Autonomous Robots manuscript No.
(will be inserted by the editor)

Predicting the Global Structure of Indoor Environments: A Constructive Machine Learning Approach

Matteo Luperto · Francesco Amigoni

Received: date / Accepted: date

Abstract Consider a mobile robot exploring an initially unknown school building and assume that it has already discovered some corridors, classrooms, offices, and bathrooms. What can the robot infer about the presence and the locations of other classrooms and offices and, more generally, about the structure of the rest of the building? This paper presents a system that makes a step towards providing an answer to the above question. The proposed system is based on a generative model that is able to represent the topological structures and the semantic labeling schemas of buildings and to generate plausible hypotheses for unvisited portions of these environments. We represent the buildings as undirected graphs, whose nodes are rooms and edges are physical connections between them. Given an initial knowledge base of graphs, our approach, relying on constructive machine learning techniques, *segments* each graph for finding significant subgraphs and *clusters* them according to their similarity, which is measured using graph kernels. A graph representing a new building or an unvisited part of a building is eventually generated by *sampling* subgraphs from clusters and connecting them.

Keywords Semantic mapping · Constructive machine learning · Graph kernels

1 INTRODUCTION

The use of *semantic knowledge* has been shown to improve the performance of autonomous robots executing tasks such as exploration and search (Quattrini Li et al, 2016; Stachniss et al, 2006). The semantic knowledge regarding indoor environments takes usually the form of a *semantic map* that associates semantic labels to spatial portions of environments (Hemachandra et al, 2014; Mozos et al, 2005; Pronobis et al, 2010). For example, some areas can be labeled as 'rooms', while other areas could be labeled as 'corridors', according to the so-called *place classification* (Pronobis et al, 2010).

Most of the methods for place classification follow an approach that starts from the data perceived by the sensors mounted on-board mobile robots (e.g., laser range scanners and cameras), extracts some features from these data, and classifies the area from which the data have been acquired using (supervised) machine learning techniques. This approach has been demonstrated to be very effective in labeling parts of environments already visited by the robots, but usually does not address the problem on inferring new knowledge on the labels and, more generally, on the structure of *unvisited* parts of environments (apart from some remarkable examples, as that by Pronobis and Jensfelt (2012)).

Availability of semantic knowledge about unvisited portions of environments can be useful for online planning tasks, like exploration and search. Aydemir et al (2013) show how search methods could be improved using a probabilistic model of the search environment able to perform place classification and make local predictions on the neighbouring unexplored spaces. Oßwald et al (2016) show how knowing a rough topo-metric graph of the environment improves the exploration performance in an otherwise unknown environment. In (Quattrini Li et al, 2016) a correct prediction of the labeling of the unexplored parts of an environment is shown to improve the exploration performance of a team of robots.

Matteo Luperto and Francesco Amigoni are with the Artificial Intelligence and Robotics Laboratory, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy
{matteo.luperto,francesco.amigoni}@polimi.it

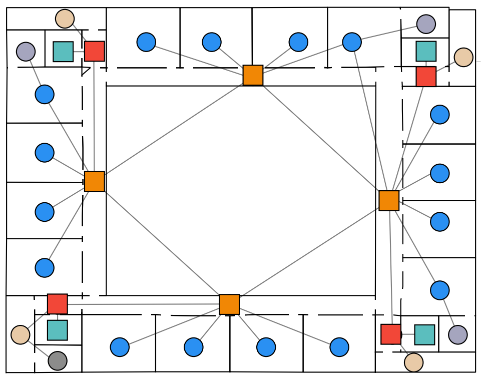
Finally, Perea Strom et al (2015) show that a prediction of the structure of the environment based on previously acquired maps can be used for better exploring challenging and repetitive environments by facilitating loop closures.

While the above studies show that an (approximate) semantic knowledge on the unexplored space is useful for many online planning applications, methods for obtaining such knowledge are still largely missing. The approach proposed by Pronobis and Jensfelt (2012) and by Aydemir et al (2013) consists in performing *local* predictions, i.e., probabilistic predictions of the labels of the unvisited rooms *directly connected* to the places already visited (and already semantically labeled) by a robot. The approach of Perea Strom et al (2015), instead, predicts the structure of unknown parts of large environments by matching the explored part with previously acquired maps of similar environments. Although the predictions of this last method are based on knowledge of the global map, they do not refer to semantic labels, but to the topo-metric structure of the environment, expressed as a Voronoi graph.

In this paper, we propose a framework for obtaining semantic knowledge on the *global* structure of a previously unknown (or partially visited) indoor environment (building). We propose to use a generative method that, following the general pattern of *Constructive Machine Learning* (CML) (Costa, 2017) is able, starting from a representative set of buildings, to model their topological structures and labeling schemas and to generate hypotheses for the structure and the labels of rooms for unknown (or partially observed) buildings. In CML, the ultimate goal of learning is not to find good models of the data, but instead to find particular instances of the domain which are likely to exhibit some desired properties of the training data, while selectively sampling from an infinite or exponentially large domain. Although our approach may not find an *exact* prediction of the structure of an unknown environment, it is nevertheless able to generate possibilities that capture some of its fundamental structural properties (that can be useful to perform several tasks, as discussed above).

Our generative model is based on a graph representation of indoor environments. More precisely, a (floor of a) building is represented as an undirected graph, whose nodes represent labeled rooms and edges represent direct physical connections between two rooms. Labels of rooms can be, for instance, ‘office’, ‘kitchen’, or ‘corridor’. Two examples of such a graph can be seen in Fig. 1, where different colors indicate different labels.

Given an initial set of buildings, our approach segments the graphs representing these buildings for finding significant subgraphs, which are then clustered according to their similarity. Similarity between graphs is calculated using *graph kernels*.



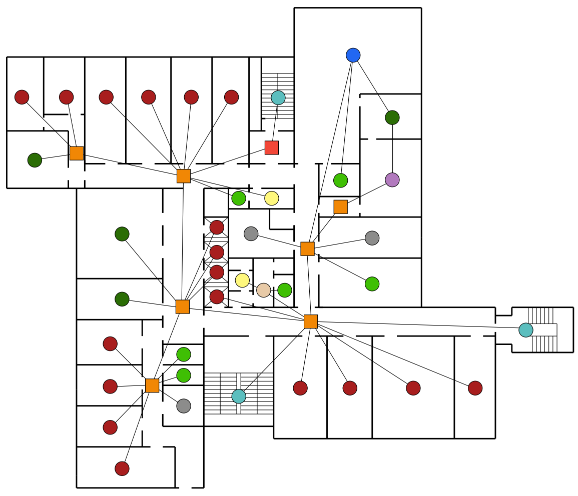


Fig. 1 Graphs representing the topological structures of a school (a) and an office (b) buildings superimposed to their floorplans. Different colors indicate different types of rooms.

In this paper, we use two graph kernels, namely the *Weisfeiler-Lehman Subtree Kernel* (K_{WL}) (Shervashidze et al, 2011) and the *Graph Hopper Kernel* (K_{GH}) (Feragen et al, 2013). A general introduction to kernels for structured data is given in the Appendix, where both K_{WL} and K_{GH} are described.

Finally, a new graph representing a predicted building (or a predicted part of a building) is generated by sampling subgraphs from clusters and connecting them. The generated graph has similar topological structure and semantic labeling schema as the graphs representing the initial buildings and in this sense we say that it represents a plausible hypothesis for (a part of) an unknown building.

Our approach distinctively focuses on *building types*, namely on specific classes of buildings that share the same functions and, consequently, the same structures, like school or office buildings (Luperto and Amigoni, 2014; Luperto et al, 2013). Moreover, our approach, similarly to that by Aydemir et al (2012), can be thought as a move from the mainstream room-level perspective, modeling the semantic relations between perceived features and rooms, to a building-level perspective, modeling the connections between rooms and the structure of buildings. Our approach does not directly rely on sensor data acquired by robots, but on a knowledge base of graph-based topological structures and labeling schemas of buildings that can be obtained from previous semantic mapping efforts or, as in our case, from other sources (like collections of blueprints). It thus relies on *a priori* knowledge rather than on incrementally-acquired sensor-based knowledge.

This paper contributes a significant extension of our previously published system (Luperto et al, 2015). In particular, although the overall architecture of the system is similar, the implementation of the steps is completely different. Specifically, here we employ a novel Monte Carlo Markov Chain sampling mechanism and, more importantly, we base our learning method on a substantial use of graph kernels. Graph kernels allow us to define a concept of similarity between two graphs that is invariant with respect to the graph layout and to the node ordering, thus performing more consistently both in graph clustering and graph sampling. This constitutes an improvement of the approach of (Luperto et al, 2015), where graph similarity is computed using the median graph, which depends on the node ordering.

The structure of the paper is the following. In Section 2 we overview related work in the field of (indoor) semantic mapping. In Section 3 we introduce the main assumptions and the goal of our approach. Section 4 describes how our model of buildings is extracted from data. Section 5 presents our sampling mechanism, which employs the model to generate new data. Experimental results are reported in Section 6. Section 7 shows a possible use of our approach. Section 8 concludes the paper.

2 RELATED WORK

As discussed before, the mainstream approach to semantic mapping involves the use of (supervised) learning methods to classify data coming from sensors mounted on robots, like laser range scanners and cameras. The classifiers are trained on data acquired by robot sensors in previous runs and already labelled. This section surveys a significant (although not exhaustive) sample of such semantic mapping systems presented in the literature. We discuss also the most interesting extensions toward the prediction of the semantic knowledge about unvisited parts of environments.

Pronobis et al (2010) present a system which exploits multiple sensors (laser range scanners and cameras) in order to associate semantic labels to areas of the environment. Different features (such as SIFT or CRFH for camera images) are extracted from sensory input taken in an environment. Three labels describing the perceived environment are then obtained independently classifying the different features describing the input data. A final label is eventually obtained by combining the three labels with a multi-modal approach.

In (Zender et al, 2008), a general framework which describes the characteristics of a semantic mapping robot is presented. The main contribution of the paper is a multi-layered spatial representation of the environment, in which different input data and classifiers can be easily integrated. A similar system, which builds a hierarchical semantic map from laser range scanner and camera data, is presented by Galindo et al (2005).

The system in (Mozos et al, 2005) classifies single laser range scans as belonging to rooms, corridors, hallways, or doorways using AdaBoost. A similar approach, which uses also features extracted from cameras, is reported by Mozos et al (2007).

Sjoo (2012) proposes a system for the automatic segmentation of two-dimensional indoor metric maps into semantic units, evaluating spatial functions based on features, such as connectivity (number of paths between rooms which cross a certain area) and functional properties (e.g., the room is a work place). These spatial functions are represented as energy functions, which are evaluated over the points of the metric map. A semantic label is then assigned to each point according to the energy functions. A refinement step adjusts the semantic labels by evaluating their spatial patterns.

A feature common to all the above studies is that the semantic maps can be used to increase the awareness of the robot only with respect to the portions of the environment already known (visited). This is due to the fact that those methods assign labels on the basis of sensor readings acquired in the visited environment. No knowledge can be obtained about the unknown (unvisited) parts of the environment.

In the last years, some proposals that deal with this limitation have been made.

For example, the system presented by Pronobis and Jensfelt (2012) is a probabilistic semantic mapping framework that applies semantic labels according to six kinds of environmental features (objects, doorways, room shapes, room size, appearance, and associated spaces). The semantic map is represented as a probabilistic chain graph model, a generalization of Bayesian networks and Markov random fields. The structure is adapted at run-time according to the perceived data. The use of a chain graph allows to consider the uncertainties of the sensory models and to estimate the unknown spatial environment. More specifically, the chain graph can predict the existence of a feature of a certain category (like a room and its label) in the unexplored space and can extend the semantic map accordingly.

Another approach, proposed by Hemachandra et al (2014), integrates metric, topological, and semantic representations with information derived from natural language descriptors obtained from human users. This is done using a factor graph formulation of the semantic properties and inferring these properties by combining natural language descriptions and image- and laser-based scene classifications. The result is a method that, for instance, from the sentence "the exit is next to the cafeteria down the hall" can infer the existence and location of an exit and a cafeteria.

In (Aydemir et al, 2012), a rather different approach is introduced, focused entirely on predicting the structure of an environment. The authors consider a knowledge base of $38,000$ rooms (representing the MIT and KTH university campuses). Each floor of the buildings is represented as a graph, where nodes are rooms labeled according to their functions (classrooms, dormrooms, offices, ...). The most frequent patterns of rooms are then extracted using gSpan, a well-known algorithm for finding the most frequent subgraphs in a set of graphs. Using this information and given a partial map of the known portion of an area, the method predicts both the topology and the labels of unvisited rooms by identifying the most common subgraphs that partially overlap the partial map.

While Aydemir et al (2012) make predictions that consist in identifying the subgraphs (portions of environment) in the knowledge base that match the part of the environment already visited, our approach generates hypotheses about the whole structure of a building, which are not necessarily limited to parts present in the initial knowledge base. We will discuss in more detail this point in Section 7.

In this paper, we describe a framework to learn the underlying model that characterizes a set of graphs representing floorplans of buildings. The resulting generative model is used to sample new graph instances. A similar task, also following the Constructive Machine Learning paradigm, is addressed in a recent paper by Costa (2017). The author develops a graph grammar to learn a distribution for Monte Carlo Markov chain sampling in a data-driven fashion. Probability estimators are implemented using one-class Support Vector Machines that classify graphs based on graph kernels. Differently from our approach, Costa (2017) uses graph grammars to propose the edit operations which result in the sampling of new graphs, while our method exploits task-dependent knowledge on the role and the label of each node in the graph for guiding the sampling.

3 OUR APPROACH

Our input representation of a floor of a building is an undirected planar graph $G = (N, E)$, where each node $n \in N$ is a room and each edge $e = (n, n') \in E$ represents a physical connection between two rooms $n, n' \in N$ (e.g., a doorway). A semantic label taken from a finite set of labels $\mathcal{L}$ is assigned to each node. Such graph G is called a *semantic map*. (See Table 1 for a summary of the notation used in this paper.)

As introduced and motivated in (Luperto and Amigoni, 2014; Luperto et al, 2013, 2015), we exploit the concept of *building type*, developing specific models for each building type. Classes of buildings that have the same function, and thus similar structures, are called building types (e.g., schools, offices, ...). Reasoning on buildings of the same type allows us to identify their common characteristics, that can be modeled effectively. We use data sets of semantic maps that we created by hand from labelled floorplans of real buildings (see (Luperto et al, 2013, 2015)) and consider the building types SCHOOL and OFFICE.

