Toward Smart Building Design Automation: Extensible CAD Framework for Indoor Localization Systems Deployment

Andrea Cirigliano, Roberto Cordone, Alessandro A. Nacci, and Marco Domenico Santambrogio, *Senior Member, IEEE*

***Abstract*—Over the last years, many smart buildings applications, such as indoor localization or safety systems, have been subject of intense research. Smart environments usually rely on several hardware nodes equipped with sensors, actuators, and communication functionalities. The high level of heterogeneity and the lack of standardization across technologies make design of such environments a very challenging task, as each installation has to be designed manually and performed *ad-hoc* for the specific building. On the other hand, many different systems show common characteristics, like the strict dependency with the building floor plan, also sharing similar requirements such as a nodes allocation that provides sensing coverage and nodes connectivity. This paper provides a computer-aided design application for the design of smart building systems based on the installation of hardware nodes across the indoor space. The tool provides a site-specific algorithm for cost-effective deployment of wireless localization systems, with the aim to maximize the localization accuracy. Experimental results from real-world environment show that the proposed site-specific model can improve the positioning accuracy of general models from the state-of-the-art. The tool, available open-source, is modular and extensible through plug-ins allowing to model building systems with different requirements.**

***Index Terms*—Indoor localization, Internet of Things, performance optimization, smart buildings design automation.**

I. Introduction

ON AVERAGE, people spend approximately 70% of their time indoors [1], such as in offices, schools, and at home. New indoor smart applications are being developed at high rate, in both research and commercial areas covering a wide range of personal and social scenarios. Smart buildings are becoming a reality with the adoption of an underlying monitoring and communication infrastructure composed by access points (APs), sensor motes, cameras, and smart devices integrated in a building management systems (BMSs). The BMS is a control system that monitors the building state and operates through actuators to increase the comfort and safety of occupants, while managing the energy efficiency at the same time.

Many smart buildings applications are based on indoor localization techniques, using location information to optimize the environment and provide context-aware services. Indoor localization systems often require the presence of wireless devices such as APs, in order to let the user identify its position by means of a mobile device. Most smart building applications have been developed in order to achieve sustainability, reducing energy waste related to energy-consuming appliances like heating, ventilation, and air-conditioning (HVAC). Some examples are [2] and [3]. Smart HVAC systems usually rely on a set of ambient sensors able to collect indoor values of temperature and humidity. This allows the control system to build thermal maps of the indoor environment, locate thermal complaint feedbacks coming from the tenants and regulate only the necessary portion of the physical system. Another target feature of complex buildings is safety, characterized by the ability to respond to crisis events limiting damages and victims. These systems are able to detect safety threats, for example from smoke detectors or heat detectors. Also in this scenario, a proper allocation of sensor nodes is essential to detect and locate the threat responsively.

The position of each node strongly affects the performance of the system, since a bad allocation could lead to unmonitored areas. The number of nodes employed, besides weighting on the installation cost, also burdens the overall energy consumption of the system, a key parameter to consider especially for energy saving systems. The choice of the hardware nodes can get more difficult by the availability on the market of several devices and components that differ in cost, power consumption and maximum range distance. Although the key role of nodes allocation, many smart building systems proposed in literature do not consider nodes amount and positioning problems in environments that differ from the original testbeds.

Without a systematic approach the design space is not well explored, which leads to inefficient solutions. In this context, the development of tools able to automatize part of the design flow of smart building systems is essential. In order to find a near-optimal allocation of nodes, the knowledge of the floor plan is required. However, for installations performed on existing buildings, administrators can encounter difficulties

Manuscript received September 14, 2016; accepted November 13, 2016. This paper was recommended by Associate Editor S. Mohanty.

A. Cirigliano, A. A. Nacci, and M. D. Santambrogio are with the Department of Electronics, Information and Bioengineering, Politecnico di
AQ1 Milano, 20133 Milan, Italy (e-mail: andrea.cirigliano@mail.polimi.it).

R. Cordone is with the Department of Computer Science, Universita' degli Studi di Milano, 20100 Milan, Italy.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2016.2638448

AQ2

TABLE I
COMPARISON BETWEEN PROPOSED DEPLOYMENT METHODS AND TOOLS FOR INDOOR WSN AND APs-BASED SYSTEMS

Deployment	Site Specific vs General Model	Heterogeneous Nodes	Application Integrated	Extensible
Zhao et al. [4]	General Model	No	Yes	No
He at al. [5]	General Model	No	No	No
Fang et. al. [6]	Site Specific Model	No	No	No
Proposed approach	Site Specific Model	Yes	Yes	Yes

in obtaining the floor plan in an easily-interpretable digital format.

To address these problems, we developed a computer-aided design (CAD) tool to assist building designers during the design of smart building systems. The application manages common requirements like the building floor plan specification. We decided to implement a node allocation algorithm for three different indoor localization systems, that searches for near-optimal allocations of nodes, from mixed hardware types, with the aim of keeping low the total cost. Due to the high level of heterogeneity and lack of standardization across systems to design, we make the system extensible through plug-ins to let new functionalities being integrated into the system. The tool[1] is developed within the QCAD[2] environment, an open-source computer-aided drafting application. The key contributions of this paper can be summarized as follows.

1) A traditional CAD interface to specify both physical building floor-plan and functional components of the smart environment.
2) An algorithm for hardware nodes allocation that provides to designers a near-optimal placement of devices. The algorithm explores combinations of different types of nodes to obtain cost-effective solutions.
3) A site-specific model for wireless indoor localization accuracy optimization that keeps into account the actual structure of the building.
4) The integration of the tool within an open-source[3] application framework able to extend the system by means of JavaScript or C++ plug-ins.

II. RELATED WORK

Building information modeling (BIM) is a consolidate process to support building constructions and renovations. BIM softwares, and in particular CAD for buildings such as ArchiCAD [7], focus on the generation and management of digital representations of the physical aspects of places. BIM tools can coordinate architectural and structural requirements, for essential tasks such as collision detection [8]. Materials employed for a construction can be represented with extremely high levels of accuracy, thanks to the several libraries developed in many years, resulting in precise cost estimations [9]. With the diffusion of integrated smart systems built to increase comfort and efficiency, buildings require the design of aspects that go beyond the mere physical design. The concept of smart environment is becoming more and more concrete with the integration of sensors, actuators and computational elements in buildings, while tools able to model smart and interactive functionalities of modern buildings are currently lacking.

The problem of the allocation of hardware nodes in a given environment can be compared, on first approximation, by the maximal cover location problem (MCLP), i.e., the problem of covering the maximum amount of demand locations with a given number of facilities. Similarly, the location set covering problem (LSCP) consists in finding the minimum set of facilities that covers all available demand locations. Each facility has the same coverage radius r; a demand point is assumed to be covered if it is within distance r of a facility. Daskin *et al*. [10], [11] gave a general formulation of the LSCP and reformulated it for network systems and emergency vehicle deployment.

The maximum sensing coverage region is a special case of the previous two problems that focuses on the research of an allocation of wireless nodes that guarantees both sensing coverage and network connectivity between nodes [12], [13]. In this scenario, the placement need to take care not only of the sensing range, but also of the communication range of each node.

For what concern the allocation in indoor environments, only minimum literature has been published so far to the best of our knowledge. Zhao *et al.* [4] proposed an AP positioning model based on the differential evolution algorithm, specific for fingerprinting localization techniques. Their model focuses on increasing the diversity of the received signal array along the indoor locations, and thus improving the positioning accuracy of fingerprinting schemes. However, the model does not take into account the effect of walls or other obstacles present in the target environment. He *et al.* [5] made use of a genetic algorithm for APs deployment model, to study the relationship between positioning error and signal space Euclidean distance. Again, the simulation results show that the error can be reduced increasing the Euclidean distance between the received signal strength (RSS) arrays of different locations. Fang and Lin [6] proposed a tool for linking the placement of APs and the positioning performance. Their algorithm maximizes signal-to-noise ratio, i.e., maximizes the signal and minimizes the noise simultaneously. However, the system is developed in a real-world environment, and requires measurements with different AP allocations that can be an expensive and time-consuming task.

A common limitation of many works described previously is the employment of simple and general models which does not take into account the actual layout and geometry of the building. The free-space path loss propagation model is often used

[1] A video demo of the tool has been published at https://youtu.be/6c6D6wolDBQ.
[2] QCAD—Open Source CAD System: http://www.qcad.org/.
[3] The source code of the system is open-source and available at https://bitbucket.org/necst/box-smartcad.

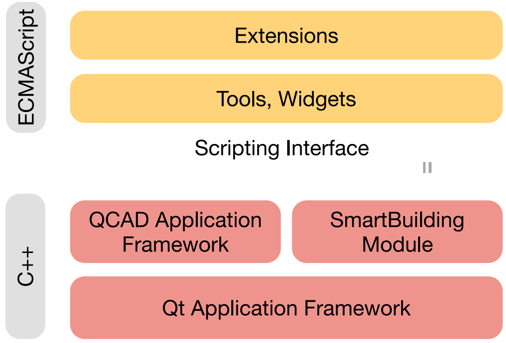

Fig. 1. Overview of the application stack. The script interpreter features standard ECMAScript functionality and on top of that provides additional classes from the Qt API, QCAD API, and the *SmartBuilding* module.

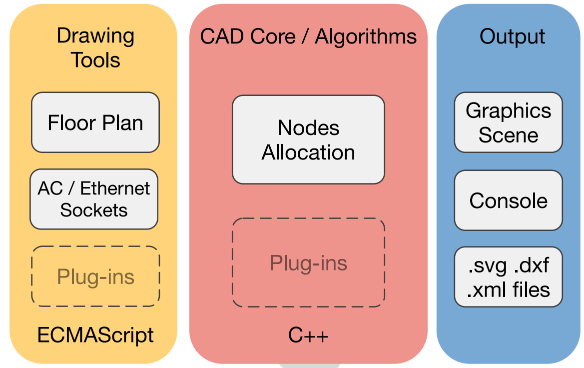

Fig. 2. Functional overview of the system components. Drawing tools and algorithms for systems deployment and simulation are extensible through ECMAScript or C++ plug-ins.

despite the presence of fixed obstructing objects like walls. Of course, none of the cited works provide a convenient way to specify geometric layout of the indoor environment. This leads the authors to validate models simply using squared or rectangular areas to represent the indoor environment, omitting the relationship between irregular areas and system coverage. In addition, none of the existing solutions takes in consideration different hardware characteristics and costs of the nodes to be deployed.

III. Proposed Application Framework

Our system has been developed on top of the QCAD application framework. The QCAD application framework consists of programming libraries and resources that provides CAD specific functionalities. An example of module provided by the QCAD application framework is the Math module that implements mathematical concepts such as vectors or matrices as well as basic geometrical classes like points, lines and so on. The QCAD Framework has been enhanced with a *SmartBuilding* module that provides some fundamental functionalities for the design of smart building systems. The module include abstract entities like rooms, walls, sockets, sensor nodes and gateways. User interface components are also provided in order to create and edit this entities (*tools*) and to specify parameters (*widgets*). Our module implements a node deployment algorithm for three commons indoor localization systems, that will be discussed later. The whole application rely on Qt, a framework that covers a lot of generic and low-level functionality for desktop applications and not directly related to CAD.

The QCAD application framework offers a very complete and powerful ECMAScript interface. The SmartBuilding module, as well as the QCAD application framework, is accessible through that scripting interface. Through the ECMAScript interface developers will be able to extend the whole application in an easy and very efficient way. The choice of a popular script language that is easy to learn enables anyone with previous programming experience to extend the application. Such extensions can for example be CAD related interactive tools like an HVAC layout construction widget, or a temperature sensor nodes deployment algorithm.

In some situations extending QCAD through scripts alone may not be possible. This is mostly the case, if the extension is based on an existing C or C++ library. In that case, it is possible to create a C++ plug-in that wraps the existing library and adds the necessary hooks to access library functionality through the script interface. Such a plug-in will be automatically loaded by QCAD on start up to add functions and classes to the script interface of QCAD. These script extensions can then be used by a script add-ons to make that functionality available as part of the application interface.

IV. Nodes Deployment for Indoor Localization

Smart environments always rely on a set of hardware nodes able to collect sensing data and communicate through cabled or wireless technologies. The number of nodes employed and the position of each one strongly affect the overall performance of the system as well as the cost of installation. In this paper, indoor localization systems have been taken as the main case study for the nodes allocation, since occupants localization and monitoring is one of the most common requirements of different smart environments.

The way in which the indoor environment must be covered by the nodes depends on the particular technology implemented; however, there can be identified three main manners.

1) Single coverage, i.e., to monitor the state of the environment with a single node for each location inside its radius. This includes for example to detect the presence of a mobile device in a proximity region [14], or to detect an RFID tag within the tags reader range [15].
2) Trilateration, to compute the position of a mobile device. This technique requires the reception of a wireless signal of at least three reference sensors with well-known positions everywhere within the covered area. We define the term k-coverage as the minimum number of sensors (or reference nodes) required in each location by a system. Single coverage systems have k-coverage $= 1$, while for trilateration $k = 3$.
3) Fingerprinting, where the number and the strength of the received signals is not fixed, but affect the localization accuracy.

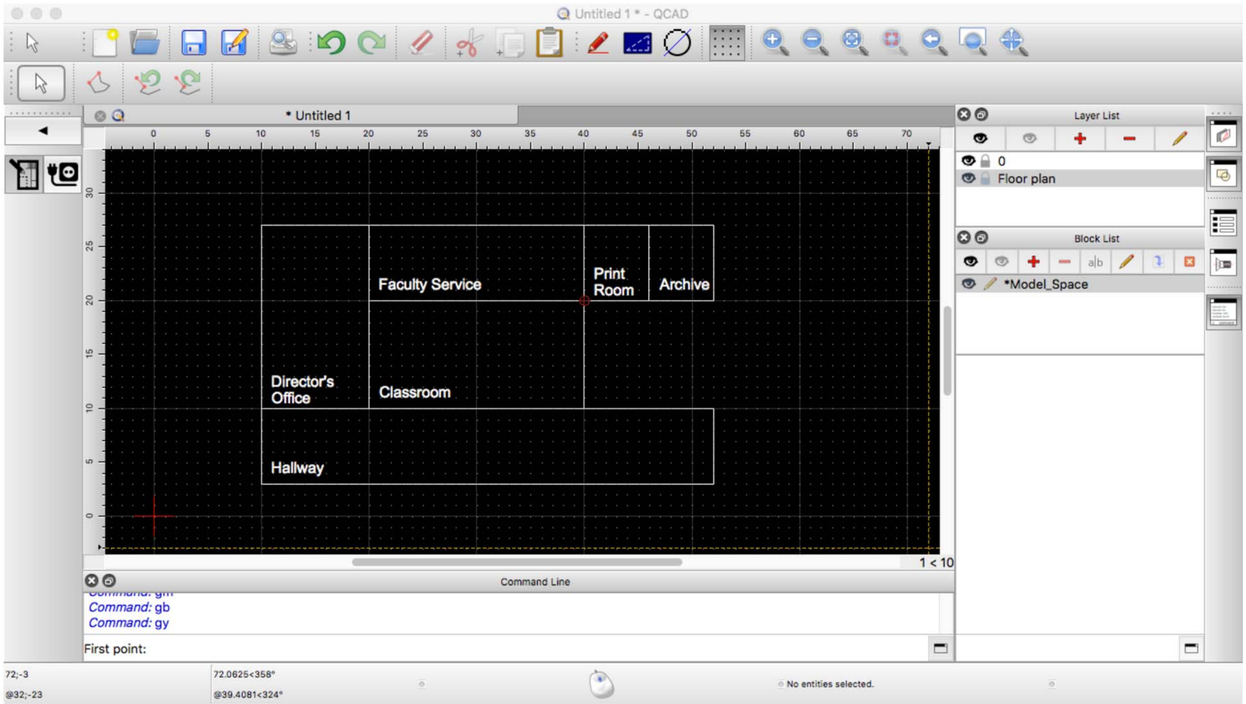

Fig. 3. Floor-plan design tool. User can specify the layout of the rooms and a possible set of candidate sites for the node placement.

Trilateration and fingerprinting usually exploit wireless technologies as Wi-Fi or Bluetooth to establish a connection between mobile and stationary nodes. Sensing regions can refer to any type of ambient sensors, such as passive infrared sensors [16], remote thermal sensors [17], but also proximity-based radio transmitters such as RFID tag readers [18] and Bluetooth low energy transmitters (BLE beacons) [19].

V. Proposed Deployment Tool

As we previously said, smart environments always rely on a set of sensor nodes, each one able to communicate through cabled or wireless technologies. Also for outdoor WSNs, a key challenge is how to achieve coverage of the target monitoring space and sufficient network connectivity between sensor nodes. Usually each sensor mote communicates with the rest of the network through technologies like Wi-Fi or ZigBee. Additional issues for outdoor WSNs are the limited battery life of each node and the power consumption required for packet transmissions. Given the availability in most (also "nonsmart") buildings of power outlets, Ethernet sockets and Wi-Fi signal, the mentioned limitations of WSNs can be solved in indoor application making use of the existing infrastructure. Differently from outdoor WSN deployments, where coverage and connectivity are always treated together, our system leaves nodes connectivity optional, focusing on providing the coverage service to the indoor locations.

The design process starts with a drafting phase in which the user specify the building floor plan as a set of rooms. During this phase the designer can restrict the possible sites for nodes allocation, selecting a set of candidate points. This can be useful when the hardware devices require power supply or Ethernet connectivity. The design interface used for both map and candidate sites specification is reported in Fig. 3.

In our model, we will refer to L as the entire set of monitoring locations to be covered, while J as the set of deployable locations where nodes can be placed. By default, $L = J$ and nodes can be positioned everywhere but as we said the set J can be restricted only to specific candidate points.

