

16-17	ADJOIN

T_{ij} of the first floor:

Design Requirements			CSSD: Decontamination zone	CSSD: Assembly/sterilization zone	CSSD: Storage/distribution zone	ENGINEERING SUPPORT	INTENSIVE CARE UNIT	GENERAL ZONE & ANAESTHESIA & INDUCTION &SCRUB-UP & OPERATION ROOMS	POST-ANAESTHESIA RECOVERY	DELIVERY UNIT	NURSERY	SERVICE-ELEVATOR	EMERGENCY-ELEVATOR	DIRTY ELEV.	OT ELEV.	PERSON ELEV.	PERSON ELEV.	CORE
accesible by wards	0	CSSD: Decontamination zone	0	1	1	0	0	0	1	1	0	1	0	1	0	0	0	0
	1	CSSD: Assembly/sterilization zone	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
accesible by wards	2	CSSD: Storage/distribution zone	1	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0
close to ot-recovery & ICU	3	ENGINEERING SUPPORT	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
close to emergency elev., acc. By person elev.	4	INTENSIVE CARE UNIT	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0
close to ot elev.	5	GENERAL ZONE & ANAESTHESIA & INDUCTION &SCRUB-UP & OPERATION ROOMS	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	6	POST-ANAESTHESIA RECOVERY	1	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0
accessible by surgical wards	7	DELIVERY UNIT	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	0
seperated from OT but accessible	8	NURSERY	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
seperated from OT but accessible	9	SERVICE-ELEVATOR	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	EMERGENCY-ELEVATOR	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	11	DIRTY ELEV.	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	12	OT ELEV.	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	13	PERSON ELEV.	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	14	PERSON ELEV.	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	15	CORE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

T_{ij} of the second floor:

Design Requirements			ADMINISTRATION	MEDICAL WARDS	SURGICAL WARDS	DOCTORS ROOM	WAITING AREA	SERVICE-ELEVATOR	EMERGENCY-ELEVATOR	DIRTY ELEV.	OT ELEV.	PERSON ELEV.	PERSON ELEV.	CORE
	0	ADMINISTRATION	0	1	1	1	0	0	0	0	0	0	0	0
facade: lighting	1	MEDICAL WARDS	1	0	0	1	1	0	1	0	0	1	1	0
facade: lighting	2	SURGICAL WARDS	1	0	0	1	1	0	1	0	0	1	1	0
	3	DOCTORS ROOM	1	1	1	0	0	0	0	0	0	0	0	0
	4	WAITING AREA	0	1	1	0	0	0	0	0	0	1	0	0
	5	SERVICE-ELEVATOR	0	0	0	0	0	0	0	0	0	0	0	0
	6	EMERGENCY-ELEVATOR	0	1	1	0	0	0	0	0	0	0	0	0
	7	DIRTY ELEV.	0	0	0	0	0	0	0	0	0	0	0	0
	8	OT ELEV.	0	0	0	0	0	0	0	0	0	0	0	0
	9	PERSON ELEV.	0	1	1	0	1	0	0	0	0	0	0	0
	10	PERSON ELEV.	0	1	1	0	0	0	0	0	0	0	0	0
	11	CORE	0	0	0	0	0	0	0	0	0	0	0	0

References

[1] D. Griffin, Hospitals: what they are and how they work, Jones & Bartlett Learn. (2011) 978–0763791094.

[2] Y. Zhou, Healthcare facility research and design, Front. Architect. Res. 3 (2014) 227, https://doi.org/10.1016/j.foar.2014.08.002.

[3] R.F. Carr, Hospital, Whole Building Design Guide. https://www.wbdg.org/building-types/health-care-facilities/hospital, 2007 (accessed July 31, 2021).

[4] S.R. Sahamir, R. Zakaria, Green assessment criteria for public hospital building development in Malaysia, Procedia Environ. Sci. 20 (2014) 106–115, https://doi.org/10.1016/j.proenv.2014.03.015.