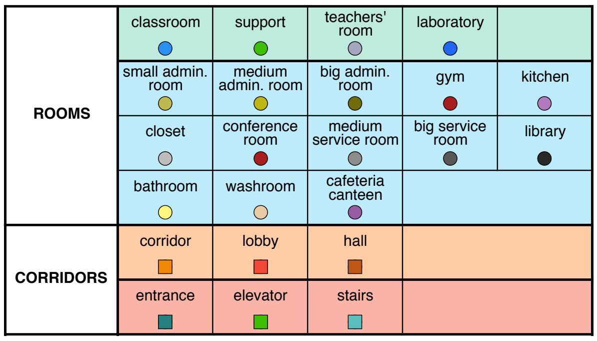
Specifically, we start from a graph database $\mathcal{G}$, which is either $\mathcal{G}_S$ or $\mathcal{G}_O$, which are composed of floorplans of 50 real schools and of 40 real office buildings, amounting to about 1800 and 1600 rooms, respectively.

The problem that we address in this paper is: given $\mathcal{G}$ representing floorplans of a given building type and a partial semantic map (graph) $\bar{G}$ describing the visited part of an environment (possibly empty), build a predicted semantic map $\hat{G}$ (or, more generally, a set of predicted semantic maps) that is maximally similar to graphs of $\mathcal{G}$ and contains $\bar{G}$ as a subgraph.

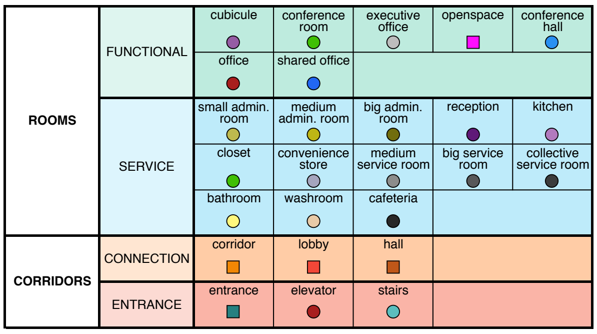
Our method builds a model of environments in $\mathcal{G}$ according to two main steps: *segmentation*, where each graph $G \in \mathcal{G}$ is segmented in a set of smaller subgraphs, and *clustering*, where similar subgraphs belonging to different graphs are grouped together according to a similarity measure (see Section 4). Once a model of graphs in $\mathcal{G}$ is available, a *sampling* process generates $\hat{G}$ (see Section 5).

symbols	meaning
$G = (N, E)$	graph representing a semantic map with rooms $n \in N$ and doorways $e = (n, n') \in E$
$\mathcal{G}$ ($\mathcal{G}_S$, $\mathcal{G}_O$)	set of semantic maps G (representing SCHOOLs and OFFICEs, respectively)
$\mathbb{G}$	set of all the possible graphs
$\bar{G}$	partially known semantic map
$\hat{G}$	predicted semantic map
$\hat{\mathcal{G}}$ ($\hat{\mathcal{G}}_\theta$)	set of predicted graphs $\hat{G}$ (according to a parameterization θ)
$\mathcal{L}$ ($\mathcal{L}_{R/C}$, $\mathcal{L}_{\text{school}}$, $\mathcal{L}_{\text{office}}$)	set of semantic labels (ROOM/CORRIDOR and labels for SCHOOLs and OFFICEs, respectively)
$S = \{\dots, s_i, \dots\}$	set of subgraphs of a semantic map G
$s_i = (N_i, E_i)$	subgraph of a semantic map G with rooms N_i and doorways E_i
$\mathcal{S}$	set of all subgraphs obtained from all the graphs in $\mathcal{G}$
$c \in E$	cut edges for G
$\mathbf{C}$	set of cut edges for all $G \in \mathcal{G}$
$H = (S, U)$	functional graph with functional areas S and their connections U
$\mathcal{H}$	set of functional graphs
$\hat{H} = (\hat{S}, \hat{U})$	predicted functional graph
$\mathcal{C} = \{\dots, C_j, \dots\}$	set of clusters obtained from $\mathcal{S}$
$\mathcal{M} = (\mathcal{G}, \mathcal{H}, \mathcal{C})$	model of buildings in $\mathcal{G}$
$V_c^G = \{\dots, v_j^G, \dots\}$	ordered set of numbers v_j^G of subgraphs of G belonging to clusters C_j
$\hat{V}_c = \{\dots, \hat{v}_j, \dots\}$	ordered set of numbers $\hat{v}_j$ of subgraphs of $\hat{G}$ belonging to clusters C_j
$\mathbf{S}_{\hat{u}}$	set of all the pairs of nodes n of $\hat{s_i}$ and n' of $\hat{s_j}$, where $\hat{u} = (\hat{s_i}, \hat{s_j})$
$\mathbf{Q}$	set of 1-neighborhood subgraphs of all $c \in \mathbf{C}$
$\bar{s_1}, \bar{s_2}, \dots$	known subgraphs of $\bar{G}$
$\bar{C_1}, \bar{C_2}, \dots$	clusters of $\bar{s_1}, \bar{s_2}, \dots$

Table 1 Meaning of the symbols denoting the main entities that are used in this paper.



(a) SCHOOL.



(b) OFFICE.

Fig. 2 Labeling schemas for SCHOOL (a) and OFFICE (b) building types. Colors and shapes associated to labels are used in the semantic maps shown through the paper.

3.1 Labeling Schema

In this paper, we use a hierarchical semantic labeling schema to have two different points of view, at two different levels of abstraction, on the semantics of the environment. At the top level, the labeling schema called $\mathcal{L}_{R/C}$ contains only two general categories:

- ROOM: a space in which an activity is performed;
- CORRIDOR: a space used to connect other spaces together.

The categorization between rooms and corridors is particularly important in our context since graphs are highly disassortative when considering the $\mathcal{L}_{R/C}$ schema (i.e., ROOMs are mostly connected to CORRIDORs). This results in CORRIDOR nodes having significantly larger degrees than ROOM nodes and suggests the use of CORRIDORS as the key nodes of our reasoning mechanism. For clarity, in the following figures, a square is used for CORRIDORS, while a circle is used for ROOMS, as in Fig. 1.

This general labeling schema is shared by all building types and constitutes the basis for specific building type labeling schemas $\mathcal{L}_{\text{type}}$. For the SCHOOL building type, we consider that a node ROOM is further specialized by labels such as CLASSROOM, LABORATORY, TEACHERS' ROOM, or by labels as ADMINISTRATIVE OFFICES, CANTEENS, BATHROOMS, ... For the OFFICE building type, we consider that a node ROOM is further specialized by labels such as OFFICE, CONFERENCE

ROOM, or RECEPTION. The complete set of labels relative to the SCHOOL building type ($\mathcal{L}_{\text{school}}$) and to the OFFICE building type ($\mathcal{L}_{\text{office}}$) have been extracted from architectural sources (Neufert and Neufert, 2012; The Whole Building Design Guide, 2015) and are reported in Fig. 2.

4 MODEL BUILDING

4.1 Segmentation

Each graph $G = (N, E) \in \mathcal{G}$ is segmented in smaller, non-overlapping subgraphs $S = \{s_1, s_2, \dots\}$, such that, for each pair $s_i, s_j \in S$ (with $s_i = (N_i, E_i)$ and $s_j = (N_j, E_j)$), it holds $N_i \cap N_j = \emptyset$ and such that $\bigcup_{s_i \in S} N_i = N$. Segmentation is performed by removing some edges, or cuts, $c \in E$ from G and aims at identifying the *functional areas*. Functional areas are groups of neighbouring rooms with a similar function within the buildings (i.e., a subgraph), which are connected to (nearby) corridors (e.g., a cluster of offices, an administrative section, a group of service rooms such as bathroom, kitchen, and vending machines).

An example of functional areas can be seen in Fig. 3. Fig. 3a displays a research office building (which is the same of Fig. 1b). We can notice that rooms with a similar function are usually close to each other and connected to the same corridor (in green). It is thus possible to separate the floor plan into functional areas, shown with different colors in Fig. 3b.

In our example, we highlight in red the parts where offices are located, in blue the parts with research labs, and in orange the parts where support rooms, such as stairs, vending machines, elevators, or bathrooms, are located. This representation can be summarized in the graph structure of Fig. 3c. Differently from Fig. 1b, in Fig. 3c each node of the *functional graph* represents a *functional area* (namely a set of connected rooms that share the same function).

The identification of functional areas introduces a new abstract perspective that better represents the structure of the buildings. For instance, the functional graph of Fig. 3c shows that research labs (blue node) are connected to offices (red nodes) only through service rooms (orange nodes), regardless of the specific rooms that are connected. In our approach, we work with a two-layered representation composed of semantic maps (like that of Fig. 1b) and functional graphs (like that of Fig. 3c).

The segmentation step is thus particularly important in our framework since it should divide each floor of a building according to the structure of the graph and the function of the rooms.

Two different unsupervised methods are used for segmentation. The first method is the Normalized Cut spectral method of Shi and Malik (2000), or `ncut`, as we used in (Luperto et al, 2015). `ncut` is a general and context-independent segmentation method that partitions a graph into components of roughly the same size. Specifically, `ncut` recursively partitions a graph G into two components $s_a = (N_a, E_a)$ and $s_b = (N_b, E_b)$ based on the magnitude of the eigenvalues of the Fiedler vector obtained after an *ad hoc* eigen-decomposition of the graph adjacency matrix. Each part is then recursively segmented until a threshold on the *normalized-cut* measure for each partition is reached. The normalized-cut measure is computed as:

$$\textit{normalized-cut}(s_a, s_b) = \frac{Cut(s_a, s_b)}{Assoc(s_a, G)} + \frac{Cut(s_a, s_b)}{Assoc(s_b, G)},$$

where $w(u, v)$ is the weight of the edge between the nodes u and v (in our setting, we use unweighted edges and so $w(u, v) = 1$ if u and v are connected by an edge and $w(u, v) = 0$ if u and v are not connected by any edge). $Cut(s_a, s_b)$ is defined as:

$$Cut(s_a, s_b) = \sum_{u \in N_a, v \in N_b} w(u, v)$$

and $Assoc(s_a, G)$ as:

$$Assoc(s_a, G) = \sum_{u \in N_a, t \in N} w(u, t).$$

Similarly for $Assoc(s_b, G)$.

The second method (that we refer to as `corr`) depends on our particular domain and is based on the distinction between ROOMs and CORRIDORs of $\mathcal{L}_{R/C}$. It follows the intuition that CORRIDORs can be considered as the hubs of the graphs. Under these premises, `corr` assigns each ROOM node to the nearest CORRIDOR node (if there is a tie, the CORRIDOR with the highest degree is selected). Each edge between two CORRIDOR nodes or between two ROOM nodes associated with two different CORRIDORs is considered as a cut c and removed by the segmentation process. This second segmentation method, unlike `ncut`, does not guarantee to produce a balanced segmentation of the graph; however, in our problem domain, both segmentation methods perform similarly in terms of number and size of subgraphs obtained after the segmentation, as shown in the example of Fig. 4.

The results of the segmentation process performed over a graph (semantic map) G can be represented as a graph $H = (S, U)$, where every node in S is a subgraph $s \in S$ and an edge $u \in U$ exists between two nodes

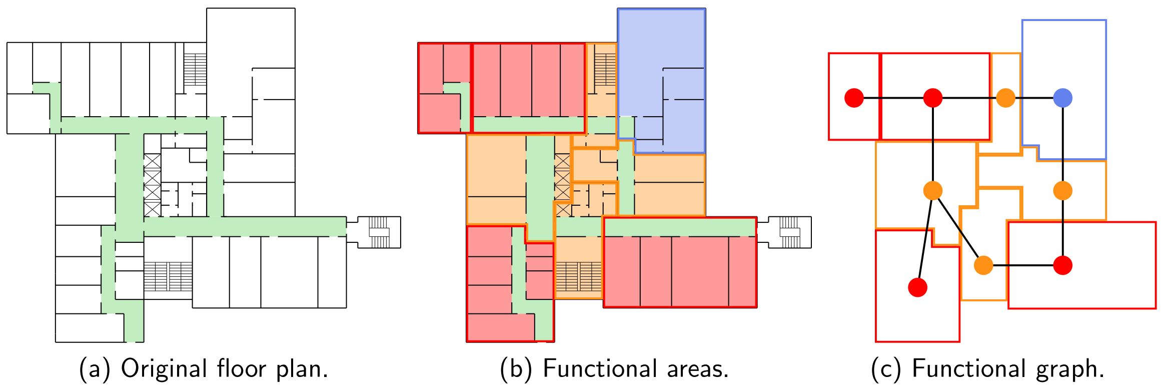

(a) Original floor plan. (b) Functional areas. (c) Functional graph.

Fig. 3 Functional analysis of a research office building floor plan. In Fig. 3a the corridors are highlighted in green. In Fig. 3b the floor plan is segmented in 9 functional parts, one for each corridor. Similar rooms are grouped together with the corridors to which they are connected to. Each part is a *functional area* and is colored according to is function: red means office, blue means research labs, orange means service rooms such as stairs, elevators, bathrooms, storage rooms, and meeting rooms. From such a segmentation, an abstract graph representation in the form of a *functional graph* is derived in Fig. 3c.

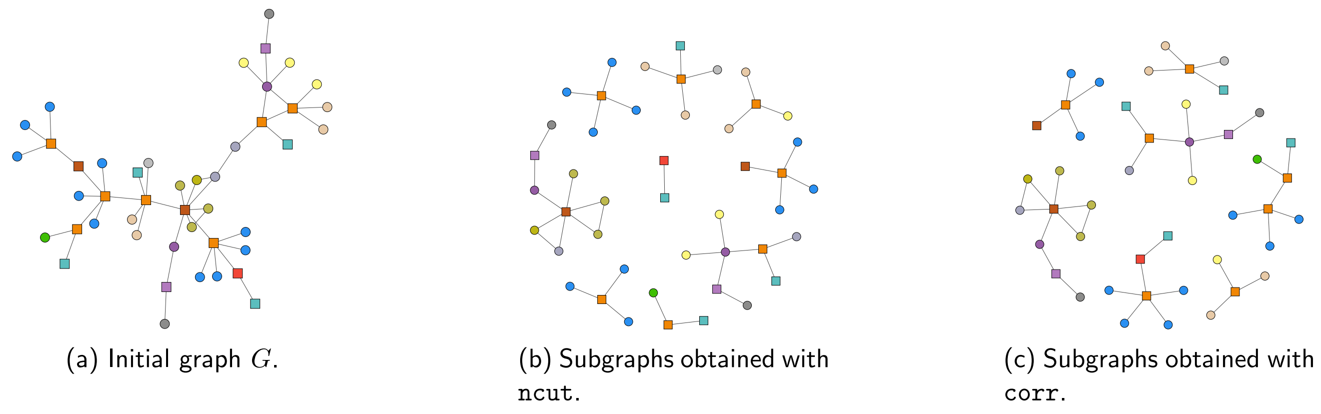

(a) Initial graph G. (b) Subgraphs obtained with `ncut`. (c) Subgraphs obtained with `corr`.

