A Minimalist Algorithm for Multirobot Continuous Coverage

Giorgio Cannata, *Member, IEEE*, and Antonio Sgorbissa

***Abstract*—This paper describes an algorithm, which has been specifically designed to solve the problem of multirobot-controlled frequency coverage (MRCFC), in which a team of robots are requested to repeatedly visit a set of predefined locations of the environment according to a specified frequency distribution. The algorithm has low requirements in terms of computational power, does not require inter-robot communication, and can even be implemented on memoryless robots. Moreover, it has proven to be statistically complete as well as easily implementable on real, marketable robot swarms for real-world applications.**

***Index Terms*—Ant-like algorithms, coverage, distributed robot systems, real-time search.**

I. Introduction

THE paper introduces the problem of multirobot-controlled frequency coverage (MRCFC), in which a team of robots are requested to repeatedly visit a set of predefined locations of the environment, by distributing their visits among locations according to a specified *frequency distribution*. The problem has a fundamental importance in many applications, e.g., surveillance and patrolling, continuous cleaning of crowded areas (e.g., malls, convention centers, restaurants, etc.), or serving food or beverages (in hospitals or in a banquet). However, differently from other problems related to multirobot coverage and exploration, it has received only a limited attention. Furthermore, the few examples in the literature (e.g., see [1]) deal only with the particular case in which all locations are visited with the same frequency, which is referred here as multirobot uniform-frequency coverage (MRUFC). The concept of *frequency distribution* of visits (or simply *visit distribution*), as used throughout this paper, describes the *ratio of visits* that are received by every location, *with respect to the total number of visits*, in a given time interval. This concept should not be confused with the *actual number of visits* received by every location *per time unit*, which is, in the following, referred to as a *visiting rate*. These two quantities are obviously correlated, for example, the case in which the visiting rate is the same for all vertices corresponds to a uniform frequency distribution.

This paper describes *PatrolGRAPH**, which is an algorithm that solves MRCFC by exhibiting additional desirable properties, i.e., it is suited for robots with low computational power and memory storage and for situations when wireless communication is not available or is severely degraded, thus being unreliable for multirobot coordination. These properties follow from the fact that *PatrolGRAPH** belongs to the class of the so-called *real-time search* and *ant-like* algorithms [2]–[5], which is able to deal with situations in which a global representation of the environment, as well as global communication capabilities, are absent. Algorithms of this class usually assume that robots navigate in a graph-like world, which is only locally known, i.e., every robot has only access to information related to the closer vertex (and, in some cases, to adjacent vertices). Starting from these assumptions, different algorithms implement different strategies to find a path to the goal state. *Real-time search* and *ant-like* algorithms are particularly interesting in that they move most of the burden of computing and memorizing from the searching agent to the vertices of the graph: the graph itself is not only a model of the topology of the environment, but it also becomes a real physical entity which—from time to time—can be built by leaving chemical trails on the floor [5], by dropping pebbles [3], etc. Toward this end, some works adopt radio-frequency-identification (RFID) tags as a reference technology (e.g., see [6]–[8]): RFID tags are low-cost, short-range, and energetically self-sustained transponders that can be distributed in the environment and can store a limited amount of information.

From a theoretical perspective, and similarly to other *real-time-search* algorithms whose behavior can be described through a navigation graph and an associated transition–probability matrix, *PatrolGRAPH** relies on Markov chains theory [9]. As shown in the following, in the context of this paper, a fundamental role is played by the so-called *inverse problem* [10], i.e., the problem of choosing transition probabilities between states to guarantee that Markov chains tend to a prescribed stationary distribution.

From a technological perspective, the practical implementation of *PatrolGRAPH** requires the presence of *smart nodes* (e.g., based on RFID technology), possibly with limited memory storage and communication range, placed in the environment prior to robot operations, and used to build a navigation graph. Every *smart node* contains navigation directions to neighboring *smart nodes*, as well as information to decide which *smart nodes* should be visited next.

The major contribution of the paper is to introduce the MRCFC problem and to propose an algorithm to solve it. This problem has been considered in the literature only under particular conditions, i.e., uniform frequency distribution [1].