After the design phase, different parameters are provided by the administrator and used to define a domain in which search for a covering solution. The parameters are as follows.

1) The covering technique (single, trilateration, or fingerprinting) that will be used to cover the locations in L.
2) A cost c_t for every type $t \in T$ of node available on the market (expressed in dollars).
3) A working range r_t for every type t of node (expressed in meters).
4) A percentage of covered area required, called target (i.e., the minimum percentage of locations $l \in L$ to be covered by the solution).

The system will return to the designer a set N of nodes n_{jt} (possibly with mixed hardware types) and their position on the building map. The outcome will have the lower cost of installation among all the inspected solutions that satisfy the target percentage of covered area. Fig. 4 shows an overview of the process explained so far.

A. Covering Techniques

Our tool provides three different ways to cover the floor-plan space, each one identified by the technique required by the system that will be installed.

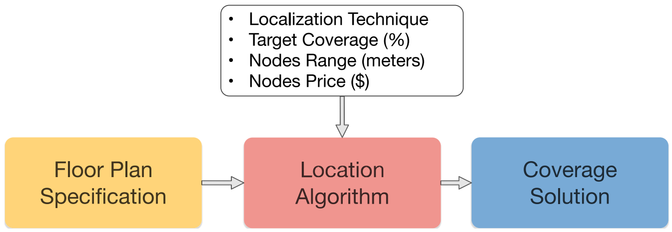

Fig. 4. System process. After the design of the floor plan, different parameters are used to define the search for an optimal allocation of nodes.

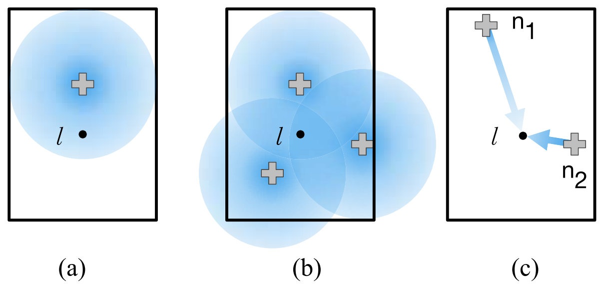

Fig. 5. Sample floor-plans with a location l covered (a) in single mode, (b) for trilateration, and (c) for fingerprinting where $\mathrm{rss}_{l,1} < \mathrm{rss}_{l,2}$.

1) Single coverage that guarantees from each position the presence of at least one reachable node. This is used for example to detect the presence of a mobile device in a proximity region. In our model, a location l of the floor-plan is considered covered if exists at least one working node n of type t within a range r_t. An example is shown in Fig. 5(a).
2) *Trilateration:* This is the process of determining the position of a point measuring its distance from three reference nodes, exploiting geometric properties of triangles. Usually, indoor trilateration systems use the strength of the signal received from a node to estimate its distance. In our model, a location l of the floor-plan is covered for trilateration if there exist at least three working nodes n_1, n_2, and n_3, each one no more distant then its corresponding range r_t. A location l served for trilateration is shown in Fig. 5(b). Although we refer only to trilateration, the same exact result can be used also for triangulation, the technique where angles are measured instead of distances.
3) *Fingerprinting:* This technique is used to estimate the position of a mobile device based on its rss vector. Each location receives the signal from k nodes, where k is not the same for all locations, but depends on how many nodes are reachable from that particular location. Each one of the k signals reaches the receiving antenna with a given power (or rss). For example, the location l shown in Fig. 5(c) perceives $k = 2$ signals so that $\mathrm{rss}_{l,1} < \mathrm{rss}_{l,2}$. We denote as $\mathrm{rss}_{l,n}$ the signal strength received at location l from a node n. The vector $\mathrm{rss}_{\mathbf{l}} = [\mathrm{rss}_{l,1}, \ldots, \mathrm{rss}_{l,k}]$ of the k signals received at run-time in location l is compared with a dataset of vectors, each one prelabeled with the corresponding position.

The comparison is usually performed by a classification algorithm using the Euclidean distance of the vectors, since rss vectors with a small Euclidean distance between them are more likely to be close also in the physical space. We have defined as $\mathrm{rss}_{l,n}$ the signal strength received at location l from a node n. The Euclidean distance between $\mathrm{rss}_{\mathbf{a}}$ and $\mathrm{rss}_{\mathbf{b}}$, both composed by k received signals, and collected, respectively, in location a and b is defined as

$$E(a, b) = \sqrt{\left(\mathrm{rss}_{a,1} - \mathrm{rss}_{b,1}\right)^2 + \cdots + \left(\mathrm{rss}_{a,k} - \mathrm{rss}_{b,k}\right)^2}. \quad (1)$$

Consider the vector $\mathrm{rss}_{\mathbf{a}}$ as the run-time sample, while the vector $\mathrm{rss}_{\mathbf{b}}$ retrieved from the stored fingerprint. The smaller is the $E(a, b)$, more confident is the localization system approximating current location of a with the stored location of b.

It has been demonstrated that maximizing the Euclidean distances of the rss arrays between all sampling points, the positioning accuracy of wireless localization systems can be improved [4], [5]. Fig. 6 is reported a graphical demonstration of the aforementioned statement. Take as an example a dataset (DS1, DS2, DS3, DS4) of stored rss vectors, where each vector is bi-dimensional ($K = 2$) and coupled with the corresponding physical position. Fig. 6(a) shows each element of the database where the Cartesian coordinates corresponds to components $\mathrm{rss}_1, \mathrm{rss}_2$. Although the plane does not represent the physical area of the floor-plan, database elements that are near between them are more likely to be close also in the physical space. Given a run-time element R, each arrow represents the Euclidean distance $E(R, \mathrm{DS}_i)$ from the surrounding dataset elements. A localization algorithm can exploit the Nearest Neighbor technique to approximate the position of R with the nearest dataset element. Unfortunately, the run-time rss measurement of R will not be constant over time, but will experience continuous fluctuations due to environmental noise. These fluctuations make the sample R move randomly to the surrounding points. Suppose that DS2 is the nearest points to R in the physical space. Fig. 6(b) shows with a green area the probability to assign R the correct (or more accurate) position, while a red (with line pattern) area represents the probability to get a wrong position from the system. Fig. 6(c) demonstrates how an increase in the rss Euclidean distance between sampling points increase the red area and the accuracy of the localization, while in Fig. 6(d) an Euclidean distance reduction will lead to poorer localizations.

The RSS has been estimated using the The WINNER II path loss model [20]

$$PL = A\ \log_{10}(d[m]) + B + C\log_{10}\left(\frac{f_c[\mathrm{GHz}]}{5.0}\right) + X \quad (2)$$

where PL is the signal path loss (in dB), f_c is the frequency in GHz, and d is the distance between the transmitter and the receiver location in meters. Values of coefficients A, B, C, and X change depending on line-of-sight (LOS) or nonline-of-sight (NLOS) propagations, and are reported in Table II. The propagation model has been used in fingerprinting coverage to maximize the Euclidean distance of the rss vectors between a location and its surrounding points, with the aim of improve the localization accuracy of the system.

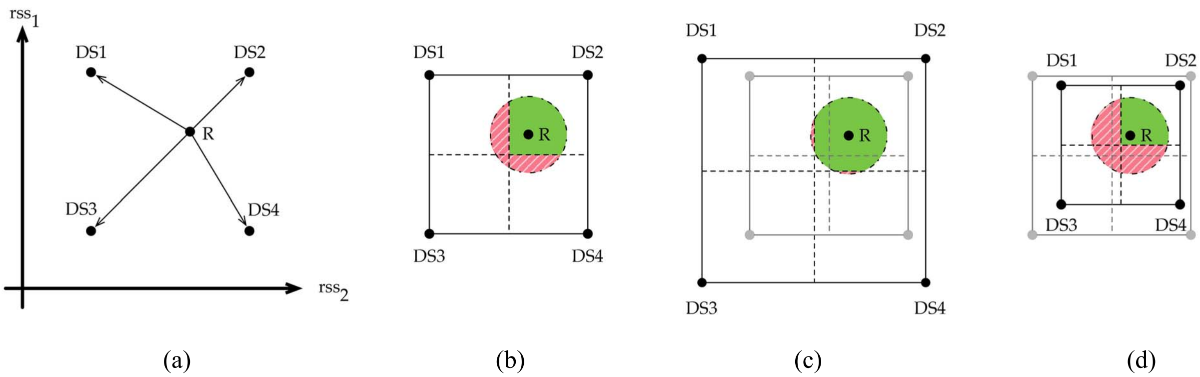

Fig. 6. (a) Bi-dimensional elements of the localization dataset are represented in Cartesian coordinates corresponding to components rss_1 and rss_2. A run-time sample *R* is shown in (b) where its circular area delineates run-time signal fluctuations. If DS2 is the nearest points to *R* in the physical space, green area is proportional to the probability of correct localization, while red dashed area represent wrong localizations. (c) Euclidean distance between sampling points has been increased, improving the correct localization. (d) Opposite effect.

TABLE II
VALUES OF COEFFICIENTS DEPENDING ON LOS OR NLOS PROPAGATIONS. VALUES HAVE BEEN TAKEN FROM THE WINNER II PATH LOSS MODEL [20]

Scenario	Path Loss Coefficients
LOS	$A = 18.7, B = 46.8, C = 20$
$NLOS$	$A = 36.8, B = 43.8, C = 20$ $X = 5(n_w - 1)$ (light walls) $X = 12(n_w - 1)$ (heavy walls)

The 2-D space of the floor plan is discretized with a length unit (default is 1 m) that is chosen by the user during the map specification phase.

As we have said, in addition to location coverage, also nodes connectivity has been modeled. In our model, a sensor node *n* is connected if exist a connected path to the gateway node. To ensure the connectivity of the whole network, the following equation must hold:

$$\forall n \in N, \text{connected}(n, \text{gateway}) = \text{true} \tag{3}$$

where

$$\begin{aligned} \text{connected}(n, n') &\stackrel{\text{def}}{=} \left|(n, n')\right| \leq \min(h, h') \\ &\vee \exists\, n_1, \ldots, n_i \in N \ (1 < i) \\ &\quad |(n, n_1)| \leq \min(h, h_1) \\ &\quad \wedge |(n_1, n_2)| \leq \min(h_1, h_2) \ \wedge \ \ldots \\ &\quad \vee \left|(n_i, n')\right| \leq \min\left(h_i, h'\right). \end{aligned} \tag{4}$$

Connected networks are managed by our allocation algorithm in the same way of nonconnected networks, with the following exception.

1) First, a manual gateway nodes allocation is required.
2) During nodes allocation, deployable points J are restricted to locations j' such that $\text{connected}(n_{j'}, \text{gateway}) = \text{true}$.
3) During deployment optimization, nodes moves are considered feasible only within the connected area.

VI. COVERING LOCATION ALGORITHM

The covering location algorithm has the purpose of placing an optimal set of nodes on the building floor plan.

TABLE III
NOTATION AND MEANING OF SYMBOLS USED FOR THE MODEL

Notation	Meaning
L	set of monitoring locations
J	set of deployable locations
c_t	cost of a node of type t
r_t	sensing range of a node of type t
h_t	communication range of a node of type t
$target$	coverage rate of L required by user (%)
n_{jt}	nodes of type t allocated in j
$rss_{l,n}$	signal strength received in l from n
$\mathbf{rss_a}$	vector of all the $rss_{a,n}$ values collected in a
$E(a,b)$	Euclidean distance between $\mathbf{rss_a}$ and $\mathbf{rss_b}$
D_l	set of locations no more distant than d from l
z	average signal space Euclidean distance
Z	objective function
b_l	reward earned for covering location l
w_l	reward weighted on the node cost
x_{jt}	allocation of node with type t in j (binary)
a_{ljt}	reachability of n_{jt} from location l (binary)
k-coverage	number of ref. nodes required by the system
k_l	current number of ref. nodes covering l
S	min. signal space Euclidean distance threshold
s_{min}	minimum number of node moves in *shaking* procedure
s_{max}	maximum number of node moves in *shaking* procedure
$Rmax$	number of restarts of the *VNS* algorithm

We have decided to implement a modified version of the multimode covering location problem [21], a generalization of the MCLP. Using a quite general and flexible reformulation of the covering problem, we have been able to adapt the algorithm at the different covering techniques described previously.

The positioning algorithm is composed by a first Greedy procedure, whose solution is then improved by a variable neighborhood search (VNS) algorithm. The positioning algorithm evaluates different solutions using a reward b_l, that is defined for each location l and will be earned only for the locations covered in that particular solution. The value of the reward depends on the coverage technique.

1) *Single Coverage:* The reward b_l will be earned if there is at least one node that covers l.
2) *Trilateration:* The reward b_l will be earned if there are at least three nodes that cover l.

$$\mathbf{rss_l} = \langle rss_{l,1}, rss_{l,2} \rangle = \langle -84, -72 \rangle \; [dB]$$
$$\mathbf{rss_s} = \langle rss_{s,1}, rss_{s,2} \rangle = \langle -67, -41 \rangle \; [dB]$$
$$E(l,s) =$$
$$= \sqrt{(rss_{l,1} - rss_{s,1})^2 + (rss_{l,2} - rss_{s,2})^2} =$$
$$= \sqrt{(-84 - (-67))^2 + (-72 - (-41))^2} =$$
$$= 35.36 \; [dB]$$

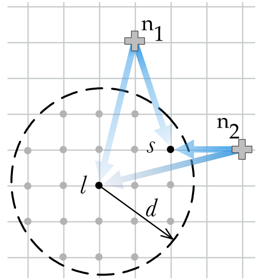

Fig. 7. Regular grid showing how is computed the mean Euclidean distance between the received rss vectors in a certain location l, and the surrounding locations s within a certain distance d.

3) *Fingerprinting:* Since this technique is often considered to be a tradeoff (in cost and accuracy) between single coverage and trilateration, we decided that the reward b_l will be earned if there are at least two nodes that covers l.

As we have said, in order to maximize the localization accuracy of the system it is possible to increase the signal space Euclidean distance between the target points. Consider the mean Euclidean distance between the received rss vector in a certain location l, and the surrounding locations s within a certain distance d

$$\frac{1}{|\, D_l \,|} \sum_{s \in D_l} E(l,s)$$
$$D_l = \{s \in L \mid \text{distance}(l,s) \leq d\}. \tag{5}$$

The distance d is used to restrict the rss comparison and diversification only to the locations that are more likely to be erroneously confuse with l by the localization system. Fig. 7 shows an example of how the Euclidean distance of a location is compared to a neighbor location.

We define the average signal space Euclidean distance z

$$z = \frac{\sum_{l \in L} \sum_{s \in D_l} \frac{E(l,s)}{|D_l|}}{|L|}. \tag{6}$$

The term z will be used by the Greedy procedure to produce a first solution with a reasonable allocation of nodes. Then, the value of z should be increased as much as possible to provide good localization accuracy to the system. However, maximize only the average does not seems fair enough, since a good system should provide a certain level of accuracy homogeneously among the target area. So we defined the objective function as difference between the term z and the signal space Euclidean variance

$$Z = z - \sqrt{\sum_{l \in L} \left(\sum_{s \in D_l} \frac{E(l,s)}{|\, D_l \,|} \right)^2}. \tag{7}$$

Maximizing the objective function Z, the intention is to provide as many target location as possible with a high signal space Euclidean distance with respect to the surrounding locations.

As we have previously introduced, we represent with L the entire set of location to be covered, while with J the set of possible positions where nodes can be placed. By default, $L = J$ and nodes can be positioned everywhere; however, its possible to restrict the J set only to specific candidate points, that represent for example power outlets or Ethernet sockets. The problem of find a near-optimal set N of nodes n_{jt} (each one located in j and having a type t) with a coverage rate $f(N)$ that satisfies the target coverage, can be formalized as follows:

$$\max Z = z - \sqrt{\sum_{l \in L} \left(\sum_{s \in D_l} \frac{E(l,s)}{|\, D_l \,|} \right)^2} \tag{8}$$

$$f(N) \geq \text{target} \tag{9}$$

$$\sum_{t \in T} x_{jt} \leq 1 \quad \forall j \in J \tag{10}$$

$$x_{jt} = 1 \quad \Longleftrightarrow n_{jt} \in N \tag{11}$$

$$f(N) = |L| / \sum_{l \in L} y_l \tag{12}$$

$$\begin{cases} y_l \leq \sum_{j \in J} \sum_{t \in T} a_{ljt} x_{jt} & \forall l \in L \; (\text{single}) \\ 2 \; y_l \leq \sum_{j \in J} \sum_{t \in T} a_{ljt} x_{jt} & \forall l \in L \; (\text{fingerprinting}) \\ 3 \; y_l \leq \sum_{j \in J} \sum_{t \in T} a_{ljt} x_{jt} & \forall l \in L \quad (\text{trilateration}). \end{cases} \tag{13}$$

The decision variable $x_{jt} = 1$ represents the allocation of a node of type t in location j; a_{ljt} is equal to 1 if location l can be reached by a node of type t placed in j, and $a_{ljt} = 0$ otherwise. $y_l = 1$ if location l is covered, $y_l = 0$ otherwise. The constraint (10) fixes to one the maximum number of nodes that can be located in each site.