Fig. 4 An example of applications of different segmentation methods to a graph G.

$s_i, s_j \in S$ (where $s_i = (N_i, E_i)$ and $s_j = (N_j, E_j)$) if there is a cut $c = (n_i, n_j) \in E$ between two nodes $n_i \in N_i$ and $n_j \in N_j$. We refer to the graph H as *functional graph.* A pair composed of a $G \in \mathcal{G}$ and of the associated $H \in \mathcal{H}$, where $\mathcal{H}$ is the set of all functional graphs, constitutes a representation of the same building at two different levels of abstraction.

4.2 Clustering

After all graphs $G \in \mathcal{G}$ have been segmented, the *clustering* step is responsible for grouping together all the subgraphs that have a similar structure, and thus to identify the same functional areas over several buildings. All the subgraphs S_ℓ obtained from all the graphs G_ℓ in $\mathcal{G}$ are now considered altogether: $\mathcal{S} = S_1 \cup S_2 \cup \cdots \cup S_{|\mathcal{G}|}$. For each pair of subgraphs s_i, s_j an affinity value $\phi(s_i, s_j)$ is computed. Affinity $\phi(s_i, s_j)$ measures the similarity between subgraphs and is computed using one of the graph kernels K (K_{WL} or K_{GH}) described in the Appendix, exploiting the fact that each subgraph $s_i = (N_i, E_i)$ is actually a graph in which each node $n \in N_i$ is described by its label in $\mathcal{L}_{\text{school}}$ or in $\mathcal{L}_{\text{office}}$ (and its correspondent label in $\mathcal{L}_{R/C}$):

$$\phi(s_i, s_j) = K(s_i, s_j),$$

so ϕ depends on both the topological structure of s_i and s_j and on the labels of their nodes.

In order to cluster together similar subgraphs, we use the *affinity propagation* clustering algorithm (Frey and Dueck, 2007). Affinity propagation is a message passing clustering algorithm which selects a subset of the data as models for their relevance. Each data element exchanges two kinds of messages with other data elements indicating how much it is suited to became a model for the other elements and how much each element is considered suited to be a model by the other nearby elements. These messages are exchanged recursively until stable clusters are formed or a maximum number of iterations is reached. Affinity propagation does not require the number of clusters as a parameter. Affinity propagation uses a damping factor α between 0 and 1 for smoothing the message update, in order to prevent oscillations and increase convergence rate to stable clusters. The use of affinity propagation clustering is a difference from our previous work (Luperto et al, 2015) where clustering is performed using Normalized Cut `ncut` (Shi and Malik, 2000), which sometimes results in a high number of small clusters and/or in a big cluster containing almost all of the data and which is experimentally outperformed by affinity propagation (data are not reported here).

The clustering step outputs a set of clusters $\mathcal{C} = \{C_1, C_2, \ldots\}$ and a function $\psi : \mathcal{S} \to \mathcal{C}$ that maps each subgraph to its cluster. Given a subgraph s, we call $C = \psi(s)$ the cluster to which s belongs. A cluster C is expected to represent a functional area and the subgraphs in the cluster are parts of (possibly different) buildings that share the similar structure and labels. The clustering results are influenced by the way in which the segmentation is performed, by the cluster algorithm, and by the graph kernel used.

The semantic maps, the functional graphs, and the clusters constitute the model $\mathcal{M} = \{\mathcal{G}, \mathcal{H}, \mathcal{C}\}$ for a specific building type constructed starting from $\mathcal{G}$.

4.3 Segmentation and Cluster Evaluation

Our segmentation and clustering steps are implemented in MATLAB R2015b. We present here a quantitative evaluation of the models built by our approach. Results are relative to $\mathcal{G}_S$ (schools), results for $\mathcal{G}_O$ are qualitatively similar and are omitted for brevity. Within our context, we consider as desirable the presence of the following features: a high intra-cluster similarity, a low inter-cluster similarity, a limited number of clusters, a number of clusters that is stable with respect to the increase of the number of graphs in the dataset $\mathcal{G}$, and a similar size for each cluster (i.e., the absence of many small clusters with less than 5 subgraphs).

We empirically find that a damping factor of $\alpha = 0.8$ for affinity propagation produces clusters with all the desired properties for our data sets. Remember that affinity ϕ is calculated according to K_{WL} or K_{GH}. Clusters obtained using K_{GH} present better features than those obtained with K_{WL}, but at the expense of a longer computing time. K_{GH} features, computed on shortest paths, capture the structure of the graphs of our domain better than the high-dimensional sparse features computed on subtrees by K_{WL}. Using K_{WL}, we obtain a higher number of clusters (27 and 12 for K_{WL} and K_{GH}, respectively, using 50 graphs in $\mathcal{G}_S$ that are segmented by `corr` into 287 subgraphs) with highly different numbers of subgraphs per cluster. On a i7QuadCore Intel 820M 16GB computer, the segmentation and clustering steps take less than one minute using K_{GH} and K_{WL} on all the 50 graphs of $\mathcal{G}_S$. Note that clustering is performed only once and then clusters are stored in the model $\mathcal{M}$ for being sampled later. From now on, we consider only K_{GH}, which performs better than K_{WL}.

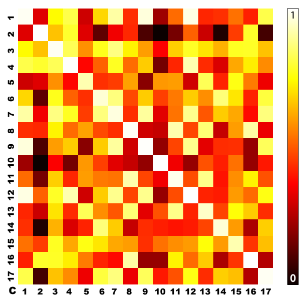
Both segmentation methods `ncut` and `corr` perform correctly for the task, producing a partition of each graph $G \in \mathcal{G}$ in components of approximatively the same size, by removing a limited number of edges. However, different segmentation methods result in slightly different clusters. `ncut` segments 50 graphs into 207 subgraphs, which are then clustered into 17 clusters. `corr` results in a higher number of subgraphs (287) but in a lower number of clusters (12).

Further results are shown in Fig. 5. Fig. 5a and Fig. 5c show intra-cluster similarity and inter-cluster similarity for clusters obtained after segmentation performed with `ncut` and `corr`, respectively. Figures show *heat maps*. A heat map is a $|\mathcal{C}| \times |\mathcal{C}|$ matrix where cell (i, j) is light (dark) if subgraphs in C_i are similar to (different from) subgraphs in C_j. Similarities between subgraphs are computed using graph kernels. High intra-cluster similarity results into light diagonal in the map, where low inter-cluster similarity results in dark cells outside the diagonal. Fig. 5b and Fig. 5d show the number of subgraphs in each cluster for `ncut` and `corr`, respectively. Each column indicates a separate cluster and its height represents the corresponding number of subgraphs.

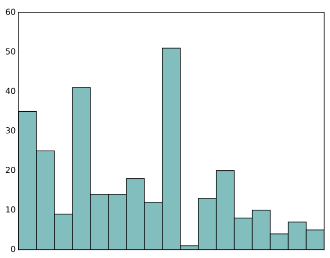
Although `corr` produces a smaller number of clusters than `ncut`, their performance reported in Fig. 5 is similar both in terms of the intra-cluster and inter-cluster similarities (heat maps) and in terms of the number of subgraphs in each cluster (bar charts). This is why the two segmentation methods are re-evaluated in Section 6, according to their impact on sampling. Note, however, that differently from `corr`, `ncut` guarantees by design that all subgraphs have a similar size, i.e., that there are no subgraphs containing only one or two nodes or, on the other side, subgraphs containing most part of the graph. Actually, using `corr`, we obtain such undesirable graph partitioning only in very few cases (see also the example of Fig. 4).

5 Sampling from the model

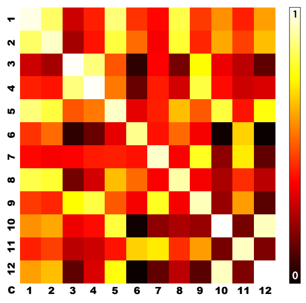
The model $\mathcal{M} = \{\mathcal{G}, \mathcal{H}, \mathcal{C}\}$ obtained in the previous section can be used for generating new graph samples representing buildings or their parts. Formally, we assume that all graphs G in our dataset $\mathcal{G}$ follow the same (unknown) graph probability distribution P that is shared



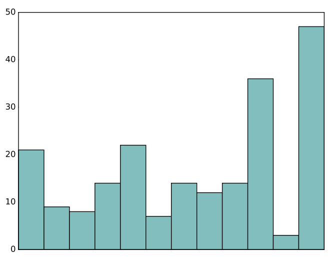
(a) Heat map using `ncut`.



(b) Number of subgraphs in clusters using `ncut`.



(c) Heat map using `corr`.



(d) Number of subgraphs in clusters using `corr`.

Fig. 5 Results of the clustering process for two different segmentation algorithms. The color of a cell (i, j) in the heat map represents if subgraphs in the cluster C_i are, on average, similar to (light color) or different from (dark color) the subgraphs in C_j (cluster numbers are reported on the axes). Bar charts have a column for every cluster in $\mathcal{C}$. Heights of the columns correspond to the number of subgraphs belonging to the clusters. The results are obtained on $\mathcal{G}_S$.

by all the buildings of the same type. This is reasonable since $\mathcal{G}$ refers to buildings of a single building type ($\mathcal{G}_S$ or $\mathcal{G}_O$ in our experiments). Sampling a new graph from an unknown graph distribution P is a Constructive Learning Problem (CLP) (Costa and De Grave, 2010), which can be seen as the task of sampling a graph from an empirical conditional probability distribution using an adaptive data-driven procedure. The model $\mathcal{M}$ is thus used to obtain an empirical probability distribution (which approximates P) for the set of graphs $\mathcal{G}$ using a hierarchical sampling process. The empirical distribution is modeled by a Markov Chain (MC) and the sampling process exploits a Monte Carlo Markov Chain (MCMC) (Koller and Friedman, 2009). In order to sample a new graph $\hat{G}$, the sampling process starts from generating its abstract structure, namely its functional graph $\hat{H}$, adding incrementally more details, following, in the reverse order, the segmentation and clustering processes of Section 4. Sampling is composed of a sequence of four steps δ, each one modeling an empirical probability distribution P_δ by using a MCMC method.

In this section, first, a more formal definition of the sampling problem is given. Then, we introduce the MCMC method that approximates the empirical probability distributions P_δ for the sampling steps. Finally, we describe each sampling step.

5.1 Generating Graphs from Empirical Distributions

Given a general domain of graphs $\mathbb{G}$, in this section we define the task of sampling a set of graphs $\hat{\mathcal{G}} \subseteq \mathbb{G}$ using an adaptive data-driven procedure. Let P be the (usually unknown) probability distribution associated to $\mathbb{G}$. Let P_θ be an approximation of P parametrized by θ. Following an approach similar to that by Costa (2017), define L as a non-negative convex function that, given two probability distributions over $\mathbb{G}$, returns zero if the two distributions are identical and positive values if the two distributions are different (the larger the values the more different the distributions). We can define an instance of the function L as:

$$L(P, P_\theta) = D_{KL}(P \| P_\theta) + D_{KL}(P_\theta \| P),$$

where D_{KL} is the Kullback-Leibler divergence:

$$D_{KL}(P \| P_\theta) = \sum_{G \in \mathbb{G}} P(G) \log \frac{P(G)}{P_\theta}.$$

(Similarly for $D_{KL}(P_\theta \| P)$.)

The sampling problem, following the Constructive Machine Learning paradigm, is to find a parametrization θ^* that results from the following optimization:

$$\theta^* = \arg\min_{\theta} \; L(P, P_\theta). \tag{1}$$

Since P and P_θ are both unknown and $\mathbb{G}$ has a possibly infinite cardinality, we consider a finite set of examples $\mathcal{G} \subseteq \mathbb{G}$, that are drawn according a probability distribution $P_{\mathcal{G}} \sim P$. Let $\hat{\mathcal{G}_\theta}$ be a sample set of graphs randomly generated according to P_θ and let $f_{\mathcal{G}}$ (respectively, $f_{\hat{\mathcal{G}_\theta}}$) be the probability distribution estimator for the samples in $\mathcal{G}$ (respectively, $\hat{\mathcal{G}_\theta}$); we can assume:

$$f_{\mathcal{G}} \sim P_{\mathcal{G}} \sim P$$
$$f_{\hat{\mathcal{G}_\theta}} \sim P_\theta.$$

Thus, we can approximate (1) with:

$$\theta^* = \arg\min_{\theta} L(f_{\mathcal{G}}, f_{\hat{\mathcal{G}_\theta}}), \tag{2}$$

and (2) returns the solution of our constructive learning problem.

The parametrization θ^* can be used to model P_{θ^*}, in order to sample from it a new set of graphs $\hat{\mathcal{G}}$. Since P_{θ^*} is unknown, in the following section we describe our proposed procedure to obtain f_{θ^*} starting from an initial set of data and we show how it can be used for sampling new data.

5.2 Monte Carlo Markov Chains

In our particular context, the set of graphs $\mathcal{G} \subseteq \mathbb{G}$ represent floors of buildings. In order to drawn samples from a distribution that approximates P we use a Metropolis Hastings algorithm, a particular type of MCMC method (Koller and Friedman, 2009). This section provides a general description of the method, which is applied, with different parameters, in the four steps of our sampling method. The detailed implementation of the method for each step is described in the following.

Generally speaking, the Metropolis Hastings algorithm is defined by a transition model $\mathcal{T}(g \to g')$ that specifies, for each pair of states (graphs, in our case) g, $g' \in \mathbb{G}$ the probability of going from g to g' according to a stationary distribution Π (note that, when graphs in $\mathbb{G}$ are intended as states for MCMC we use the lowercase symbol g):

$$\Pi(G = g') = \sum_{g \in \mathbb{G}} \Pi(G = g)\mathcal{T}(g \to g').$$

The transition model $\mathcal{T}$ is defined in terms of *transition kernel* T, that generates, using some edit function, successors of a state g and then selects randomly the next candidate g' from these successors. We can either accept the proposal (and move to g') or reject it (and stay at g), using a transition probability $\mathcal{A}(g \to g')$. Namely:

$$\mathcal{T}(g \to g') = T(g \to g')\mathcal{A}(g \to g') \quad g \neq g'$$

$$\mathcal{T}(g \to g) = T(g \to g') + \sum_{g \neq g'} T(g \to g')(1 - A(g \to g')).$$

The acceptance probability $A(g \to g')$ reduces to:

$$A(g \to g') = \min\big(1, \frac{\Pi(g')T(g' \to g)}{\Pi(g)T(g \to g')}\big).$$

Since our proposal distributions are (in first approximation) symmetric, we have that $T(g \to g') = T(g' \to g)$.