This paper is organized as follows. Section II discusses related work, and Section III describes the MRCFC problem in

Manuscript received April 7, 2010; revised September 6, 2010; accepted December 30, 2010. Date of publication February 10, 2011; date of current version April 7, 2011. This paper was recommended for publication by Associate Editor K. Kyriakopoulos and Editor L. Parker upon evaluation of the reviewers' comments

The authors are with the Department of Communication, Computer and System Sciences, University of Genova, 16145 Genova, Italy (e-mail: giorgio.cannata@unige.it; antonio.sgorbissa@unige.it).

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/TRO.2011.2104510

detail. Section IV describes the properties of known algorithms in the literature, which constitute the basis to describe *Patrol-GRAPH** in Section V, and analyzes its analytical properties in Section VI. Section VII shows the experimental results performed in simulation. Conclusions are given in Section VIII.

II. Related Work

Most approaches to coverage, either single or *multirobot*, are based on *space decomposition*. For example, Ntafos [11] assumed a single robot equipped with a squared sweeping tool and imposed a tool-based *simple-grid* (i.e., without internal holes) approximation over the work area where cells have the same dimensions as that of the tool. Then, the grid is subdivided into rectangular subgrids that can be covered optimally, and a traveling-salesman problem (TSP) is solved to compute a path that visits all subgrids (in the case of many robots, the problem can be formalized as a multiple TSP—MTSP [12]). These ideas are extended in [13] by considering *nonsimple grids* that do not possess *local-cut nodes* (i.e., nodes whose removal locally disconnects the graph induced by the grid). In [14], a complete coverage algorithm is described, which is based on a new type of exact cellular-decomposition method, which is termed as the *boustrophedon cellular decomposition* (i.e., based on back-and-forth ox-like motions). In this approach, cells are computed in such a way to guarantee that a robot can easily plan a boustrophedon motion in each cell. The approach has been extended to multirobot coverage [15], toward the end of minimizing the number of regions that are covered multiple times under different communication models (line-of-sight communication versus global communication). Similar approaches have been proposed, among others, in [16]–[19]. Complete coverage of unknown rectilinear environments using a team of square robots with contact sensing is achieved in [20] by performing an online decomposition, where each cell can be covered completely by back-and-forth motions that are performed parallel to one of the walls of the environment. In [21], a planning algorithm is proposed for multiple mobile robots to sweep an area by relocating portable obstacles; a total path is initially created that would allow a single robot to cover the whole area, and a path for each robot is generated next by decomposing and distributing the total path.

Differently from the previous approaches, which mostly rely on the idea that the work area is decomposed into subregions and that each robot is assigned a subregion (or set of subregions) to cover, *spanning-tree coverage* (*STC*) envisions a situation in which all robots periodically cover the whole environment with the purpose of guaranteeing that all areas are visited with *uniform frequency*. *STC* for a single robot was first proposed in [22] and [23], again by assuming a robot equipped with an ideal sweeping tool, and by imposing a tool-based grid approximation. The approach relies on the idea of building the minimum spanning tree over the grid and moving along the Hamiltonian cycle that follows the tree around. The authors show that each cell in the grid is visited repeatedly at the same frequency. These ideas have been extended to *multirobot STC* (*MSTC*) in [1]; after computing the Hamiltonian cycle, robots are positioned uniformly along the path and are instructed to walk along the cycle in equidistant relative positions. According to Zheng *et al.* [24], computing *MSTC* trajectories in order to minimize the coverage time is a nondeterministic-polynomial-time (NP) complete problem; to deal with this limitation, this paper presents *multirobot forest coverage* (*MFC*), which is a polynomial-time multirobot-coverage heuristic technique providing an improved solution. In [25], an online variant of *MSTC* is proposed, which assumes that the robots do not have *a priori* knowledge of the work area and proves to be complete and robust in presence of noisy sensor data. In [26], a different method is described to generate Hamiltonian exploration paths for multiple mobile robots in a restricted working environment. Approaches not based on space decomposition are also proposed in [27], which introduces the concept of "spurious obstacles" (i.e., already covered areas) to limit the amount of repeated coverage for a team of robots in unknown environments, and in [28], where a neural network is used for planning the path of multiple cleaning robots in an arbitrarily varying environment. A solution to fully decentralized patrolling in the framework of behavioral control has been recently proposed in [29].