A. Greedy Procedure

The positioning algorithm starts with a Greedy procedure with the purpose of find a reasonable number of reference nodes, for both coverage and localization accuracy. The procedure generate a first solution N positioning a set of $k = |N|$ nodes, each one with a type $t \in T$. For all three coverage techniques, the reward b_l is weighted with the cost of the current node n^* selected for the coverage

$$w_l = \frac{b_l}{c_t}; \quad \left\{ n^* = n_{jt} \; \wedge \; \text{distance}(j,l) \leq r_t \right\}. \tag{14}$$

The weighted reward w_l will be used by the Greedy algorithm so that on equal covered area, the cheapest node type has the priority over the others. We denote as L_{jt} the subset of locations that are reachable by a reference node n of type t placed at location j. At each iteration, the algorithm places a node n of type t^* at position j^* that covers the subset of locations $L_{j^*t^*}$ with the maximum reward. The term

$$1 - \frac{k_l}{k - \text{coverage}} \tag{15}$$

is used to prioritize the covering of locations with a lower "temporary" k-coverage (called k_l) with respect to the k-coverage required by the current techniques. In this way, Greedy procedure tends to avoid the placement of nodes very

Algorithm 1 Greedy(L, J, T, w, target)

$N := \varnothing$;
$L_{jt} := \{l \in L \mid l$ is covered by node in j with type $t\}$;
while $(f(N) < target) \wedge (z < S)$ **do**
  $j^* := \arg\max_{j \in J} \sum_{l \in L_{jt}} w_l \; (1 - \frac{k_l}{k\text{-}coverage})$;
  $t^* := \arg\max_{t \in T} \sum_{l \in L_{jt}} w_l \; (1 - \frac{k_l}{k\text{-}coverage})$;
  $N := N \cup \{n_{j^* t^*}\}$;
  $L_{jt} := L_{jt} \setminus L_{j^* t}$ for all $j \in J$;
return N;

close to one other which can lead, especially for trilateration systems, to poor localization accuracy. It is important to notice that the purpose of the Greedy procedure is to find a reasonable number of nodes for the localization service. The starting positioning is made on a best-effort basis, that will be improved by the successive VNS. After a node allocation, all subsets L_{jt} are updated according to the coverage technique. In trilateration for example, a location l is removed from L_{jt} only if there exist, other than the current $n_{j^* t^*}$, other two nodes that are already covering l.

The Greedy procedure ends when the target coverage is satisfied, and when the average signal space Euclidean distance z reaches the threshold S. In our implementation we set the threshold $S = 4.5$ that has been proven to be the average Euclidean distance for which the positioning error is limited to 2 m [5]. How we will see in Section VII, the Greedy procedure is able to provide an average Euclidean distance not so far from the final best known. However, thanks to the low complexity of the Greedy procedure, additional time can be used to improve the solution. In addition, the Euclidean distance variance will be strongly improved.

B. Variable Neighborhood Search

The method called VNS has been used to improve the solution coming from the Greedy procedure. The VNS approach empowers the classical local search framework with a restart mechanism that extends the search after a local optimum has been achieved by generating new starting solutions in progressively enlarged neighborhoods of the current best known solution. The key elements of the VNS (reported in Algorithm 2) are a starting solution N with a hierarchy of size-increasing neighborhoods, and a local search procedure, i.e., the criterion to select the incumbent solution from the neighborhood. These components are used to restart the search every time that the procedure reaches a local optimum. Fig. 8 shows an overview of the VNS process. A first local search procedure is applied to the solution produced by the Greedy procedure. At each iteration, the *shaking* procedure is used to generate a new starting solution, which is then improved by the execution of the local search. The shaking procedure perturbs s node allocations of the current solution N^* replacing them with s unused nodes. The behavior of the shaking parameter s, that depends on the result of the local search, is explained in Fig. 9. The parameter s starts from a minimum value $s_{\min}$ (in the example $s_{\min} = 1$) and every time that the local search does not improve the best known solution, s is increased by 1. Differently, when the local search succeeds, the best solution N^* is updated and s goes back to $s_{\min}$.

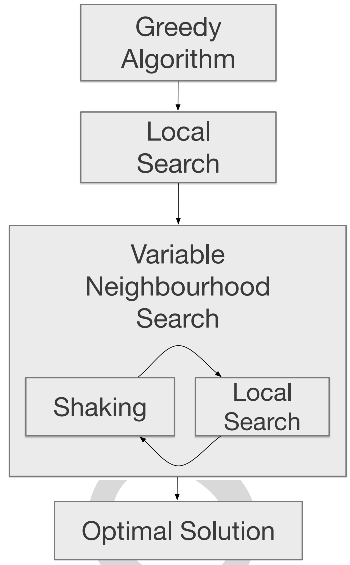

Fig. 8. Location algorithm. The solution found by the *Greedy* algorithm is improved applying iteratively a *Local Search* for an optimal solution and a *Shaking* procedure that perturbs the current solution.

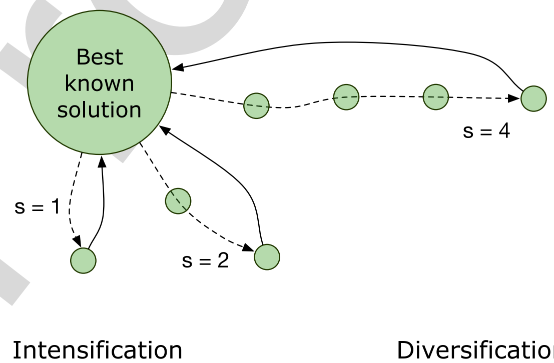

Fig. 9. Shaking procedure: the parameter s is increased when the solution does not improve (dashed line) and restarts when a new optimum is found (continuous line).

The purpose of the shaking procedure is to first explore new starting solutions that are more similar to the best known result, so that the search is *intensified* in a promising neighborhood of the entire domain. If these local searches fail, the shaking procedure moves the search from intensification to *diversification*, generating starting solutions that are more and more different from the incumbent one. Whenever a new best solution is found, the shaking procedure comes back to $s_{\min}$, to intensify the search near the just updated N^*. In principle, the shaking parameter s can be increased until $k = |N^*|$, changing all the node allocations. However, we experimented running different configurations that excessively moving away from the best known solution can be unproductive, causing a useless waste of computational time. We have fixed a reasonable value of $s_{\max} = \lfloor (2/3)k \rfloor$.

Algorithm 2 VNS(L, J, T, w, target, $s_{\min}, s_{\max}, R_{\max}$)

$N := Greedy(L, J, T, w, target)$;
$N^0 := LocalSearch(L, J, T, w, target)$;
$N^* := N^0$;
$s := s_{min}$;
for $r := 1$ to R_{max} **do**
    $N := Shaking(N^*, s, L, J, T, w, target)$
    $N^0 := LocalSearch(L, J, T, w, target)$
    if $(Z(N^0) > Z(N^*))$ **then**
        $s := s_{min}$;
        $N^* := N^0$;
    else
        $s := s + 1$;
        if $(s > s_{max})$ **then**
            $s := s_{min}$;
return N^*;

The VNS algorithm terminates when the total number of restarts reaches a given value $R_{\max}$.

As we have said, the local search is the heuristic that proceeds from an initial solution to its neighborhood by a sequence of local changes, trying to improve each time the value of the objective function until a local optimum is found. The neighborhood of the adopted approach is given by cyclic sequences of moves, where each move consists in locating a new node, removing a node or changing the type of the node. A cyclic move is considered feasible only if the new covering rate respects the target coverage, and the total cost of the solution does not increase. Of course, each site must continue to hosts at maximum one node [constraint (10)]. A cyclic move can be visualized on a graph $G = (N, A)$, where each node of the graph is a possible allocation of a hardware node. Each node of the graph is characterized by a location j, and a state that indicates if the node is active or inactive. A node n_{jt} currently allocated in location j, is represented on the graph with an active node n_j, labeled with its hardware type t. Note that index t does not appear because at most one type can be active in each node, and the type is specified by the label. Inactive nodes are instead left unlabeled. An arc (n_j, n_k) can represent the following.

1) The allocation of a hardware node in site j, if n_j is inactive and n_k is active.
2) The removal of a hardware node in site j, if n_j is active and n_k is inactive.
3) An hardware node n_j changing its hardware type, if both nodes are active.

In both 1) and 2), the new node takes the hardware type of the head label (t of n_k). A cyclic exchange corresponds to a directed cycle on the improvement graph, as depicted in Fig. 10. Each move, and so each arc (n_j, n_k), determines a variation δZ in the value of the objective function Z. The purpose is to represent a group of moves so that a cyclic exchange represents an increase in the current objective function. However, the total variation δZ is non additive with respect to the sequence of δZ values coming from single moves. This is caused by the interdependence between different hardware nodes with overlapping covering regions, that lead to nonadditive moves. To overcome this drawback, every cycle has been evaluated using an own temporary function Z' updated step by step from the end of the path to its starting node. In this way, all the cycles with a positive total weight bring improvements on the starting solution.

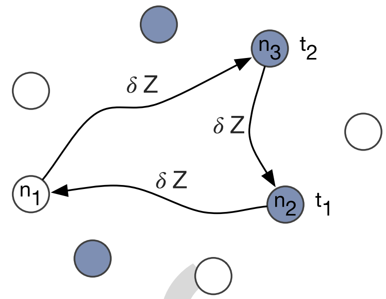

Fig. 10. Improvement graph: colored nodes represent current allocations, while empty nodes are possible allocations. All active nodes are labeled with their corresponding type. Each arc is a change (move) on the allocations.

The search for the cyclic exchange with maximum weight is performed with exhaustive breadth-first exploration of the paths of graph G.

VII. Experimental Results

Presented experimental results are initially focused on the usability of the tool, testing the ability to provide a solution in a reasonable time. Then, the performances of the model have been evaluated, in terms of localization accuracy through realistic indoor environment experiments, and in terms of cost-effectiveness of the suggested deployments.

A. Computational Experience

The tool has been evaluated running several different configurations. Every test reported in this section has been executed with a spatial resolution of the floor plan equal to 1 m. A first analysis can be done on the execution times of the proposed solution. Although the execution time can be tuned by the parameter $R_{\max}$, which represents the maximum number of restarts of the VNS algorithm, an idea on the order of magnitude is given by Fig. 11, where the time is represented as a function of the floor-plan dimension. In the given example, $R_{\max}$ has been fixed to 20 restarts, the target coverage equals to 95% of the total area, a single node type available with a range of 8 m, covering floor-plans with rectangular areas. The graph shows that for single coverage and fingerprinting the processing time grows approximately linearly with the floor plan area.

A numeric comparison of the same tests is reported in Table IV, where execution times are reported in seconds for increasing floor plans. For single coverage, the execution time is low even for areas of 3000 squared meters. For trilateration and fingerprinting, the execution times become high from floor-plan of 2500 m^2. However, the tests represent a bad case in which the map dimension is very large while the node range available and the spatial resolution are small (respectively,

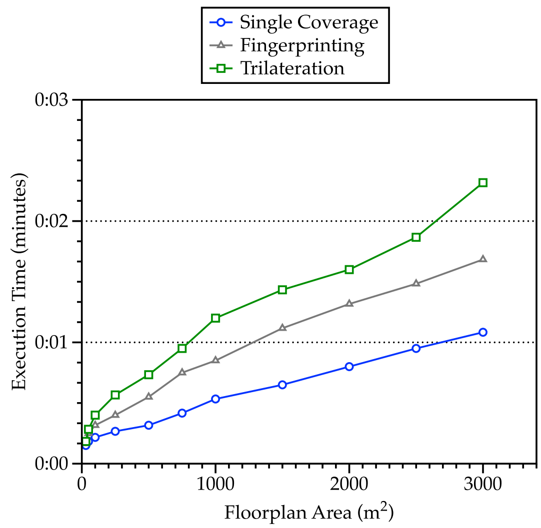

Fig. 11. Execution time of the tool with floor plans of different areas, for each covering technique ($R_{\max} = 20$, target = 95%, and $r_t = 8$).

TABLE IV
EXECUTION TIME OF THE TOOL FOR INCREASING FLOOR PLAN AREAS ($R_{\max} = 20$, target = 95%, AND $r_t = 8$)

Floor Plan Area (m^2)	**Execution Time (s)**		
	Single	Fingerprinting	Trilateration
30	9.07	10.53	11.49
50	11.31	15.10	17.67
100	13.05	19.41	24.22
250	16.57	24.89	34.16
500	19.18	33.48	44.66
750	25.43	45.32	57.94
1000	32.19	51.68	72.83
1500	39.37	67.12	86.27
2000	48.30	79.49	96.11
2500	57.11	89.47	112.34
3000	65.41	101.24	139.18

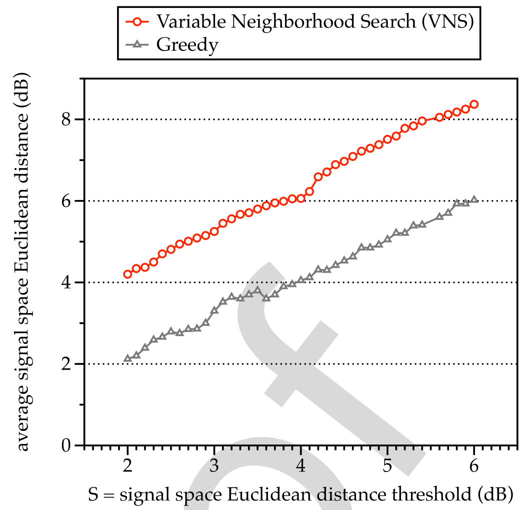

Fig. 12. Average signal space Euclidean distance (z) obtained with the Greedy execution and compared with the z value after the VNS optimization. z values expressed as a function of the threshold S. Floor-plan area = 2500 m^2, $R_{\max} = 20$, target = 100%, and $r_t = 12$.

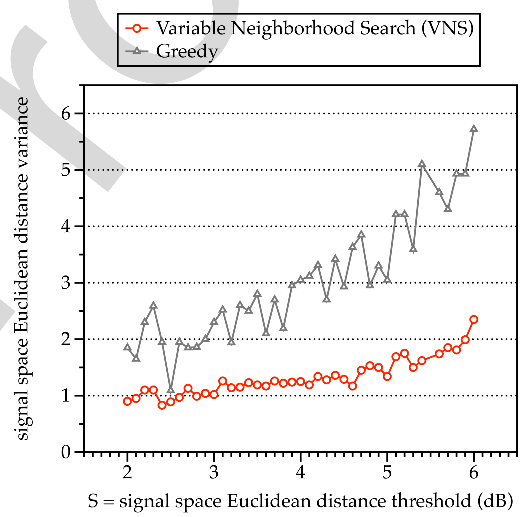

Fig. 13. Signal space Euclidean distance variance obtained with the Greedy execution and compared with the z value after the VNS optimization. Values expressed as a function of the threshold S. Floor-plan area = 2500 m^2, $R_{\max} =$ 20, target = 100%, and $r_t = 12$.

8 and 1 m). Increasing the range or the resolution, the instance of the problem decrease, resulting in faster executions.

A key aspect that characterizes the goodness of the proposed approach is the improvement of the objective function achieved by the VNS algorithm with respect to the first Greedy configuration. For this test we have run the tool several times with a floor-plan area of 2500 m^2 and a node range of 12 m. The number of reference nodes allocated is determined by the Greedy procedure and increase with S, while the number of VNS restarts $R_{\max}$ has been fixed to 35.

In Fig. 12, we reported the value of z, i.e., the average signal space Euclidean distance obtained with the first Greedy execution, compared with the z value after the VNS optimization. The graph reports the z values as a function of the threshold S, described in Section VI-A as the minimum value of average signal space Euclidean distance (z) required during the Greedy procedure. The graph shows that moving the threshold within the range (2, 6)dB the VNS is able to improve the z value constantly around 2 dB. Although the VNS improvement is not astonishing for what regard the average value, Fig. 13 shows that the variance is strongly improved. This has been achieved moving from the objective function z used in Greedy procedure to the Z function of the VNS. The Z objective function has in fact the purpose to provide as many target location as possible with a high signal space Euclidean distance w.r.t. the surrounding locations.

B. Experimental Setup and Accuracy Evaluation

The proposed tool was evaluated using data collected from a real-world environment, the NECST Lab, located at the basement of DEIB Department at the Politecnico di Milano.

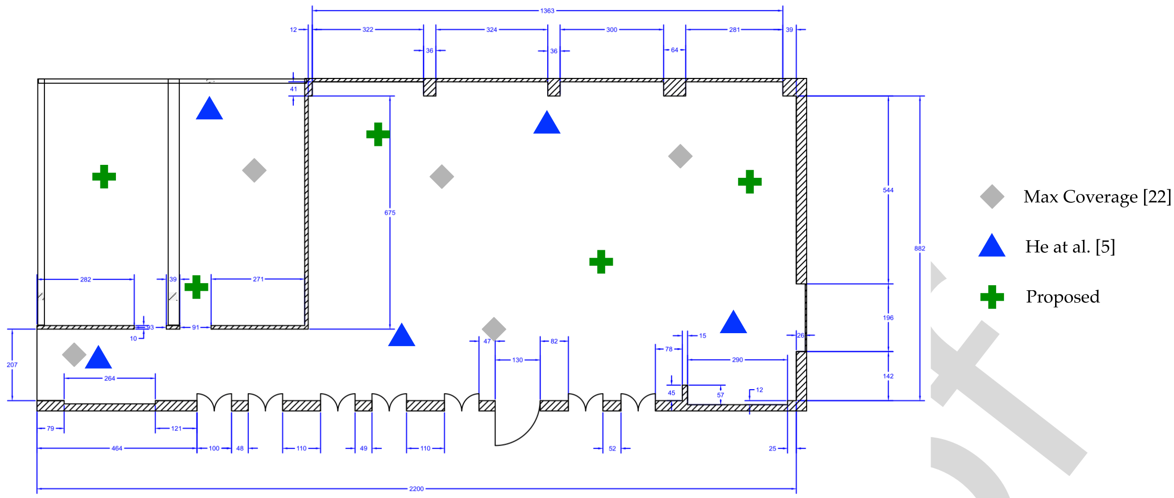

Fig. 14. NECST Laboratory floor-plan, located at the basement of DEIB Department at the Politecnico di Milano. Each allocation corresponds to a BLE beacon with a range of 7 m. Green crosses indicates allocations provided by our algorithm, gray rhombus represent allocations from [5] while blue triangle positions have been computed maximizing the coverage [22].