Finally, we approximate $\Pi(g)$ with a data-driven adaptive Boltzmann-like function b:

$$\Pi(g) \sim b(g) = e^{-\alpha \Gamma(g, \mathcal{G})}, \tag{3}$$

where $\mathcal{G}$ is our set of samples, Γ is a cost function that penalizes graphs g that are different from those in $\mathcal{G}$, and α is a user-defined parameter (which is set to 1 if not indicated differently). For each step of the sampling process described in the following, a definition of T and of Γ will be provided.

The number of iterations of each MCMC is selected in order to have reasonable chances that the mixing time is reached (i.e., that the transition model reaches a stationary distribution). We use an heuristic based on the observation that different sets of samples collected at different times should, if the mixing time has been reached, show similar variance in each of the chains. More formally, we run K separate chains for $\tau + M$ steps starting from different starting points. After discarding the first τ samples from each chain, the convergence is checked (see (Koller and Friedman, 2009, p. 522-523) for details). If the mixing time has been reached, then τ is chosen as the minimum number of iterations, otherwise τ is increased by a fixed amount and the test is repeated.

5.3 Hierarchical Sampling

The hierarchical sampling process that creates a new graph $\hat{G}$ from a graph model $\mathcal{M} = (\mathcal{G}, \mathcal{H}, \mathcal{C})$ is divided in four steps:

1. cluster configuration sampling,
2. functional graph sampling,
3. subgraphs sampling,
4. node connections sampling.

A running example of these four steps is shown in Fig. 6 and used as a reference in the following.

Hierarchical sampling allows us to change the structure of a graph in a fast way during functional graph sampling (e.g., by adding or removing a subgraph or by connecting two subgraphs), in order to obtain rather different graphs in a small number of iterations. Directly sampling node labels and node connections at the level of semantic maps requires a larger number of iterations, since graph kernels compute similarities between graphs by considering the entire graph structure, and not only the single labels. Hence, adding, removing, or changing the labels of the single nodes in a semantic map result in similar graphs (according to graph kernels), thus requiring several iterations for covering the solution space. Moreover, note that worst-case computational complexity of graph kernels is strongly related to the number of nodes and edges of the graph population (see Appendix), and functional graphs have far less nodes than the graphs representing semantic maps (see also Table 4 and the relative discussion reported below).

5.3.1 Cluster configuration sampling

The first step is the cluster configuration sampling that returns the composition $\hat{V}_c$ of the sampled graph $\hat{G}$ in terms of how many subgraphs for each cluster are present in $\hat{G}$.

The cluster configuration of each original graph $G \in \mathcal{G}$ is a vector:

$$V_c^G = \{v_1^G, v_2^G, \dots, v_{|\mathcal{C}|}^G\},\ v_i^G \in \mathbb{N},$$

where v_i^G represents the number of subgraphs of G (in which G has been segmented) belonging to the cluster $C_i \in \mathcal{C}$.

Using the Metropolis Hastings algorithm, we start from a random initialization of $\hat{V}_c = \{\hat{v}_1, \hat{v}_2, \dots, \hat{v}_{|\mathcal{C}|}\}$ and employ a transition kernel T_1 where, at each step, one component of the vector $\hat{V}_c$ is selected at random and increased or decreased by one unit. Values of each $\hat{v}_i$ are bounded in the interval $[0, \max_{G\in\mathcal{G}} {v_i}^G]$.

The cost function Γ_1 is defined as:

$$\Gamma_1(\hat{V}_c) = \sum_{G\in\mathcal{G}} d_{pw}(\hat{V}_c, {V_c}^G),$$

where d_{pw} is the pairwise vector distance and V_c^G is the vector representing the graph G in $\mathcal{G}$. In (3), α is set to 0.7 in case of `ncut` segmentation and to 0.55 in the case of `corr` segmentation. After $\tau_1 = 300$ iterations the sampled vector $\hat{V}_c$ is considered as final and the sampling continues with the next step.

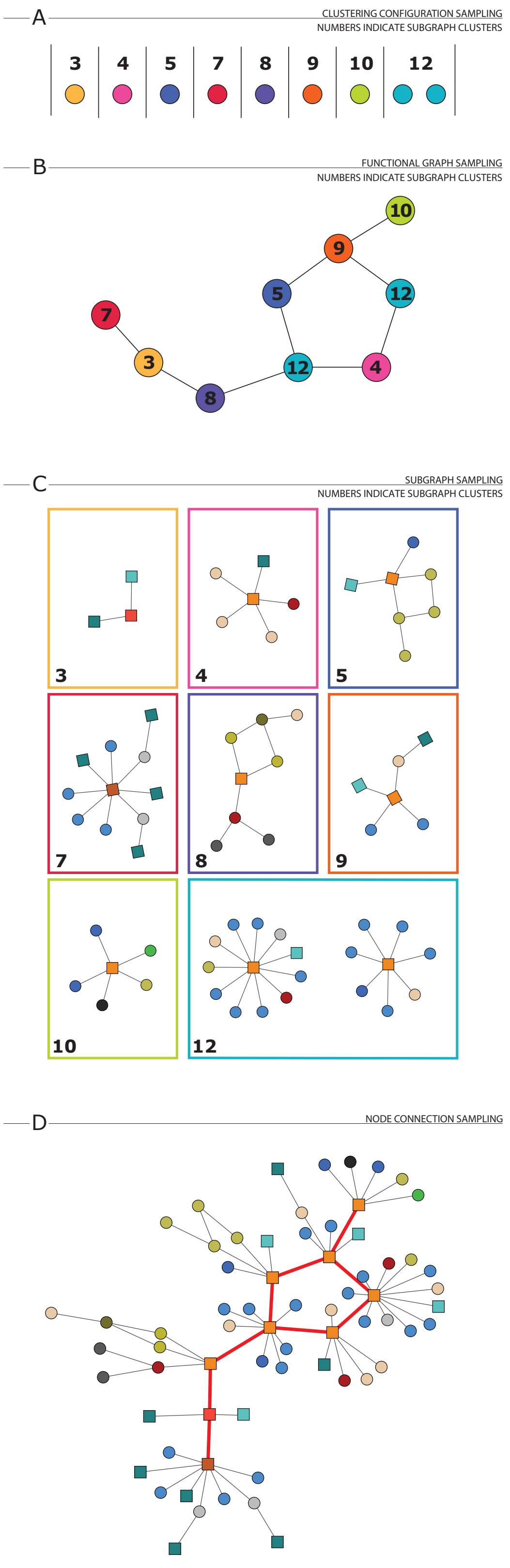

A
CLUSTERING CONFIGURATION SAMPLING
NUMBERS INDICATE SUBGRAPH CLUSTERS
3
4
5
7
8
9
10
12
B
FUNCTIONAL GRAPH SAMPLING
NUMBERS INDICATE SUBGRAPH CLUSTERS
10
9
5
12
7
3
12
4
8
C
SUBGRAPH SAMPLING
NUMBERS INDICATE SUBGRAPH CLUSTERS
3
4
5
7
8
9
10
12
D
NODE CONNECTION SAMPLING

Fig. 6 An example of the sampling steps for generating a graph $\hat{G}$. Subfigure A shows the sampled cluster configuration. Subfigure B represents the sampled functional graph. Subfigure C displays the sampled subgraphs for each cluster. Subfigure D shows the final sampled semantic map. Numbers denote clusters of subgraphs.

In the example of Fig. 6, the sampled cluster configuration is:

$$\hat{V}_c = \{0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 2\}$$

where $|\mathcal{C}| = 12$, and is graphically represented in Fig. 6A.

5.3.2 Functional graph sampling

In the second sampling step, the sampled functional graph $\hat{H} = (\hat{S}, \hat{U})$ is computed from the cluster configuration $\hat{V}_c$. $\hat{H}$ is obtained creating a node $\hat{s}_i \in \hat{S}$ for each subgraph indicated by $\hat{v}_i$ and sampling an edge $\hat{u} = (\hat{s}_i, \hat{s}_j)$ between two subgraphs $\hat{s}_i$ and $\hat{s}_j$ that will be later connected.

This step is performed using the Metropolis Hasting algorithm introduced in Section 5.2 by selecting a random initialization for the edges of $\hat{H}$. The transition kernel T_2 is symmetric and consist in three moves: ADD an edge to the graph (randomly selecting the two nodes $\hat{s}_i$ and $\hat{s}_j$ to be connected), REMOVE an existing edge (selected randomly), and SWAP a node in an edge $\hat{u}$ (randomly select $\hat{u} = (\hat{s}_i, \hat{s}_j)$ and replace $\hat{s}_j$ with a different node $\hat{s}_t$, thus transitioning from $\hat{u}$ to $\hat{u}' = (\hat{s}_i, \hat{s}_t)$). Each move is chosen using the following probabilities:

$$\begin{aligned} P_{\text{ADD}} &= 1 - e^{-\lambda_1 \frac{|\hat{S}|}{|\hat{U}|}}, \\ \epsilon &= 1 - e^{\lambda_2 \frac{|\hat{U}|}{|\hat{S}|}}, \\ P_{\text{REMOVE}} &= (1 - P_{\text{ADD}})\epsilon, \text{and} \\ P_{\text{SWAP}} &= (1 - P_{\text{ADD}})(1 - \epsilon), \end{aligned}$$

where $|\hat{S}|$ and $|\hat{U}|$ are the number of nodes and edges in $\hat{H}$, respectively, and λ_1 and λ_2 are user defined parameters that are set empirically to reduce the mixing time of the MCMC. In our experiments, $\lambda_1 = \lambda_2 = 0.1$.

A transition is either accepted or rejected by evaluating a cost function $\varGamma_2$:

$$\varGamma_2(\hat{H}) = \sum_{H_i \in \mathcal{H}} K(\hat{H}, H_i)$$

where K is a graph kernel. For this step we use K_{WL} as graph kernel, since it guarantees similar performance but at a significant lower computational cost than K_{GH}, and the cost function $\varGamma_2$ should be evaluated once for every iteration of the MCMC. The total number of iterations τ_2 for this step is selected to be a random number higher than 2000.

In the example of Fig. 6, the generated functional graph $\hat{H}$ is shown in Fig. 6B, where the numbers indicate the cluster of each subgraph (node of the functional graph).

5.3.3 Subgraphs sampling

In the third sampling step, a specific subgraph $\hat{s}_i \in C_i$ for each node of $\hat{H}$ is selected. This is done by picking with uniform probability a subgraph $\hat{s}_i$ from cluster C_i until $\hat{v}_i$ subgraphs are sampled from C_i. Note that a cluster C can contain several repetitions of the same subgraph, due to its repeated presence in the graphs $\mathcal{G}$. The subgraphs sampling step is performed atomically and does not use the MCMC framework.

In the example of Fig. 6, the sampled subgraphs are listed in Fig. 6C. Boxes are highlighted using the same colors for nodes in Figs. 6A and 6B, and the cluster number is shown at the bottom left in each box.

5.3.4 Node connections sampling

The last step is the node connections sampling, where the final graph (semantic map) $\hat{G} = (\hat{N}, \hat{E})$ is obtained from the functional graph $\hat{H} = (\hat{S}, \hat{U})$. $\hat{N}$ is the set of all the nodes $\hat{N}_i$ of the subgraphs $\hat{s}_i = (\hat{N}_i, \hat{E}_i)$ sampled in the previous step, while $\hat{E}$ is initialized with the subgraphs edges $\hat{E}_i$ (and with no edges between nodes of different subgraphs). Then, for each edge $\hat{u} = (\hat{s}_i, \hat{s}_j) \in \hat{U}$ in the functional graph $\hat{H}$, an edge $e = (\hat{n}_k, \hat{n}_l) \in \hat{E}$ between two nodes $\hat{n}_k \in \hat{s}_i$ and $\hat{n}_l \in \hat{s}_j$ should be created (here, with a slight notation overload, we use $\hat{n}_k \in \hat{s}_i$ instead of $\hat{n}_k \in \hat{N}_i$, where $\hat{s}_i = (\hat{N}_i, \hat{E}_i)$). Ideally, this step is the opposite of the graph segmentation procedure described in Section 3, and a new connection e should be added where a cut c would have been performed during segmentation.

Adding a connection e is a local edit operation on the graph $\hat{G}$ which results in a global change in the graph's layout and in its structural properties. Given a connection $\hat{u} = (\hat{s}_i, \hat{s}_j)$ between two subgraphs $\hat{s}_i$ and $\hat{s}_j$ defined in the functional graph, call $\mathbf{S}_{\hat{u}}$ the set of all pairs of nodes belonging to the two subgraphs:

$$\mathbf{S}_{\hat{u}} = \{(n, n') : n \in \hat{s}_i, n' \in \hat{s}_j\}.$$

Each pair of nodes in $\mathbf{S}_{\hat{u}}$ is a potential candidate for adding a connection between the two subgraphs connected by $\hat{u}$ in the functional graph $\hat{H}$. Given a pair of nodes $(n, n') \in \mathbf{S}_{\hat{u}}$ and the local neighbourhoods of n and n' in $\hat{G}$, the function $\varPhi : \mathbf{S}_{\hat{u}} \to [0, 1]$ represents the probability that a connection exists between the two nodes. $\varPhi$ is computed as a product of different potentials, exploiting the local structure of the neighbours of the nodes:

$$\varPhi(n, n') = \varXi(n, n') \cdot \varLambda(n) \cdot \varLambda(n') \cdot \varUpsilon(\mathcal{L}(n), \mathcal{L}(n')).$$

The first potential, $\varXi(n, n)$, is based on the local neighbourhood of the candidate edge (n, n'). We define as $\mathbf{N}_\epsilon(e)$ the ϵ-neighbourhood of an edge $e = (n, n')$, which is a subgraph composed of the edge e, its two nodes n and n', and all the nodes and edges that are at distance ϵ from n or n' (distance is the number of hops). Given the set $\mathbf{C}$ of all cuts c performed on the original graphs in $\mathcal{G}$, we compute the set $\mathbf{Q}$ of all their 1-neighbourhoods, $\mathbf{Q} = \{\mathbf{N}_1(c) : c \in \mathbf{C}\}$. The potential Ξ is then computed as:

$$\Xi(n, n') = \frac{\sum_{x \in \mathbf{Q}} k(d(\mathbf{N}_1(n, n'), x), \bar{d}_x)}{\sum_{x \in \mathbf{Q}} \sum_{y \in \mathbf{Q}} k(d(y, x), \bar{d}_x)},$$

where $\bar{d}_x = \frac{1}{|\mathbf{Q}|} \sum_{y \in \mathbf{Q}} d(x, y)$, $d(x, y)$ is the distance between two graphs x and y computed using a graph kernel (K_{WL} in our case), and k is a Gaussian kernel $k(d', d'') = e^{-\frac{\|d'-d''\|^2}{2\sigma^2}}$ ($\sigma = 0.5$ in our experiments).