[5] C. Cubukcuoglu, P. Nourian, M.F. Tasgetiren, I.S. Sariyildiz, S. Azadi, Hospital layout design renovation as a Quadratic Assignment Problem with geodesic distances, J. Build. Eng. 44 (2021) 102952, https://doi.org/10.1016/j.jobe.2021.102952.

[6] P. Nourian, Configraphics: Graph theoretical methods for design and analysis of spatial configurations, A + BE | Architecture and the, Built Environ. 14 (2016), 978–94–6186-720-9.

[7] P. Pérez-Gosende, J. Mula, M. Diaz-Madroñero, Facility layout planning. an extended literature review, Int. J. Prod. Res. 59 (2021) 3777–3816, https://doi.org/10.1080/00207543.2021.1897176.

[8] J. Huo, J. Liu, H. Gao, An NSGA-II algorithm with adaptive local search for a new double-row model solution to a multi-floor hospital facility layout problem, Appl. Sci. 11 (2021) 1758, https://doi.org/10.3390/app11041758.

[9] N. Jamali, R.K. Leung, S. Verderber, A review of computerized hospital layout modelling techniques and their ethical implications, Front. Architect. Res. 9 (2020), https://doi.org/10.1016/j.foar.2020.01.003.

[10] C.-H. Peng, Y.-L. Yang, F. Bao, D. Fink, D.-M. Yan, P. Wonka, N.J. Mitra, Computational network design from functional specifications, ACM Trans. Graph. 35 (2016) 1–12, https://doi.org/10.1145/2897824.2925935.

[11] H. Hosseini-Nasab, S. Fereidouni, S.M.T. Fatemi Ghomi, M.B. Fakhrzad, Classification of facility layout problems: a review study, Int. J. Adv. Manuf. Technol. 94 (2018) 957–977, https://doi.org/10.1007/s00170-017-0895-8.

[12] M. Rajasekharan, B.A. Peters, T. Yang, A genetic algorithm for facility layout design in flexible manufacturing systems 36 (1998) 95–110, https://doi.org/10.1080/002075498193958.

[13] Q. Liu, R.D. Meller, A sequence-pair representation and MIP-model-based heuristic for the facility layout problem with rectangular departments, IIE Trans. 39 (2007) 377–394, https://doi.org/10.1080/07408170600844108.

[14] Y.A. Bozer, C.T. Wang, A graph-pair representation and MIP-model-based heuristic for the unequal-area facility layout problem, Eur. J. Oper. Res. 218 (2012) 382–391, https://doi.org/10.1016/j.ejor.2011.10.052.

[15] D. Koşucuoğlu, Ü. Bilge, Material handling considerations in the FMS loading problem with full routing flexibility, Int. J. Prod. Res. 50 (2012) 6530–6552, https://doi.org/10.1080/00207543.2011.653837.

[16] C.C. Murray, X.Q. Zuo, A.E. Smith, An extended double row layout problem, in: 12th IMHRC Proceedings, 2012, pp. 554–569. https://digitalcommons.georgiasouthern.edu/pmhr_2012/33/ (accessed July 31, 2021).

[17] S. Kulturel-Konak, A. Konak, Linear programming based genetic algorithm for the unequal area facility layout problem, Int. J. Prod. Res. 51 (2013) 4302–4324, https://doi.org/10.1080/00207543.2013.774481.

[18] Y. Xiao, Y. Seo, M. Seo, A two-step heuristic algorithm for layout design of unequal-sized facilities with input/output points, Int. J. Prod. Res. 51 (2013) 4200–4222, https://doi.org/10.1080/00207543.2012.752589.