The dimension of the test-bed is 198 squared meters (9×22 m). We collected BLE signal data coming from BLE beacons with a coverage radius of 7 m. Signal data has been collected using a Nexus 5 smartphone running Android 6.0.1. First, the NECST Laboratory floor-plan has been designed using our tool, obtaining the optimal number of beacons ($|N| = 5$) and their allocation for fingerprinting localization. $R_{\max}$ has been fixed to 20 restarts, the target coverage equals to 100% of the total area, a single node type available with a range of 7 m, and the threshold $S = 4, 5$. We collected 40 training samples for the localization algorithm using the obtained allocation. Then, the test samples were collected at distinct positions changing the phone orientation and the way in which user was keeping it, for example by hand or in a pocket. For the entire duration of training and test phase, the number of occupants and their enabled wireless devices has changed, from a minimum of 3 to a maximum of 17 people. This variation affects the accuracy performances, but at the same time contributes in obtaining realistic results. The training and test phase has been repeated with two configurations coming from different allocation algorithms: maximization of the coverage [22] and the allocation algorithm proposed by He *et al.* [5]. For these two algorithms, the number of employed nodes has been fixed to 5. KNN with $K = 3$ has been employed as the fingerprinting algorithm.

A first result is shown in Fig. 15. The cumulative error distribution function shows that from 1.5 m our approach performs better. Under 1.5 m, He *et al.* [5] approach performs better, but the difference in accuracy is marginal.

Fig. 16 shows the mean positioning accuracy divided into different error ranges: (0, 0.5], (0.5, 1], (1, 1.5], (1.5, 2], (2, 2.5], (2.5, 3], (3, 3.5], and (3.5, 4]. It is possible to notice that the majority of the localization errors appears within the (1.5, 2] m. The test-bed floor-plan, composed by three rooms, has been reported in Fig. 14. Green crosses indicates allocations provided by our algorithm, gray rhombus represent allocations from [5] while blue triangle positions have been computed maximizing the coverage [22].

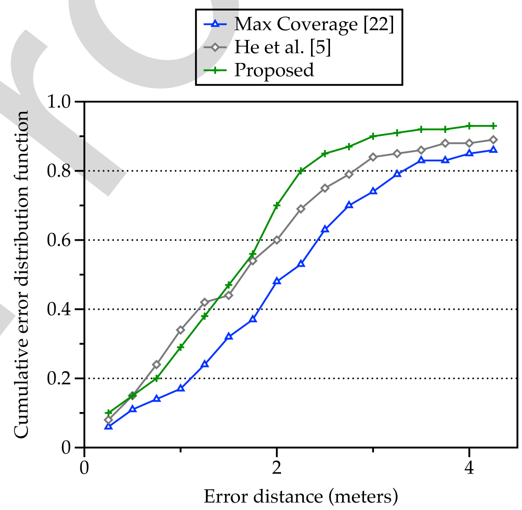

Fig. 15. Cumulative error distribution function experienced by our approach ad compared with two different solutions from the state-of-the-art.

C. Cost-Effectiveness Analysis

A feature of our tool interesting for testing is the possibility to obtain solutions from mixed node types, with different characteristics and costs. In particular, given two types t_1 and t_2 characterized by two ranges r_i, and two costs c_i, it is possible to compare the total cost of a homogeneous solution with the cost of a mixed solution. Given a baseline type of node with a range $r_1 = 8$ m and a cost of $c_1 = 60$ \$, we can assume the presence on the market of a second type of hardware, with the half of the range distance ($r_2 = 4$ m). The area covered by t_1 (≈ 200 m^2) is four times bigger than the coverage of t_2 (≈ 50 m^2). In order to obtain a fair test, the cost of t_2 should be $c_2 \geq c_1/4$, and so we set $c_2 = 20$ \$. This test has been performed with a target coverage of 95% on a rectangular map of 1000 m^2.

From Table V, it is possible to observe that, although hardware nodes of type t_2 have a lower convenience in terms of

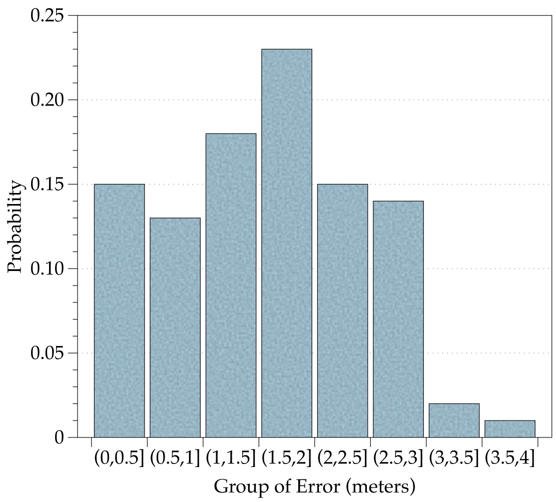

Fig. 16. Mean positioning accuracy of the proposed allocation algorithm divided into different error ranges.

TABLE V
COST OF HOMOGENEOUS AND MIXED SOLUTIONS ($A = 1000$ m^2, target = 95%, $r_1 = 8$ m, $r_2 = 4$ m, $c_1 = 60$ \$, AND $c_2 = 20$ \$)

Node types	Solution Costs (in \$)		
	Single	Trilateration	Fingerprinting
$T = \{t_1\}$	480	1440	840
$T = \{t_2\}$	500	1620	880
$T = \{t_1, t_2\}$	440	1280	760

(*area/price*) (t_1 outperform t_2 in homogeneous solutions), the mixed strategy can use the smaller range nodes to reduce the total cost. This because less powerful nodes of type t_2 are employed to cover small portions of the floor-plan, like corners or small regions left uncovered by the larger range nodes.

The amount of saving in the total cost of the mixed solution does not depend only on the nodes range and price, but also on the irregularity of the floor plan perimeter. A distinguish feature of the proposed tool respect to other works is the possibility to cover spaces that are not necessarily rectangular or squared. The level of irregularity of a floor plan can be identified by the minimum number of rectangles that compose the shape. In Fig. 17 for example, the index of the floor plan irregularity is $I = 4$. We experimented the behavior of the tool increasing the level of irregularity, while maintaining a constant total area of 1000 m^2. The test has been done with the same nodes configuration used in Table V (homogeneous $T = t_1$, mixed $T = t_1, t_2$). The results shown in Table VI proven that increasing the floor-plan irregularity, the cost difference between homogeneous and mixed solution becomes higher. This is caused by the increasing number of corners in the map, that can be covered with less powerful nodes.

In conclusion, experimental results show that for most of the problem instances, a solution can be obtained in reasonable execution times. Depending on the available hardware types, homogeneous solutions could be improved with the employment of different type of nodes.

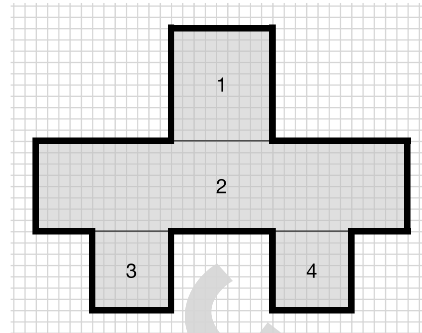

Fig. 17. Irregularity of the floor-plan perimeter summarized by the minimum number of rectangles.

TABLE VI
COST DIFFERENCES (IN \$) BETWEEN HOMOGENEOUS AND MIXED SOLUTION INCREASING THE FLOOR PLAN IRREGULARITY (AREA FIXED TO 1000 m^2)

I	Single		Trilater.		Fingerprint.	
	homog.	mixed	homog.	mixed	homog.	mixed
1	480	440	1440	1280	840	760
2	480	440	1500	1320	840	780
4	600	500	1560	1380	900	820
8	720	580	1680	1480	1200	920

VIII. CONCLUSION

In this paper, we tried to explain the challenges faced by designers during the installation of smart building systems that require the positioning of several hardware nodes. A common limitation of existing models is the lack of a convenient way to specify geometric information of the indoor map. This also leads to the employment of less accurate general models for signal propagation, instead of site-specific models. The design phase is made more difficult by the availability on the market of different hardware nodes, with different power transmissions and costs.

For these reasons we propose an integrated tool for both floor plan specification and node positioning, developed within an open-source CAD environment extensible through plug-ins. The tool is able to provide a near-optimal solution of node allocations, possibly with mixed types, with the aim to reduce the installation costs. The results suggest that, for most of the problem instances, a solution can be obtained in a reasonable execution time. Depending on the available hardware types, total cost of the solution could be improved moving from homogeneous to mixed type allocation.

A limitation of the proposed approach resides in the propagation model used to compute near-optimal solutions for localization systems. The model implemented is site-specific, and take in consideration walls for LOS and NLOS propagations. However, the approach do not consider refraction or diffraction effects. Another limitation is the inability of the system to model the signal propagation between different floors of the building, managing each level independently. For future work, we plan to improve the system with an indoor signal propagation model able to consider refraction and diffraction effects of the indoor environment like walls and floors. In addition, we will try to apply the model to

3-D designing tools, becoming suitable also for multifloor environments.

References

[1] C.-A. Roulet, "Indoor environment quality in buildings and its impact on outdoor environment," *Energy Build.*, vol. 33, no. 3, pp. 183–191, 2001.

[2] V. L. Erickson, M. Á. Carreira-Perpiñán, and A. E. Cerpa, "OBSERVE: Occupancy-based system for efficient reduction of HVAC energy," in *Proc. 10th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Chicago, IL, USA, 2011, pp. 258–269. [Online]. Available: http://ACMBuildSys2015.com

[3] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal, "Sentinel: Occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings," in *Proc. 11th ACM Conf. Embedded Netw. Sensor Syst.*, Rome, Italy, 2013, p. 17.

[4] Y. Zhao, H. Zhou, and M. Li, "Indoor access points location optimization using differential evolution," in *Proc. Int. Conf. Comput. Sci. Softw. Eng.*, Wuhan, China, 2008, pp. 382–385. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4721767

[5] Y. He, W. Meng, L. Ma, and Z. Deng, "Rapid deployment of APs in WLAN indoor positioning system," in *Proc. 6th Int. ICST Conf. Commun. Netw. China (CHINACOM)*, Harbin, China, 2011, pp. 268–273.

[6] S.-H. Fang and T.-N. Lin, "A novel access point placement approach for WLAN-based location systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Sydney, NSW, Australia, 2010, pp. 1–4.

[7] *ArchiCAD—The Architectural BIM CAD Software*. [Online]. Available: http://www.graphisoft.com/archicad/ AQ3

[8] J. P. Zhang and Z. Z. Hu, "BIM-and 4D-based integrated solution of analysis and management for conflicts and structural safety problems during construction: 1. Principles and methodologies," *Autom. Construct.*, vol. 20, no. 2, pp. 167–180, 2011.

[9] Y. G. Xu, C. Qian, W.-P. Sung, J. C. M. Kao, and R. Chen, "Lean cost analysis based on BIM modeling for construction project," *Front. Mech. Eng. Mater. Eng. II*, vols. 457–458, pp. 1444–1447, 2014. [Online]. Available: http://www.scientific.net/AMM.457-458.1444.pdf AQ4

[10] M. S. Daskin, "A maximum expected covering location model: Formulation, properties and heuristic solution," *Transp. Sci.*, vol. 17, no. 1, pp. 48–70, 1983. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-0020707868{&}partnerID=tZOtx3y1 AQ5

[11] M. S. Daskin and E. H. Stern, "A hierarchical objective set covering model for emergency medical service vehicle deployment," *Transp. Sci.*, vol. 15, no. 2, pp. 137–152, 1981. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-0019565514{&}partnerID=tZOtx3y1

[12] V. T. Quang and T. Miyoshi, "An algorithm for sensing coverage problem in wireless sensor networks," in *Proc. IEEE Sarnoff Symp.*, Princeton, NJ, USA, 2008, pp. 1–5.

[13] A. M.-C. So and Y. Ye, "On solving coverage problems in a wireless sensor network using Voronoi diagrams," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (LNCS 3828). Heidelberg, Germany: Springer, 2005, pp. 584–593. AQ6

[14] T. Andersson. (2014). *Bluetooth Low Energy and Smartphones for Proximity-Based Automatic Door Locks*. [Online]. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2:723899{&}dswid=-9677

[15] A. S. Paul *et al.*, "MobileRF: A robust device-free tracking system based on a hybrid neural network HMM classifier," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Seattle, WA, USA, 2014, pp. 159–170. [Online]. Available: http://doi.acm.org/10.1145/2632048.2632097

[16] V. L. Erickson, S. Achleitner, and A. E. Cerpa, "POEM: Power-efficient occupancy-based energy management system," in *Proc. 12th Int. Conf. Inf. Process. Sensor Netw.*, Philadelphia, PA, USA, 2013, pp. 203–216.

[17] A. Beltran, V. V. L. Erickson, and A. E. A. Cerpa, "ThermoSense: Occupancy thermal based sensing for HVAC control," in *Proc. 5th ACM Workshop Embedded Syst. Energy Efficient Build.*, Rome, Italy, 2013, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/2528282.2528301$ndelimiter"026E30F$nhttp://dl.acm.org/citation.cfm?id=2528301

[18] Y. Zhao, A. LaMarca, and J. R. Smith, "A battery-free object localization and motion sensing platform," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. UbiComp Adjunct*, Seattle, WA, USA, 2014, pp. 255–259. [Online]. Available: http://dx.doi.org/10.1145/2632048.2632078$ndelimiter"026E30F$nhttp://dl.acm.org/citation.cfm?doid=2632048.2632078

[19] A. Corna, L. Fontana, A. A. Nacci, and D. Sciuto, "Occupancy detection via iBeacon on android devices for smart building management," in *Proc. Des. Autom. Test Eur. Conf. Exhibit.*, Grenoble, France, 2015, pp. 629–632.

[20] P. Kyösti *et al.*, "IST-4-027756 WINNER II D1. 1.2 V1. 2 WINNER II channel models.pdf," *Projectscelticinitiativeorg*, vol. 1, no. 82, p. 82, 2008. [Online]. Available: http://projects.celtic-initiative.org/winner+/WINNER2-Deliverables/D1.1.2v1.2.pdf

[21] F. Colombo, R. Cordone, and G. Lulli, "The multimode covering location problem," *Comput. Oper. Res.*, vol. 67, pp. 25–33, Mar. 2016. [Online]. Available: http://dx.doi.org/10.1016/j.cor.2015.09.003

[22] M. Kouakou, S. Yamamoto, K. Yasumoto, and M. Ito "Cost-efficient deployment for full-coverage and connectivity in indoor 3D WSNs," in *Proc. IPSJ Dicomo*, 2010, pp. 1975–1982.

Andrea Cirigliano received the B.Sc. degree in computer engineering from the Politecnico di Milano, Milan, Italy, in 2013, where he is currently pursuing the M.Sc. degree.

He joined the NECST Laboratory, Politecnico di Milano, in 2015, where he is currently researching on nonintrusive indoor localization systems and occupancy detection systems. His current research interests include design of smart building systems, wireless indoor localization, and pervasive data management.

Roberto Cordone received the Dr.Eng. degree in electronic engineering and the Ph.D. degree in computer science and control theory from the Politecnico di Milano, Milan, Italy, in 1996 and 2000, respectively.

He is currently an Assistant Professor with the Universita' degli Studi di Milano, Milan. His current research interests include operations research and algorithm design and analysis.

Dr. Cordone is a member of the Italian Association of Operations Research.

Alessandro A. Nacci received the B.Sc. and M.Sc. degrees in computer engineering from the Politecnico di Milano, Milan, Italy, in 2009 and 2012, respectively, where he is currently pursuing the Ph.D. degree.

He was with EPFL, Lausanne, Switzerland. He was with the Telecom Italia Joint Open Laboratory S-Cube and the NECST Laboratory on the smart buildings topic with the Politecnico di Milano, where he has been a Research Affiliate and a Teaching Assistant, since 2016. He was a Post-Doctoral Research Fellow with the University of California at San Diego, San Diego, CA, USA, for six months, researching at the Synergy Laboratory on the smart complex buildings. In 2014, he started two companies within the Internet of Things and smart building market.

Marco Domenico Santambrogio (SM'XX) AQ7 received the laurea (M.Sc. equivalent) degree in computer engineering from the Politecnico di Milano, Milan, Italy, in 2004, the second M.Sc. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2005, and the Ph.D. degree in computer engineering from the Politecnico di Milano, in 2008.

He is an Assistant Professor with the Politecnico di Milano. He was a Post-Doctoral Fellow with CSAIL, MIT, Cambridge, MA, USA, and has also held visiting positions at the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA, in 2006 and 2007, and Heinz Nixdorf Institute, Paderborn, Germany, in 2006. He has been with the NECST Laboratory, Politecnico di Milano, where he founded the Dynamic Reconfigurability in Embedded System Design project in 2004 and the CHANGE (self-adaptive computing system) project in 2010. His current research interests include reconfigurable computing, self-aware and autonomic systems, hardware/software co-design, embedded systems, and high performance processors and systems.

Dr. Santambrogio is a Senior Member of ACM.

AUTHOR QUERIES
AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ1: Please confirm that the e-mail id for the author "A. Cirigliano" is correct as set.
AQ2: Please provide the in-text citation for "Tables I and III," "Figs. 1 and 2," and "Algorithm 1."
AQ3: Please provide the accessed date for Reference [7].
AQ4: Please provide the issue number or month for Reference [9].
AQ5: Please confirm that the edits made to References [10]–[13] and [22] are correct as set.
AQ6: Please confirm that the location and publisher information for Reference [13] is correct as set.
AQ7: Please provide the membership year for the author "M. D. Santambrogio."