The second potential, $\Lambda(n)$, is based on the *preferential attachment* principle introduced by Barabási and Albert (1999) (and empirically observed in graphs representing indoor buildings in Aydemir et al (2012)), which assigns to each node n a probability of having a connection that is proportional to its degree $D(n)$:

$$\Lambda(n) = \frac{D(n)}{\sum_{n'' \in \bar{N}_c} D(n'')},$$

where $\bar{N}_c$ is the set of the nodes that appear in a cut $c \in \mathbf{C}$. $\Lambda(n')$ is computed similarly.

The third (and last) potential function, $\Upsilon(\mathcal{L}(n), \mathcal{L}(n'))$, is evaluated considering all the cuts in $\mathbf{C}$. Considering the nodes n and n' and their labels $\mathcal{L}(n)$ and $\mathcal{L}(n')$, Υ calculates the frequency of a cut $c \in \mathbf{C}$ between two nodes with the same labels.

The probability $\Phi(n, n')$ is then used to select, for each edge $\hat{u}$ between two subgraphs in a functional graph $\hat{H}$, which pair of nodes n, n' belonging to the two subgraphs should be connected in $\hat{G}$. This step is performed using the MCMC algorithm. A transition kernel T_4 selects randomly an edge $\hat{u}$ from the set of edges $\hat{U}$ of the functional graph $\hat{H}$ and selects randomly a pair of nodes from $\mathbf{S}_{\hat{u}}$ to be connected according to the following probability distribution:

$$P_{conn}(n, n') = \frac{\Phi(n, n')}{\sum_{\dot{n}, \dot{n}' \in \mathbf{S}_{\hat{u}}} \Phi(\dot{n}, \dot{n}')}.$$

The transition obtained with this edit operation is accepted or rejected according to a cost function Γ_4:

$$\Gamma_4(\hat{G}) = \sum_{G \in \mathcal{G}} K(\hat{G}, G), \qquad (4)$$

where K is a graph kernel. For this last step, the MCMC is run for more than 100 iterations, after starting from a random initialization of the connections between two nodes in $\mathbf{S}_{\hat{u}}$ for each $\hat{u} \in \hat{U}$ computed using P_{conn}. We use K_{WL} as graph kernel in (4), since it guarantees similar performance, but at a significant lower computational cost, than K_{GH}, and the cost function Γ_4 should be evaluated once for each iteration.

In the example of Fig. 6, the final sampled semantic map is shown in Fig. 6D. The edges added in the last sampling step are highlighted in red.

5.4 Availability of Initial Knowledge

The sampling process depicted in this section assumes that no *a priori* knowledge about the generated building is available, but samples a completely new instance of a building from an empirical distribution. It is easy to adapt the process to the case in which some initial knowledge is available.

Specifically, assume that a portion of a building is known in form of some subgraphs $\{\bar{s}_1, \bar{s}_2, \ldots\}$ which are assigned to clusters $\{\bar{C}_1, \bar{C}_2, \ldots\}$. The cluster configuration sampling is thus performed as reported in Section 5.3.1, but considering the value of $\hat{v}_{\bar{C}_1}$ relative to $\bar{C}_1$ as bounded in the interval $[1, \max_{G \in \mathcal{G}}(v_i{}^G)]$ (the interval starts from 1 since $\bar{s}_1$ has been observed and will belong to $\hat{G}$). Similarly, for $\hat{v}_{\bar{C}_2}$ relative to $\bar{C}_2$ and so on. In the sampling of the functional graph, the known edges between $\{\bar{s}_1, \bar{s}_2, \ldots\}$ are not modified. Subgraphs $\{\bar{s}_1, \bar{s}_2, \ldots\}$ are considered fixed also during subgraphs sampling. Connection between known and unknown nodes are also preserved; for instance, if a corridor connected to five rooms has been explored and only three of them have been already explored, the two edges connected to the still unknown rooms are preserved and sampled rooms are connected to them. Finally, in sampling node connections, known connections between nodes of $\{\bar{s}_1, \bar{s}_2, \ldots\}$ are considered as fixed and are not modified during this last sampling step.

6 EXPERIMENTAL RESULTS

In this section, a quantitative evaluation of our method for sampling new instances of graphs is presented. The proposed method has been implemented[1] in MATLAB R2015b and executed on a i7QuadCore Intel 820QM 16GB computer.

We use either $\mathcal{G}_S$ (with 50 labeled graphs representing schools) or $\mathcal{G}_O$ (with 40 labeled graphs representing offices) and we generate sampled graphs $\hat{\mathcal{G}}$. For each

[1] Code is available at: `https://github.com/goldleaf3i/generativeCMLgraphs`.

metric	$\mathcal{G}_S$	$\hat{\mathcal{G}}_{\texttt{corr}}$	$\hat{\mathcal{G}}_{\texttt{ncut}}$
nodes	35.24 (18.33)	34.15 (14.73)	35.27 (14.56)
nodes R	27.84 (15.05)	26.72 (11.55)	27.88 (12.00)
nodes C	7.42 (4.45)	7.43 (3.89)	7.39 (3.49)
path-length	3.33 (0.81)	3.29 (0.55)	3.38 (0.65)
diameter	6.34 (2.30)	6.25 (1.59)	6.67 (1.87)
art-points	8.8 (5.81)	8.61 (4.27)	9.08 (4.15)
assortativity	-0.51 (0.20)	-0.50 (0.14)	-0.48 (0.12)
betw-cen	0.039 (0.011)	0.040 (0.012)	0.040 (0.015)
betw-cen R	0.005 (0.006)	0.006 (0.006)	0.010 (0.010)
betw-cen C	0.181 (0.090)	0.172 (0.066)	0.157 (0.058)
closn-cen	0.328 (0.086)	0.322 (0.056)	0.317 (0.063)
closn-cen R	0.309 (0.077)	0.302 (0.051)	0.298 (0.055)
closn-cen C	0.418 (0.160)	0.401 (0.088)	0.389 (0.097)
eig-cen	0.256 (0.078)	0.254 (0.066)	0.260 (0.083)
eig-cen R	0.197 (0.062)	0.190 (0.052)	0.197 (0.064)
eig-cen C	0.498 (0.181)	0.497 (0.141)	0.503 (0.172)
katz-cen	0.181 (0.045)	0.179 (0.042)	0.177 (0.046)
katz-cen R	0.166 (0.045)	0.165 (0.039)	0.164 (0.045)
katz-cen C	0.241 (0.076)	0.236 (0.056)	0.232 (0.056)

Table 2 Values of metrics relative to the SCHOOL building type for original graphs $\mathcal{G}_S$ and graphs generated with the proposed approach using `corr` and `ncut` segmentation ($\hat{\mathcal{G}}_{\texttt{corr}}$ and $\hat{\mathcal{G}}_{\texttt{ncut}}$, respectively). Entries report average μ over the graphs and standard deviation σ (in parenthesis). R means ROOM and C means CORRIDOR.

metric	$\mathcal{G}_O$	$\hat{\mathcal{G}}_{\texttt{corr}}$	$\hat{\mathcal{G}}_{\texttt{ncut}}$
nodes	39.64 (18.44)	41.63 (18.69)	35.62 (14.05)
nodes R	30.83 (15.47)	32.5 (15.01)	27.57 (11.77)
nodes C	8.81 (4.38)	9.08 (4.32)	8.05 (3.43)
path-length	3.33 (0.62)	3.38 (0.56)	3.23 (0.51)
diameter	6.37 (1.74)	6.52 (1.66)	6.16 (1.54)
art-points	8.97 (4.23)	9.38 (4.48)	8.02 (3.13)
assortativity	-0.55 (0.14)	-0.55 (0.12)	-0.54 (0.10)
betw-cen	0.035 (0.012)	0.035 (0.014)	0.038 (0.015)
betw-cen R	0.005 (0.007)	0.003 (0.004)	0.006 (0.011)
betw-cen C	0.141 (0.042)	0.154 (0.065)	0.151 (0.056)
closn-cen	0.321 (0.071)	0.294 (0.049)	0.326 (0.054)
closn-cen R	0.303 (0.068)	0.294 (0.049)	0.306 (0.047)
closn-cen C	0.387 (0.093)	0.387 (0.090)	0.399 (0.084)
eig-cen	0.251 (0.079)	0.239 (0.085)	0.257 (0.083)
eig-cen R	0.188 (0.065)	0.174 (0.085)	0.189 (0.068)
eig-cen C	0.471 (0.131)	0.482 (0.173)	0.492 (0.156)
katz-cen	0.166 (0.040)	0.164 (0.046)	0.175 (0.045)
katz-cen R	0.152 (0.039)	0.149 (0.044)	0.159 (0.044)
katz-cen C	0.220 (0.045)	0.221 (0.059)	0.231 (0.052)

Table 3 Values of metrics relative to the OFFICE building type for original graphs $\mathcal{G}_O$ and graphs generated with the proposed approach using `corr` and `ncut` segmentation ($\hat{\mathcal{G}}_{\texttt{corr}}$ and $\hat{\mathcal{G}}_{\texttt{ncut}}$, respectively). Entries report average μ over the graphs and standard deviation σ (in parenthesis). R means ROOM and C means CORRIDOR.

data set $\mathcal{G}$, we use either `ncut` or `corr` for segmentation and affinity propagation for clustering, together with K_{GH}, generating $\hat{\mathcal{G}}_{\texttt{corr}}$ and $\hat{\mathcal{G}}_{\texttt{ncut}}$ composed of 200 graphs each. Sampling is performed using K_{WL}.

From preliminary tests we observed that the segmentation methods impact on sampling (indirectly, via formed clusters) and, for this reason, we consider both segmentation methods in our tests.

We define a set of metrics representing structural properties of a graph. To assess the similarity between the original graphs $\mathcal{G}$ (either $\mathcal{G}_S$ or $\mathcal{G}_O$) and the sampled graphs $\hat{\mathcal{G}}$ we compare the average μ and standard deviation σ of these metrics computed on both sets. This allows us to check whether the structure of the generated graphs is consistent with that of the real buildings present in $\mathcal{G}$.

Metrics can be divided in two groups. Those of the first group consider standard graph measures: the number of nodes (nodes), the average length of a shortest path between each pair of nodes in the graph (path-length), the maximum distance between two nodes along their shortest path (diameter), the number of articulation points (art-points; an articulation point is a node whose removal separates the graph into two distinct components), and the degree assortativity (assortativity) (Newman, 2003), which indicates whether nodes are connected to other nodes with a similar degree (1) or not (−1):

$$\mathsf{assortativity} = \sum_{i,j} Dg_i Dg_j \frac{e_{Dg_i,Dg_j} - a_{Dg_i} a_{Dg_j}}{\sigma_a \sigma_b},$$

where Dg_i and Dg_j are the degrees of nodes i and j, respectively, e_{Dg_i,Dg_j} is the fraction of edges between a node with degree Dg_i and a node with degree Dg_j, a_{Dg_i} is the fraction of the edges where at least one node has a degree of Dg_i (a_{Dg_j} is defined similarly), and σ_a and σ_b are the standard deviations of the empirical distributions of a_{Dg_i} and a_{Dg_j}, respectively.

The second group of metrics measure the *centrality* of each node in the graph. Centrality of a node indicates its role within the graph relatively to the connections between nodes. High centrality values indicate important hub nodes, whose presence greatly impacts on the graph layout, while low centrality values indicate peripheral nodes. Since centrality is a metric calculated for each node, we compute the average centrality μ and its standard deviation σ for each graph, by averaging over the number of nodes in the graph. Moreover, since in our domain CORRIDOR nodes are usually the most important (and thus should result in higher centrality values), while ROOM nodes are usually connected only to a corridor, the centrality measures are evaluated also separately for these two types of nodes. In the literature there exist several metrics for computing node centrality. We select four centrality measures among the most used ones.

Betweenness centrality betw-cen for a node $n \in N$ of a graph $G = (N, E)$ is defined as the number of shortest paths between two nodes $u \neq t$ ($\neq n$) that pass through n:

$$\mathsf{betw\text{-}cen}(n) = \sum_{u,t \in N, u \neq n \neq t} \frac{\tau_{ut}(n)}{\tau_{ut}},$$

where τ_{ut} is the total number of shortest paths from u to t and $\tau_{ut}(n) \leq \tau_{ut}$ is the number of these paths that pass through n.

Closeness centrality (closn-cen) for a node n is defined according to the shortest distance between n and all other nodes:

$$\mathsf{closn\text{-}cen}(n) = \sum_{t \in N \setminus \{n\}} 2^{-d_G(n,t)},$$

where $d_G(n,t)$ is the length of the shortest path on G between the nodes n and t. Eigenvector centrality (eig_cen) assigns a relative score to all nodes in the graph based on the idea that connections to high-scoring nodes contribute more to the score of a node than connections to low-scoring nodes. It is calculated as:

$$\mathsf{eig\_cen}(n) = \frac{1}{\lambda} \sum_{t \in N} m_{t,n}\, \mathsf{eig\_cen}(k),$$

where $\lambda \neq 0$ is a constant and $m_{t,n}$ is the element (t,n) of the adjacency matrix M of the graph G.

Katz centrality (katz) computes the centrality of a node n based on that of its adjacent nodes:

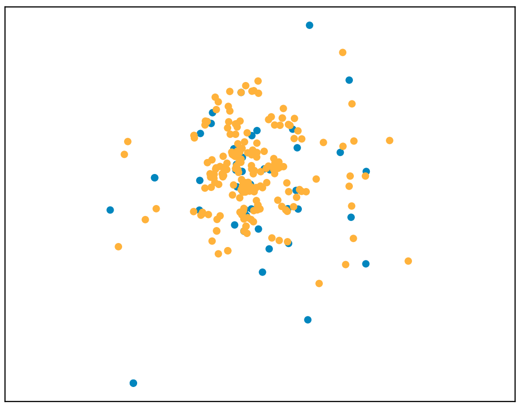
$$\mathsf{katz}(n) = \alpha \sum_{n' \in N} m_{n',n} \mathsf{katz}(n') + \beta,$$

where α and β are constants and $m_{n',n}$ is the element corresponding to the (n', n) element in the adjacency matrix M of G.

