

Partitioning of Intelligent Buildings for Distributed Contaminant Detection and Isolation

Alexis Kyriacou, *Student Member, IEEE*, Stelios Timotheou, *Member, IEEE*, Michalis Michaelides, *Member, IEEE*, Christos Panayiotou, *Senior Member, IEEE*, and Marios Polycarpou, *Fellow Member, IEEE*

Abstract—Intelligent buildings are responsible for ensuring indoor air quality for their occupants under normal operation as well as under possibly harmful contaminant events. An emerging environmental application involves the monitoring of intelligent buildings against harmful events by incorporating various sensing technologies and using sophisticated algorithms to detect and isolate such events. In this context, both centralized and distributed approaches have been proposed, with the latter having significant benefits in terms of complexity, scalability, reliability and performance. This paper considers the automatic partitioning of the building into subsystems, which enables the distributed simulation, modeling, analysis and management of the intelligent building while ensuring the effective detection and isolation of contaminants in the building interior. Specifically, we develop both a high-quality heuristic algorithm and an optimal Mixed Integer Linear Programming (MILP) formulation for the building partitioning problem. The MILP formulation is based on graph partitioning techniques, while the heuristic is based on matrix clustering techniques. Both approaches partition the building into subsystems while ensuring (i) maximum decoupling between the various subsystems, (ii) strong connectivity between the zones of each subsystem and (iii) control of the size of the subsystems with respect to the number of allocated zones. A combination of the two approaches is also proposed for reconfiguring an initial partitioning composition in real time in order to accommodate partitioning needs that arise from dynamic system changes.

Index Terms—Building Partitioning, Mixed-Integer Linear Programming, Distributed Monitoring Algorithms, Online optimization, Matrix Reordering Techniques, Contaminant Detection and Isolation

I. INTRODUCTION

Intelligent Buildings (IB) are all about ensuring the occupants' comfort, productivity and safety, and at the same time maximizing the building's efficiency, by autonomously governing and adapting the building environment. This is achieved using modern information and communication technologies (ICT), such as sensors (e.g., temperature, occupancy and gas emissions), distributed microprocessors and computers that have sensing, communication, computation, and control capabilities. Sensing is necessary to measure and monitor the state of the building environment, while communication

enables information exchange between various ICT devices. Computation is needed to dynamically process collected information, perceive the environment and make autonomous decisions. These control decisions are then applied to the building environment to meet the design objectives of the considered application.

One such application concerns the monitoring of indoor air quality (IAQ). IAQ is considered one of the three most important factors (the other two are visual and thermal comfort) influencing occupants' quality of life in building environments [1]. IAQ is often compromised by various airborne contaminants that are either generated indoors or penetrate into the indoor environment with passive or active airflows [2], [3]. Under these safety-critical conditions it becomes of paramount importance to promptly detect the presence of the contaminant event and identify the location of the source for the occupants' safety. Passive protective measures have been used for years and include architectural features, physical security and air filtration.

Recently, the emergence of IBs has enabled the application of active protective measures. In this context, IAQ sensors are deployed inside IBs to measure and monitor contaminants that propagate through the different building zones (e.g. rooms) based on the existing airflows. The collected data are gathered at a central computing unit for processing using Contaminant Detection and Isolation (CDI) algorithms which detect the presence of the contaminant source (detection) and estimate its location (isolation) (e.g. building zone). Promising centralized CDI approaches proposed include the adjoint probability method [3], the Bayesian updating technique [4] and the state-space method [5].

Nonetheless, for large-scale buildings' implementations, centralized CDI approaches suffer from several issues, related to realization complexity, scalability and reliability [6]. Therefore, development of distributed CDI schemes is important in order to benefit from the advantages of distributed fault diagnosis architectures and frameworks which have been documented in [7], [8], [9]. In the context of IBs, a distributed CDI scheme was developed in [10], by considering the building as a collection of interconnected subsystems and designing contaminant event monitoring agents for different subsystems. Each monitoring agent aims to detect the contaminant introduced in the underlying subsystem and isolate the building zone where the contaminant source is located. Moreover, each agent is allowed to exchange information with its neighbouring agents. Close investigation of distributed CDI algorithms reveals that the modeling and interconnection uncertainties that

This work was partially supported by the European Research Council (ERC) under the ERC Advanced Grant through the FAULT-ADAPTIVE Project.

A. Kyriacou, S. Timotheou, C. Panayiotou and M. Polycarpou are with the KIOS Research Center for Intelligent Systems and Networks and the Department of Electrical and Computer Engineering, University of Cyprus, e-mail: {kyriacou.alexis, timotheou.stelios, christosp, mpolycar} @ucy.ac.cy

M. Michaelides is with the Department of Electrical Engineering, Computer Engineering and Informatics, Cyprus University of Technology e-mail: michalis.michaelides@cut.ac.cy

directly affect the performance of the algorithms (e.g. speed and accuracy of detection and isolation) strongly depend on the airflows between the zones and the relative size of the subsystems. Hence, one important problem that needs to be addressed involves the appropriate selection of building zones for each subsystem, in order to maximize the performance of the CDI algorithms.

In this paper, we address this building partitioning problem. For its effective solution, we consider three main design criteria: (i) minimize the coupling between subsystems in terms of airflows, (ii) ensure subsystem connectivity, (iii) balance the size between various subsystems with respect to the number of allocated zones. The first criterion directly affects the performance of the CDI algorithms by minimizing the interconnection uncertainty. The second criterion ensures the continuity of the building area comprising a specific subsystem. The third criterion aims to balance the uncertainty across subsystems and the computational effort of each monitoring agent. An additional requirement is that any developed algorithm should be able to quickly repartition the building due to dynamically varying airflows which occur because of either environmental (temperature, wind direction and velocity) or structural (opening/closing of doors/windows) changes.

According to the aforementioned design criteria, we develop two different approaches for the solution of the building partitioning problem: a Mixed Integer Linear Programming (MILP) approach based on an exact mathematical formulation and a matrix-clustering heuristic approach termed the Building Partitioning Heuristic (BPH). The BPH algorithm is based on matrix clustering and incorporates matrix reordering techniques in order to decide on the best partitions. Moreover, it utilizes maximum and minimum size constraints to control the resulting partition size and provides high quality results. Even though it does not guarantee solution optimality, it is able to re-partition the building given an initial solution in a timely manner and in this way adapt to the changing airflows in the building's interior. On the other hand, the proposed MILP is based on graph partitioning techniques. It is able to completely control the formation of the resulting subsystems and provide optimal solutions for the building partitioning problem. Moreover, it employs a novel way of adaptively defining a source zone in every resulting subsystem. The selected sources are then combined with the theory of network flows, to ensure that the formed subsystems are connected.

In summary, the main contributions of this work are the following:

- 1) Formulation of the building partitioning problem as a mathematical constrained optimization problem, the first formulation that enables the effective application of distributed CDI algorithms, and development of a MILP approach for its solution.
- 2) Development of the BPH algorithm for building partitioning which adheres to connectivity and size constraints.
- 3) Combination of the two approaches (MILP and BPH) to obtain real-time and effective solutions to problems that arise from dynamically changing airflows in the interior of the building.

The organization of this paper is as follows: Section II presents the related work on matrix clustering and graph partitioning approaches using connectivity and size constraints. Section III introduces the problem of building partitioning and gives an overview of the problems' objectives and constraints. Section IV presents a novel MILP formulation for building partitioning. Section V describes the two-phases of the BPH approach and discusses the restrictions of each phase, while Section VI discusses the combination of MILP and BPH to achieve on-line optimization capabilities. Section VII illustrates the capabilities of the distributed CDI algorithm which utilizes the partitioning solution of the proposed approaches, evaluates and compares the performance of the two approaches for both real and random buildings in terms of optimality and execution time, and also analyzes the quality of the on-line optimization solution that combines the two approaches. Finally, Section VIII concludes the paper.

Notation: Calligraphic letters denote sets, superscript $(\cdot)^T$ denotes the transpose and $|\mathcal{Z}|$ denotes the cardinality of set \mathcal{Z} .

II. RELATED WORK

Partitioning problems appear in many fields including engineering, machine learning, pattern recognition and economics. Despite the fact that partitioning problems have the same goal, they arise in the literature with different names (e.g. decomposition, clustering, grouping, partitioning), depending on the field of application and the solution approach. Partitioning techniques can be grouped into two main categories: (i) *Decomposition* techniques that consider the system as a whole and try to separate it into subsystems, and (ii) *Clustering or grouping* techniques that consider individual segments of the system and try to combine them in order to form subsystems. Although many partitioning techniques have been proposed in recent decades, there is no universal solution that deals with all associated problems as the vast majority of approaches are subject to different constraints and objective functions. In addition, the *impossibility theorem* [11] confirms the difficulty in the development of a universal partitioning scheme. Differences between schemes arise due to the representation of data (e.g., points in a 2D/3D plane, relational data matrices, graphs), the resulting number of clusters (e.g., known from the start, derived from the result), the execution time requirements and the problem objective. Next, we present both clustering and graph partitioning techniques that are related to the IB partitioning problem, as well as techniques that explicitly consider the connectivity of the resulting partitions.

A. Matrix Clustering

For the automatic partitioning of buildings, the most relevant techniques found in the literature are clustering schemes that incorporate *matrix reordering techniques* which are commonly used in the area of machine-component grouping in production flow [12], [13], [14], [15], [16], [17]. Machine component grouping arranges machines of different purposes into clusters, to suite production of specific component families, depending on the similarity of operations that are meant to be performed. The dataset consists of a matrix with binary values which

indicates the connections between the different machines-components of the system. The goal is to form square blocks of values indicating the clusters, while maximizing the values enclosed in the blocks. Therefore, machines that execute sequential or similar jobs are clustered together. This is achieved through rearranging the rows and columns of the initial matrix. Each row and its corresponding column are rearranged simultaneously, preserving in this way the initial connectivity between rows and columns. Matrix clustering techniques also ensure the connectivity inside the resulting clusters. Additionally, they allow easy application of size constraints to the formed clusters, by disallowing a row to enter a cluster that has reached maximum size; hence, matrix reordering techniques make a good candidate for building partitioning problems. The proposed BPH utilizes maximum and minimum size constraint in combination with a novel row moving policy to achieve block formation and connectivity conservation. An important feature of the BPH, missing from the aforementioned approaches, is the swapping of rows between different blocks. This feature renders the algorithm suitable for online optimization, i.e. it allows the algorithm to obtain fast and effective building partitions under dynamically varying conditions by incorporating good initial solutions.

B. Exact graph partitioning

Buildings can easily be modeled as graphs, where each zone corresponds to a vertex and an airflow path connecting two zones indicates the existence of a weighted edge between them. Therefore, the building partitioning problem can be directly related to graph partitioning schemes which partition an initial graph into subgraphs by removing a number of its edges. It is worth mentioning that many fast heuristic algorithms for large graphs can be found in the literature (see for example [18], [19], [20], [21]). However, they do not offer the same level of performance as exact algorithms that provide optimal solutions. Examples of exact algorithms can be found in [22], [23], [24] where they mainly target and find exact solutions for the balanced bisection problem, while also providing bounds for other small number of resulting partitions. Since the graph partitioning problem is NP-Hard [25], exact algorithms can only target relatively small-sized problems. Although these algorithms include various size constraints, to the best of the authors' knowledge, they do not require connectivity among vertices of the same partition, which is one of the main constraints in the building partitioning problem. In our work, the proposed MILP formulation allows full control over the size of the partitions and explicitly enforces connectivity between zones in the same subsystem.