Toward Smart Building Design Automation: Extensible CAD Framework for Indoor Localization Systems Deployment

Andrea Cirigliano, Roberto Cordone, Alessandro A. Nacci, and Marco Domenico Santambrogio, *Senior Member, IEEE*

***Abstract*—Over the last years, many smart buildings applications, such as indoor localization or safety systems, have been subject of intense research. Smart environments usually rely on several hardware nodes equipped with sensors, actuators, and communication functionalities. The high level of heterogeneity and the lack of standardization across technologies make design of such environments a very challenging task, as each installation has to be designed manually and performed *ad-hoc* for the specific building. On the other hand, many different systems show common characteristics, like the strict dependency with the building floor plan, also sharing similar requirements such as a nodes allocation that provides sensing coverage and nodes connectivity. This paper provides a computer-aided design application for the design of smart building systems based on the installation of hardware nodes across the indoor space. The tool provides a site-specific algorithm for cost-effective deployment of wireless localization systems, with the aim to maximize the localization accuracy. Experimental results from real-world environment show that the proposed site-specific model can improve the positioning accuracy of general models from the state-of-the-art. The tool, available open-source, is modular and extensible through plug-ins allowing to model building systems with different requirements.**

***Index Terms*—Indoor localization, Internet of Things, performance optimization, smart buildings design automation.**

I. Introduction

On AVERAGE, people spend approximately 70% of their time indoors [1], such as in offices, schools, and at home. New indoor smart applications are being developed at high rate, in both research and commercial areas covering a wide range of personal and social scenarios. Smart buildings are becoming a reality with the adoption of an underlying monitoring and communication infrastructure composed by access points (APs), sensor motes, cameras, and smart devices integrated in a building management systems (BMSs). The BMS is a control system that monitors the building state and operates through actuators to increase the comfort and safety of occupants, while managing the energy efficiency at the same time.

Many smart buildings applications are based on indoor localization techniques, using location information to optimize the environment and provide context-aware services. Indoor localization systems often require the presence of wireless devices such as APs, in order to let the user identify its position by means of a mobile device. Most smart building applications have been developed in order to achieve sustainability, reducing energy waste related to energy-consuming appliances like heating, ventilation, and air-conditioning (HVAC). Some examples are [2] and [3]. Smart HVAC systems usually rely on a set of ambient sensors able to collect indoor values of temperature and humidity. This allows the control system to build thermal maps of the indoor environment, locate thermal complaint feedbacks coming from the tenants and regulate only the necessary portion of the physical system. Another target feature of complex buildings is safety, characterized by the ability to respond to crisis events limiting damages and victims. These systems are able to detect safety threats, for example from smoke detectors or heat detectors. Also in this scenario, a proper allocation of sensor nodes is essential to detect and locate the threat responsively.

The position of each node strongly affects the performance of the system, since a bad allocation could lead to unmonitored areas. The number of nodes employed, besides weighting on the installation cost, also burdens the overall energy consumption of the system, a key parameter to consider especially for energy saving systems. The choice of the hardware nodes can get more difficult by the availability on the market of several devices and components that differ in cost, power consumption and maximum range distance. Although the key role of nodes allocation, many smart building systems proposed in literature do not consider nodes amount and positioning problems in environments that differ from the original testbeds.

Without a systematic approach the design space is not well explored, which leads to inefficient solutions. In this context, the development of tools able to automatize part of the design flow of smart building systems is essential. In order to find a near-optimal allocation of nodes, the knowledge of the floor plan is required. However, for installations performed on existing buildings, administrators can encounter difficulties

Manuscript received September 14, 2016; accepted November 13, 2016. This paper was recommended by Associate Editor S. Mohanty.

A. Cirigliano, A. A. Nacci, and M. D. Santambrogio are with the Department of Electronics, Information and Bioengineering, Politecnico di Milano, 20133 Milan, Italy (e-mail: andrea.cirigliano@mail.polimi.it).

R. Cordone is with the Department of Computer Science, Universita' degli Studi di Milano, 20100 Milan, Italy.

AQ1

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2016.2638448

TABLE I

AQ2

COMPARISON BETWEEN PROPOSED DEPLOYMENT METHODS AND TOOLS FOR INDOOR WSN AND APS-BASED SYSTEMS

Deployment	**Site Specific vs General Model**	**Heterogeneous Nodes**	**Application Integrated**	**Extensible**
Zhao et al. [4]	General Model	No	Yes	No
He at al. [5]	General Model	No	No	No
Fang et. al. [6]	Site Specific Model	No	No	No
Proposed approach	Site Specific Model	Yes	Yes	Yes

in obtaining the floor plan in an easily-interpretable digital format.

To address these problems, we developed a computer-aided design (CAD) tool to assist building designers during the design of smart building systems. The application manages common requirements like the building floor plan specification. We decided to implement a node allocation algorithm for three different indoor localization systems, that searches for near-optimal allocations of nodes, from mixed hardware types, with the aim of keeping low the total cost. Due to the high level of heterogeneity and lack of standardization across systems to design, we make the system extensible through plug-ins to let new functionalities being integrated into the system. The tool[1] is developed within the QCAD[2] environment, an open-source computer-aided drafting application. The key contributions of this paper can be summarized as follows.

1) A traditional CAD interface to specify both physical building floor-plan and functional components of the smart environment.
2) An algorithm for hardware nodes allocation that provides to designers a near-optimal placement of devices. The algorithm explores combinations of different types of nodes to obtain cost-effective solutions.
3) A site-specific model for wireless indoor localization accuracy optimization that keeps into account the actual structure of the building.
4) The integration of the tool within an open-source[3] application framework able to extend the system by means of JavaScript or C++ plug-ins.

II. RELATED WORK

Building information modeling (BIM) is a consolidate process to support building constructions and renovations. BIM softwares, and in particular CAD for buildings such as ArchiCAD [7], focus on the generation and management of digital representations of the physical aspects of places. BIM tools can coordinate architectural and structural requirements, for essential tasks such as collision detection [8]. Materials employed for a construction can be represented with extremely high levels of accuracy, thanks to the several libraries developed in many years, resulting in precise cost estimations [9]. With the diffusion of integrated smart systems built to increase comfort and efficiency, buildings require the design of aspects that go beyond the mere physical design. The concept of smart environment is becoming more and more concrete with the integration of sensors, actuators and computational elements in buildings, while tools able to model smart and interactive functionalities of modern buildings are currently lacking.

The problem of the allocation of hardware nodes in a given environment can be compared, on first approximation, by the maximal cover location problem (MCLP), i.e., the problem of covering the maximum amount of demand locations with a given number of facilities. Similarly, the location set covering problem (LSCP) consists in finding the minimum set of facilities that covers all available demand locations. Each facility has the same coverage radius r; a demand point is assumed to be covered if it is within distance r of a facility. Daskin *et al.* [10], [11] gave a general formulation of the LSCP and reformulated it for network systems and emergency vehicle deployment.

The maximum sensing coverage region is a special case of the previous two problems that focuses on the research of an allocation of wireless nodes that guarantees both sensing coverage and network connectivity between nodes [12], [13]. In this scenario, the placement need to take care not only of the sensing range, but also of the communication range of each node.

For what concern the allocation in indoor environments, only minimum literature has been published so far to the best of our knowledge. Zhao *et al.* [4] proposed an AP positioning model based on the differential evolution algorithm, specific for fingerprinting localization techniques. Their model focuses on increasing the diversity of the received signal array along the indoor locations, and thus improving the positioning accuracy of fingerprinting schemes. However, the model does not take into account the effect of walls or other obstacles present in the target environment. He *et al.* [5] made use of a genetic algorithm for APs deployment model, to study the relationship between positioning error and signal space Euclidean distance. Again, the simulation results show that the error can be reduced increasing the Euclidean distance between the received signal strength (RSS) arrays of different locations. Fang and Lin [6] proposed a tool for linking the placement of APs and the positioning performance. Their algorithm maximizes signal-to-noise ratio, i.e., maximizes the signal and minimizes the noise simultaneously. However, the system is developed in a real-world environment, and requires measurements with different AP allocations that can be an expensive and time-consuming task.

A common limitation of many works described previously is the employment of simple and general models which does not take into account the actual layout and geometry of the building. The free-space path loss propagation model is often used

[1] A video demo of the tool has been published at https://youtu.be/6c6D6wolDBQ.

[2] QCAD—Open Source CAD System: http://www.qcad.org/.

[3] The source code of the system is open-source and available at https://bitbucket.org/necst/box-smartcad.

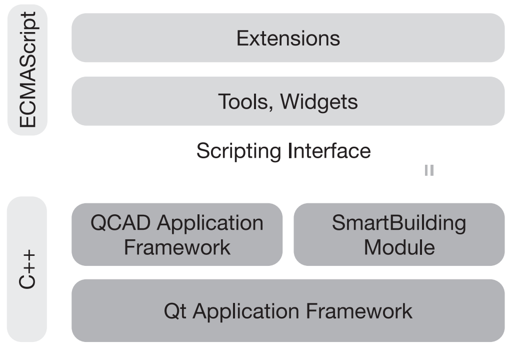

Fig. 1. Overview of the application stack. The script interpreter features standard ECMAScript functionality and on top of that provides additional classes from the Qt API, QCAD API, and the *SmartBuilding* module.

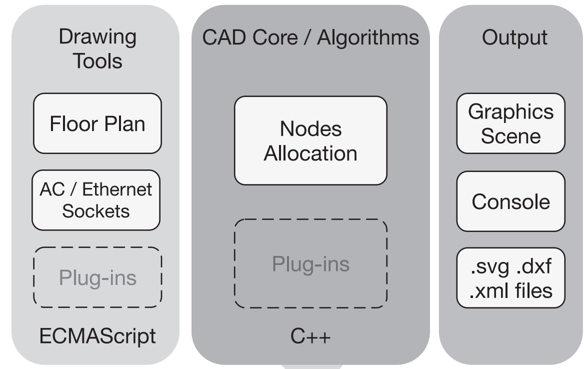

Fig. 2. Functional overview of the system components. Drawing tools and algorithms for systems deployment and simulation are extensible through ECMAScript or C++ plug-ins.

despite the presence of fixed obstructing objects like walls. Of course, none of the cited works provide a convenient way to specify geometric layout of the indoor environment. This leads the authors to validate models simply using squared or rectangular areas to represent the indoor environment, omitting the relationship between irregular areas and system coverage. In addition, none of the existing solutions takes in consideration different hardware characteristics and costs of the nodes to be deployed.

III. Proposed Application Framework

Our system has been developed on top of the QCAD application framework. The QCAD application framework consists of programming libraries and resources that provides CAD specific functionalities. An example of module provided by the QCAD application framework is the Math module that implements mathematical concepts such as vectors or matrices as well as basic geometrical classes like points, lines and so on. The QCAD Framework has been enhanced with a *SmartBuilding* module that provides some fundamental functionalities for the design of smart building systems. The module include abstract entities like rooms, walls, sockets, sensor nodes and gateways. User interface components are also provided in order to create and edit this entities (*tools*) and to specify parameters (*widgets*). Our module implements a node deployment algorithm for three commons indoor localization systems, that will be discussed later. The whole application rely on Qt, a framework that covers a lot of generic and low-level functionality for desktop applications and not directly related to CAD.

The QCAD application framework offers a very complete and powerful ECMAScript interface. The SmartBuilding module, as well as the QCAD application framework, is accessible through that scripting interface. Through the ECMAScript interface developers will be able to extend the whole application in an easy and very efficient way. The choice of a popular script language that is easy to learn enables anyone with previous programming experience to extend the application. Such extensions can for example be CAD related interactive tools like an HVAC layout construction widget, or a temperature sensor nodes deployment algorithm.

In some situations extending QCAD through scripts alone may not be possible. This is mostly the case, if the extension is based on an existing C or C++ library. In that case, it is possible to create a C++ plug-in that wraps the existing library and adds the necessary hooks to access library functionality through the script interface. Such a plug-in will be automatically loaded by QCAD on start up to add functions and classes to the script interface of QCAD. These script extensions can then be used by a script add-ons to make that functionality available as part of the application interface.

IV. Nodes Deployment for Indoor Localization

Smart environments always rely on a set of hardware nodes able to collect sensing data and communicate through cabled or wireless technologies. The number of nodes employed and the position of each one strongly affect the overall performance of the system as well as the cost of installation. In this paper, indoor localization systems have been taken as the main case study for the nodes allocation, since occupants localization and monitoring is one of the most common requirements of different smart environments.

The way in which the indoor environment must be covered by the nodes depends on the particular technology implemented; however, there can be identified three main manners.

1) Single coverage, i.e., to monitor the state of the environment with a single node for each location inside its radius. This includes for example to detect the presence of a mobile device in a proximity region [14], or to detect an RFID tag within the tags reader range [15].
2) Trilateration, to compute the position of a mobile device. This technique requires the reception of a wireless signal of at least three reference sensors with well-known positions everywhere within the covered area. We define the term *k*-coverage as the minimum number of sensors (or reference nodes) required in each location by a system. Single coverage systems have k-coverage $= 1$, while for trilateration $k = 3$.
3) Fingerprinting, where the number and the strength of the received signals is not fixed, but affect the localization accuracy.

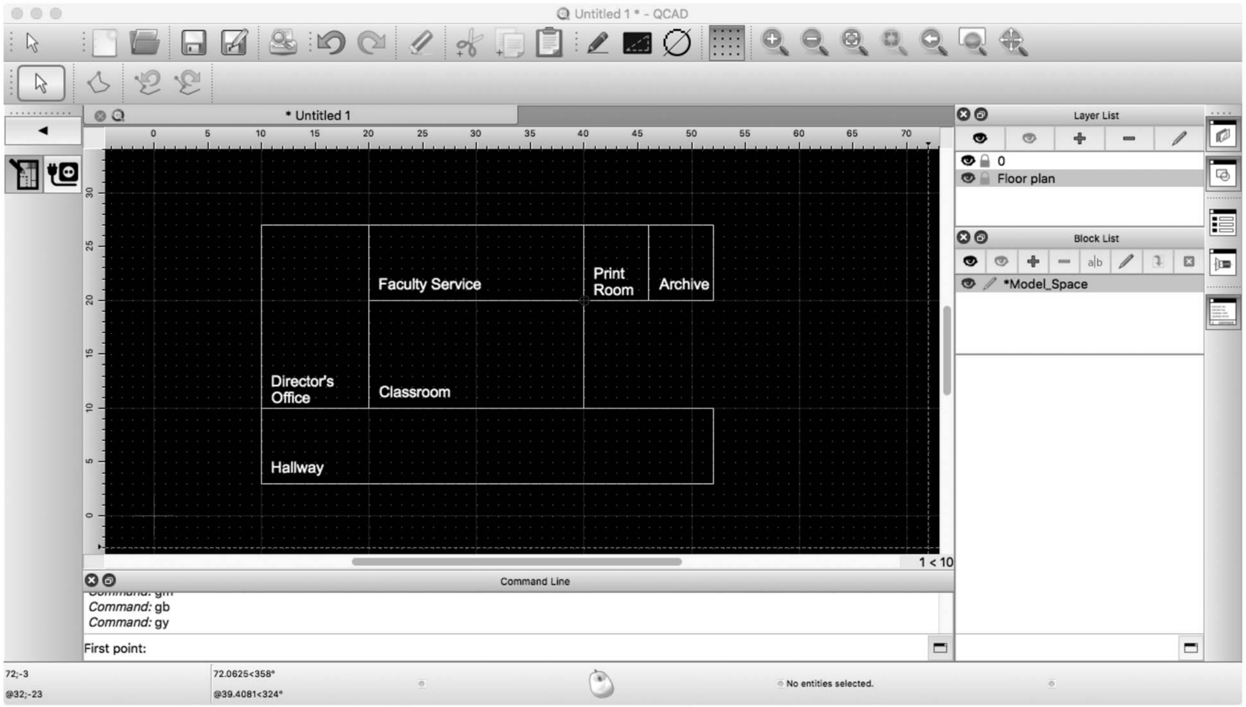

Fig. 3. Floor-plan design tool. User can specify the layout of the rooms and a possible set of candidate sites for the node placement.

Trilateration and fingerprinting usually exploit wireless technologies as Wi-Fi or Bluetooth to establish a connection between mobile and stationary nodes. Sensing regions can refer to any type of ambient sensors, such as passive infrared sensors [16], remote thermal sensors [17], but also proximity-based radio transmitters such as RFID tag readers [18] and Bluetooth low energy transmitters (BLE beacons) [19].

V. Proposed Deployment Tool

As we previously said, smart environments always rely on a set of sensor nodes, each one able to communicate through cabled or wireless technologies. Also for outdoor WSNs, a key challenge is how to achieve coverage of the target monitoring space and sufficient network connectivity between sensor nodes. Usually each sensor mote communicates with the rest of the network through technologies like Wi-Fi or ZigBee. Additional issues for outdoor WSNs are the limited battery life of each node and the power consumption required for packet transmissions. Given the availability in most (also "nonsmart") buildings of power outlets, Ethernet sockets and Wi-Fi signal, the mentioned limitations of WSNs can be solved in indoor application making use of the existing infrastructure. Differently from outdoor WSN deployments, where coverage and connectivity are always treated together, our system leaves nodes connectivity optional, focusing on providing the coverage service to the indoor locations.

The design process starts with a drafting phase in which the user specify the building floor plan as a set of rooms. During this phase the designer can restrict the possible sites for nodes allocation, selecting a set of candidate points. This can be useful when the hardware devices require power supply or Ethernet connectivity. The design interface used for both map and candidate sites specification is reported in Fig. 3.

In our model, we will refer to L as the entire set of monitoring locations to be covered, while J as the set of deployable locations where nodes can be placed. By default, $L = J$ and nodes can be positioned everywhere but as we said the set J can be restricted only to specific candidate points.