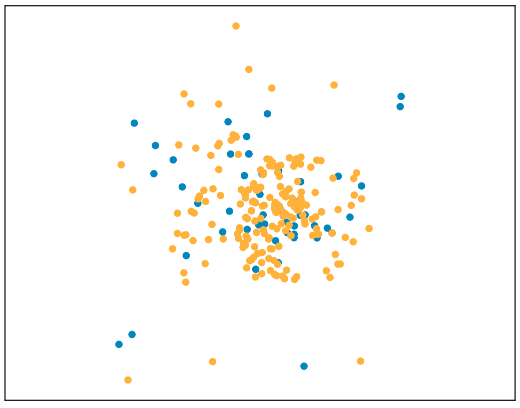
Tables 2 and 3 shows that the generated graphs $\hat{\mathcal{G}}$ are similar to the original ones in $\mathcal{G}$, according to all the metrics. Statistical equivalence is checked by performing the Wilcoxon test for all metrics of Tables 2 and 3. The test is successfully passed in all cases except for the case of betw-cen R for OFFICEs using `corr` and for the case of betw-cen R for SCHOOLs using `ncut`.

The results show that our method can sample new graphs with a similar structure as the original ones. In particular, node importance in graphs $\hat{\mathcal{G}}$ is consistent with the node importance of the real graphs $\mathcal{G}$. It is important to point out that centrality measures are relative to the entire structure of the graph and are not dependent on any local pattern nor on single nodes. Similar centrality measures for $\mathcal{G}$ and $\hat{\mathcal{G}}$ mean that they actually contain similar graphs. Moreover, the different values of all four centrality metrics (although more evident for betw-cen) for nodes ROOM and CORRIDOR (with low and high values, respectively) indicates that also the roles of the nodes are distributed similarly over $\mathcal{G}$ and $\hat{\mathcal{G}}$.

Fig. 7 displays a visual representation of how graphs in $\mathcal{G}_S$ and $\hat{\mathcal{G}}$ are distributed, for both segmentation techniques `corr` and `ncut`. Using K_{WL} we compute the similarity between each graph in $\bar{\mathcal{G}} = \mathcal{G}_S \cup \hat{\mathcal{G}}$, obtaining a $|\bar{\mathcal{G}}| \times |\bar{\mathcal{G}}|$ similarity matrix. This high-dimensional feature space is then reduced to a two-dimensional one using a *t-distributed stochastic neighbor embedding* (t-SNE) (van der Maaten and Hinton, 2008). t-SNE models each high-dimensional object as a two-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points. In both Fig. 7a and Fig. 7b the original graphs (in blue) and the sampled ones (in orange) are similarly distributed, thus providing a visual confirmation that the two graph sets contain similar graphs. Results from $\mathcal{G}_O$ are similar and data are not reported here.

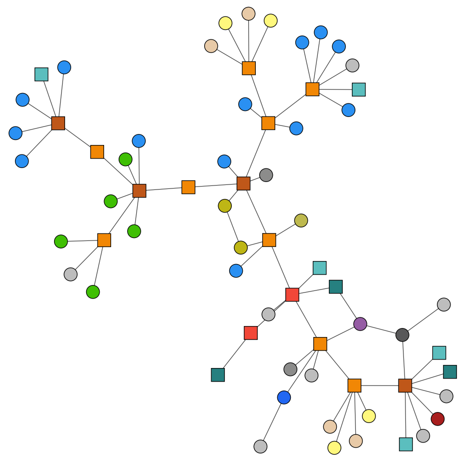


(a) `corr`

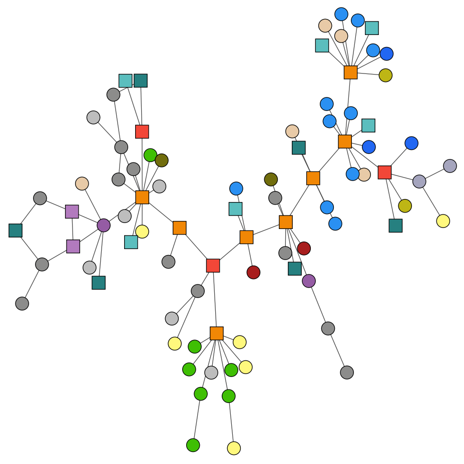


(b) `ncut`

Fig. 7 2D representation of graph distributions for graphs in $\mathcal{G}_S$ (blue) and in $\hat{\mathcal{G}}$ (orange) using t-SNE.



(a) real



(b) sampled

Fig. 8 A graph representing a real school building from $\mathcal{G}_S$ (a) and a graph $\hat{G}$ sampled using our method (b).

graph kernel	step 1	step 2	step 3	step 4
K_{WL}	10^{-1}	10^{-1}	10^{-3}	10
K_{GH}	10^{-1}	10	10^{-3}	10^{2}

Table 4 Computing time (order of magnitude, in seconds) for each sampling step. Step 1 indicates cluster configuration sampling, step 2 indicates functional graph sampling, step 3 indicates subgraphs sampling, and step 4 indicates node connections sampling. For steps 1, 2, and 4 the time is intended for a single iteration of the MCMC method explained in Section 5. Steps 1 and 3 do not use graph kernels.

Fig. 8 shows a sampled semantic map $\hat{G}$ and a similar semantic map from $\mathcal{G}_S$. The two graphs share the same structure composed of a corridor tree to which other rooms are attached and this visually confirms the outcome of quantitative analysis that they represent two instances of the same class of buildings. In Fig. 9, we report some examples of sampled graphs in $\hat{\mathcal{G}}_{\texttt{corr}}$ relative to the SCHOOL building type, in order to show the variety of generated environments, both in structure (e.g., with or without loops of corridors) and in size (number of nodes).

Table 4 reports computing times for the sampling of graphs using our approach. It emerges that node connections sampling is the most expensive operation, since, for every iteration of the method and for every connection in the functional graph between two subgraphs $\hat{s}_i$, $\hat{s}_j$, it computes a score for connections between every possible pair of nodes $\hat{n}_k$ and $\hat{n}_l$ belonging to $\hat{s}_i$ and $\hat{s}_j$, respectively.

Sampling results obtained using `ncut` as segmentation method are similar to those obtained using `corr`. However, when the task is to sample new graphs from scratch, it is advisable to use `ncut`, since it divides the original graphs into subgraphs of similar size and per-

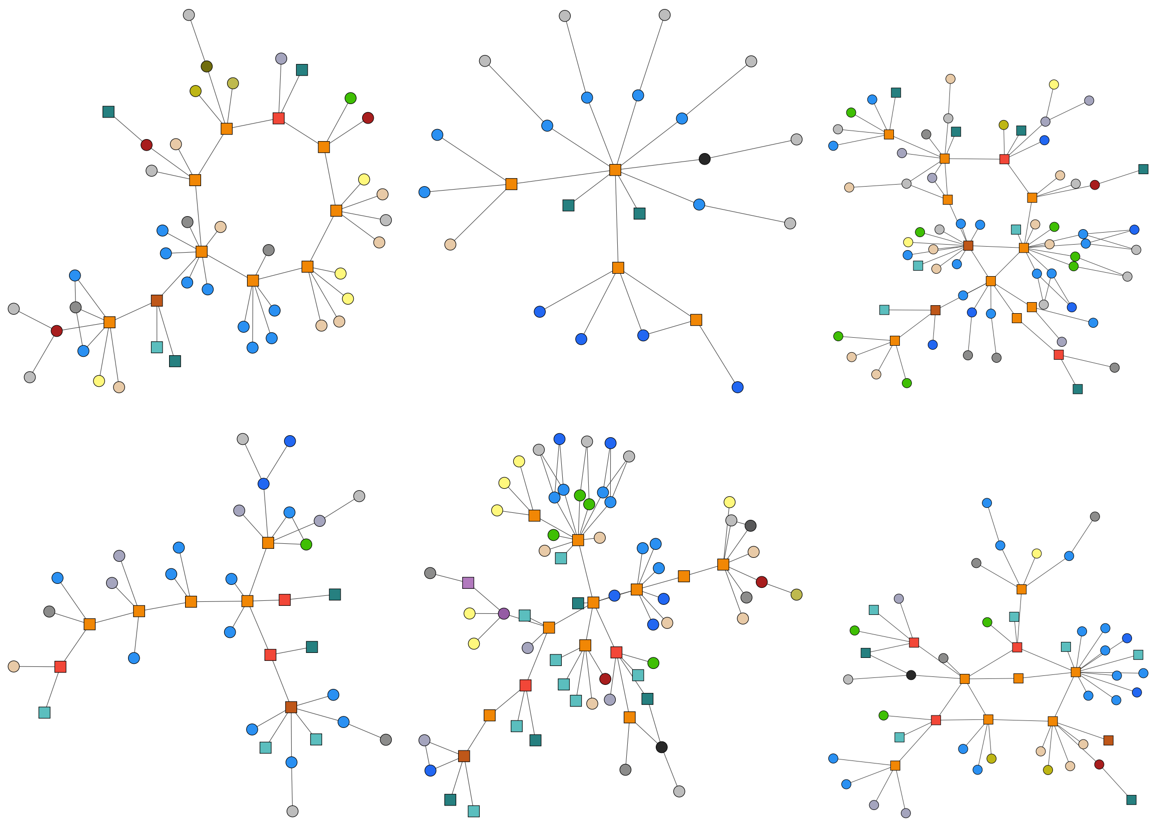


Fig. 9 Six sampled graphs randomly selected from $\hat{\mathcal{G}}_{\text{corr}}$ obtained fro the SCHOOL building type.

forms cuts between nodes independently of their semantic labels, thus providing more different results when sampling subgraphs and reconnecting nodes. Segmentation using `corr` is designed to facilitate the recognition of a subgraph by a robot during the (online) exploration of a building, and it should be employed when our method is used to generate hypotheses about the structure of a partially visited building given the available semantic map. An example of this application is provided in the next section.

7 PREDICTION

We envisage at least two perspective applications of the proposed approach, in simulation and in exploration. In the first application, the proposed generative model can be used to create new realistic environments from scratch (without initial knowledge), which can be embedded in a simulator to ease the test of the performance of autonomous mobile robots in several settings. This ides is described in Amigoni et al (2014) and is not discussed further here. Instead, we focus on the second application, exploration, in which the proposed generative model can be used to generate hypotheses on the structure and the labeling of the unexplored parts of a partially known environment, in order to provide information that can speed up the exploration, for example by better coordinating robots (Quattrini Li et al, 2016). Our goal here is not to provide a robotic system for exploration, but just to illustrate a potential use of the method we propose in this paper.

Consider a simplified setting in which an initially unknown (floor of a) school building is being explored by one or more robots. We assume that the exploration proceeds as follows:

1. exploration starts from one of the rooms labeled as ENTRANCE, ELEVATOR, or STAIRS;
2. the robots move to the first CORRIDOR room connected to the entrance;
3. each room connected to the corridor is explored using a breadth-first exploration strategy; the semantic map is expanded accordingly with the correct semantic labels for the explored rooms;
4. after all the rooms connected to the first corridor have been explored the robots predict the structure of the unexplored part of the building;
5. the robots move to the nearest corridor discovered and repeat from step 3, until the entire building has been explored.

For this setup we use `corr` as segmentation method. This allows us to automatically detect the subgraphs during exploration, since, when step 3 above is executed, the robots explore a new subgraph (as explained in Section 3, `corr` segments the graphs assuming that subgraphs are formed by a CORRIDOR and its neighbouring ROOMs). Under these premises, possible structures of the building are generated every time a new subgraph is explored. Prediction is tested on the same $\mathcal{G}_S$ composed of 50 graphs of Section 6 using leave-one-out (the explored graph for which the predictions are made is removed from the data set when training the model $\mathcal{M}$).

In order to show how our approach can be applied to the above setting, we present in detail two examples. Given the importance of corridors for exploration (see, for instance, (Quattrini Li et al, 2016; Stachniss et al, 2006)), we are particularly interested in generating predictions of the structure of the corridors of the unvisited part of the environment. For each example, we display the original graph (representing the environment being explored), its corridor structure, the predicted graph, and its corridor structure during an incremental exploration of the environment. The known (already explored) part of the original graph is highlighted with a grey overlay.

As a first example, consider the environment of Fig. 10a, whose cross-shaped "skeleton" of corridors is shown in Fig. 10b (refer to Fig. 2a for the labels). Figs. 11a-11b show the predictions after the exploration of the initial corridor and the connected rooms. The rest of Fig. 11 shows three other predictions made after the exploration of larger and larger parts of the environment. As expected, the more the initial knowledge used for sampling an environment (see Section 5.4), the more accurate the generated hypotheses when compared to the real environment of Fig. 10. This holds especially for the corridor "skeleton" of the building, which is correctly predicted to be cross-shaped.

We also evaluate how well our approach can predict the labels distribution (i.e., the percentage of rooms with a specific label) in the unvisited part of the environment. Figs. 11c, 11f, 11i, and 11l show the labels distribution of the real graph of Fig. 10 (in yellow), the average labels distribution for all the graphs in $\mathcal{G}_S$ (in green), and the predicted labels distribution at each step (in purple). Ideally, the purple bars should be more similar to the yellow ones than to the green ones, meaning that our approach is able to actually predict the distribution of labels for a given environment and does not simply return a "blind" prediction based on the average of labels distributions of the initial graphs. This is actually the case for the most relevant labels, especially when the exploration proceeds. For instance, see the corridors and the classrooms in Fig. 11i and Fig. 11l.

The second example, relative to a building with loops of corridors (two loops of corridors and a third one closed by a cafeteria), is shown in Figs. 12 and 13. Also in this case, the loops of corridors that represent the "skeleton" of the environment are predicted correctly by the hypotheses generated with our approach after step 1 of exploration. Note that in both the examples (and in several other tests that are not shown here), the prediction is not always accurate with respect to the exact shape of the unvisited environment, but nevertheless it captures well the structure of the building (respectively, the cross shape of the first example and the loop shape of the second example). This is sound with the observation at the core of the idea of building type (see Section 3), namely that every school building is different but all school buildings share some similarities, which are actually captured by our model.