Finally, in the surveillance scenario, many works deal with multirobot coverage in a game theoretic framework, with the purpose of finding the minimum number of mobile pursuers and an optimal search strategy, which guarantee to find an evader that tries to avoid capture within a bounded region. Approaches of this class have been initially proposed for searching graphs [30], and they have been extended to polygonal [31], [32], curved [33], and unknown planar environments in absence of localization capabilities [34].

Remark 1: None of the previous approaches deal with the MR-CFC problem; a few of them deal explicitly with the less-general MRUFC problem (e.g., see [1]). Furthermore, all the previous approaches have two drawbacks. First, the trajectory followed by the robot(s) is easily predictable; this is an acceptable property for application, like cleaning, but is unacceptable in other contexts, like surveillance.[1] This issue is explicitly addressed in [35]–[38]. Second, these approaches, with few exceptions, require the robot(s) to have a complete map of the environment *a priori* or to build it in run time, with obvious consequences on the computational power required onboard for sensor fusion and planning. □

Algorithms that do not exhibit repeated motion patterns and do not rely on a complete representation of the environment exist: they belong to the domain of the so-called *real-time search* or *ant-like* methods, which have been designed to provide a general solution to complex search problems but have been exploited for robot coverage and exploration as well. The simpler of these methods is perhaps *random walk*. Let us assume that the environment is modeled as an oriented graph[2]; when the robot is in a vertex, it selects a departing edge randomly with uniform probability, which guarantees complete coverage in a

[1]Contracts with security companies explicitly state that patrol rounds must be randomly varied, since repeatability allows intruders to infiltrate more easily.

[2]A grid can be mapped onto an oriented graph simply by doubling all edges that connect neighboring cells and by assigning them opposite directions.

statistical sense as the exploration time tends to infinite. *Edge counting* [39] is a deterministic variant of this idea in which a robot, in a given vertex, counts the number of times that each departing edge has been selected, and chooses different edges in circular order in subsequent visits. *Node count* [2] exhibits an improved behavior by relying on the idea of associating a value with each vertex of the graph, which counts how often each vertex has been already visited so far. When a robot enters a vertex, it increases the value of the vertex by one; next, it moves to the adjacent vertex that has received less visits up to present time. Variants of this simple idea exist (e.g., see [2], [40], and [41]); in particular, *learning real-time A** [2] requires a look-ahead of one edge traversal but guarantees better performance [4].

Real-time-search or *ant-like* methods, which are especially crafted for robotic applications, have been proposed. In [3], [42], and [43], a graph-exploration strategy is described, which relies on the physical deployment of *pebbles* that the robot can drop or collect to recognize already-visited vertices. In [5] and [44], a family of ant-like approaches is proposed in which robots are able to cover the floor of an unmapped building by physically leaving pheromone-like chemical odor traces that evaporate with time. In [45], a variant of *node count* is proposed, which is especially designed for multirobot patrolling and is based on the concept of idleness, i.e., the amount of time elapsed since a vertex has received the visit of an agent.

Remark 2: Existing *real-time-search* and *ant-like* approaches can be easily extended to *multirobot* systems. However, they have at least two drawbacks. First, they neither deal explicitly with the MRCFC problem, nor with the less-general MRUFC problem. Second, an implementation of a real sensor that reliably evaluates the strength of smell of an evaporating chemical trace turns out to be technically very challenging, and the same can be said for a servomechanism that deploys and collects pebbles. Therefore, these processes are usually emulated through a global memory that is shared among all robots, which is in obvious contradiction with the principles for which the approaches have been designed in the first place (a counterexample is described in [46]). □