C. Subsystem Connectivity

Enforcing connectivity inside each partition has been explicitly investigated in other graph partitioning problems. In [26], an iterative method is proposed for partitioning 2D- and 3D-meshes while minimizing the number of edges that are not included in a partition. In the aforementioned work, connectivity is preserved by constructing iteratively different subsets of partitions called fronts, by accumulating only nodes

that share a connection in each subset. The theory of network flows has also been a major contributor of exactly modeling connectivity in graph partitioning problems. Connectivity preservation using network flows is achieved by ensuring that a unit of flow initiated from any node of one partition can reach any other node of the same partition. This is accomplished by having a source node send one unit of flow to all other nodes (sinks) [27]. In the general case this is quite challenging, as the nodes in each partition and hence the source nodes are not known a priori. Solution to a special case where only one partition is formed from a set of vertices is given in [28].

In contrast to the aforementioned literature, the developed MILP approach provides extensive flexibility over the partitioning solutions by allowing the incorporation of general connectivity and size constraints. Although the BPH allows for less control, it can incorporate initial exact solutions from the MILP approach to provide effective and fast (suitable for real-time application) building re-partitioning when airflow changes occur.

III. PROBLEM STATEMENT

Consider an n -zone building as a centralized system, where the contaminant dispersion in the indoor environment of the building is described by the following multi-zone state-space model¹ [29]:

$$\begin{aligned}\Sigma: \quad \dot{\chi}(t) &= A\chi(t) + Q^{-1}Bv(t) + Q^{-1}G\gamma(t) \\ \psi(t) &= C\chi(t) + \omega(t)\end{aligned}$$

where $\chi \in \mathbb{R}^n$ represents the concentration of the contaminant in the building zones, while $A \in \mathbb{R}^{n \times n}$ is the state transition matrix with its elements $A_{i,j}$ modeling the changes in contaminant concentration between zone i and zone j , primarily as a result of the air-flows; note that $A_{i,j} = A_{j,i} = 0$ if there is no leakage path connecting zones i and j . The controllable inputs in the form of doors, windows, fans and air handling units are represented by $v \in \mathbb{R}^p$, $B \in \mathbb{B}^{n \times p}$ is a zone index matrix concerning their location with $\mathbb{B} = \{0, 1\}$, while matrix $Q \in \mathbb{R}^{n \times n}$ holds the volumes of the zones in its diagonal. The last term indicates the location and characteristics of the contaminant sources, represented by $G \in \mathbb{B}^{n \times s}$ and $\gamma \in \mathbb{R}^s$, respectively. The outputs of the sensors monitoring Σ are represented by $\psi \in \mathbb{R}^m$, while $C \in \mathbb{B}^{m \times n}$ is a zone index matrix for the sensor locations and $\omega \in \mathbb{R}^m$ characterizes the additive measurement noise.

Now consider the same building decomposed into K interconnected subsystems $\Sigma_I, I \in \{1, \dots, K\}$, where each subsystem consists of n_I zones. The corresponding distributed model¹ for contaminant dispersion inside the building is given in [10] as:

$$\begin{aligned}\Sigma_I: \dot{\chi}_I(t) &= A_I\chi_I(t) + Q_I^{-1}B_Iv_I(t) + Q_I^{-1}G_I\gamma_I(t) + H_I\zeta_I(t) \\ \psi_I(t) &= C_I\chi_I(t) + \omega_I(t)\end{aligned}$$

Terms $\{\chi_I, v_I, \gamma_I, \psi_I, \omega_I\}$ and $\{A_I, B_I, C_I, Q_I, G_I\}$ are local sub-vectors and sub-matrices of subsystem Σ_I of appropriate

¹The state-space model representation deviates from the traditional notation in order to be distinguishable from the rest of the paper notation.

dimensions, having the same meaning with the corresponding terms of the centralized system Σ . In addition, $\zeta_I \in \mathbb{R}^l$ denotes the vector of interconnection flows in subsystem Σ_I and H_I is a matrix composed of the elements of A associated with the interconnection flows in subsystem Σ_I .

A key observation regarding distributed CDI is that the detection and isolation performance depends significantly on the sum of all interconnection flows. By reducing the interconnection flows, the inter-dependencies between subsystems become weaker. Thus, possible contaminant events in one subsystem do not propagate to neighboring subsystems, leading to improved performance. Hence, given a state transition matrix A of a building system Σ , the objective is to minimize the Partitioning Cost (PC) of splitting the building into K subsystems in order to minimize the interconnection airflows. PC is expressed as:

$$PC = \sum_{(i,j) \notin \mathcal{A}_I, I \in \{1, \dots, K\}} A_{i,j} = \sum_{I=1}^K \sum_{(i,j)} H_{I,i,j} \quad (1)$$

where \mathcal{A}_I denotes the set of elements (i, j) of matrix A which define matrix A_I of subsystem I .

In addition to the minimization of the objective defined in (1), the considered problem has two main constraints. The first, is that all zones in each subsystems should be interconnected to ensure continuity of the building area comprising each subsystem. The second constraint controls the size of each subsystem so that all subsystems have more than \underline{M} and less than \overline{M} zones and/or the relative size difference of the subsystems does not exceed R . Restricting the subsystem size maintains the computational effort of the CDI algorithms under control; this is particularly important as the computational complexity of the CDI algorithms increases exponentially with the number of involved zones. A final requirement for the considered problem is that the developed partitioning algorithms should be able to quickly re-optimize the problem when environmental and structural changes affecting the building airflows take place. Note that, in a real-world scenario, such changes may be detected and their effects estimated through the utilization of advance information processing and classification approaches [30], [31].

IV. EXACT MILP FORMULATION

In this section we describe the graph-based MILP formulation. The formulation adheres to three sets of constraints: “Basic partitioning constraints” to partition the graph, “Connectivity constraints” to ensure connectivity of zones in each partition and “Size constraints” to control the size of the partitions.

Consider an undirected, connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a matrix $D = A + A^T \in \mathbb{R}^{N \times N}$, $N = |\mathcal{V}|$ with edge weights $d_{i,j}$, $(i, j) \in \mathcal{E}$ defined in terms of the state transition matrix A so that the edge weights are symmetric. The objective is to partition the graph into $K \in \mathbb{Z}^+$ subgraphs while minimizing the edge weights that are not included in any subgraph which represent the PC defined in (1). Let variables $z_{i,j}$, $(i, j) \in \mathcal{E}$ indicate whether edge (i, j) belongs to any subgraph ($z_{i,j} = 1$)

or not ($z_{i,j} = 0$). The objective of the problem can then be expressed as:

$$\min_{\{x,y,z,f,w,u,s,g\}} \sum_{(i,j) \in \mathcal{E}} \frac{1}{2} (1 - z_{i,j}) d_{i,j}. \quad (2)$$

The above objective is subject to various constraints imposed by the set of variables $\{x, y, z, f, w, u, s, g\}$ defined in the sequel.

Basic partitioning constraints (3a)-(3g) enforce the formation of subgraphs. The set of binary variables $x_{i,h}$, $i \in \mathcal{V}$, $h \in \mathcal{K}$ indicates whether vertex i belongs to subgraph \mathcal{G}_h and is the output of the formulation. Auxiliary variables $w_{i,j,h}$ indicate whether edge (i, j) belongs to subgraph \mathcal{G}_h ($w_{i,j,h} = 1$) or not ($w_{i,j,h} = 0$) as indicated by (3a)-(3b) while the values of $z_{i,j}$ are enforced by constraints (3c)-(3d). Note that although both $w_{i,j,h}$ and $z_{i,j}$ are defined as continuous variables they always take binary values. The case $w_{i,j,h} = 0$ is enforced when either $x_{i,h} = 0$ or $x_{j,h} = 0$ from constraints (3a)-(3b). When $x_{i,h} = 1$ and $x_{j,h} = 1$, then $w_{i,j,h}$ can take values in the range $[0, 1]$; nonetheless, the most beneficial value is $w_{i,j,h} = 1$ because it ensures that $z_{i,j} = 1$ (see (3d)) which minimizes the objective (2). Finally, equality (3e) ensures that a vertex is only included in one subgraph.

Basic partitioning constraints

$$w_{i,j,h} \leq x_{i,h}, \quad (i, j) \in \mathcal{E}, h \in \mathcal{K} \quad (3a)$$

$$w_{i,j,h} \leq x_{j,h}, \quad (i, j) \in \mathcal{E}, h \in \mathcal{K} \quad (3b)$$

$$z_{i,j} = z_{j,i}, \quad (i, j) \in \mathcal{E} \quad (3c)$$

$$z_{i,j} = \sum_{h \in \mathcal{K}} w_{i,j,h}, \quad (i, j) \in \mathcal{E} \quad (3d)$$

$$\sum_{k \in \mathcal{K}} x_{i,k} = 1, \quad i \in \mathcal{V} \quad (3e)$$

$$w_{i,j,h} \in [0, 1], \quad z_{i,j} \in [0, 1], \quad (i, j) \in \mathcal{E}, h \in \mathcal{K} \quad (3f)$$

$$x_{i,h} \in \{0, 1\}, \quad i \in \mathcal{V}, h \in \mathcal{K} \quad (3g)$$

Solving the problem with the Basic partitioning constraints (3a)-(3g) results in K subgraphs with minimum partitioning cost. However, connectivity inside the subgraphs may not be preserved. To ensure connectivity between vertices (i.e. building zones) inside a subgraph the theory of *network flows* has been used. This is achieved by sending one unit of flow from one source vertex in each subgraph to all other vertices of the same subgraph. Nonetheless, this task is challenging because the resulting subgraphs are not predetermined; hence the identification and selection of sources is not trivial. A novel way of adaptively recognizing source vertices and ensuring connectivity is presented in the (4a)-(4h).