For a quantitative evaluation, Table 5 reports the values of the same metrics of Section 6 now calculated for the original graphs $\mathcal{G}_S$ and for 224 graphs ($\hat{\mathcal{G}}_{\texttt{corr}}$) generated from partially explored graphs $\bar{G}$. Using leave-one-out, we simulate three exploration runs for each graph $G \in \mathcal{G}_S$. Each simulated exploration run on G produces some randomly-selected graphs $\bar{G}$ (from 1, for small graphs G, to $5-6$, for large graphs G, depending on the number of corridors/subgraphs present in G). Each $\bar{G}$ results in a prediction obtained by generating a $\hat{G}$ with our approach, and all $\hat{G}$ are collected in $\hat{\mathcal{G}}_{\texttt{corr}}$. Note that, from the table, although graphs $\hat{\mathcal{G}}_{\texttt{corr}}$ are significantly larger than those in $\mathcal{G}_S$ (since larger graphs contribute with several $\bar{G}$ and, thus, with more graphs $\hat{G}$ in $\hat{\mathcal{G}}_{\texttt{corr}}$), the values for the centrality measures are similar. These results show that the hypotheses generated for the unknown part of a graph G are plausible, namely are consistent with graphs in $\mathcal{G}_S$ (recall that G is excluded).

In our discussion, we assumed to have a perfect knowledge of the labels of the already explored subgraph $\bar{G}$, which can be difficult to obtain in a real world scenario. We now justify this assumption by showing that similarity between graphs computed using graph kernels, which is at the basis of our approach, is robust wrt changes of labels of few nodes, because it considers the global structure of the graphs. If we consider $\mathcal{G}$ and we normalize the distance $K_{GH}(G, G')$ between any two graphs G and G' of $\mathcal{G}$ to $[0, 1]$, then the distance $K_{GH}(G, G')$ between G and its copy G' in which 10% of the labels are randomly changed is 0.002 (on average over $\mathcal{G}$). The distance $K_{GH}(G, G')$ between G and its copy G' where 5% of the nodes are changed

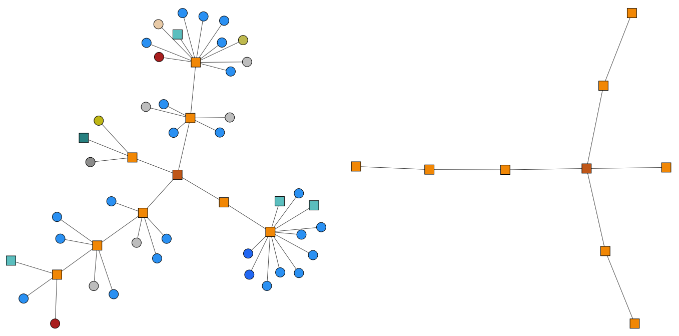


Fig. 10 An environment being explored (first example).

metric	$\mathcal{G}_S$	$\hat{\mathcal{G}}_{\texttt{corr}}$
nodes	35.24 (18.33)	45.23 (17.17)
nodes R	27.84 (15.05)	35.95 (14.17)
nodes C	7.42 (4.45)	9.28 (4.14)
path-length	3.33 (0.81)	3.69 (0.60)
diameter	6.34 (2.30)	7.29 (1.73)
art-points	8.8 (5.81)	11.09 (5.31)
assortativity	-0.51 (0.20)	-0.50 (0.14)
betw-cen	0.039 (0.011)	0.034 (0.011)
betw-cen R	0.005 (0.006)	0.006 (0.006)
betw-cen C	0.181 (0.090)	0.152 (0.052)
closn-cen	0.328 (0.086)	0.287 (0.048)
closn-cen R	0.309 (0.077)	0.272 (0.043)
closn-cen C	0.418 (0.160)	0.347 (0.074)
eig-cen	0.256 (0.078)	0.214 (0.060)
eig-cen R	0.197 (0.062)	0.162 (0.049)
eig-cen C	0.498 (0.181)	0.427 (0.139)
katz-cen	0.181 (0.045)	0.153 (0.033)
katz-cen R	0.166 (0.045)	0.141 (0.032)
katz-cen C	0.241 (0.076)	0.205 (0.045)

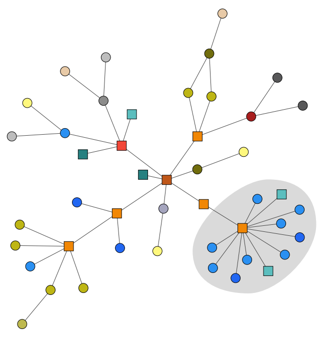
Table 5 Values of metrics relative to the SCHOOL building type for original graphs $\mathcal{G}_S$ and graphs generated after partial explorations with the proposed approach using `corr` segmentation ($\hat{\mathcal{G}}_{\texttt{corr}}$). Entries report average μ over the graphs and standard deviation σ (in parenthesis). R means ROOM and C means CORRIDOR.

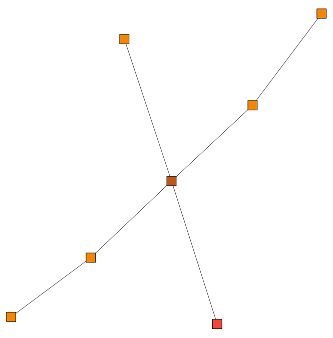
(randomly attached to other nodes, thus changing the graph structure), results in an average value of 0.05.

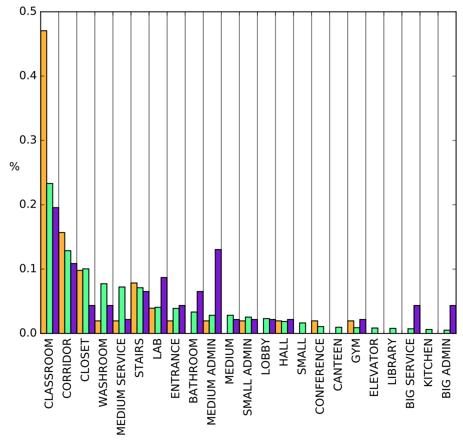
Note, finally, that the generation of hypotheses for unknown parts of environments made with our approach and discussed in this section cannot be attained with the methods performing local predictions (see Sections 1 and 2), because they can only predict the presence and the labels of nodes adjacent to the known nodes. Moreover, the predictions obtained with our approach are not limited to be equal to portions of the initial buildings in $\mathcal{G}$ (as in Aydemir et al (2012)), but can be arbitrary compositions of the subgraphs of the initial buildings.

8 CONCLUSIONS

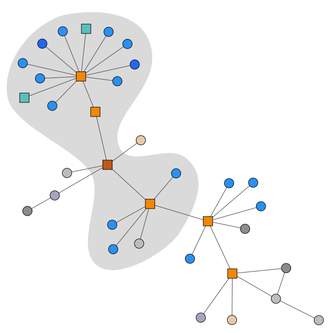
This paper has presented an approach that builds a generative model of graphs representing the topological structure and the semantic labeling of indoor environments and that uses the model to create new instances of (parts of) buildings. The three main steps of the proposed approach, namely segmentation, clustering, and sampling, are based on the use of graph kernels to evaluate the similarity between graphs and on a hierarchical MCMC sampling algorithm. The experimental validation shows that the generated buildings, although not identical, share many features with the original buildings used to create the model and, significantly, have similar structures. The approach can be used to generate hypotheses on unknown parts of partially explored buildings or to generate new buildings from scratch.

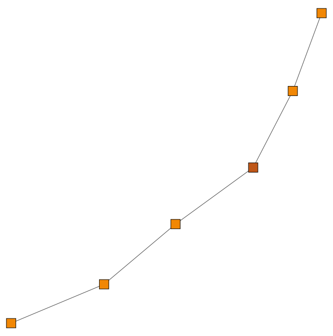

(a) Predicted semantic graph after step 1.

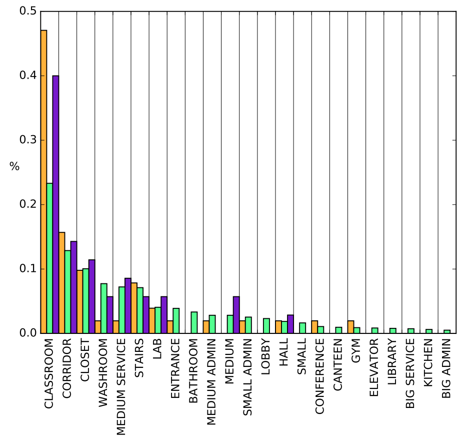

(b) Predicted corridor structure after step 1.



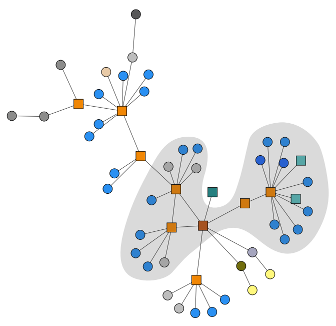
(c) Labels distribution after step 1.

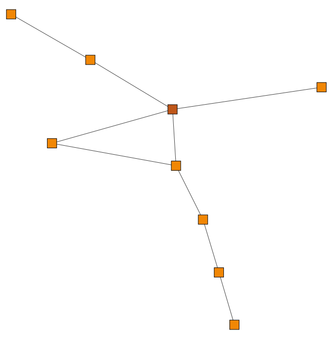

(d) Predicted semantic graph after step 2.

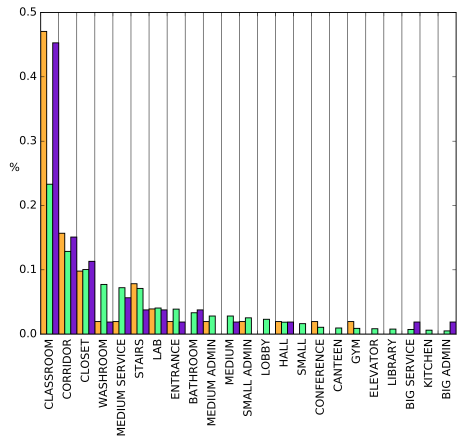

(e) Predicted corridor structure after step 2.



(f) Labels distribution after step 2.

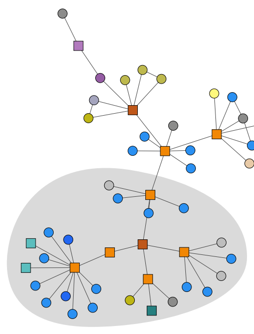

(g) Predicted semantic graph after step 3.

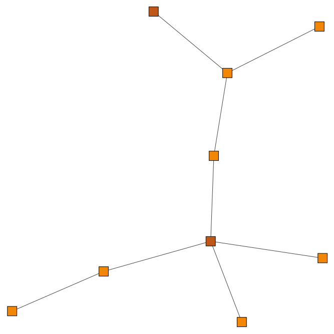

(h) Predicted corridor structure after step 3.

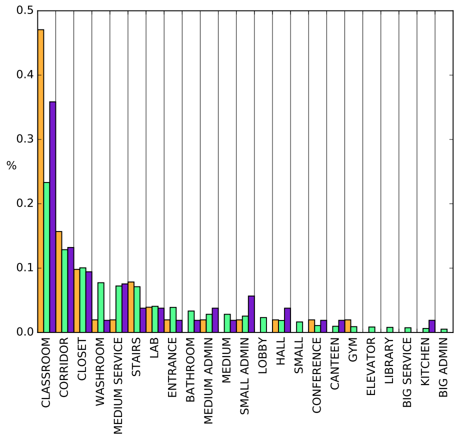


(i) Labels distribution after step 3.

Fig. 11 Predictions made during exploration of the environment of Fig. 10. (Continues on the next page.)

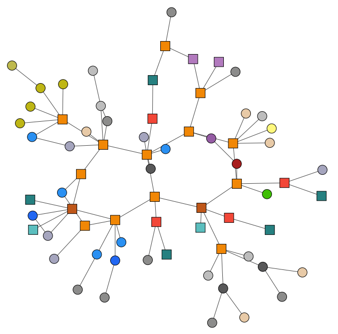

(j) Predicted semantic graph after step 4.


(k) Predicted corridor structure after step 4.



(l) Labels distribution after step 4.

Fig. 11 Predictions made during exploration of the environment of Fig. 10. (Continued from the previous page.)


(a) Real environment.

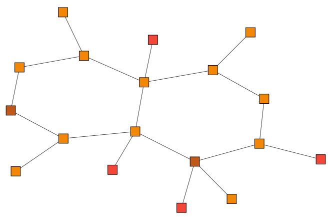

(b) Corridor structure.

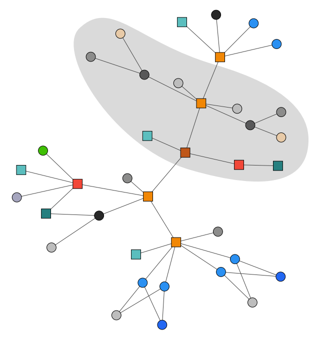
Fig. 12 An environment being explored (second example).

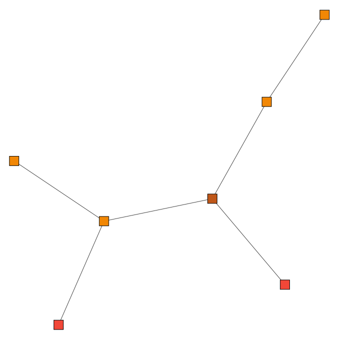
Possible future work includes the further assessment of the proposed approach, applying it to other building types and possibly refining the methods that implement its three main steps. We expect that the method can be smoothly applied to other structured environments (like hospitals), while its application to environments that are less structured (like houses, which are also relatively small) could be less straightforward. In addition, we will investigate the actual benefits to autonomous exploration of unknown environments, starting from the results reported in the previous section and embedding our approach in exploring robot systems like those of Wurm et al (2008) and Solanas and Garcia (2004). More generally, our aim is to develop methods that provide reliable predictions about unvisited parts of buildings.

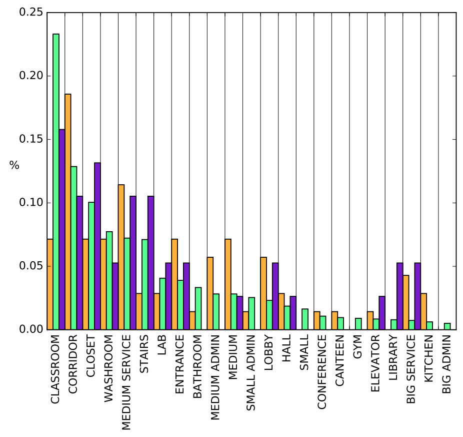
Appendix: Graph Kernels

The task of learning a model of structured data like graphs, as we do in this paper, is often complex and requires *ad hoc* techniques since standard machine learning techniques cannot be naturally adopted due to the dimensionality of the data (De Raedt, 2008). We make a substantial use of *graph kernel* methods, which use a decompositional approach to measure the similarity between two graphs. In this appendix we provide a general overview on the concept of graph kernel, as introduced by Haussler (1999), and we describe the graph kernels that we use in our method. Graph kernel methods are the equivalent for structured data of kernel methods (Gärtner et al, 2004), which are typically defined in a vectorial space. Among the graph kernel families, the general class of *convolutional kernels* proposed by Haussler (1999) represents the guiding principle in kernel design for structured objects (Costa and De Grave, 2010; Gärtner et al, 2004). Convolutional kernels are based on the idea that the structure of a graph can be captured by a relation R between the graph and its parts. A graph kernel is then defined as a composition of kernels defined on different parts of the graph.