[19] D. Hong, Y. Seo, Y. Xiao, A concurrent approach for facility layout and AMHS design in semiconductor manufacturing, International, J. Ind. Eng. 21 (2014)

231–242. https://koreauniv.pure.elsevier.com/en/publications/a-concurrent-approach-for-facility-layout-and-amhs-design-in-semi (accessed July 31, 2021).
[20] A.W.A. Hammad, D. Rey, A. Akbarnezhad, A cutting plane algorithm for the site layout planning problem with travel barriers, Comput. Oper. Res. 82 (2017) 36–51, https://doi.org/10.1016/j.cor.2017.01.005.
[21] T.A. Lacksonen, Static and dynamic layout problems with varying areas, J. Oper. Res. Soc. 45 (1994) 47–58, https://doi.org/10.1057/jors.1994.7.
[22] I.J. Leno, S.S. Sankar, S.G. Ponnambalam, MIP model and elitist strategy hybrid GA–SA algorithm for layout design, J. Intell. Manuf. 29 (2018) 369–387, https://doi.org/10.1007/s10845-015-1113-x.
[23] X. Wan, X. Zuo, X. Li, X. Zhao, A hybrid multiobjective GRASP for a multi-row facility layout problem with extra clearances, Int. J. Prod. Res. (2020) 1–20, https://doi.org/10.1080/00207543.2020.1847342.
[24] S. Liu, Z. Zhang, C. Guan, L. Zhu, M. Zhang, P. Guo, An improved fireworks algorithm for the constrained single-row facility layout problem, Int. J. Prod. Res. 59 (2021) 2309–2327, https://doi.org/10.1080/00207543.2020.1730465.
[25] IBM ILOG CPLEX User''s manual. https://www.ibm.com/docs/en/SSSA5P_12.8.0/ilog.odms.studio.help/pdf/usrcplex.pdf (accessed July 31, 2021).
[26] K. Keatruangkamala, K. Sinapiromsaran, Optimizing architectural layout design via mixed integer programming, Comp. Aided Architect. Design Futures (2005) 175–184, https://doi.org/10.1007/1–4020-3698-1_16.
[27] K. Keatruangkamala, P. Nilkaew, Strong valid inequality constraints for architectural layout design optimization, in: Proceedings of the 11th International Conference on Computer Aided Architectural Design Research, 2006, pp. 179–185. http://papers.cumincad.org/cgi-bin/works/BrowseTreefield=seriesorder=AZ/Show?caadria2006_179 (accessed July 31, 2021).
[28] K. Keatruangkamala, K. Sinapiromsaran, Mixed integer programming model with non circular and guided constraints for architectural layout design optimization, Songklanakarin J. Sci. Technol. 30.5 (2008) 673–686. https://eds.a.ebscohost.com/eds/pdfviewer/pdfviewer?vid=0&sid=b7e9210b-c987-4c94-82b4-65ffcc080810%40sessionmgr4006 (accessed July 31, 2021).
[29] G.B. Benitez, G.J.C. Da Silveira, F.S. Fogliatto, Layout planning in healthcare facilities: a systematic review, HERD: Health Environ. Res. Design J. 12 (2019) 31–44, https://doi.org/10.1177/1937586719855336.
[30] A.R.S. Amaral, The corridor allocation problem, Comput. Oper. Res. 39 (2012) 3325–3330, https://doi.org/10.1016/j.cor.2012.04.016.
[31] S. Helber, D. Böhme, F. Oucherif, S. Lagershausen, S. Kasper, A hierarchical facility layout planning approach for large and complex hospitals, Flex. Serv. Manuf. J. 2 (2014) 5–29, https://doi.org/10.1007/s10696-015-9214-6.
[32] A. Chraibi, S. Kharraja, I.H. Osman, O. Elbeqqali, Multi-agent system for solving dynamic operating theatre facility layout problem, IFAC-PapersOnLine 28 (2015) 1146–1151, https://doi.org/10.1016/j.ifacol.2015.06.238.
[33] I. Acar, S.E. Butt, Modelling nurse-patient assignments considering patient acuity and travel distance metrics, J. Biomed. Inform. 64 (2016) 192–206, https://doi.org/10.1016/j.jbi.2016.10.006.