In the connectivity constraints, the role of (4a)-(4e) is the selection of the source nodes for each subgraph. In these constraints, variables $y_{j,h}$, $j \in \mathcal{V}$, $h \in \mathcal{K}$ indicate whether a vertex with smaller or equal index than j belongs to \mathcal{G}_h , as expressed by (4a)-(4b). Variables $u_{j,h}$, $j \in \mathcal{V}$, $h \in \mathcal{K}$ indicate whether vertex j has the smallest index of all vertices belonging to \mathcal{G}_h ($u_{j,h} = 1$) or not ($u_{j,h} = 0$), as expressed

Connectivity constraints

$$\sum_{i=1}^j \frac{x_{i,h}}{N} \leq y_{j,h} \leq \sum_{i=1}^j x_{i,h}, \quad j \in \mathcal{V}, h \in \mathcal{K} \quad (4a)$$

$$y_{j,h} \geq x_{j,h}, \quad j \in \mathcal{V}, h \in \mathcal{K} \quad (4b)$$

$$u_{j,h} = y_{j,h} - y_{j-1,h}, \quad j \in \mathcal{V} \setminus \{1\}, h \in \mathcal{K} \quad (4c)$$

$$u_{1,h} = y_{1,h}, \quad h \in \mathcal{K} \quad (4d)$$

$$\sum_{j \in \mathcal{V}} u_{j,h} = 1, \quad h \in \mathcal{K} \quad (4e)$$

$$u_{j,h} \sum_{i \in \mathcal{V}} x_{i,h} - x_{j,h} + \sum_{\substack{v,j \in \mathcal{V} \\ (v,j) \in \mathcal{E}}} f_{v,j,h} = \sum_{\substack{v,j \in \mathcal{V} \\ (j,v) \in \mathcal{E}}} f_{j,v,h} \quad j \in \mathcal{V}, h \in \mathcal{K} \quad (4f)$$

$$0 \leq f_{i,j,h} \leq \overline{M} z_{i,j} \quad (i,j) \in \mathcal{E} \quad (4g)$$

$$y_{j,h} \in [0, 1], \quad u_{j,h} \in [0, 1] \quad j \in \mathcal{V}, h \in \mathcal{K} \quad (4h)$$

Linearization of constraint (4f)

$$s_h = \sum_{i \in \mathcal{V}} x_{i,h} \quad h \in \mathcal{K} \quad (5a)$$

$$\underline{M} u_{j,h} \leq g_{j,h} \leq \overline{M} u_{j,h} \quad j \in \mathcal{V}, h \in \mathcal{K} \quad (5b)$$

$$g_{j,h} \leq s_h - (1 - u_{j,h}) \underline{M} \quad j \in \mathcal{V}, h \in \mathcal{K} \quad (5c)$$

$$s_h - (1 - u_{j,h}) \overline{M} \leq g_{j,h} \quad j \in \mathcal{V}, h \in \mathcal{K} \quad (5d)$$

$$(g_{j,h} - x_{j,h}) + \sum_{v \in \mathcal{V}} f_{v,j,h} = \sum_{v \in \mathcal{V}} f_{j,v,h} \quad j \in \mathcal{V}, h \in \mathcal{K} \quad (5e)$$

Size constraints

$$\underline{M} \leq s_h \leq \overline{M} \quad h \in \mathcal{K} \quad (6a)$$

$$-R \leq s_h - s_k \leq R \quad h, k \in \mathcal{K}, h \neq k \quad (6b)$$

by (4c)-(4e). According to these definitions, we define the source node j_h of subgraph \mathcal{G}_h as the one with the smallest index, i.e. $u_{j_h,h} = 1$. The role of (4f) - (4h) is to ensure the flow conservation constraints in each subgraph \mathcal{G}_h when sending one unit of flow from j_h to the other nodes of the same subgraph. In these constraints, continuous variables $f_{j,v,h}$, $(j,v) \in \mathcal{E}, h \in \mathcal{K}$ indicate the flow of edge (j,v) for subgraph \mathcal{G}_h . The first two terms of (4f) denote the amount of flow originating and consumed at node j in subgraph \mathcal{G}_h , respectively. When $u_{j,h} = 1$, i.e. j is the source in subgraph \mathcal{G}_h , $\sum_{i \in \mathcal{V}} x_{i,h}$ units of flow are sent to the other nodes of the subgraph. When $x_{j,h} = 1$, i.e. j is a node that belongs to subgraph \mathcal{G}_h , j is a destination node that consumes one unit of flow. The last two terms represent the total inflow and the total outflow of the vertex. Finally, constraints (4g) ensures that the flows only propagate through edges that belong to particular subgraphs, as indicated by variables $z_{i,j}$.

The presence of products of continuous and binary variables in (4f) makes the problem nonlinear and hence not suitable for Mixed Integer Linear/Quadratic Programming (MILP/MIQP) solvers. To resolve this issue, the first term of (4f) has been linearized. Considering that $u_{j,h} \in \{0, 1\}$ and setting $s_h = \sum_{i \in \mathcal{V}} x_{i,h}$, the product $g_{j,h} = u_{j,h} s_h$ can be expressed using linear constraints through (5a)–(5e).

The third set of constraints involve bounding the absolute or relative size of subgraphs. Having defined the size of each subgraph with s_h in (5a), these constraints can be easily expressed using (6a)–(6b). Note that these constraints are inactive when $\underline{M} = 1$, $\overline{M} = N$ and $R = N$, respectively. Constraints (6a) and (6b) are meant to be used separately according to the requirements of each problem. In case the two constraints are used together, \underline{M} , \overline{M} and R should be chosen carefully to avoid infeasible solutions.

To illustrate the importance of connectivity and size constraints, consider the example of Fig. 1, which represents a six-zone building with airflows as indicated by the edge weights.

The optimal solution derived when $\underline{M} = 2$, $\overline{M} = 4$ and $R = 2$ is given by Cut 1 with $PC = 10$. The best solution without the connectivity constraints is given by Cut 2 with $PC = 2$. Finally, if both size and connectivity constraints are excluded, then the solution is given by Cut 3 with $PC = 1$. From the three cases, it can be deduced that connectivity and size constraints reduce the quality of the solution in terms of PC, but balance the subsystem size while ensuring connectivity constraints.

To summarize, the building partitioning problem is the minimization of the partitioning cost (2) subject to the basic partitioning constraints (3a)–(3g), the connectivity constraints (4a)–(4e), (5a)–(5e), (4g)–(4h) and the size constraint (6a) and/or (6b). Even though the exact solutions of partitioning problems that include connectivity and size constraints are NP-Hard and hence of exponential complexity, they are valuable for finding solutions for small sized problems and for the proper evaluation of heuristic solutions obtained by other algorithms.

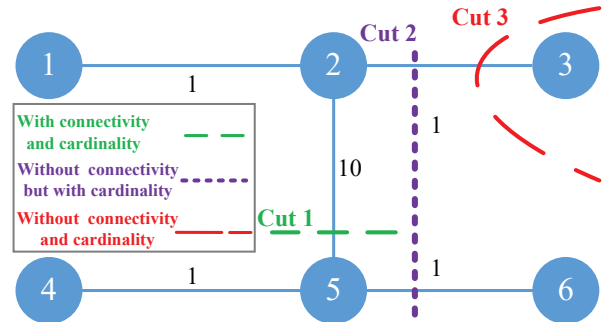


Fig. 1. Example of results with both the constraints of connectivity and cardinality (Cut 1), with only the cardinality constraint (Cut 2) and with no constraint (Cut 3).

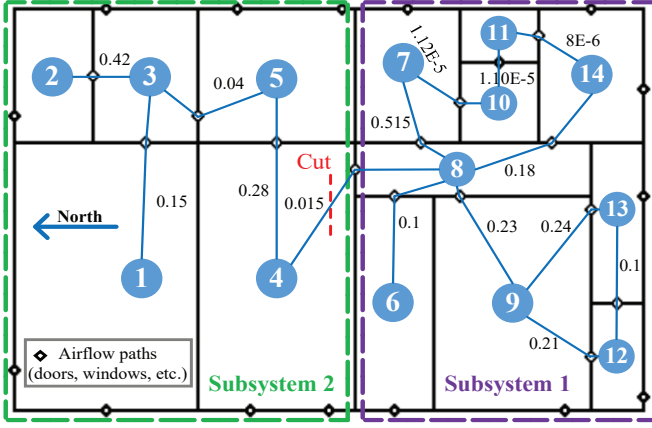


Fig. 2. Holmes House building partitioning using the MILP approach for $K = 2$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-0.15	0	0.15	0	0	0	0	0	0	0	0	0	0	0
2	0	-0.42	0.42	0	0	0	0	0	0	0	0	0	0	0
3	0	0	-0.9	0	0.04	0	0	0	0	0	0	0	0	0
4	0	0	0	-0.28	0.28	0	0	0	0	0	0	0	0	0
5	0	0	0	0	-0.563	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	-0.1	0	0.1	0	0	0	0	0	0
7	0	0	0	0	0	0	-0.252	0	0	0	0	0	0	0
8	0	0	0	0.015	0	0	0.515	-0.71	0	0	0	0	0	0.18
9	0	0	0	0	0	0	0	0.23	-0.23	0	0	0	0	0
10	0	0	0	0	0	0	1.12E-05	0	0	-1.12E-05	0	0	0	0
11	0	0	0	0	0	0	0	0	0	1.10E-05	-1.10E-05	0	0	0
12	0	0	0	0	0	0	0	0	0.21	0	0	-0.31	0.1	0
13	0	0	0	0	0	0	0	0	0.24	0	0	0	-0.24	0
14	0	0	0	0	0	0	0	0	0	0	8.0E-06	0	0	-0.3

Fig. 3. State transition matrix A .

A. Illustrative Example

To better comprehend the MILP formulation, consider a building simulation example using *Holmes house*² shown in Figure 2. In the scenario considered, the wind is blowing from the north-east with speed 10 m/s, the ambient temperature is 25° C, while the doors and windows are in the fully open position. When simulated using CONTAM [33], a multizone airflow and contaminant transport analysis software, the particular environmental and building conditions create the state transition matrix A that appears in Fig. 3. Each matrix entry $A_{i,j}$ denotes the airflow created from zone j to zone i . Note that the sum of incoming flows should be equal to the sum of outgoing flows for each zone. Hence, the entries in each row of matrix A should sum up to zero unless there are external incoming flows. In the considered scenario, zones Z3, Z5, Z7 and Z14 are affected by external incoming flows due to the wind direction. Hereafter, the diagonal elements are set to zero since they do not affect the partitioning solution.

Each room in the house is one zone so that the particular building is comprised of 14 zones and hence 14 vertices as depicted in Figure 2. The weight edges, $d_{i,j}$, are equal to the sum of the airflows in both directions between corresponding zones, i.e. $d_{i,j} = A_{i,j} + A_{j,i}$. The optimal building partitioning obtained for $\underline{M} = 3$, $\overline{M} = 9$ and $K = 2$ is also shown in Figure 2. It is easy to see that the minimum cut partitioning

the graph into two subgraphs includes only the edge (4,8), with $PC=0.015$. Note that, there are solutions with smaller PC for $K = 2$, however, none satisfies the size constraint of each subsystem ($\underline{M} = 3$).

To gain a better insight to the basic partitioning and connectivity constraints, let us examine more closely the solution for the first subsystem. The optimal solution obtained from the MILP formulation for \mathcal{V}_1 is $\mathbf{x}_1 = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]^T$. Combining the solution with constraints (4a) and (4b), the vector $\mathbf{y}_1 = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]^T$ is formed where the first non-zero element appears in the 6th index. Clearly, the first non-zero element corresponds to the vertex with the smallest index in \mathcal{G}_1 . In order to distinguish that particular vertex, constraints (4c)-(4e) ensure that $u_{6,1} = 1$ while $u_{j,1} = 0, j \in \mathcal{V} \setminus \{6\}$. Note that, the utilization of (4e) is crucial since it forces $u_{j,h}$ to 0 for $j > j_h$, where j_h is the vertex with the smallest index in subgraph \mathcal{G}_h . Also, constraint (4d) complements (4c), especially when the globally smallest index vertex is included in a subgraph, as is the case for the second subgraph of the example. With regards to the basic partitioning constraints, it is true that all variables $z_{i,j}$ have value equal to one except from $z_{4,8}$ which is equal to zero, as $w_{4,8,1} = 0$ and $w_{4,8,2} = 0$.