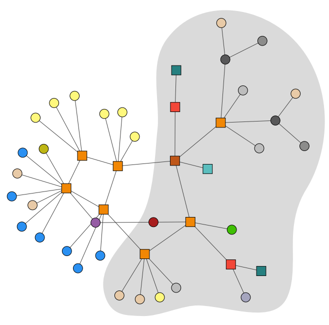
Let $G = (N, E) \in \mathcal{G}$ be a graph and $\{g_1, \ldots, g_D\}$ one of its decompositions into (possibly overlapping) parts (sub-

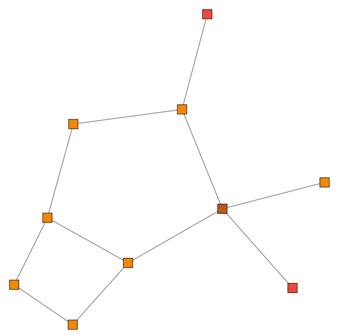

(a) Predicted semantic graph after step 1.

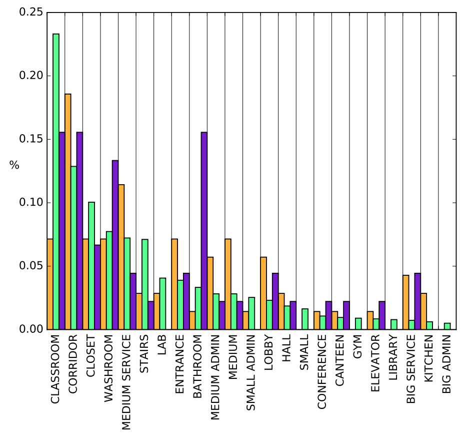

(b) Predicted corridor structure after step 1.



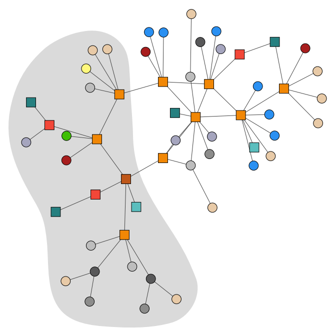
(c) Labels distribution after step 1.

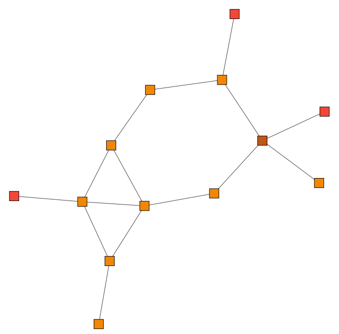

(d) Predicted semantic graph after step 2.

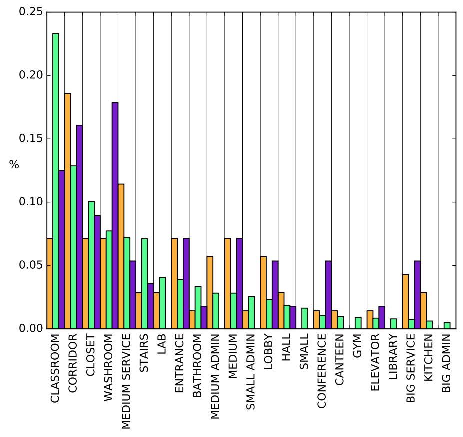

(e) Predicted corridor structure after step 2.



(f) Labels distribution after step 2.


(g) Predicted semantic graph after step 3.


(h) Predicted corridor structure after step 3.



(i) Labels distribution after step 3.

Fig. 13 Predictions made during exploration of the environment of Fig. 12.

graphs). Each part g_d in an element of a countable set G_d, for $d = 1, \ldots, D$, $D \geq 1$. Consider a relation R defined on $G_1 \times \ldots \times G_D \times \mathcal{G}$, where $R(g_1, \ldots, g_D, G)$ is true if the set $\{g_1, g_2, \ldots, g_D\}$ is one of the possible sets of the parts (subgraphs) of G (i.e., in which G can be decomposed). Given R, we can define the function R^{-1} as the decomposition function which returns the sets of parts (decompositions) of G: $R^{-1}(G) = \{g_1, \ldots, g_D : R(g_1, \ldots, g_D, G)\}$. Consider now any positive definite kernel function K_d over $G_d \times G_d$, $d = 1, \ldots, D$. For two graphs $G, Q \in \mathcal{G}$, we can define a *convolutional* or *decomposition kernel* on graphs as the function:

$$K(G, Q) = \sum_{\substack{g_1, \ldots, g_D \in R^{-1}(G) \\ q_1, \ldots, q_D \in R^{-1}(Q)}} \prod_{d=1}^{D} K_d(g_d, q_d).$$

Given a generic graph kernel K, the similarity measure between two graphs $G, Q \in \mathcal{G}$ can be defined as the normalized version of the graph kernel:

$$K_{normalized}(G, Q) = \frac{K(G, Q)}{\sqrt{K(G, G)K(Q, Q)}}.$$

In this work we use two instances of convolutional graph kernels: the *Weisfeiler-Lehman Subtree Kernel* (K_{WL}) (Shervashidze et al, 2011) and the *Graph Hopper Kernel* (K_{GH}) (Feragen et al, 2013). Several other kernels, such as those by Costa and De Grave (2010), Menchetti et al (2005), and Kashima et al (2003) have been empirically tested with less convincing results (numerical results are not reported here).

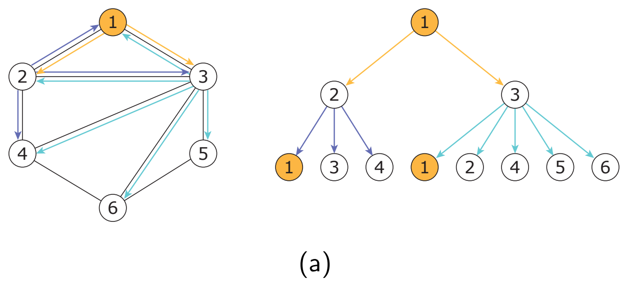


Fig. 14 An example of a $d = 2$ subtree rooted from node 1.

The K_{WL} belongs to the family of Weisfeiler-Lehman graph kernels (Shervashidze et al, 2011), that exploit the Weisfeiler-Lehman test of isomorphism on graphs as a rapid feature extraction scheme. K_{WL} is implemented as an iterative procedure in which, at each iteration, the node labels are augmented by concatenating to the current label of each node the sorted list of labels of its adjacent nodes. The resulting augmented label list is then compressed into a new, shorter label list using an hash function. Each graph G is thus mapped to a sequence of graphs $\{G_0, G_1, \ldots, G_h\} = \{(N, E, L_0), (N, E, L_1), \ldots, (N, E, L_h)\}$, where G_i and L_i are the graph and the label set (one label per node) after i iterations of the algorithm, respectively, and h is the total number of iterations. Given any graph kernel K_b called *base kernel*, K_{WL} is then computed as:

$$K_{WL}(G, Q) = K_b(G_0, Q_0) + K_b(G_1, Q_1) + \ldots + K_b(G_h, Q_h).$$

In our setting we use as base kernel $K_b(G_i, Q_i)$ a subtree kernel. It computes all the rooted subtrees of $G = (N, E)$ (a rooted subtree is an acyclic sub-graph of G with a given depth d and rooted on a node $n \in N$, where nodes $n' \in N$ can be repeated in different branches of the tree; an example of a rooted subtree is shown in Fig. 14). These rooted subtrees are the outcome of the decomposition R^{-1}. For each iteration i, $i = 1, \ldots, h$, K_b counts the number of common labels between all the subtrees of G_i and Q_i rooted in two nodes with the same label $l^* \in L_i$. The complexity of K_{WL} when applied to a set of N graphs is $O(Nhm + N^2hn)$, where N is the total number of graphs, and n and m are the number of nodes and edges of the graphs respectively (assumed equal for all graphs for simplicity).

In K_{GH} (Feragen et al, 2013), the decomposition relation R^{-1} is based on the shortest path between each pair of nodes:

$$K_{GH}(G, Q) = \sum_{\pi \in \mathcal{P}, \pi' \in \mathcal{P}'} k_p(\pi, \pi'),$$

where k_p is a kernel defined on paths and $\mathcal{P}$ and $\mathcal{P}'$ are the sets of shortest paths between all pairs of nodes of G and Q, respectively. The path-kernel k_p is defined on two paths π and π' as:

$$k_p(\pi, \pi') = \begin{cases} \sum_{j=1}^{|\pi|} k_n(\pi(j), \pi'(j)) & \text{if } |\pi| = |\pi'| \\ 0 & \text{else} \end{cases},$$

where $\pi(i) = n_i$ indicates the i-th node in the path $\pi = (n_1, \ldots, n_{|\pi|})$. We use a linear node kernel $k_n(n_i, n_j)$ that returns 1 if n_i and n_j have the same label and 0 otherwise. Total complexity of K_{GH} is $O(N^2(n^2(m + \log n + d + \delta^2)))$ where N, n, and m are defined as in K_{WL}, δ is the length of the longest shortest path, and d is a constant.

Acknowledgment

Authors gratefully thank Modestino Cucciniello and Mattia Di Vitto for their contributions to the implementation of the system presented in this paper.

References

Amigoni F, Luperto M, Quattrini Li A (2014) Towards more realistic indoor environments for the virtual robot competition. In: RoboCup2014 CD

Aydemir A, Jensfelt P, Folkesson J (2012) What can we learn from 38,000 rooms? Reasoning about unexplored space in indoor environments. In: Proc. IROS, pp 4675–4682

Aydemir A, Pronobis A, Gobelbecker M, Jensfelt P (2013) Active visual object search in unknown environments using uncertain semantics. IEEE Transactions on Robotics 29(4):986–1002

Barabási AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512

Costa F (2017) Learning an efficient constructive sampler for graphs. Artificial Intelligence 244:217–238

Costa F, De Grave K (2010) Fast neighborhood subgraph pairwise distance kernel. In: Proc. ICML, pp 255–262

De Raedt L (2008) Logical and Relational Learning. Springer

Feragen A, Kasenburg N, Petersen J, de Bruijne M, Borgwardt K (2013) Scalable kernels for graphs with continuous attributes. In: Proc. NIPS, pp 216–224

Frey BJ, Dueck D (2007) Clustering by passing messages between data points. Science 315(5814):972–976

Galindo C, Saffiotti A, Coradeschi S, Buschka P, Fernandez-Madrigal J, González J (2005) Multi-hierarchical semantic maps for mobile robotics. In: Proc. IROS, pp 2278–2283

Gärtner T, Lloyd JW, Flach PA (2004) Kernels and distances for structured data. Machine Learning 57(3):205–232

Haussler D (1999) Convolution kernels on discrete structures. Tech. rep., University of California, Santa Cruz, USA

Hemachandra S, Walter M, Tellex S, Teller S (2014) Learning spatial-semantic representations from natural language descriptions and scene classifications. In: Proc. ICRA, pp 2623–2630

Kashima H, Tsuda K, Inokuchi A (2003) Marginalized kernels between labeled graphs. In: Proc. ICML, pp 321–328

Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT press

Luperto M, Amigoni F (2014) Exploiting structural properties of buildings towards general semantic mapping systems. In: Proc. IAS-13, pp 375–387

Luperto M, Quattrini Li A, Amigoni F (2013) A system for building semantic maps of indoor environments exploiting the concept of building typology. In: Proc. RoboCup, pp 504–515

Luperto M, D'Emilio L, Amigoni F (2015) A generative spectral model for semantic mapping of buildings. In: Proc. IROS, pp 4451–4458

Menchetti S, Costa F, Frasconi P (2005) Weighted decomposition kernels. In: Proc. ICML, pp 585–592

Mozos O, Stachniss C, Burgard W (2005) Supervised learning of places from range data using AdaBoost. In: Proc. ICRA, pp 1730–1735

Mozos O, Triebel R, Jensfelt P, Rottmann A, Burgard W (2007) Supervised semantic labeling of places using information extracted from sensor data. Robotics and Autonomous Systems 55(5):391–402

Neufert E, Neufert P (2012) Architects' data. Wiley-Blackwell

Newman ME (2003) Mixing patterns in networks. Physical Review E 67(2):026,126–1 – 026,126–13

Oßwald S, Bennewitz M, Burgard W, Stachniss C (2016) Speeding-up robot exploration by exploiting background information. IEEE Robotics and Automation Letters 1(2):716–723

Perea Strom D, Nenci F, Stachniss C (2015) Predictive exploration considering previously mapped environments. In: Proc. ICRA, pp 2761–2766

Pronobis A, Jensfelt P (2012) Large-scale semantic mapping and reasoning with heterogeneous modalities. In: Proc. ICRA, pp 3515–3522

Pronobis A, Mozos O, Caputo B, Jensfelt P (2010) Multi-modal semantic place classification. International Journal of Robotics Research 29(2-3):298–320

Quattrini Li A, Cipolleschi R, Giusto M, Amigoni F (2016) A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings. Autonomous Robots 40(4):581–597

Shervashidze N, Schweitzer P, Van Leeuwen EJ, Mehlhorn K, Borgwardt KM (2011) Weisfeiler-Lehman graph kernels. The Journal of Machine Learning Research 12:2539–2561

Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8):888–905

Sjoo K (2012) Semantic map segmentation using function-based energy maximization. In: Proc. ICRA, pp 4066–4073

Solanas A, Garcia M (2004) Coordinated multi-robot exploration through unsupervised clustering of unknown space. In: Proc. IROS, pp 717–721

Stachniss C, Mozos O, Burgard W (2006) Speeding-up multi-robot exploration by considering semantic place information. In: Proc. ICRA, pp 1692–1697

The Whole Building Design Guide (2015) `https://www.wbdg.org`, [Online; accessed 29-September-2017]

van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. Journal of Machine Learning Research 9:2579–2605

Wurm K, Stachniss C, Burgard W (2008) Coordinated multi-robot exploration using a segmentation of the environment. In: Proc. IROS, pp 1160–1165

Zender H, Mozos O, Jensfelt P, Kruijff G, Burgard W (2008) Conceptual spatial representations for indoor mobile robots. Robotics and Autonomous Systems 56(6):493–502