Assume that a *real-time-search* algorithm is adopted for which place-to-place navigation can be represented as a transition on a navigation graph, and described through a transition-probability matrix (as it happens, e.g., with *random walk*). MRCFC can be ideally dealt with by solving the so-called inverse problem, i.e., the problem of choosing transition probabilities between vertices to guarantee that the corresponding Markov chain [9] tends to the prescribed frequency distribution.[3] The inverse problem has been formally defined for the first time in [10], which also demonstrates that the set of transition probability matrices that solve the inverse problem is a compact and convex set. In subsequent years, a significant research effort has been spent to determine the extreme points of this set, by mostly focusing on matrices with a uniform stationary distribution (i.e., doubly stochastic matrices, e.g., see [47] and [48]), and by considering matrices with a generic stationary distribution only in a few cases (e.g., see [49]). A very simple solution to the inverse problem can be found by applying the so-called metropolis rule [9], which was first presented in the seminal paper [50]. The metropolis rule basically states that, if an initial navigation graph and a transition matrix are given, it is possible to obtain a new transition matrix with the desired steady-state distribution by adding self-loops to the original graph (i.e., edges departing from/arriving to the same vertex), and by modifying transitions probabilities to adjacent vertices accordingly. However, this is not adequate for many applications, such as cleaning; it does not really make sense to clean the same location twice in a row, and therefore, self-loops would cause a significant wastage of time. In the following, the inverse problem is considered under the additional constraints that the topology of the graph is given *a priori* and unmodifiable, i.e., when some transition probabilities are constrained to zero, and others are constrained to a positive value. Finally, some authors refer to the inverse problem as the problem of estimating transition probabilities, which are more likely to produce a set of observed aggregated data, i.e., the proportion of individuals observed in each state at each moment of time [51], [52]. However, the problem is only partially related to the one addressed here.

The recent RFID technology provides a solution to implement *real-time-search* and *ant-like* approaches. RFID tags are low-cost, short-range, energetically self-sustained transponders that can be distributed in the environment and can store a limited amount of information. In [53], it is proposed to exploit RFID tags dispersed in an environment as a sort of distributed memory in which to store *digital pheromones*: RFID reader devices carried by humans or by robots could deploy pheromone trails in the environment simply by writing *pheromone values* on existing RFID tags, and they could sense such trails by simply reading values on nearby RFID tags. Ziparo *et al.* [6] and Kleiner *et al.* [7] proposed to deploy RFID tags while exploring the environment in rescue applications toward the end of building a graph-like structure that will subsequently allow robots to plan paths. A similar idea has been proposed in [8]. The use of RFID tags and other deployable sensor networks in robotics have been recently investigated also to solve general robotic problems (i.e., navigation and localization [54]–[56]), as well as for specific applications (e.g., cleaning [57] and exploration [58]). The feasibility of RFID-based navigation has been also tested on a team of ROOMBA robots [59], [60], which are developed within the ROBOSWARM project [61]. Then, RFID tags have been proven to provide a significant aid in different kinds of robotic tasks, both when they are *a priori* distributed during a setup phase preceding task execution, and when they are directly deployed by the robots in the environment.

III. Problem Statement: Multirobot-Controlled Frequency Coverage

The MRCFC problem consists finding a decision procedure that allows a team of robots to navigate in a workspace modeled as a navigation graph, thereby ensuring that vertices of the graph are visited according to a prescribed frequency distribution. We have the following assumptions.

[3]To the best authors' knowledge, this possibility has never been explored in multirobot coverage.

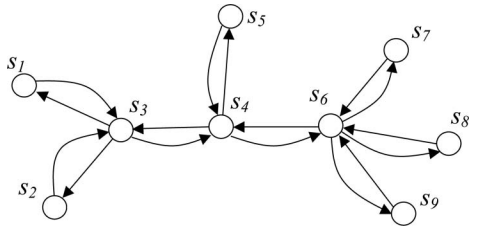

Fig. 1. Oriented graph $\hat{G}_N$ built from G_N.

1) G_N is a connected, nonoriented graph of arbitrary order, possibly containing cycles, which represents the topology of the free space, which is referred to as the *navigation graph*. The navigation graph is better represented through a strongly connected, oriented graph $\hat{G}_N$, which is derived from G_N by doubling all its edges and assigning them opposite directions (see Fig. 1).
2) $S = \{s_i\}$ denotes the finite set of N vertices in $\hat{G}_N$. Each vertex s_i is associated with a location in the workspace.
3) $A_i = \{a_{ij}\} \neq 0$ is the finite, nonempty set of directed edges that leave vertex $s_i \in S$. Each edge a_{ij} is defined through a pair of indices (i, j), which, respectively, identify the corresponding start and end vertices. $|A_i|$ is the dimension of the set, i.e., the number of edges departing from s_i.
4) $R = \{r_i\}$ is a set of M robots. Robots are allowed to move in the workspace from s_i to s_j in $\hat{G}_N$ only if $a_{ij} \in A_i$, i.e., if the two vertices are adjacent.
5) $\lambda = [\lambda_1, \cdots, \lambda_N]^T \in \Re^N$ is a vector that describes the average visiting rate to each vertex $s_i \in S$, which is expressed as *number of robots* per *time unit*.
6) $\lambda^* = [\lambda_1^*, \cdots, \lambda_N^*]^T \in \Re^N, 0 \leq \lambda_i^* \leq 1$, and $\sum_1^N \lambda_i^* = 1$ is a vector that describes the prescribed frequency distribution of visits.