Note that the particular example, with the corresponding state transition matrix depicted in Figure 3 will also be used to explain the operation of the BPH algorithm.

V. BUILDINGS PARTITIONING HEURISTIC

In this section, we develop the Buildings Partitioning Heuristic (BPH) to solve the considered problem which is based on matrix reordering techniques. Although the particular heuristic does not always provide optimal solutions, the advantage of using BPH over MILP solvers is that it can obtain fast close-to-optimal solutions in polynomial complexity time, whereas the solution to the MILP is of exponential computational complexity in the worst case. The BPH is a modified version of the matrix clustering algorithm proposed in [17] that can also handle size and relative difference size constraints. Even though the problem could also be addressed using other computational intelligence approaches such as genetic algorithms, the benefit of the BPH is that it can also be used on-line (see Section VI).

The heuristic rearranges the rows and columns of the state transition matrix A in such a way that blocks of values are formed indicating the resulting subsystems. Its objective, is to minimize the values that are not included in any block (partitioning cost) and retain their individual connectivity while restricting the maximum and minimum block size. Furthermore, possibly its biggest advantage, is its ability to utilize an existing subsystem configuration as a starting point and optimize it on-line; hence, compensating for the dynamic nature of the buildings' airflows.

The heuristic algorithm is composed of two phases: (a) the "Dividing" and (b) the "Regrouping" phase. In the first phase, "Dividing", the matrix is separated into blocks that satisfy the constraint regarding the maximum number of zones inside a

²Holmes house is a low-rise residential house model used by J.D. Holmes and then scaled up for simulating airflow and contaminant transport in and around buildings [32].

subsystem while also minimizing the PC. In the second phase, “Regrouping”, the smaller blocks are merged to form bigger ones in order to satisfy the constraint regarding the minimum number of zones inside a subsystem.

The heuristic algorithm uses the index function $f : \mathcal{N} \rightarrow \mathcal{K}$, where $\mathcal{N} = \{1, \dots, n\}$, $\mathcal{K} = \{1, \dots, K\}$, to indicate the block which includes row $i \in \mathcal{N}$. For instance, $f(2) = 3$ means that row 2 is in block 3. Also, \mathcal{M}_k is a set that includes all row indices included in the k^{th} block, while $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_K\}$ is a structure containing all sets \mathcal{M}_k . Note that with each row rearrangement the corresponding column is rearranged simultaneously for preservation of the airflow connections. For example, $\mathcal{M}_k = \{2, 4\}$ implies that block k includes elements (2, 2), (2, 4), (4, 2) and (4, 4). In addition, $\sigma_i^k = \sum_{l \in \mathcal{M}_k} (A_{i,l} + A_{l,i})$ is the sum of flows of all elements associated with row and column i inside block k and denotes the benefit of having row i into block k . Moreover, $S_{i \rightarrow k} = \sigma_i^k - \sigma_i^{f(i)}$ is used to indicate the relative benefit of moving row i from its current block, $f(i)$, to block k . Finally, the sum of values that are not included in any block form the Partitioning Cost (PC) as defined in (1). The main objective of the BPH is to minimize the PC.

A. Dividing Phase

The dividing phase aims to split matrix A into blocks of elements in order to maximize the sum of element values inside all blocks – or equivalently minimize the sum of element values outside all blocks, i.e. the PC – while also satisfying a maximum block size constraint. Towards this direction, a greedy policy is used for changing the blocks of particular rows that reduce the PC (see function CHANGEBLK). Changing the block of row i from $f(i)$ to k may occur as a result of either *moving* row i to block k (lines 2-4 of CHANGEBLK), if there is space in block k , or *replacing* another row of the particular block with row i , if block k has maximum size (lines 5-12 of CHANGEBLK). This replacement capability renders the heuristic capable of optimizing on-line an initial solution and increases the possibility of reaching optimality.

```

1: function CHANGEBLK( $i, k, \mathcal{M}, Imp$ )
   Considers changing the block of row  $i$  to  $k$ .
2:   if ( $|\mathcal{M}_k| < \bar{M}$ ) then
3:     Move row  $i$  to  $\mathcal{M}_k$ 
4:     Set  $Imp \leftarrow True$ 
5:   else
6:     for all  $l \in \mathcal{M}_k$  do
7:        $\hat{\sigma}_l = \sigma_i^k - \sigma_l^k - (A_{i,l} + A_{l,i})$ 
8:        $\hat{\sigma}_l^* \leftarrow \max_l \hat{\sigma}_l$ ;  $\hat{l} \leftarrow \arg \max_l \hat{\sigma}_l$ 
9:       if ( $\hat{\sigma}_l^* > 0$ ) then
10:        Replace row  $\hat{l}$  with  $i$  in  $\mathcal{M}_k$ 
11:        Create a singleton block for  $\hat{l}$ 
12:        Set  $Imp \leftarrow True$ 
13:   return  $\mathcal{M}, Imp$ 
14: end function

```

In more detail, at the beginning of the dividing phase, each row of the matrix is inserted into a block of its own

Dividing Phase

```

1: Set the maximum block size  $\bar{M}$ 
2: for all rows  $i$  do
3:   Generate singleton block  $\mathcal{M}_i = \{i\}$ 
4: repeat
5:   Set  $Imp \leftarrow False$ 
6:   for all rows  $i$  do
7:     for all rows  $j$  do
8:       if  $f(i) \neq f(j)$  then
9:         Compute  $S_{i \rightarrow f(j)}$  and  $S_{j \rightarrow f(i)}$ 
10:        if ( $S_{i \rightarrow f(j)} > 0$  OR  $S_{j \rightarrow f(i)} > 0$ ) then
11:          if ( $S_{i \rightarrow f(j)} > S_{j \rightarrow f(i)}$ ) then
12:            [ $\mathcal{M}, Imp$ ] = CHANGEBLK( $i, f(j), \mathcal{M}, Imp$ )
13:          else if ( $S_{i \rightarrow f(j)} < S_{j \rightarrow f(i)}$ ) then
14:            [ $\mathcal{M}, Imp$ ] = CHANGEBLK( $j, f(i), \mathcal{M}, Imp$ )
15:          else if ( $S_{i \rightarrow f(j)} = S_{j \rightarrow f(i)}$ ) then
16:            if ( $|\mathcal{M}_{f(i)}| \geq |\mathcal{M}_{f(j)}|$ ) then
17:              [ $\mathcal{M}, Imp$ ] = CHANGEBLK( $i, f(j), \mathcal{M}, Imp$ )
18:            else ( $|\mathcal{M}_{f(i)}| < |\mathcal{M}_{f(j)}|$ )
19:              [ $\mathcal{M}, Imp$ ] = CHANGEBLK( $j, f(i), \mathcal{M}, Imp$ )
20:   until ( $Imp = False$ )
21: Determine connected blocks

```

(referred to as *singleton block*) (lines 2-3 of the Dividing Phase). Then, a series of movements or replacements between rows of different blocks are performed, in order to maximize the values inside each block while preserving the maximum block size constraint. To achieve this, all pair combinations of rows (i, j) belonging to different blocks are examined (lines 5-7). For each examined pair, the relative benefit of moving rows i to the block of j is computed, i.e. $S_{i \rightarrow f(j)}$; similarly $S_{j \rightarrow f(i)}$ is also computed (line 8). If either of the two relative benefits is positive (line 9), one out of three cases is engaged:

- **Case 1:** If $S_{i \rightarrow f(j)} > S_{j \rightarrow f(i)}$, then it is considered to change the block of row i to $f(j)$ (lines 10-11).
- **Case 2:** If $S_{i \rightarrow f(j)} < S_{j \rightarrow f(i)}$, then it is considered to change the block of row j to $f(i)$ (lines 12-13).
- **Case 3:** If $S_{i \rightarrow f(j)} = S_{j \rightarrow f(i)}$, then it is considered to change the block of the row which belongs to the largest block among $f(i)$ and $f(j)$ (lines 14-18).

This procedure is repeated until no beneficial change of block exists. The final stage of the Dividing Phase is to check the connectivity of the formed blocks (lines 21). In case a block is unconnected, then smaller blocks need to be formed that are composed of only connected components. This procedure can be easily and efficiently implemented by constructing the graph of the non-zero elements of the particular block and grouping together only connected components [34, p.499–501]. Note that during this process the PC remains unaltered. The result of the Dividing Phase is a number of connected blocks.

Considering the example of Section IV-A, Fig. 4 represents a snapshot of the Dividing phase after a number of steps. In this snapshot four blocks are formed as indicated by the shaded areas: $\mathcal{M}_1 = \{1, 2, 3, 4, 5\}$, $\mathcal{M}_2 = \{6, 7, 8, 9, 10, 11, 12\}$, $\mathcal{M}_3 = \{13\}$ and $\mathcal{M}_4 = \{14\}$. The

	1	3	2	5	4	8	6	7	9	10	11	12	13	14
1	0	0.15	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0.04	0	0	0	0	0	0	0	0	0	0
2	0	0.42	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0.28	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0.015	0	0	0.515	0	0	0	0	0	0.18
6	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0.23	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1.12E-05	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1.10E-05	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0.21	0	0	0	0.1	0
13	0	0	0	0	0	0	0	0	0.24	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	8E-06	0	0	0	0

Fig. 4. Transition matrix A for Holmes house during the execution of the Dividing phase. The screenshot depicts the processing of rows 12 and 13 for the computation of $S_{12 \rightarrow 3}$ and $S_{13 \rightarrow 2}$. The solid and dashed red circles represent the elements associated with the calculation of σ_{12}^2 and σ_{13}^3 respectively, towards the computation of $S_{12 \rightarrow 3}$. Similarly, the solid and dashed green rectangles represent σ_{13}^3 and σ_{12}^2 respectively, towards the computation of $S_{13 \rightarrow 2}$.

	1	3	2	5	4	8	6	7	9	10	11	12	13	14
1	0	0.15	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0.04	0	0	0	0	0	0	0	0	0	0
2	0	0.42	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0.28	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0.015	0	0	0.515	0	0	0	0	0	0.18
6	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0.23	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1.12E-05	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	1.10E-05	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0.21	0	0	0	0.1	0
13	0	0	0	0	0	0	0	0	0.24	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	8E-06	0	0	0	0

Fig. 5. Transition matrix A for Holmes house during the replacement operation of function $\text{CHANGE_BLK}(13, 2, \mathcal{M})$. The solid and dashed circles depict the elements added and subtracted respectively when computing $\hat{\sigma}_{11}$.

current value of the partitioning cost is $PC = 0.535$. Rows $i = 12$ and $j = 13$ are currently processed which belong to blocks $f(12) = 2$ and $f(13) = 3$, respectively. In this example the maximum block size considered is equal to $\overline{M} = 7$.