After the design phase, different parameters are provided by the administrator and used to define a domain in which search for a covering solution. The parameters are as follows.

1) The covering technique (single, trilateration, or fingerprinting) that will be used to cover the locations in L.
2) A cost c_t for every type $t \in T$ of node available on the market (expressed in dollars).
3) A working range r_t for every type t of node (expressed in meters).
4) A percentage of covered area required, called target (i.e., the minimum percentage of locations $l \in L$ to be covered by the solution).

The system will return to the designer a set N of nodes n_{jt} (possibly with mixed hardware types) and their position on the building map. The outcome will have the lower cost of installation among all the inspected solutions that satisfy the target percentage of covered area. Fig. 4 shows an overview of the process explained so far.

A. Covering Techniques

Our tool provides three different ways to cover the floor-plan space, each one identified by the technique required by the system that will be installed.

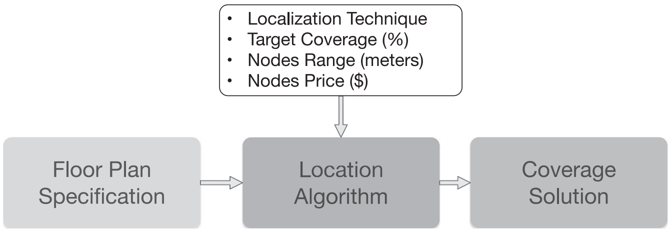

Fig. 4. System process. After the design of the floor plan, different parameters are used to define the search for an optimal allocation of nodes.

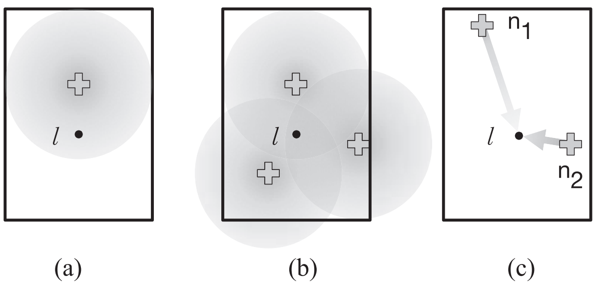

Fig. 5. Sample floor-plans with a location l covered (a) in single mode, (b) for trilateration, and (c) for fingerprinting where $\mathrm{rss}_{l,1} < \mathrm{rss}_{l,2}$.

1) Single coverage that guarantees from each position the presence of at least one reachable node. This is used for example to detect the presence of a mobile device in a proximity region. In our model, a location l of the floor-plan is considered covered if exists at least one working node n of type t within a range r_t. An example is shown in Fig. 5(a).
2) *Trilateration:* This is the process of determining the position of a point measuring its distance from three reference nodes, exploiting geometric properties of triangles. Usually, indoor trilateration systems use the strength of the signal received from a node to estimate its distance. In our model, a location l of the floor-plan is covered for trilateration if there exist at least three working nodes n_1, n_2, and n_3, each one no more distant then its corresponding range r_t. A location l served for trilateration is shown in Fig. 5(b). Although we refer only to trilateration, the same exact result can be used also for triangulation, the technique where angles are measured instead of distances.
3) *Fingerprinting:* This technique is used to estimate the position of a mobile device based on its rss vector. Each location receives the signal from k nodes, where k is not the same for all locations, but depends on how many nodes are reachable from that particular location. Each one of the k signals reaches the receiving antenna with a given power (or rss). For example, the location l shown in Fig. 5(c) perceives $k = 2$ signals so that $\mathrm{rss}_{l,1} < \mathrm{rss}_{l,2}$. We denote as $\mathrm{rss}_{l,n}$ the signal strength received at location l from a node n. The vector $\mathrm{rss}_{\mathbf{l}} = [\mathrm{rss}_{l,1}, \dots, \mathrm{rss}_{l,k}]$ of the k signals received at run-time in location l is compared with a dataset of vectors, each one prelabeled with the corresponding position.

The comparison is usually performed by a classification algorithm using the Euclidean distance of the vectors, since rss vectors with a small Euclidean distance between them are more likely to be close also in the physical space. We have defined as $\mathrm{rss}_{l,n}$ the signal strength received at location l from a node n. The Euclidean distance between $\mathrm{rss}_{\mathbf{a}}$ and $\mathrm{rss}_{\mathbf{b}}$, both composed by k received signals, and collected, respectively, in location a and b is defined as

$$E(a, b) = \sqrt{\left(\mathrm{rss}_{a,1} - \mathrm{rss}_{b,1}\right)^2 + \cdots + \left(\mathrm{rss}_{a,k} - \mathrm{rss}_{b,k}\right)^2}. \quad (1)$$

Consider the vector $\mathrm{rss}_{\mathbf{a}}$ as the run-time sample, while the vector $\mathrm{rss}_{\mathbf{b}}$ retrieved from the stored fingerprint. The smaller is the $E(a, b)$, more confident is the localization system approximating current location of a with the stored location of b.

It has been demonstrated that maximizing the Euclidean distances of the rss arrays between all sampling points, the positioning accuracy of wireless localization systems can be improved [4], [5]. Fig. 6 is reported a graphical demonstration of the aforementioned statement. Take as an example a dataset (DS1, DS2, DS3, DS4) of stored rss vectors, where each vector is bi-dimensional ($K = 2$) and coupled with the corresponding physical position. Fig. 6(a) shows each element of the database where the Cartesian coordinates corresponds to components rss_1, rss_2. Although the plane does not represent the physical area of the floor-plan, database elements that are near between them are more likely to be close also in the physical space. Given a run-time element R, each arrow represents the Euclidean distance $E(R, \mathrm{DS}_i)$ from the surrounding dataset elements. A localization algorithm can exploit the Nearest Neighbor technique to approximate the position of R with the nearest dataset element. Unfortunately, the run-time rss measurement of R will not be constant over time, but will experience continuous fluctuations due to environmental noise. These fluctuations make the sample R move randomly to the surrounding points. Suppose that DS2 is the nearest points to R in the physical space. Fig. 6(b) shows with a green area the probability to assign R the correct (or more accurate) position, while a red (with line pattern) area represents the probability to get a wrong position from the system. Fig. 6(c) demonstrates how an increase in the rss Euclidean distance between sampling points increase the red area and the accuracy of the localization, while in Fig. 6(d) an Euclidean distance reduction will lead to poorer localizations.

The RSS has been estimated using the The WINNER II path loss model [20]

$$PL = A\ \log_{10}(d[m]) + B + C \log_{10}\left(\frac{f_c[\mathrm{GHz}]}{5.0}\right) + X \quad (2)$$

where PL is the signal path loss (in dB), f_c is the frequency in GHz, and d is the distance between the transmitter and the receiver location in meters. Values of coefficients A, B, C, and X change depending on line-of-sight (LOS) or nonline-of-sight (NLOS) propagations, and are reported in Table II. The propagation model has been used in fingerprinting coverage to maximize the Euclidean distance of the rss vectors between a location and its surrounding points, with the aim of improve the localization accuracy of the system.

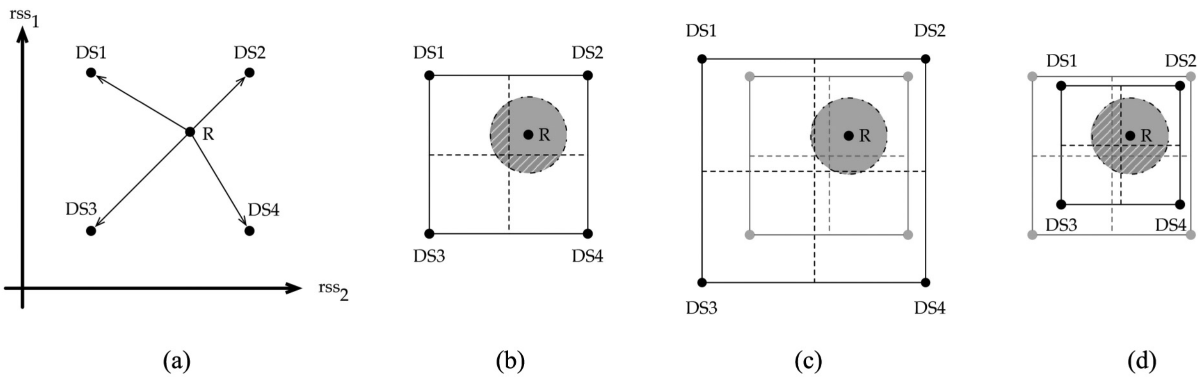

Fig. 6. (a) Bi-dimensional elements of the localization dataset are represented in Cartesian coordinates corresponding to components rss_1 and rss_2. A run-time sample *R* is shown in (b) where its circular area delineates run-time signal fluctuations. If DS2 is the nearest points to *R* in the physical space, green area is proportional to the probability of correct localization, while red dashed area represent wrong localizations. (c) Euclidean distance between sampling points has been increased, improving the correct localization. (d) Opposite effect.

TABLE II
VALUES OF COEFFICIENTS DEPENDING ON LOS OR NLOS PROPAGATIONS. VALUES HAVE BEEN TAKEN FROM THE WINNER II PATH LOSS MODEL [20]

Scenario	Path Loss Coefficients
LOS	$A = 18.7, B = 46.8, C = 20$
$NLOS$	$A = 36.8, B = 43.8, C = 20$ $X = 5(n_w - 1)$ (light walls) $X = 12(n_w - 1)$ (heavy walls)

TABLE III
NOTATION AND MEANING OF SYMBOLS USED FOR THE MODEL

Notation	Meaning
L	set of monitoring locations
J	set of deployable locations
c_t	cost of a node of type t
r_t	sensing range of a node of type t
h_t	communication range of a node of type t
$target$	coverage rate of L required by user (%)
n_{jt}	nodes of type t allocated in j
$rss_{l,n}$	signal strength received in l from n
$\mathbf{rss_a}$	vector of all the $rss_{a,n}$ values collected in a
$E(a,b)$	Euclidean distance between $\mathbf{rss_a}$ and $\mathbf{rss_b}$
D_l	set of locations no more distant than d from l
z	average signal space Euclidean distance
Z	objective function
b_l	reward earned for covering location l
w_l	reward weighted on the node cost
x_{jt}	allocation of node with type t in j (binary)
a_{ljt}	reachability of n_{jt} from location l (binary)
k-coverage	number of ref. nodes required by the system
k_l	current number of ref. nodes covering l
S	min. signal space Euclidean distance threshold
s_{min}	minimum number of node moves in *shaking* procedure
s_{max}	maximum number of node moves in *shaking* procedure
$Rmax$	number of restarts of the *VNS* algorithm

The 2-D space of the floor plan is discretized with a length unit (default is 1 m) that is chosen by the user during the map specification phase.

As we have said, in addition to location coverage, also nodes connectivity has been modeled. In our model, a sensor node n is connected if exist a connected path to the gateway node. To ensure the connectivity of the whole network, the following equation must hold:

$$\forall n \in N, \text{connected}(n, \text{gateway}) = \text{true} \tag{3}$$

where

$$\begin{aligned} \text{connected}(n, n') \overset{\text{def}}{=} & \left|(n, n')\right| \le \min(h, h') \\ & \vee \; \exists \; n_1, \ldots, n_i \; \in \; N \; (1 < i) \\ & \quad |(n, n_1)| \; \le \; \min(h, h_1) \\ & \quad \wedge \; |(n_1, n_2)| \; \le \; \min(h_1, h_2) \; \wedge \; \ldots \\ & \quad \vee \; \left|(n_i, n')\right| \; \le \; \min\big(h_i, h'\big). \end{aligned} \tag{4}$$

Connected networks are managed by our allocation algorithm in the same way of nonconnected networks, with the following exception.

1) First, a manual gateway nodes allocation is required.
2) During nodes allocation, deployable points J are restricted to locations j' such that connected$(n_{j'}, \text{gateway}) = \text{true}$.
3) During deployment optimization, nodes moves are considered feasible only within the connected area.

VI. COVERING LOCATION ALGORITHM

The covering location algorithm has the purpose of placing an optimal set of nodes on the building floor plan. We have decided to implement a modified version of the multimode covering location problem [21], a generalization of the MCLP. Using a quite general and flexible reformulation of the covering problem, we have been able to adapt the algorithm at the different covering techniques described previously.

The positioning algorithm is composed by a first Greedy procedure, whose solution is then improved by a variable neighborhood search (VNS) algorithm. The positioning algorithm evaluates different solutions using a reward b_l, that is defined for each location l and will be earned only for the locations covered in that particular solution. The value of the reward depends on the coverage technique.

1) *Single Coverage:* The reward b_l will be earned if there is at least one node that covers l.
2) *Trilateration:* The reward b_l will be earned if there are at least three nodes that cover l.

$$\mathbf{rss_l} = \langle rss_{l,1}, rss_{l,2} \rangle = \langle -84, -72 \rangle \ [dB]$$
$$\mathbf{rss_s} = \langle rss_{s,1}, rss_{s,2} \rangle = \langle -67, -41 \rangle \ [dB]$$
$$E(l,s) =$$
$$= \sqrt{(rss_{l,1} - rss_{s,1})^2 + (rss_{l,2} - rss_{s,2})^2} =$$
$$= \sqrt{(-84 - (-67))^2 + (-72 - (-41))^2} =$$
$$= 35.36 \ [dB]$$

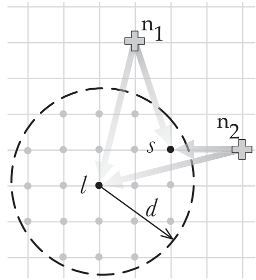

Fig. 7. Regular grid showing how is computed the mean Euclidean distance between the received rss vectors in a certain location l, and the surrounding locations s within a certain distance d.

3) *Fingerprinting:* Since this technique is often considered to be a tradeoff (in cost and accuracy) between single coverage and trilateration, we decided that the reward b_l will be earned if there are at least two nodes that covers l.

As we have said, in order to maximize the localization accuracy of the system it is possible to increase the signal space Euclidean distance between the target points. Consider the mean Euclidean distance between the received rss vector in a certain location l, and the surrounding locations s within a certain distance d

$$\frac{1}{\mid D_l \mid} \sum_{s \in D_l} E(l,s)$$
$$D_l = \{s \in L \mid \text{distance}(l,s) \leq d\}. \quad (5)$$

The distance d is used to restrict the rss comparison and diversification only to the locations that are more likely to be erroneously confuse with l by the localization system. Fig. 7 shows an example of how the Euclidean distance of a location is compared to a neighbor location.

We define the average signal space Euclidean distance z

$$z = \frac{\sum_{l \in L} \sum_{s \in D_l} \frac{E(l,s)}{|D_l|}}{|L|}. \quad (6)$$

The term z will be used by the Greedy procedure to produce a first solution with a reasonable allocation of nodes. Then, the value of z should be increased as much as possible to provide good localization accuracy to the system. However, maximize only the average does not seems fair enough, since a good system should provide a certain level of accuracy homogeneously among the target area. So we defined the objective function as difference between the term z and the signal space Euclidean variance

$$Z = z - \sqrt{\sum_{l \in L} \left(\sum_{s \in D_l} \frac{E(l,s)}{\mid D_l \mid} \right)^2}. \quad (7)$$

Maximizing the objective function Z, the intention is to provide as many target location as possible with a high signal space Euclidean distance with respect to the surrounding locations.

As we have previously introduced, we represent with L the entire set of location to be covered, while with J the set of possible positions where nodes can be placed. By default, $L = J$ and nodes can be positioned everywhere; however, its possible to restrict the J set only to specific candidate points, that represent for example power outlets or Ethernet sockets. The problem of find a near-optimal set N of nodes n_{jt} (each one located in j and having a type t) with a coverage rate $f(N)$ that satisfies the target coverage, can be formalized as follows:

$$\max Z = z - \sqrt{\sum_{l \in L} \left(\sum_{s \in D_l} \frac{E(l,s)}{\mid D_l \mid} \right)^2} \quad (8)$$
$$f(N) \geq \text{target} \quad (9)$$
$$\sum_{t \in T} x_{jt} \leq 1 \quad \forall j \in J \quad (10)$$
$$x_{jt} = 1 \iff n_{jt} \in N \quad (11)$$
$$f(N) = |L| / \sum_{l \in L} y_l \quad (12)$$
$$\begin{cases} y_l \leq \sum_{j \in J} \sum_{t \in T} a_{ljt} x_{jt} & \forall l \in L \ \text{(single)} \\ 2\ y_l \leq \sum_{j \in J} \sum_{t \in T} a_{ljt} x_{jt} & \forall l \in L \ \text{(fingerprinting)} \\ 3\ y_l \leq \sum_{j \in J} \sum_{t \in T} a_{ljt} x_{jt} & \forall l \in L \quad \text{(trilateration).} \end{cases} \quad (13)$$

The decision variable $x_{jt} = 1$ represents the allocation of a node of type t in location j; a_{ljt} is equal to 1 if location l can be reached by a node of type t placed in j, and $a_{ljt} = 0$ otherwise. $y_l = 1$ if location l is covered, $y_l = 0$ otherwise. The constraint (10) fixes to one the maximum number of nodes that can be located in each site.