[34] T.W. Butler, K.R. Karwan, J.R. Sweigart, Multi-level strategic evaluation of hospital plans and decisions, J. Operat. Res. Soc. 43 (2017) 665–675, https://doi.org/10.1057/jors.1992.99.
[35] S. Safarzadeh, H. Koosha, Solving an extended multi-row facility layout problem with fuzzy clearances using GA, Appl. Soft Comput. 61 (2017) 819–831, https://doi.org/10.1016/j.asoc.2017.09.003.
[36] X. Wang, X. Gong, N. Geng, Z. Jiang, L. Zhou, Metamodel-based simulation optimisation for bed allocation, Int. J. Prod. Res. 58 (2020) 6315–6335, https://doi.org/10.1080/00207543.2019.1677962.
[37] E. Lee, J. Daugherty, J. Selga, U. Schmidt, Enhancing patients' wayfinding and visitation experience improves quality of care, J. PeriAnesthesia Nurs. 35 (3) (2020) 250–254, https://doi.org/10.1016/j.jopan.2019.11.003.
[38] S. Aripin, Healing architecture: daylight in hospital design, Conf. Sust. Build. South East Asia 5 (7) (2007) 173–181. https://www.academia.edu/696902/_HEALING_ARCHITECTURE_DAYLIGHT_IN_HOSPITAL_DESIGN (accessed 31 July 2021).
[39] C. Alalouch, P.A. Aspinall, H. Smith, Design criteria for privacy-sensitive healthcare buildings, Int. J. Eng. Technol. 8 (2016) 32–39, https://doi.org/10.7763/IJET.2016.V8.854.
[40] W.-C. Chiang, Visual facility layout design system, Int. J. Prod. Res. 39 (2001) 1811–1836, https://doi.org/10.1080/00207540110035192.
[41] P. Merrell, E. Schkufza, V. Koltun, Computer-generated residential building layouts, ACM SIGGRAPH Asia (2010) 1–12, https://doi.org/10.1145/1866158.1866203.
[42] W. Wu, L. Fan, L. Liu, P. Wonka, MIQP-based layout design for building interiors, Comp. Graphics Forum 37 (2) (2018) 511–521, https://doi.org/10.1111/cgf.13380.
[43] R.M. Karp, Reducibility among combinatorial problems, Complex. Comp. Comput. (1972) 85–103, https://doi.org/10.1007/978-1-4684-2001-2_9.
[44] H. Varawalla, V. Desai, Hospital Design Guide: How to Get Started. https://www.academia.edu/4237726/Hospital_Design_Guide_How_to_get_started_Contents.
[45] World Health Organization, District Hospitals: Guidelines for Development, WHO regional publications: Western Pacific series 4, 1996, 9789290611172.
[46] V. Desai, Hospital planning and project management, Symbiosis Centre of Health Care (SCHC), https://www.scribd.com/document/152068065/Hospital-Planning-and-Project-Management (accessed July 31, 2021).
[47] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (4) (2007) 395–416, https://doi.org/10.1007/s11222-007-9033-z.
[48] P. Nourian, S. Rezvani, I.S. Sariyildiz, F.D. van der Hoeven, Spectral modelling for spatial network analysis, Proceedings of the Symposium on Simulation for Architecture and Urban Design (simAUD 2016), https://www.researchgate.net/profile/Pirouz-Nourian/publication/303944600_Spectral_Modelling_for_Spatial_Network_Analysis/links/575fdce708ae9a9c9561215c/Spectral-Modelling-for-Spatial-Network-Analysis.pdf (accessed July 31, 2021).
[49] C. Cubukcuoglu, P. Nourian, I.S. Sariyildiz, M.F. Tasgetiren, A discrete event simulation procedure for validating programs of requirements: The case of hospital space planning, SoftwareX 12 (100539) (2020) 1–8, https://doi.org/10.1016/j.softx.2020.100539.
[50] B. Medjdoub, B. Yannou, Separating topology and geometry in space planning, Comput. Aided Des. 32 (2000) 39–61, https://doi.org/10.1016/S0010-4485(99)00084-6.