At first, the two relative benefits are computed yielding $S_{12 \rightarrow 3} = \sigma_{12}^3 - \sigma_{12}^2 = A_{12,13} - A_{12,9} = 0.1 - 0.21 = -0.11$ and $S_{13 \rightarrow 2} = \sigma_{13}^2 - \sigma_{13}^3 = (A_{13,9} + A_{12,13}) - A_{13,13} = 0.24 + 0.1 - 0 = 0.34$ illustrated in Fig. 4. Because $S_{13 \rightarrow 2}$ is positive and $S_{13 \rightarrow 2} > S_{12 \rightarrow 3}$ the function $\text{CHANGE_BLK}(13, 2, \mathcal{M}, \text{Imp})$ is called. During the function call, the replacement operation is activated because the size of block 2 is equal to the maximum block size, as shown in Fig. 5. The sum of values in block 2 for every possible replacement of one of its rows with row 13 are calculated yielding $\hat{\sigma}_l = \{0.24, -0.175, -0.505, -0.34, 0.33998, 0.33999, 0.03\}$, $l \in \{6, 7, 8, 9, 10, 11, 12\}$ (lines 6-7 of CHANGE_BLK). The largest positive value is $\hat{\sigma}_{11} = 0.33999$, and hence row 11 in block 2 is replaced by the candidate row 13, while row 11 is moved to a singleton block as shown in Fig. 6. The resulting new blocks are $\mathcal{M}_1 = \{1, 2, 3, 4, 5\}$, $\mathcal{M}_2 = \{6, 7, 8, 9, 10, 11, 13\}$, $\mathcal{M}_3 = \{14\}$ and $\mathcal{M}_4 = \{12\}$ with cost equal to $PC = 0.195$. The partitioning of the transition matrix A at the end of the Dividing Phase is shown in Fig. 7 with cost equal to $PC = 0.0150192$.

Overall, the Dividing phase manages to form the blocks by grouping the matrix values according to the maximum size constraint while minimizing the values that remain outside all blocks, i.e. the PC. However, as the minimum block size

	1	3	2	5	4	11	8	6	7	9	10	13	12	14
1	0	0.15	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0.04	0	0	0	0	0	0	0	0	0	0
2	0	0.42	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0.28	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1.10E-05	0	0	0
8	0	0	0	0	0.015	0	0	0	0.515	0	0	0	0	0.18
6	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0.23	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	1.12E-05	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0.24	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0.21	0	0.1	0	0
14	0	0	0	0	0	8E-06	0	0	0	0	0	0	0	0

Fig. 6. Transition matrix A for Holmes house during the execution of the Dividing phase. The screenshot depicts the result from replacing row 11 with row 13 in block 2, and creating a singleton block for row 11.

Regrouping Phase

- 1: Set Minimum Zone Size \underline{M}
- 2: **while** Blocks with size $< \underline{M}$ exist **do**
- 3: **for all** Blocks i with size $< \underline{M}$ **do**
- 4: **for all** Blocks $j \neq i$ **do**
- 5: $C_{i,j} = \sum_{l \in \mathcal{M}_i} \sum_{k \in \mathcal{M}_j} (A_{l,k} + A_{k,l})$
- 6: $j^* = \text{argmax}_{j \in \{1, \dots, n\}} C_{i,j}$
- 7: **if** $(C_{i,j^*} > 0)$ **then**
- 8: Merge block i with block j^*

constraints are not examined, the size of certain blocks may be below \underline{M} while there may also exist singleton blocks. The solution to this issue is given by the Regrouping phase presented in the next section.

B. Regrouping Phase

The Regrouping Phase is an iterative procedure that merges different blocks to ensure satisfaction of the minimum size constraints. In each iteration, the benefit of merging one *small* block i (with size smaller than \underline{M}) with another block is examined and the benefit of merging it with another block j , $C_{i,j} = \sum_{l \in \mathcal{M}_i} \sum_{k \in \mathcal{M}_j} (A_{l,k} + A_{k,l})$ is computed. The value of $C_{i,j}$ is essentially the sum of all interconnection flows between blocks i and j ; hence, a zero value implies that the particular blocks are not connected. In this process, the block j^* with the maximum $C_{i,j}$ value is merged with block i if the two blocks are connected, i.e. $C_{i,j^*} > 0$. The procedure is repeated until all small blocks are eliminated. Note that, the regrouping phase retains block connectivity so that BPH results in a number of connected blocks.

Consider for example the transition matrix A at the end of the Dividing Phase depicted in Fig. 7, with minimum block size of $\underline{M} = 3$. This matrix is comprised of three blocks of size $|\mathcal{M}_1| = 5$, $|\mathcal{M}_2| = 2$ and $|\mathcal{M}_3| = 7$, with block 2 being a small block as $|\mathcal{M}_2| < \underline{M}$. In this setting, the merging benefit with blocks 1 and 3 is equal to $C_{2,1} = 0$ and $C_{2,3} = 1.82E - 05$ and block 2 is merged with block 3 yielding a partitioning cost $PC = 0.015$. Notice that $C_{2,1} = 0$ which implies that blocks 1 and 2 are not connected.

C. BPH properties

One interesting characteristic of the BPH is that different permutations of the transition matrix (A) and input parameters

	1	3	2	5	4	10	11	8	6	7	9	14	13	12
1	0	0.15	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0.04	0	0	0	0	0	0	0	0	0	0
2	0	0.42	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0.28	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1.12E-05	0	0	0	0
11	0	0	0	0	0	1.1E-05	0	0	0	0	0	0	0	0
8	0	0	0	0	0.015	0	0	0	0	0.515	0	0.18	0	0
6	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0.23	0	0	0	0	0	0
14	0	0	0	0	0	0	8E-06	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0.24	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0.21	0	0.1	0

Fig. 7. Screenshot of Holmes house after the execution of the Dividing phase with maximum block size set to 7.

$(\underline{M}, \overline{M})$ create partitioning solutions with varying characteristics in terms of the number and size of blocks. Therefore, by executing the BPH for various configurations, the best solutions can be stored for different parameters $(K, \underline{M}, \overline{M})$. Consequently, a set of solutions is produced that satisfies a wide range of size constraints (both absolute and relative) for different numbers of blocks.

Another important characteristic of the BPH is its flexibility in handling application specific constraints during the Regrouping Phase. The reason is that the Regrouping Phase allows to easily incorporate size constraints when merging different blocks or to explicitly keep track of the number of blocks until a desirable number is reached. Possible customizations include: a) regulating the number of resulting blocks by forcing them to keep merging until the desirable number is reached and b) connecting smaller blocks together until all block sizes are almost equal. Both example customizations can be incorporated with no additional complexity.

A third important characteristic involves the ability of the Dividing Phase to replace rows inside a block once the maximum size has been reached. This alleviates problems created from blocks that attract rows simply because they are larger in size, a phenomenon commonly referred to as *chaining* [35]. This characteristic is especially useful for improving upon an initial partitioning solution that has blocks of maximum size through the replacement operation.

It is important to mention that the aforementioned characteristics combined, allow for online partitioning of buildings under dynamic changes of the transition matrix.

VI. ONLINE OPTIMIZATION USING BPH

Airflows in buildings are dynamic and depend on ambient wind speed and direction, temperature, closing and opening doors. Hence, the benefits of the initial subsystem configuration will gradually degrade as the changing airflows call for a different building partitioning. Moreover, due to the exponential complexity of the MILP solvers, it may not be possible to compute the subsystems in a timely fashion with respect to the rate of change of the airflows. Therefore, a combination of the MILP and BPH solutions is proposed to achieve real-time dynamic building re-partitioning.

The characteristics mentioned in Section V-C render the BPH capable of re-optimizing a given initial solution based on a new flow matrix (A') . At the beginning, an initial building partitioning is derived by optimally solving the MILP

formulation. Then, the heuristic algorithm is used in order to optimize on-line the solution based on the changes in the airflow values and the initial solution. In this way, the execution of the MILP formulation is restricted to cases where major changes of the airflows are detected.

The incorporation of initial solutions into the heuristic and the consideration of online optimization requires slight modifications to the two BPH phases. In the Dividing Phase, lines 2-3 need to be updated so that instead of initially creating singleton blocks, the initial solution blocks are adopted. In the Regrouping Phase, appropriate modifications need to be made to maintain the desired configuration parameters of the initial solution $(K, \underline{M}, \overline{M})$. In order to enforce both the new resulting number $|\mathcal{M}'|$ and size of the blocks the Modified Regrouping phase introduces a penalty and a variable minimum size constraint indicated by \underline{M}' . When the merging of two blocks produces a block of size greater than \overline{M} , then the penalty divides the merging benefit $C_{i,j}$ by the overhead block size, hindering the formation of blocks larger than the maximum. The maximum size constraint was introduced as soft rather than a hard constraint to avoid infeasibility problems. Finally, the variable minimum size constraint \underline{M}' allows the formation of exactly K blocks by increasing its value until the desirable number of resulting blocks has been reached.

Modified Regrouping phase

- 1: Set \underline{M}'
 - 2: **while** True **do**
 - 3: **for all** Blocks i with size $< \underline{M}'$ **do**
 - 4: **for all** Connected blocks j **do**
 - 5: $C_{ij} = \sum_{l \in \mathcal{M}_i} \sum_{k \in \mathcal{M}_j} (A_{l,k} + A_{k,l})$
 - 6: **if** $|\mathcal{M}_j| + |\mathcal{M}_i| > \overline{M}$ **then**
 - 7: $C_{ij} = \frac{C_{ij}}{\left(\left(|\mathcal{M}_j| + |\mathcal{M}_i|\right) - \overline{M}\right) + 1}$
 - 8: $j^* = \operatorname{argmax}_{j \in \mathcal{N}} C_{i,j}$
 - 9: Merge block i with block j^*
 - 10: **if** $|\mathcal{M}'| \leq K$ **then return**
 - 11: $\underline{M}' \leftarrow \underline{M}' + 1$
-

VII. SIMULATION RESULTS

This section illustrates the capabilities of the CDI scheme and examines the performance of the proposed partitioning algorithms. Section VII-A demonstrates the performance of the CDI scheme by utilizing the partitioning solution of the proposed partitioning algorithms. Section VII-B examines the effect of different parameters regarding the number and size of partitions. Section VII-C compares the performance between the MILP solution obtained through Gurobi optimization solver [36] and the BPH solution obtained through the developed algorithm in Section V. The comparison investigates both the PC and the execution time of the two algorithms for both real and artificial building models. Finally, Section VII-D demonstrates the ability of BPH to perform online partitioning under dynamically varying environmental and building conditions.

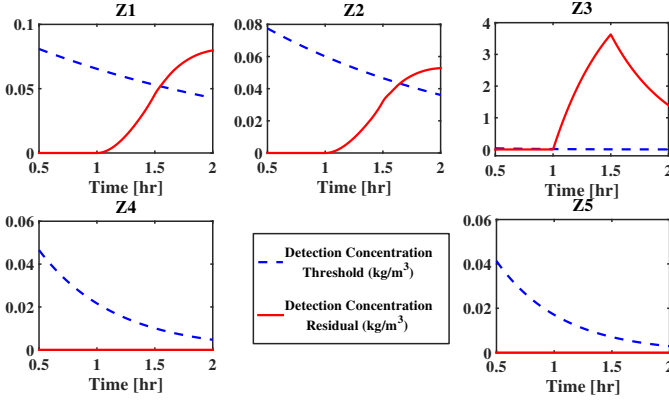


Fig. 8. Detection residual and threshold for subsystem Σ_2 of Holmes house illustrated in Fig. 2. The system has been partitioned into two subsystems. The contaminant source is located in zone Z3 with a release rate of 300 g/hr and release time of $t = 1$ hr.