A. Greedy Procedure

The positioning algorithm starts with a Greedy procedure with the purpose of find a reasonable number of reference nodes, for both coverage and localization accuracy. The procedure generate a first solution N positioning a set of $k = |N|$ nodes, each one with a type $t \in T$. For all three coverage techniques, the reward b_l is weighted with the cost of the current node n^* selected for the coverage

$$w_l = \frac{b_l}{c_t}; \quad \left\{ n^* = n_{jt} \ \wedge \ \text{distance}(j,l) \leq r_t \right\}. \quad (14)$$

The weighted reward w_l will be used by the Greedy algorithm so that on equal covered area, the cheapest node type has the priority over the others. We denote as L_{jt} the subset of locations that are reachable by a reference node n of type t placed at location j. At each iteration, the algorithm places a node n of type t^* at position j^* that covers the subset of locations $L_{j^*t^*}$ with the maximum reward. The term

$$1 - \frac{k_l}{k - \text{coverage}} \quad (15)$$

is used to prioritize the covering of locations with a lower "temporary" k-coverage (called k_l) with respect to the k-coverage required by the current techniques. In this way, Greedy procedure tends to avoid the placement of nodes very

Algorithm 1 Greedy(L, J, T, w, target)

$N := \varnothing$;
$L_{jt} := \{l \in L \mid l$ is covered by node in j with type $t\}$;
while $(f(N) < target) \wedge (z < S)$ **do**
  $j^* := \arg\max_{j \in J} \sum_{l \in L_{jt}} w_l \, (1 - \frac{k_l}{k\text{-}coverage})$;
  $t^* := \arg\max_{t \in T} \sum_{l \in L_{jt}} w_l \, (1 - \frac{k_l}{k\text{-}coverage})$;
  $N := N \cup \{n_{j^*t^*}\}$;
  $L_{jt} := L_{jt} \setminus L_{j^*t}$ for all $j \in J$;
return N;

close to one other which can lead, especially for trilateration systems, to poor localization accuracy. It is important to notice that the purpose of the Greedy procedure is to find a reasonable number of nodes for the localization service. The starting positioning is made on a best-effort basis, that will be improved by the successive VNS. After a node allocation, all subsets L_{jt} are updated according to the coverage technique. In trilateration for example, a location l is removed from L_{jt} only if there exist, other than the current $n_{j^*t^*}$, other two nodes that are already covering l.

The Greedy procedure ends when the target coverage is satisfied, and when the average signal space Euclidean distance z reaches the threshold S. In our implementation we set the threshold $S = 4.5$ that has been proven to be the average Euclidean distance for which the positioning error is limited to 2 m [5]. How we will see in Section VII, the Greedy procedure is able to provide an average Euclidean distance not so far from the final best known. However, thanks to the low complexity of the Greedy procedure, additional time can be used to improve the solution. In addition, the Euclidean distance variance will be strongly improved.

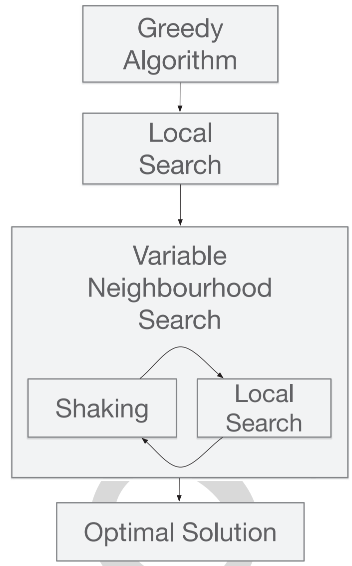

Fig. 8. Location algorithm. The solution found by the *Greedy* algorithm is improved applying iteratively a *Local Search* for an optimal solution and a *Shaking* procedure that perturbs the current solution.

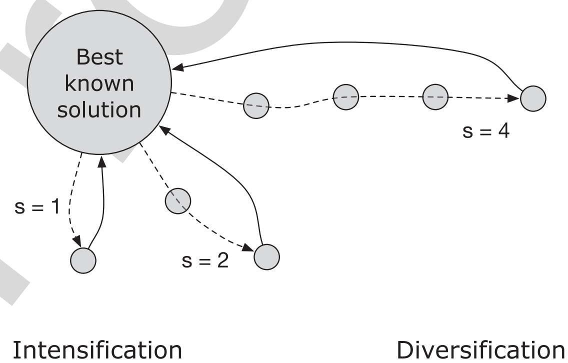

Fig. 9. Shaking procedure: the parameter s is increased when the solution does not improve (dashed line) and restarts when a new optimum is found (continuous line).

B. Variable Neighborhood Search

The method called VNS has been used to improve the solution coming from the Greedy procedure. The VNS approach empowers the classical local search framework with a restart mechanism that extends the search after a local optimum has been achieved by generating new starting solutions in progressively enlarged neighborhoods of the current best known solution. The key elements of the VNS (reported in Algorithm 2) are a starting solution N with a hierarchy of size-increasing neighborhoods, and a local search procedure, i.e., the criterion to select the incumbent solution from the neighborhood. These components are used to restart the search every time that the procedure reaches a local optimum. Fig. 8 shows an overview of the VNS process. A first local search procedure is applied to the solution produced by the Greedy procedure. At each iteration, the *shaking* procedure is used to generate a new starting solution, which is then improved by the execution of the local search. The shaking procedure perturbs s node allocations of the current solution N^* replacing them with s unused nodes. The behavior of the shaking parameter s, that depends on the result of the local search, is explained in Fig. 9. The parameter s starts from a minimum value $s_{\min}$ (in the example $s_{\min} = 1$) and every time that the local search does not improve the best known solution, s is increased by 1. Differently, when the local search succeeds, the best solution N^* is updated and s goes back to $s_{\min}$.

The purpose of the shaking procedure is to first explore new starting solutions that are more similar to the best known result, so that the search is *intensified* in a promising neighborhood of the entire domain. If these local searches fail, the shaking procedure moves the search from intensification to *diversification*, generating starting solutions that are more and more different from the incumbent one. Whenever a new best solution is found, the shaking procedure comes back to $s_{\min}$, to intensify the search near the just updated N^*. In principle, the shaking parameter s can be increased until $k = |N^*|$, changing all the node allocations. However, we experimented running different configurations that excessively moving away from the best known solution can be unproductive, causing a useless waste of computational time. We have fixed a reasonable value of $s_{\max} = \lfloor (2/3)k \rfloor$.

Algorithm 2 VNS(L, J, T, w, target, $s_{\min}, s_{\max}, R_{\max}$)

$N := Greedy(L, J, T, w, target)$;
$N^0 := LocalSearch(L, J, T, w, target)$;
$N^* := N^0$;
$s := s_{min}$;
for $r := 1$ to R_{max} **do**
  $N := Shaking(N^*, s, L, J, T, w, target)$
  $N^0 := LocalSearch(L, J, T, w, target)$
  if $(Z(N^0) > Z(N^*))$ **then**
    $s := s_{min}$;
    $N^* := N^0$;
  else
    $s := s + 1$;
    if $(s > s_{max})$ **then**
      $s := s_{min}$;
return N^*;

The VNS algorithm terminates when the total number of restarts reaches a given value $R_{\max}$.

As we have said, the local search is the heuristic that proceeds from an initial solution to its neighborhood by a sequence of local changes, trying to improve each time the value of the objective function until a local optimum is found. The neighborhood of the adopted approach is given by cyclic sequences of moves, where each move consists in locating a new node, removing a node or changing the type of the node. A cyclic move is considered feasible only if the new covering rate respects the target coverage, and the total cost of the solution does not increase. Of course, each site must continue to hosts at maximum one node [constraint (10)]. A cyclic move can be visualized on a graph $G = (N, A)$, where each node of the graph is a possible allocation of a hardware node. Each node of the graph is characterized by a location j, and a state that indicates if the node is active or inactive. A node n_{jt} currently allocated in location j, is represented on the graph with an active node n_j, labeled with its hardware type t. Note that index t does not appear because at most one type can be active in each node, and the type is specified by the label. Inactive nodes are instead left unlabeled. An arc (n_j, n_k) can represent the following.

1) The allocation of a hardware node in site j, if n_j is inactive and n_k is active.
2) The removal of a hardware node in site j, if n_j is active and n_k is inactive.
3) An hardware node n_j changing its hardware type, if both nodes are active.

In both 1) and 2), the new node takes the hardware type of the head label (t of n_k). A cyclic exchange corresponds to a directed cycle on the improvement graph, as depicted in Fig. 10. Each move, and so each arc (n_j, n_k), determines a variation δZ in the value of the objective function Z. The purpose is to represent a group of moves so that a cyclic exchange represents an increase in the current objective function. However, the total variation δZ is non additive with respect to the sequence of δZ values coming from single moves. This is caused by the interdependence between different hardware

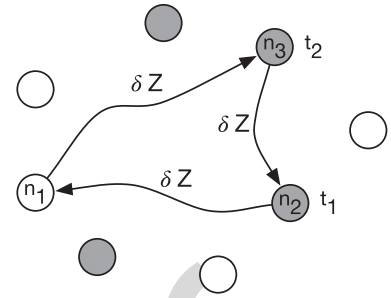

Fig. 10. Improvement graph: colored nodes represent current allocations, while empty nodes are possible allocations. All active nodes are labeled with their corresponding type. Each arc is a change (move) on the allocations.

nodes with overlapping covering regions, that lead to nonadditive moves. To overcome this drawback, every cycle has been evaluated using an own temporary function Z' updated step by step from the end of the path to its starting node. In this way, all the cycles with a positive total weight bring improvements on the starting solution.

The search for the cyclic exchange with maximum weight is performed with exhaustive breadth-first exploration of the paths of graph G.

VII. Experimental Results

Presented experimental results are initially focused on the usability of the tool, testing the ability to provide a solution in a reasonable time. Then, the performances of the model have been evaluated, in terms of localization accuracy through realistic indoor environment experiments, and in terms of cost-effectiveness of the suggested deployments.

A. Computational Experience

The tool has been evaluated running several different configurations. Every test reported in this section has been executed with a spatial resolution of the floor plan equal to 1 m. A first analysis can be done on the execution times of the proposed solution. Although the execution time can be tuned by the parameter $R_{\max}$, which represents the maximum number of restarts of the VNS algorithm, an idea on the order of magnitude is given by Fig. 11, where the time is represented as a function of the floor-plan dimension. In the given example, $R_{\max}$ has been fixed to 20 restarts, the target coverage equals to 95% of the total area, a single node type available with a range of 8 m, covering floor-plans with rectangular areas. The graph shows that for single coverage and fingerprinting the processing time grows approximately linearly with the floor plan area.

A numeric comparison of the same tests is reported in Table IV, where execution times are reported in seconds for increasing floor plans. For single coverage, the execution time is low even for areas of 3000 squared meters. For trilateration and fingerprinting, the execution times become high from floor-plan of 2500 m^2. However, the tests represent a bad case in which the map dimension is very large while the node range available and the spatial resolution are small (respectively,

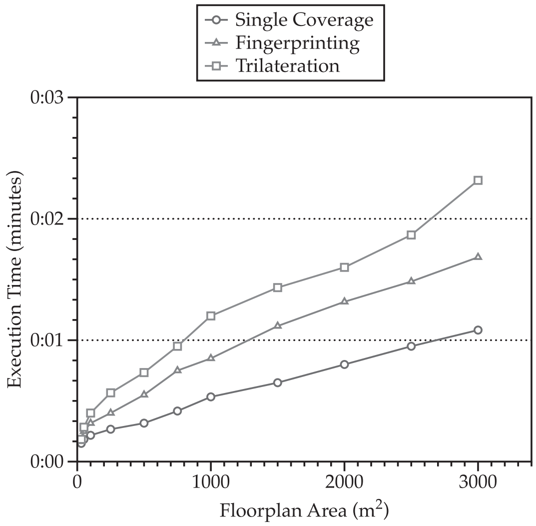

Fig. 11. Execution time of the tool with floor plans of different areas, for each covering technique ($R_{\max} = 20$, target = 95%, and $r_t = 8$).

TABLE IV
EXECUTION TIME OF THE TOOL FOR INCREASING FLOOR PLAN AREAS ($R_{\max} = 20$, target = 95%, AND $r_t = 8$)

Floor Plan Area (m^2)	Execution Time (s)		
	Single	Fingerprinting	Trilateration
30	9.07	10.53	11.49
50	11.31	15.10	17.67
100	13.05	19.41	24.22
250	16.57	24.89	34.16
500	19.18	33.48	44.66
750	25.43	45.32	57.94
1000	32.19	51.68	72.83
1500	39.37	67.12	86.27
2000	48.30	79.49	96.11
2500	57.11	89.47	112.34
3000	65.41	101.24	139.18

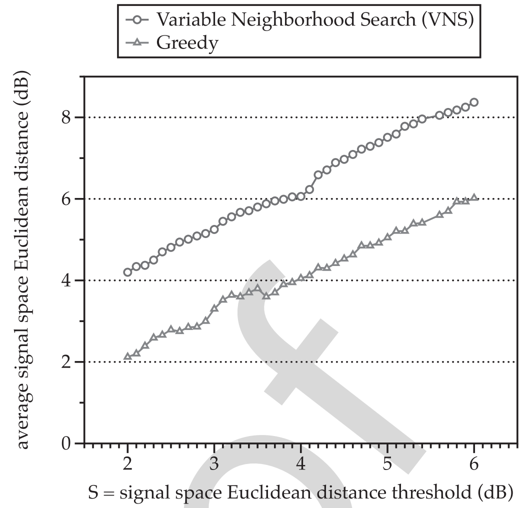

Fig. 12. Average signal space Euclidean distance (z) obtained with the Greedy execution and compared with the z value after the VNS optimization. z values expressed as a function of the threshold S. Floor-plan area = 2500 m^2, $R_{\max} = 20$, target = 100%, and $r_t = 12$.

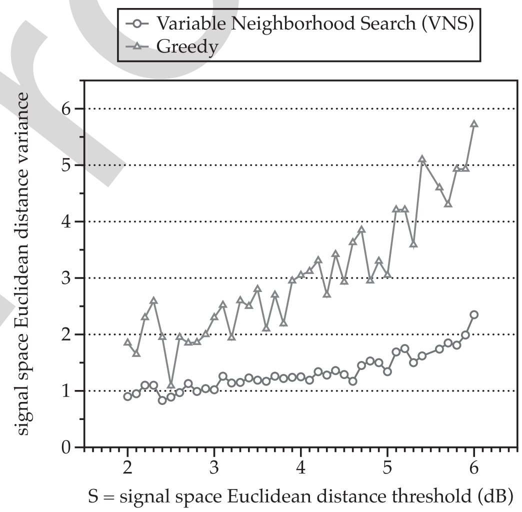

Fig. 13. Signal space Euclidean distance variance obtained with the Greedy execution and compared with the z value after the VNS optimization. Values expressed as a function of the threshold S. Floor-plan area = 2500 m^2, $R_{\max} =$ 20, target = 100%, and $r_t = 12$.

8 and 1 m). Increasing the range or the resolution, the instance of the problem decrease, resulting in faster executions.

A key aspect that characterizes the goodness of the proposed approach is the improvement of the objective function achieved by the VNS algorithm with respect to the first Greedy configuration. For this test we have run the tool several times with a floor-plan area of 2500 m^2 and a node range of 12 m. The number of reference nodes allocated is determined by the Greedy procedure and increase with S, while the number of VNS restarts $R_{\max}$ has been fixed to 35.

In Fig. 12, we reported the value of z, i.e., the average signal space Euclidean distance obtained with the first Greedy execution, compared with the z value after the VNS optimization. The graph reports the z values as a function of the threshold S, described in Section VI-A as the minimum value of average signal space Euclidean distance (z) required during the Greedy procedure. The graph shows that moving the threshold within the range (2, 6)dB the VNS is able to improve the z value constantly around 2 dB. Although the VNS improvement is not astonishing for what regard the average value, Fig. 13 shows that the variance is strongly improved. This has been achieved moving from the objective function z used in Greedy procedure to the Z function of the VNS. The Z objective function has in fact the purpose to provide as many target location as possible with a high signal space Euclidean distance w.r.t. the surrounding locations.

B. Experimental Setup and Accuracy Evaluation

The proposed tool was evaluated using data collected from a real-world environment, the NECST Lab, located at the basement of DEIB Department at the Politecnico di Milano.

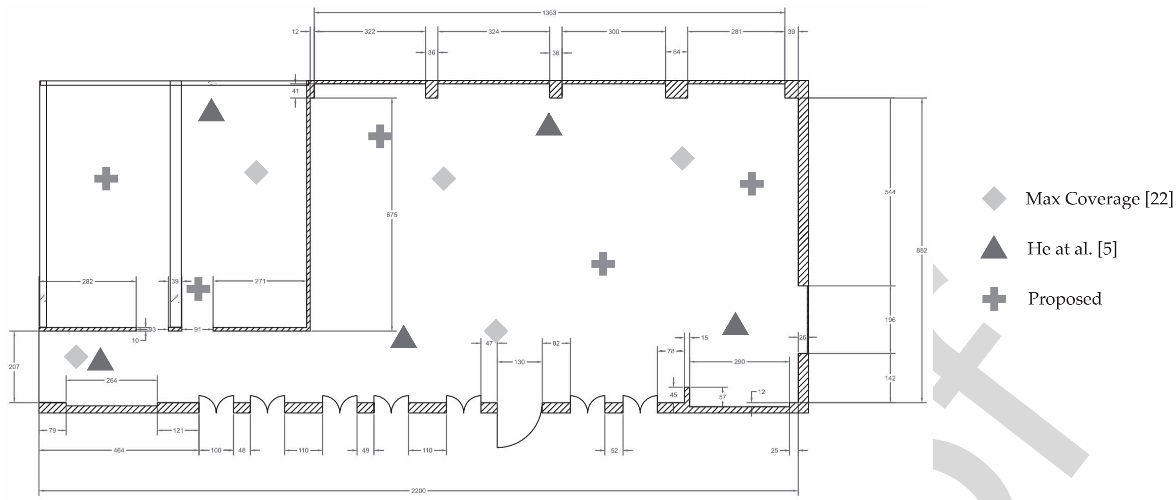

Fig. 14. NECST Laboratory floor-plan, located at the basement of DEIB Department at the Politecnico di Milano. Each allocation corresponds to a BLE beacon with a range of 7 m. Green crosses indicates allocations provided by our algorithm, gray rhombus represent allocations from [5] while blue triangle positions have been computed maximizing the coverage [22].