[51] H. Huang, N.-C. Lin, L. Barrett, D. Springer, H.-C. Wang, M. Pomplun, L.-F. Yu, Way to Go! Automatic Optimization of Wayfinding Design, ArXiv e-prints. https://arxiv.org/abs/1706.08891, 2017 (accessed July 31, 2021).
[52] C. Calori, D. Vanden-Eynden, Signage and wayfinding design: a complete guide to creating environmental graphic design systems, John Wiley & Sons, 2015, 987-1-118-69299-8.
[53] R.P. Darken, J.L. Sibert, Wayfinding strategies and behaviors in large virtual worlds, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 1996, pp. 142–149. https://dl.acm.org/doi/fullHtml/10.1145/238386.238459 (accessed July 31, 2021).
[54] M.A. Foltz, Designing navigable information spaces, Massachusetts Institute of Technology, MS Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1998. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.6582&rep=rep1&type=pdf (accessed July 31, 2021).
[55] A. Uebele, Signage Systems & Information Graphics: A Professional Sourcebook, Thames & Hudson New York, 2007, 978-0500513798.
[56] P. Arthur, R. Passini, Wayfinding: people, signs, and architecture, McGraw-Hill, 1st edition, 1992, 978–0075510161.
[57] A.P. Payne, Understanding Change in Place: Spatial Knowledge Acquired by Visually Impaired Users Through the Change in Footpath Materials, PhD thesis, North Carolina State University, 2009. https://www.proquest.com/docview/304961418?pq-origsite=gscholar&fromopenview=true (accessed July 31, 2021).
[58] R.G. Golledge, Wayfinding Behavior: Cognitive Mapping and other Spatial Processes, JHU press, 1999, 9781421402895.
[59] R.P. Darken, B. Peterson, Spatial orientation, wayfinding, and representation, in: Handbook of Virtual Environments, 2014, 9780429163937.
[60] M. Duckham, L. Kulik, "Simplest" paths: automated route selection for navigation, Int. Conf. Spatial Info. Theory (2003) 169–185, https://doi.org/10.1007/978-3-540-39,923-0_12.
[61] C.E. Miller, A.W. Tucker, R.A. Zemlin, Integer programming formulation of traveling salesman problems, J. ACM 7 (4) (1960) 326–329, https://doi.org/10.1145/321043.321046.
[62] U. Pferschy, R. Stanëk, Generating subtour elimination constraints for the TSP from pure integer solutions, CEJOR 25 (1) (2017) 231–260, https://doi.org/10.1007/s10100-016-0437-8.
[63] A. Asef-Vaziri, M. Kazemi, Covering and connectivity constraints in loop-based formulation of material flow network design in facility layout, Eur. J. Oper. Res. 264 (3) (2018) 1033–1044, https://doi.org/10.1016/j.ejor.2017.07.019.
[64] Google OR-tools. https://developers.google.com/optimization (accessed July 25, 2021).
[65] Republic of Turkey Ministry of Health, Türkiye Sağlık Yapıları Asgari Tasarım Standartları 2010 Yılı Klavuzu. https://sbu.saglik.gov.tr/Ekutuphane/kitaplar/s.b.2010_klavuz_lowres_23092010.pdf, 2010 (accessed July 25, 2021).
[66] Permanent Modular Construction 2015 Annual Report. http://modular.org/ (accessed December 13, 2020).
[67] P. Steadman, Why are most buildings rectangular? Archit. Res. Q. 10 (2) (2006) 119–130, https://doi.org/10.1017/S1359135506000200.
[68] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, S. Angel, A Pattern Language: Towns, Buildings, Construction, Oxford University Press, New York, 1977, pp. 978–0195019193.
[69] F. Rothlauf, Optimization methods, Design of Modern Heuristics Natural Computing Series, Springer, Berlin, Heidelberg, 2011, pp. 45–102, https://doi.org/10.1007/978-3-540-72,962-4_3.