In all simulation experiments conducted, the airflow values are generated randomly when dealing with artificial buildings. On the contrary, for the partitioning of real buildings for CDI purposes, realistic airflows are generated using the Matlab-CONTAM Toolbox [37] for specific environmental and building conditions.

A. Contaminant detection and isolation using CDI scheme

To demonstrate the effectiveness of the partitioning algorithms as well as present the capabilities of the aforementioned distributed CDI scheme, the Holmes house example of Section IV-A and its corresponding partitioning solution illustrated in Fig. 2 are used for a contaminant event scenario. For the purpose of this scenario, a contaminant source with release rate of 300 g/hr and duration of 0.5 hr is placed in zone Z3 and released at time $t = 1$ hr. For contaminant detection and isolation we used the distributed CDI scheme proposed in [10]. In this scheme, each subsystem is assigned a monitoring agent, which is essentially a software program that processes the contaminant sensor measurements and aims to detect the existence of a contaminant and isolate the zone where its source is located. The decision logic implemented in a contaminant event monitoring agent is based on estimating the state (contaminant concentration) of the subsystems through distributed observer schemes. Observer based residuals and adaptive thresholds are utilized for the contaminant detection and isolation decisions.

The detection decision for subsystem Σ_2 appears in Fig. 8. As can be seen from the figure, the contaminant is successfully detected (i.e., the detection concentration residual surpasses the corresponding threshold) in zone Z3 almost instantly (7 seconds). Note that the contaminant is also detected by zones Z1 and Z2 due to their high airflow connection with zone Z3. While this effect is desirable for zones of the same subsystem, a similar reaction between the zones of different subsystems may degrade the performance of the CDI approach. In this respect, the proposed partitioning algorithms hinder the appearance of such undesirable effects by ensuring the minimization of the airflow dependencies between subsystems.

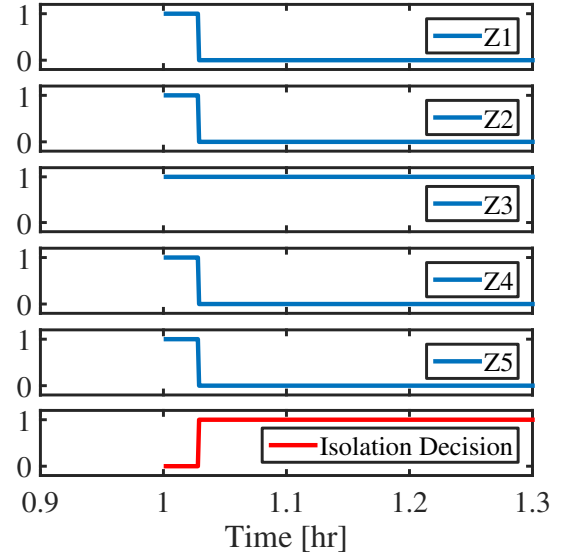


Fig. 9. Contaminant isolation results for subsystem Σ_2 . The bottom figure displays the isolation decision signal while the other figures display the isolation logic signals for individual zones.

Immediately after the contaminant detection, the isolation procedure is activated in order to locate its source zone. The resulting isolation logic for each zone of subsystem Σ_2 as well as the overall contaminant isolation decision are illustrated in Fig. 9. At the beginning of the isolation procedure all zones are considered as contaminated (logic signal equals to 1). An isolation decision is reached when *all zones except one* are eliminated (logic signal equals to 0) from the assumption of containing the contaminant source. According to Fig. 9, the contaminant is correctly isolated after 93 seconds from the start of the isolation procedure, where the isolation decision signal becomes equal to one (i.e., only the logic signal of Z3 remains equal to one).

B. Varying Parameter Effect in MILP

In order to investigate the effect of different parameters on the optimal PC (based on the MILP solution), simulation experiments were conducted for varying minimum, maximum and relative subsystem size \underline{M} , \overline{M} and R , respectively. All simulation experiments were conducted based on a three story building (DH15 [38]) consisting of 21 zones and 93 leakage paths between zones. The flows were generated by an air handling unit. Figs. 10 and 11 depict the effect of the minimum and maximum subsystem size for different values of K . From the figures a number of important observations can be deduced:

- Increasing the number of subsystems K leads to a worse PC, as more interconnection airflows are left out of the partitions.
- The selected values of size constraints can introduce feasibility issues. In the case of parameter \underline{M} , infeasibility is certain if $\underline{M} > \lfloor N/K \rfloor$, as all subsystems need to have size at least equal to \underline{M} . Otherwise, feasibility is ensured if there exists a combination of connected subsystems of appropriate size. For example, for $N = 21$ and $K = 4$,

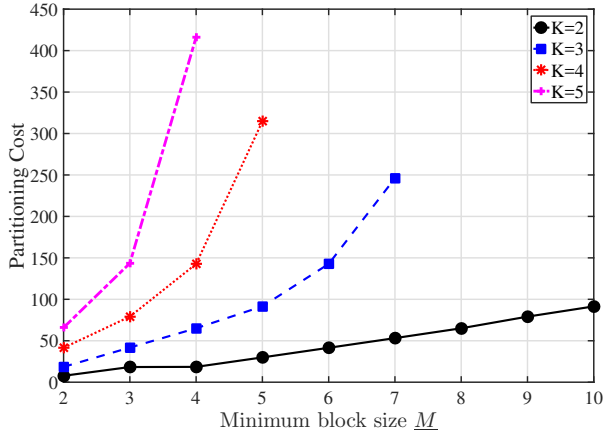


Fig. 10. Effects of the minimum size constraint \underline{M} and number of subsystems K to the PC of building DH15.

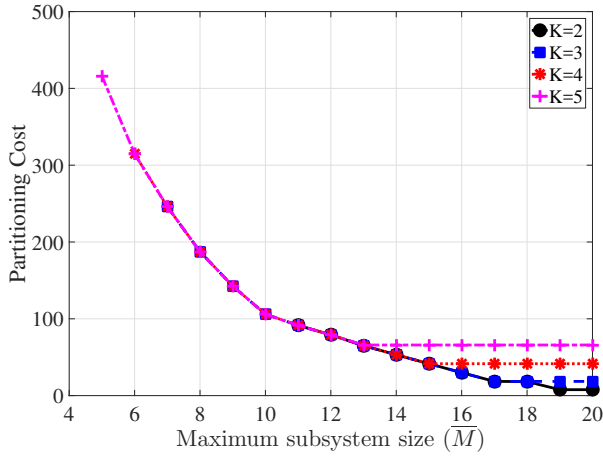


Fig. 11. Effects of the maximum size constraint \overline{M} and number of subsystems K to the PC of building DH15.

the problem is infeasible for $\underline{M} > \lfloor 21/4 \rfloor = 5$, as clearly depicted in Fig. 10. In the case of parameter \overline{M} , infeasibility is certain if $\overline{M} < \lceil N/K \rceil$, as all zones need to be allocated to at least one subsystem. For example, for $N = 21$ and $K = 4$, the problem is infeasible for $\overline{M} < \lceil 21/5 \rceil = 6$, as clearly depicted in Fig. 11.

- As infeasibility is approached (for increasing \underline{M} and decreasing \overline{M}) the PC is significantly increased, which indicates that the selection of size constraints should not be very restrictive.

Fig. 12 illustrates the effect of the relative size constraints on the PC for different K . Contrary to the minimum and maximum size, the relative size constraints are less prone to infeasibility; in the considered simulations no infeasible points were observed, even for $R = 1$. For this reason, we have adopted the relative size constraints to achieve balanced subsystems with respect to the number of zones. Another interesting observation is that $R = 5$ provides a good trade-off between subsystem size balance and performance since the PC is reduced by more than 50% compared to its maximum values.

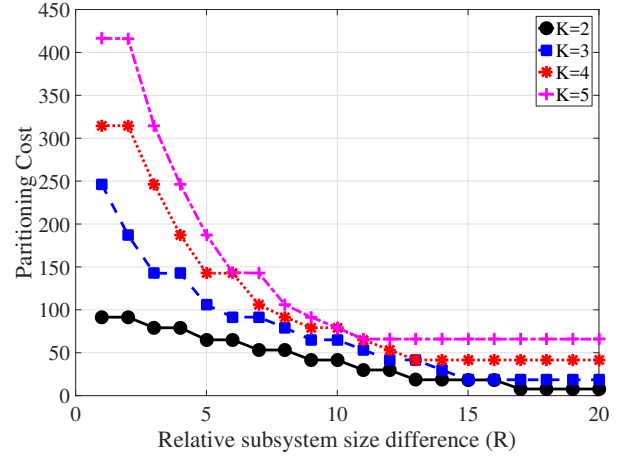


Fig. 12. Effects of the relative size constraint R and number of subsystems K on the PC of building DH15.

C. Performance comparison between MILP and BPH

The performance between MILP and BPH was compared in terms of the PC and the execution time for both artificial and real building configurations.

1) *Artificial buildings:* In the first set of experiments, randomly generated graphs were created to resemble buildings by imposing building characteristics such as low node connectivity and connectivity *only* with neighbouring nodes. More specifically, 200 undirected graphs were generated of 30 nodes and 50 edges each, with random integer values $w \in [200, 1000]$ assigned for the edges weights. The two developed algorithms (MILP and BPH) were used to partition the resulting artificial buildings for a ranging number of subsystems, $K \in \{2, \dots, 7\}$. For the comparison, relative subsystem size constraints were imposed with $R = \lceil N/K \rceil$. Hence, only the results adhering to the specific constraints were kept for the BPH, while the rest were discarded. It is worth pointing out that in only 4 out of the 200 random examples and for $K = 7$, the BPH provided an infeasible solution and those examples were removed from the comparison.

Fig. 13 shows the percent relative average deviation (PRAD) of the Partitioning Cost (1) of the BPH solution from the optimal one calculated using MILP formulation for each example. PRAD is defined as:

$$PRAD = \frac{1}{200} \sum_{m=1}^{200} \frac{PC_m^{BPH} - PC_m^{MILP}}{PC_m^{MILP}} \times 100\%,$$

where PC_m^{MILP} and PC_m^{BPH} denote the PC of the MILP and BPH approaches of the m -th example for a specific number of subsystems. It can be seen that, for $K = 2$, BPH provides always optimal results while for $K = 3$ it is very close to optimal. As expected, as the number of resulting subsystems grows, the BPH moves further away from optimality because the number of possible subsystem configurations increases. However, even for $K = 7$ in a building consisting of 30 rooms the BPHs' deviation from optimality is within 15.5%.

Fig. 14 illustrates the mean execution time of the BPH algorithm and the MILP solver for varying number of subsystems.