The dimension of the test-bed is 198 squared meters (9×22 m). We collected BLE signal data coming from BLE beacons with a coverage radius of 7 m. Signal data has been collected using a Nexus 5 smartphone running Android 6.0.1. First, the NECST Laboratory floor-plan has been designed using our tool, obtaining the optimal number of beacons ($|N| = 5$) and their allocation for fingerprinting localization. $R_{\max}$ has been fixed to 20 restarts, the target coverage equals to 100% of the total area, a single node type available with a range of 7 m, and the threshold $S = 4, 5$. We collected 40 training samples for the localization algorithm using the obtained allocation. Then, the test samples were collected at distinct positions changing the phone orientation and the way in which user was keeping it, for example by hand or in a pocket. For the entire duration of training and test phase, the number of occupants and their enabled wireless devices has changed, from a minimum of 3 to a maximum of 17 people. This variation affects the accuracy performances, but at the same time contributes in obtaining realistic results. The training and test phase has been repeated with two configurations coming from different allocation algorithms: maximization of the coverage [22] and the allocation algorithm proposed by He *et al.* [5]. For these two algorithms, the number of employed nodes has been fixed to 5. KNN with $K = 3$ has been employed as the fingerprinting algorithm.

A first result is shown in Fig. 15. The cumulative error distribution function shows that from 1.5 m our approach performs better. Under 1.5 m, He *et al.* [5] approach performs better, but the difference in accuracy is marginal.

Fig. 16 shows the mean positioning accuracy divided into different error ranges: $(0, 0.5]$, $(0.5, 1]$, $(1, 1.5]$, $(1.5, 2]$, $(2, 2.5]$, $(2.5, 3]$, $(3, 3.5]$, and $(3.5, 4]$. It is possible to notice that the majority of the localization errors appears within the $(1.5, 2]$ m. The test-bed floor-plan, composed by three rooms, has been reported in Fig. 14. Green crosses indicates allocations provided by our algorithm, gray rhombus represent allocations from [5] while blue triangle positions have been computed maximizing the coverage [22].

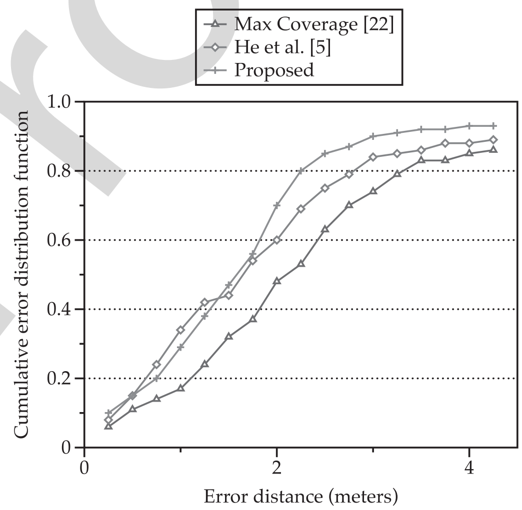

Fig. 15. Cumulative error distribution function experienced by our approach ad compared with two different solutions from the state-of-the-art.

C. Cost-Effectiveness Analysis

A feature of our tool interesting for testing is the possibility to obtain solutions from mixed node types, with different characteristics and costs. In particular, given two types t_1 and t_2 characterized by two ranges r_i, and two costs c_i, it is possible to compare the total cost of a homogeneous solution with the cost of a mixed solution. Given a baseline type of node with a range $r_1 = 8$ m and a cost of $c_1 = 60$ \$, we can assume the presence on the market of a second type of hardware, with the half of the range distance ($r_2 = 4$ m). The area covered by t_1 (≈ 200 m^2) is four times bigger than the coverage of t_2 (≈ 50 m^2). In order to obtain a fair test, the cost of t_2 should be $c_2 \geq c_1/4$, and so we set $c_2 = 20$ \$. This test has been performed with a target coverage of 95% on a rectangular map of 1000 m^2.

From Table V, it is possible to observe that, although hardware nodes of type t_2 have a lower convenience in terms of

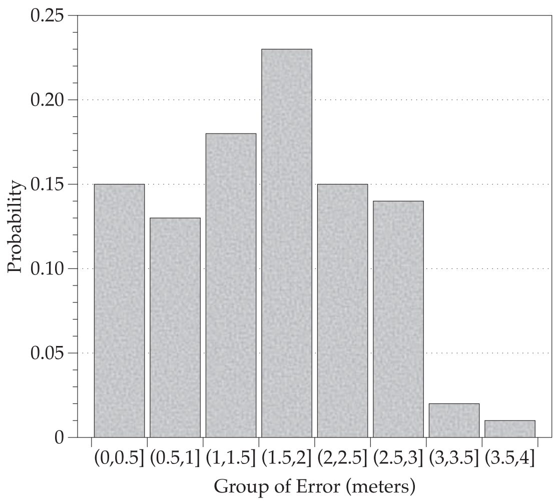

Fig. 16. Mean positioning accuracy of the proposed allocation algorithm divided into different error ranges.

TABLE V
COST OF HOMOGENEOUS AND MIXED SOLUTIONS ($A = 1000$ m^2, target = 95%, $r_1 = 8$ m, $r_2 = 4$ m, $c_1 = 60$ \$, AND $c_2 = 20$ \$)

Node types	Solution Costs (in \$)		
	Single	Trilateration	Fingerprinting
$T = \{t_1\}$	480	1440	840
$T = \{t_2\}$	500	1620	880
$T = \{t_1, t_2\}$	440	1280	760

(*area*/*price*) (t_1 outperform t_2 in homogeneous solutions), the mixed strategy can use the smaller range nodes to reduce the total cost. This because less powerful nodes of type t_2 are employed to cover small portions of the floor-plan, like corners or small regions left uncovered by the larger range nodes.

The amount of saving in the total cost of the mixed solution does not depend only on the nodes range and price, but also on the irregularity of the floor plan perimeter. A distinguish feature of the proposed tool respect to other works is the possibility to cover spaces that are not necessarily rectangular or squared. The level of irregularity of a floor plan can be identified by the minimum number of rectangles that compose the shape. In Fig. 17 for example, the index of the floor plan irregularity is $I = 4$. We experimented the behavior of the tool increasing the level of irregularity, while maintaining a constant total area of 1000 m^2. The test has been done with the same nodes configuration used in Table V (homogeneous $T = t_1$, mixed $T = t_1, t_2$). The results shown in Table VI proven that increasing the floor-plan irregularity, the cost difference between homogeneous and mixed solution becomes higher. This is caused by the increasing number of corners in the map, that can be covered with less powerful nodes.

In conclusion, experimental results show that for most of the problem instances, a solution can be obtained in reasonable execution times. Depending on the available hardware types, homogeneous solutions could be improved with the employment of different type of nodes.

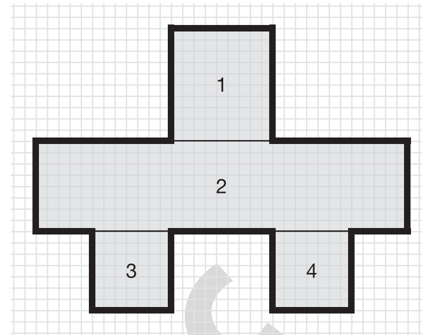

Fig. 17. Irregularity of the floor-plan perimeter summarized by the minimum number of rectangles.

TABLE VI
COST DIFFERENCES (IN \$) BETWEEN HOMOGENEOUS AND MIXED SOLUTION INCREASING THE FLOOR PLAN IRREGULARITY (AREA FIXED TO 1000 m^2)

I	Single		Trilater.		Fingerprint.	
	homog.	mixed	homog.	mixed	homog.	mixed
1	480	440	1440	1280	840	760
2	480	440	1500	1320	840	780
4	600	500	1560	1380	900	820
8	720	580	1680	1480	1200	920

VIII. CONCLUSION

In this paper, we tried to explain the challenges faced by designers during the installation of smart building systems that require the positioning of several hardware nodes. A common limitation of existing models is the lack of a convenient way to specify geometric information of the indoor map. This also leads to the employment of less accurate general models for signal propagation, instead of site-specific models. The design phase is made more difficult by the availability on the market of different hardware nodes, with different power transmissions and costs.

For these reasons we propose an integrated tool for both floor plan specification and node positioning, developed within an open-source CAD environment extensible through plug-ins. The tool is able to provide a near-optimal solution of node allocations, possibly with mixed types, with the aim to reduce the installation costs. The results suggest that, for most of the problem instances, a solution can be obtained in a reasonable execution time. Depending on the available hardware types, total cost of the solution could be improved moving from homogeneous to mixed type allocation.

A limitation of the proposed approach resides in the propagation model used to compute near-optimal solutions for localization systems. The model implemented is site-specific, and take in consideration walls for LOS and NLOS propagations. However, the approach do not consider refraction or diffraction effects. Another limitation is the inability of the system to model the signal propagation between different floors of the building, managing each level independently. For future work, we plan to improve the system with an indoor signal propagation model able to consider refraction and diffraction effects of the indoor environment like walls and floors. In addition, we will try to apply the model to

3-D designing tools, becoming suitable also for multifloor environments.

REFERENCES

[1] C.-A. Roulet, "Indoor environment quality in buildings and its impact on outdoor environment," *Energy Build.*, vol. 33, no. 3, pp. 183–191, 2001.

[2] V. L. Erickson, M. Á. Carreira-Perpiñán, and A. E. Cerpa, "OBSERVE: Occupancy-based system for efficient reduction of HVAC energy," in *Proc. 10th ACM/IEEE Int. Conf. Inf. Process. Sensor Netw.*, Chicago, IL, USA, 2011, pp. 258–269. [Online]. Available: http://ACMBuildSys2015.com

[3] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal, "Sentinel: Occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings," in *Proc. 11th ACM Conf. Embedded Netw. Sensor Syst.*, Rome, Italy, 2013, p. 17.

[4] Y. Zhao, H. Zhou, and M. Li, "Indoor access points location optimization using differential evolution," in *Proc. Int. Conf. Comput. Sci. Softw. Eng.*, Wuhan, China, 2008, pp. 382–385. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4721767

[5] Y. He, W. Meng, L. Ma, and Z. Deng, "Rapid deployment of APs in WLAN indoor positioning system," in *Proc. 6th Int. ICST Conf. Commun. Netw. China (CHINACOM)*, Harbin, China, 2011, pp. 268–273.

[6] S.-H. Fang and T.-N. Lin, "A novel access point placement approach for WLAN-based location systems," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Sydney, NSW, Australia, 2010, pp. 1–4.

[7] *ArchiCAD—The Architectural BIM CAD Software*. [Online]. Available: http://www.graphisoft.com/archicad/

[8] J. P. Zhang and Z. Z. Hu, "BIM-and 4D-based integrated solution of analysis and management for conflicts and structural safety problems during construction: 1. Principles and methodologies," *Autom. Construct.*, vol. 20, no. 2, pp. 167–180, 2011.

[9] Y. G. Xu, C. Qian, W.-P. Sung, J. C. M. Kao, and R. Chen, "Lean cost analysis based on BIM modeling for construction project," *Front. Mech. Eng. Mater. Eng. II*, vols. 457–458, pp. 1444–1447, 2014. [Online]. Available: http://www.scientific.net/AMM.457-458.1444.pdf

[10] M. S. Daskin, "A maximum expected covering location model: Formulation, properties and heuristic solution," *Transp. Sci.*, vol. 17, no. 1, pp. 48–70, 1983. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-0020707868{&}partnerID=tZOtx3y1

[11] M. S. Daskin and E. H. Stern, "A hierarchical objective set covering model for emergency medical service vehicle deployment," *Transp. Sci.*, vol. 15, no. 2, pp. 137–152, 1981. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-0019565514{&}partnerID=tZOtx3y1

[12] V. T. Quang and T. Miyoshi, "An algorithm for sensing coverage problem in wireless sensor networks," in *Proc. IEEE Sarnoff Symp.*, Princeton, NJ, USA, 2008, pp. 1–5.

[13] A. M.-C. So and Y. Ye, "On solving coverage problems in a wireless sensor network using Voronoi diagrams," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (LNCS 3828). Heidelberg, Germany: Springer, 2005, pp. 584–593.

[14] T. Andersson. (2014). *Bluetooth Low Energy and Smartphones for Proximity-Based Automatic Door Locks*. [Online]. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2:723899{&}dswid=-9677

[15] A. S. Paul *et al.*, "MobileRF: A robust device-free tracking system based on a hybrid neural network HMM classifier," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, Seattle, WA, USA, 2014, pp. 159–170. [Online]. Available: http://doi.acm.org/10.1145/2632048.2632097

[16] V. L. Erickson, S. Achleitner, and A. E. Cerpa, "POEM: Power-efficient occupancy-based energy management system," in *Proc. 12th Int. Conf. Inf. Process. Sensor Netw.*, Philadelphia, PA, USA, 2013, pp. 203–216.

[17] A. Beltran, V. V. L. Erickson, and A. E. A. Cerpa, "ThermoSense: Occupancy thermal based sensing for HVAC control," in *Proc. 5th ACM Workshop Embedded Syst. Energy Efficient Build.*, Rome, Italy, 2013, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/2528282.2528301$ndelimiter"026E30F$nhttp://dl.acm.org/citation.cfm?id=2528301

[18] Y. Zhao, A. LaMarca, and J. R. Smith, "A battery-free object localization and motion sensing platform," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput. UbiComp Adjunct*, Seattle, WA, USA, 2014, pp. 255–259. [Online]. Available: http://dx.doi.org/10.1145/2632048.2632078$ndelimiter"026E30F$nhttp://dl.acm.org/citation.cfm?doid=2632048.2632078

[19] A. Corna, L. Fontana, A. A. Nacci, and D. Sciuto, "Occupancy detection via iBeacon on android devices for smart building management," in *Proc. Des. Autom. Test Eur. Conf. Exhibit.*, Grenoble, France, 2015, pp. 629–632.

[20] P. Kyösti *et al.*, "IST-4-027756 WINNER II D1. 1.2 V1. 2 WINNER II channel models.pdf," *Projectsculticinitiativeorg*, vol. 1, no. 82, p. 82, 2008. [Online]. Available: http://projects.celtic-initiative.org/winner+/WINNER2-Deliverables/D1.1.2v1.2.pdf

[21] F. Colombo, R. Cordone, and G. Lulli, "The multimode covering location problem," *Comput. Oper. Res.*, vol. 67, pp. 25–33, Mar. 2016. [Online]. Available: http://dx.doi.org/10.1016/j.cor.2015.09.003

[22] M. Kouakou, S. Yamamoto, K. Yasumoto, and M. Ito "Cost-efficient deployment for full-coverage and connectivity in indoor 3D WSNs," in *Proc. IPSJ Dicomo*, 2010, pp. 1975–1982.

Andrea Cirigliano received the B.Sc. degree in computer engineering from the Politecnico di Milano, Milan, Italy, in 2013, where he is currently pursuing the M.Sc. degree.

He joined the NECST Laboratory, Politecnico di Milano, in 2015, where he is currently researching on nonintrusive indoor localization systems and occupancy detection systems. His current research interests include design of smart building systems, wireless indoor localization, and pervasive data management.

Roberto Cordone received the Dr.Eng. degree in electronic engineering and the Ph.D. degree in computer science and control theory from the Politecnico di Milano, Milan, Italy, in 1996 and 2000, respectively.

He is currently an Assistant Professor with the Universita' degli Studi di Milano, Milan. His current research interests include operations research and algorithm design and analysis.

Dr. Cordone is a member of the Italian Association of Operations Research.

Alessandro A. Nacci received the B.Sc. and M.Sc. degrees in computer engineering from the Politecnico di Milano, Milan, Italy, in 2009 and 2012, respectively, where he is currently pursuing the Ph.D. degree.

He was with EPFL, Lausanne, Switzerland. He was with the Telecom Italia Joint Open Laboratory S-Cube and the NECST Laboratory on the smart buildings topic with the Politecnico di Milano, where he has been a Research Affiliate and a Teaching Assistant, since 2016. He was a Post-Doctoral Research Fellow with the University of California at San Diego, San Diego, CA, USA, for six months, researching at the Synergy Laboratory on the smart complex buildings. In 2014, he started two companies within the Internet of Things and smart building market.

Marco Domenico Santambrogio (SM'XX) received the laurea (M.Sc. equivalent) degree in computer engineering from the Politecnico di Milano, Milan, Italy, in 2004, the second M.Sc. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2005, and the Ph.D. degree in computer engineering from the Politecnico di Milano, in 2008.

He is an Assistant Professor with the Politecnico di Milano. He was a Post-Doctoral Fellow with CSAIL, MIT, Cambridge, MA, USA, and has also held visiting positions at the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA, in 2006 and 2007, and Heinz Nixdorf Institute, Paderborn, Germany, in 2006. He has been with the NECST Laboratory, Politecnico di Milano, where he founded the Dynamic Reconfigurability in Embedded System Design project in 2004 and the CHANGE (self-adaptive computing system) project in 2010. His current research interests include reconfigurable computing, self-aware and autonomic systems, hardware/software co-design, embedded systems, and high performance processors and systems.

Dr. Santambrogio is a Senior Member of ACM.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ1: Please confirm that the e-mail id for the author “A. Cirigliano” is correct as set.
AQ2: Please provide the in-text citation for “Tables I and III,” “Figs. 1 and 2,” and “Algorithm 1.”
AQ3: Please provide the accessed date for Reference [7].
AQ4: Please provide the issue number or month for Reference [9].
AQ5: Please confirm that the edits made to References [10]–[13] and [22] are correct as set.
AQ6: Please confirm that the location and publisher information for Reference [13] is correct as set.
AQ7: Please provide the membership year for the author “M. D. Santambrogio.”