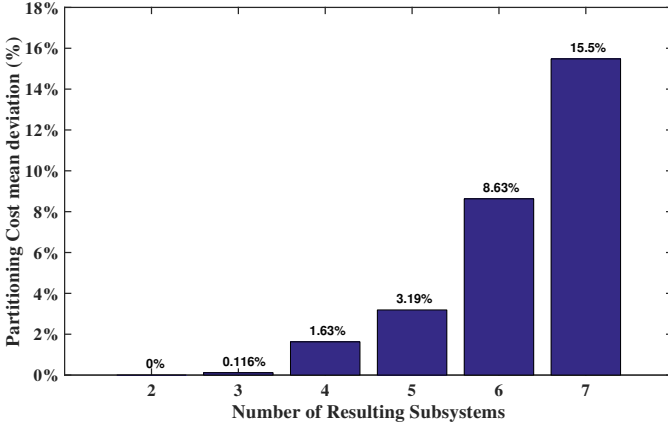


Fig. 13. Mean deviation percentage between the BPH and the MILP PC for 200 undirected random graphs.

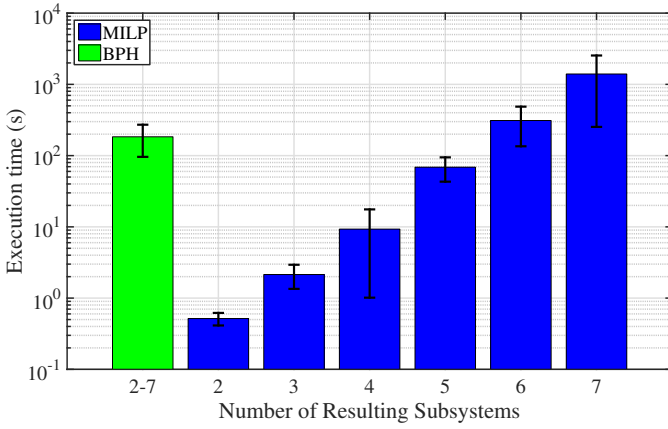


Fig. 14. Comparison of mean execution time between the MILP and the BPH for 200 undirected random graphs when $K \in \{2, \dots, 7\}$. Since BPH provides a range of results for different resulting numbers of subsystems after one run, its execution time is presented once for all K . The error bars correspond to the one standard deviation.

Note that the two approaches cannot be directly compared for individual K , since the BPH produces multiple solutions with varying K in one run. Therefore, the mean execution time for each K was calculated only for the MILP and compared with the resulting mean execution time of BPH which covers all K . In the figure, the error bars denote the one standard deviation from the mean for all cases considered. From the figure, it becomes evident that for $K \in \{2, 3, 4, 5\}$ the execution time of MILP is lower than the BPH. For $K = 6$ the execution time of the BPH (200s) is 50% better than that of the MILP (300s), while for $K \in \{7\}$ the BPH execution time is almost an order of magnitude better than the MILP.

2) *Real buildings*: The model of a health-care facility [39] is simulated, which consists of 126 zones (mainly examination, waiting and storage rooms) and 420 leakage paths consisting of doors and windows. Natural ventilation is assumed with the air flowing from 240° relative to the north with a speed of 10 m/s. No air handling units are considered in this example. Fig. 15 presents an optimal partitioning solution with three balanced subsystems. The solution is derived using the MILP formulation for $K = 3$ and $R = 5$ in 11.79 seconds. The

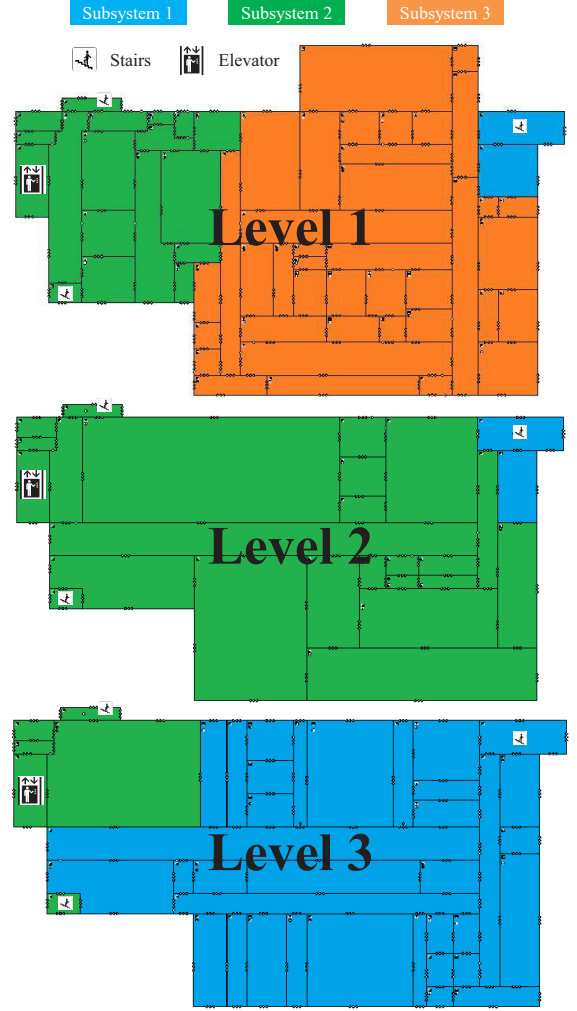


Fig. 15. Example partitioning of a health-care facility [39] for 3 resulting subsystems with balanced number of zones.

resulting optimal partitioning configuration is comprised of two subsystems that span all three floor levels (1 and 2) while subsystem 3 consists only of 3rd level zones.

This case study demonstrates for one thing the ability of the proposed partitioning algorithms to handle large-scale complex buildings. Additionally, it shows that partitioning solutions for these cases can be far from trivial (e.g. assigning one partition for each floor level). In fact, as demonstrated in this case, the resulting partition can usually span several floors, as it is influenced by the airflows connecting different floors through open spaces such as staircases and also by the existence of one or more air handling units, which not only arbitrarily connect zones from the same level but can also connect zones from different levels.

D. On-line partitioning

The unpredictability of the MILP execution time renders it incapable of ensuring that results will be produced in a timely manner. However, its optimality and versatility are important in providing a solution under multiple constraints and it is mostly suited for applications having no execution

Wind Direction	PC MILP	PC BPH	Time MILP (s)	Time BPH (s)	PRAD
54°	57.32	—	51.64	—	—
97°	58.36	58.61	67.83	1.165	0.43 %
140°	53.38	54.86	15036.67	1.372	2.70 %
183°	48.55	49.12	2501.2	1.572	1.16 %
226°	56.36	57.79	17769.21	1.707	2.47 %
269°	60.28	61.55	53.16	1.473	2.07 %
312°	48.74	48.74	49.38	0.337	0 %
355°	44.62	44.62	57.89	0.351	0 %

TABLE I
RE-OPTIMIZATION RESULTS FOR A HOSPITAL BUILDING CONSISTING OF 97 ZONES.

time requirements. On the other hand, the BPH has more predictable execution time but has sub-optimal performance. Therefore, the combination of both can ensure predictable execution time as well as maintain the enforced constraints of the MILP. Under the on-line partitioning perspective, a hospital building [39] with 97 rooms comprised of 6 floors and 530 airflow leakages is analyzed. The building model includes 4 different air handling units systems that span all the building. The problem of separating the building into subsystems, is initially solved using the MILP for $K = 4$ and $R = 5$. It is assumed that natural ventilation is the dominant cause of air movement in the building with an ambient wind speed of 10 m/s. In the initial solution a wind direction of 54° from the north is used. Then, the derived subsystem composition is inserted as the initial state M' of the BPH on-line partitioning formulation and solved separately for wind direction increments of 43° with maximum subsystem size of $\bar{M} = 27$. The initial minimum subsystem size is set to $\underline{M}' = 5$. The results for each wind direction are compared with the MILP results in the terms of partitioning cost and execution time (Table I).

The results indicate that, partitioning on-line with BPH takes far less time than solving the problem from the beginning. In particular, the BPH manages to provide a viable result following the constraints of the MILP for a building with 97 zones in under 2 seconds. In [40], the authors demonstrated in a test chamber using real sensors that a sensor sampling time of less than 10 seconds has negligible or no improvement on the contaminant identification process. Since the BPH is able to re-optimize a partitioning solution in under 2 seconds, it can be considered suitable for real world applications with changing airflows. It is important to emphasize that the fast execution time does not compromise the quality of the solution since BPH deviates at most 2.7% from the optimal MILP solution. Note that in 2 out of 7 runs the solutions are actually identical.

VIII. CONCLUSIONS

Distributed CDI approaches require a procedure for defining the subsystems while minimizing their interdependencies,

controlling their size and preserving their initial structural connectivity. In this paper we have developed both an exact MILP formulation and a matrix-based heuristic algorithm (BPH) for building partitioning with the above considerations. Both approaches can effectively partition buildings following size and connectivity constraints in such a way as to ensure the successful deployment of distributed CDI approaches. Simulation results demonstrate that the BPH yields performance close to optimal and simultaneously provides solutions to partitioning problems for different numbers of subsystems. Most importantly, it is able to optimize the problem in real-time based on previous solutions in order to compensate for the dynamic nature of building and environmental conditions.

Future work includes, incorporating observability constraints to the algorithms to address the CDI problem with reduced number of sensors, assigning a risk factor in each zone and incorporating this new information in the building partitioning problem so as to further increase the performance of the CDI in high risk zones. Another future work direction is to consider other computational intelligence techniques, such as genetic algorithms, for the solution of this problem that intelligently incorporate the size and connectivity constraints in a way that improves solution quality and avoids excessive infeasibility over successive iterations. Finally, another exciting direction regards the concurrent control of the building (e.g., opening/closing doors and windows) and the air handling units in combination with the proper partitioning solution in order to minimize contamination effects.

REFERENCES

- [1] A. I. Dounis and C. Caraiscos, "Advanced control systems engineering for energy and comfort management in a building environment - A review," *Renewable and Sustainable Energy Reviews*, vol. 13, no. 6, pp. 1246–1261, 2009.
- [2] X. Liu and Z. Zhai, "Inverse modeling methods for indoor airborne pollutant tracking: Literature review and fundamentals," *Indoor Air*, vol. 17, no. 6, pp. 419–438, 2007.
- [3] X. Liu and Z. J. Zhai, "Prompt tracking of indoor airborne contaminant source location with probability-based inverse multi-zone modeling," *Building and Environment*, vol. 44, no. 6, pp. 1135–1143, 2009.
- [4] M. D. Sohn, R. G. Sextro, A. J. Gadgil, and J. M. Daisey, "Responding to sudden pollutant releases in office buildings: 1. Framework and analysis tools," *Indoor Air*, vol. 13, no. 3, pp. 267–276, 2003.

- [5] M. P. Michaelides, V. Reppa, M. Christodoulou, C. G. Panayiotou, and M. M. Polycarpou, "Contaminant event monitoring in multi-zone buildings using the state-space method," *Building and Environment*, vol. 71, pp. 140–152, 2014.
- [6] J. E. Braun, "Intelligent Building Systems - Past, Present, and Future," in *Proceedings of American Control Conference*, 2007, pp. 4374–4381.
- [7] X. Zhang and Q. Zhang, "Distributed fault diagnosis in a class of interconnected nonlinear uncertain systems," *International Journal of Control*, vol. 85, no. 11, pp. 1644–1662, 2012.
- [8] R. M. Ferrari, T. Parisini, and M. M. Polycarpou, "Distributed fault detection and isolation of large-scale discrete-time nonlinear systems: An adaptive approximation approach," *IEEE Transactions on Automatic Control*, vol. 57, no. 2, pp. 275–290, 2012.
- [9] F. Boem, R. M. Ferrari, T. Parisini, and M. M. Polycarpou, "Distributed fault diagnosis for continuous-time nonlinear systems: The input-output case," *Annual Reviews in Control*, vol. 37, no. 1, pp. 163–169, 2013.
- [10] V. Reppa, M. Michaelides, M. Christodoulou, C. G. Panayiotou, and M. Polycarpou, "A distributed framework for contaminant event detection and isolation in multi-zone intelligent buildings," in *Proceedings of IEEE European Control Conference (ECC)*, 2014, pp. 1637–1642.
- [11] T. Van Laarhoven and E. Marchiori, "Local quality functions for graph clustering with non-negative matrix factorization," *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 90, no. 6, pp. 446–453, 2014.
- [12] A. Kusiak, "The generalized group technology concept," *International Journal of Production Research*, vol. 25, no. 4, pp. 561–569, 1987.
- [13] J.L. Burbidge, "Production flow analysis," *Production Engineer*, vol. 50, no. 8, p. 352, 1971.
- [14] M. P. Chandrasekharan and R. Rajagopalan, "MODROC: an extension of rank order clustering for group technology," *International Journal of Production Research*, vol. 24, no. 5, pp. 1221–1233, 1986.
- [15] —, "ZODIAC an algorithm for concurrent formation of part-families and machine-cells," *International Journal of Production Research*, vol. 25, no. 6, pp. 835–850, 1987.
- [16] A. Kusiak, "The part families problem in flexible manufacturing systems," *Annals of Operations Research*, vol. 3, no. 6, pp. 277–300, 1985.
- [17] A. Zakarian, "A new nonbinary matrix clustering algorithm for development of system architectures," *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, vol. 38, no. 1, pp. 135–141, 2008.
- [18] P. K. Chan, M. D. F. Schlag, and J. Y. Zien, "Spectral K-way ratio-cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1088–1096, 1994.
- [19] G. Karypis and V. Kumar, "Multilevel k-way Partitioning Scheme for Irregular Graphs," *Journal of Parallel and Distributed Computing*, vol. 48, no. 1, pp. 96–129, 1998.
- [20] C. Aykanat, B. B. Cambazoglu, and B. Uçar, "Multi-level direct K-way hypergraph partitioning with multiple constraints and fixed vertices," *Journal of Parallel and Distributed Computing*, vol. 68, no. 5, pp. 609–625, 2008.
- [21] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, 2006, pp. 475–486.
- [22] M. Boule, "Compact mathematical formulation for graph partitioning," *Optimization and Engineering*, vol. 5, no. 3, pp. 315–333, 2004.
- [23] W. W. Hager, D. T. Phan, and H. Zhang, "An exact algorithm for graph partitioning," *Mathematical Programming*, vol. 137, no. 1-2, pp. 531–556, 2013.
- [24] L. Liberti, "Compact linearization for bilinear mixed-integer problems," Technical Report 1124, Opt. Online, Tech. Rep., 2005.
- [25] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoretical Computer Science*, vol. 1, no. 3, pp. 237–267, 1976.
- [26] P. C. Jr and F. Lamour, "On the validity of a front-oriented approach to partitioning large sparse graphs with a connectivity constraint," *Numerical Algorithms*, vol. 12, no. 1, pp. 193–214, 1996.
- [27] B. Dilkina and C. P. Gomes, "Solving connected subgraph problems in wildlife conservation," in *Proceedings of International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*. Springer, 2010, pp. 102–116.
- [28] T. Shirabe, "A Model of Contiguity for Spatial Unit Allocation," *Geographical Analysis*, vol. 37, no. 1, pp. 2–16, 2005.
- [29] M. Michaelides, V. Reppa, C. G. Panayiotou, and M. Polycarpou, "Contaminant event monitoring in intelligent buildings using a multi-zone formulation," in *Proceedings of 8th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes (SAFEPROCESS)*, vol. 8, 2012, pp. 492–497.
- [30] C. Alippi, D. Liu, D. Zhao, and L. Bu, "Detecting and reacting to changes in sensing units: the active classifier case," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 3, pp. 353–362, 2014.
- [31] L. Bu, C. Alippi, and D. Zhao, "A pdf-free change detection test based on density difference estimation," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.
- [32] L. L. Wang, W. S. Dols, and Q. Chen, "Using CFD Capabilities of CONTAM 3.0 for Simulating Airflow and Contaminant Transport in and around Buildings," *HVAC&R Research*, vol. 16, no. 6, pp. 749–763, 2010.
- [33] G. N. Walton and W. S. Dols, "CONTAM 3.0 user guide and program documentation," *National Institute of Standards and Technology Technical Report NISTIR*, vol. 7251, 2010.
- [34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press Cambridge, 2001, vol. 6.
- [35] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," in *The elements of statistical learning*. Springer, 2009, pp. 485–585.
- [36] G. O. Inc., "Gurobi Optimizer reference manual," p. 572, 2014. [Online]. Available: <http://www.gurobi.com>
- [37] M. P. Michaelides, D. G. Eliades, M. Christodoulou, M. Kyriakou, C. G. Panayiotou, and M. Polycarpou, "A matlab-contam toolbox for contaminant event monitoring in intelligent buildings," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2013, pp. 605–614.
- [38] A. K. Persily, A. Musser, and D. D. Leber, *A collection of homes to represent the US housing stock*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2006.
- [39] L. C. Ng, A. Musser, A. K. Persily, and S. J. Emmerich, "Airflow and indoor air quality models of DOE reference commercial buildings," *Gaithersburg, MD, National Institute of Standards and Technology*, vol. 163, 2012.
- [40] X. Shao, X. Li, and H. Ma, "Identification of constant contaminant sources in a test chamber with real sensors," *Indoor and Built Environment*, vol. 25, no. 6, pp. 997–1010, 2016.



Alexis Kyriacou (S13) received his B.Sc. degree in 2013 and he is currently pursuing the Ph.D. degree, all from the Department of Electrical and Computer Engineering at University of Cyprus. Since 2014, he has been employed as a Researcher at KIOS Research Center for Intelligent Systems and Networks. His research interests include fault detection and diagnosis, optimization techniques and contaminant diagnosis and accommodation in Intelligent Buildings.

Alexis is the Secretary of IEEE University of Cyprus Student Branch from Feb. 2015 and the Chair of IEEE Cyprus Computational Intelligence Chapter from Nov. 2016.



Stelios Timotheou (S'04-M'10) received a B.Sc. from the Electrical and Computer Engineering (ECE) School of the National Technical University of Athens, and an M.Sc. and Ph.D. from the Electrical and Electronic Engineering Department of Imperial College London. He is currently a Research Associate at the KIOS Research Center for Intelligent Systems and Networks of the University of Cyprus (UCY). In previous appointments, he was a Visiting Lecturer at the ECE Department of UCY, a Research Associate at the Computer Laboratory

of the University of Cambridge and a Visiting Scholar at the Intelligent Transportation Systems Center & Testbed, University of Toronto. His research focuses on the modeling and system-wide solution of problems in complex and uncertain environments that require real-time and close to optimal decisions by developing optimisation, machine learning and computational intelligence techniques. Application areas of his work include intelligent transportation systems, communication systems, and smart buildings.



Michalis P. Michaelides (S'04-M'9) is a tenure-track Lecturer with the Department of Electrical Engineering, Computer Engineering and Informatics at the Cyprus University of Technology in Limassol, Cyprus (since January 2012). He has received a BSEE (1997) and a MSEE (1998), both with honors, from the School of Electrical and Computer Engineering at Purdue University, West Lafayette, IN, USA. During 1999-2007 he has worked in the Cyprus industry as a Medical Service Engineer and as an Information Technology Officer. In 2009, he

received his PhD from the Department of Electrical and Computer Engineering at the University of Cyprus and during the next two years he was employed as a research fellow at the KIOS Research Center for Intelligent Systems and Networks at the University of Cyprus. Michalis research interests include wireless sensor networks, event detection and localization, fault detection and diagnosis, fault tolerance, collaborative signal and information processing, environmental monitoring, intelligent irrigation systems and intelligent buildings. In 2014, he received the Elsevier Building and Environment Journal Best Paper Award.



Christos Panayiotou (SM06) is an Associate Professor with the Electrical and Computer Engineering (ECE) Department at the University of Cyprus (UCY). He is also the Deputy Director of the KIOS Research Center for Intelligent Systems and Networks for which he is also a founding member. Christos has received a B.Sc. and a Ph.D. degree in Electrical and Computer Engineering from the University of Massachusetts at Amherst, in 1994 and 1999 respectively. He also received an MBA from the Isenberg School of Management, at the

aforementioned university in 1999. Before joining UCY in 2002, he was a Research Associate at the Center for Information and System Engineering (CISE) and the Manufacturing Engineering Department at Boston University (1999 - 2002). His research interests include distributed and intelligent control systems, wireless, ad hoc and sensor networks, computer communication networks, fault diagnosis, optimization and control of discrete-event systems, resource allocation, transportation networks and intelligent buildings.

Prof. Panayiotou has published more than 190 papers in international refereed journals and conferences and is the recipient of the 2014 Best Paper Award for the journal Building and Environment (Elsevier). He is an Associate Editor for the Conference Editorial Board of the IEEE Control Systems Society, the IEEE Transactions on Control Systems Technology, the Journal of Discrete Event Dynamical Systems and the European Journal of Control. He held several positions in organizing committees of numerous international conferences.



Marios Polycarpou (F06) is a Professor of Electrical and Computer Engineering and the Director of the KIOS Research Center for Intelligent Systems and Networks at the University of Cyprus. He received the B.A degree in Computer Science and the B.Sc. in Electrical Engineering, both from Rice University, USA in 1987, and the M.S. and Ph.D. degrees in Electrical Engineering from the University of Southern California, in 1989 and 1992 respectively. His teaching and research interests are in intelligent systems and networks, adaptive and

cooperative control systems, computational intelligence, fault diagnosis and distributed agents. Dr. Polycarpou has published more than 300 articles in refereed journals, edited books and refereed conference proceedings, and co-authored 7 books. He is also the holder of 6 patents.

Prof. Polycarpou is a Fellow of IEEE and IFAC. He is the recipient of the 2016 IEEE Neural Networks Pioneer Award and the 2014 Best Paper Award for the journal Building and Environment (Elsevier). He has served as the President of the IEEE Computational Intelligence Society (2012-2013), and as the Editor-in-Chief of the IEEE Transactions on Neural Networks and Learning Systems (2004-2010). He is currently the Vice President of the European Control Association (EUCA). Prof. Polycarpou has participated in more than 60 research projects/grants, funded by several agencies and industry in Europe and the United States, including the prestigious European Research Council (ERC) Advanced Grant.