The following objective must be achieved; the *M* robots must guarantee the coverage of $\hat{G}_N$ according to the prescribed visit distribution λ^*, i.e., robots must move in such a way that, for all $s_i, \lambda_i = C\lambda_i^*$, where C is a scaling factor that is expressed dimensionally as number of robots per time units.

Remark 3: The MRCFC problem does not deal with the average visiting rate at s_i expressed as number of robots per time units; the latter depends on the scaling factor C, which measures how many times per second it is possible to observe one of the M robots arriving to one of the N vertices and depends on many factors, including the navigation time to move between nodes, the number of robots, etc. Instead, MRCFC guarantees a prescribed frequency distribution of visits, for example, to guarantee that, whichever is the average visiting rate λ_i at s_i, the average visiting rate λ_j at s_j is doubled, i.e., $\lambda_j = 2\lambda_i$.

Remark 4: In the following, it will be shown that, depending both on λ^* and on the topology of $\hat{G}_N$, an exact solution to the MRCFC problem does not necessarily exist; in such a case, only approximate solutions of the form $\lambda_i \approx C\lambda_i^*$ will be considered.

Remark 5: The particular MRUFC problem corresponds to the case when λ^* has a uniform distribution, i.e., for all $s_i, \lambda_i^* = 1/N$.

Remark 6: For a real-world implementation, it is assumed that robots are equipped with proper algorithms for vertex-to-vertex navigation, as well as for obstacle avoidance and localization. In particular, it is assumed that vertex s_i is linked to an adjacent vertex s_j through a_{ij}, whenever it is possible to reach s_j starting from s_i through a "simple motion," e.g., moving along a straight line.[4]

In this study, additional desirable properties are taken into account, which are not necessarily related to the MRCFC problem but can play an important role to allow implementation on affordable and dependable robots with minimal computational, memory, and communication capabilities.

1) *Low computational cost.* The algorithm that solves MRCFC should be executable in parallel on very simple robots with limited computational power.
2) *Local memory.* The graph $\hat{G}_N$ should possibly never be stored in robots memory and, in general, a central repository where $\hat{G}_N$ is stored should not be accessible to robots during operations. Instead, all the information concerning a generic vertex, as well as the edges departing from it, should be stored into a *smart node* opportunely located in the environment. To help robots to physically navigate in the workspace, every *smart node* can store navigation directions to reach neighboring *smart nodes*.
3) *Local communication.* Robots should be able to communicate only with *smart nodes* within a very short communication range and to indirectly communicate with other robots by writing to/reading from *smart nodes*. It is assumed that: a robot cannot directly communicate with another robot; a *smart node* cannot directly communicate with another *smart node*; a robot cannot communicate with two *smart nodes* at the same time.
4) *Unpredictability.* The path followed by robots should be hardly predictable by an external observer, which is a fundamental feature for applications like surveillance and patrolling.

Remark 7: The properties above can have a relevance from the practical point of view. For example, embedding information in the environment, by using *smart nodes*, allows one to seamlessly add robots to the team without requiring a configuration phase. In addition, for security-related applications, if *smart nodes* with local communication capabilities store only local information, it is more difficult for a potential intruder to get an overall picture of the system behavior; on the other side, storage of global information, possibly on board of the robots, is a potential source of security hassles.

IV. Random Walk and Edge Counting for Multirobot Coverage

In this section, known properties of the *random walk* and *edge counting* algorithms are recalled by focusing on the multirobot-coverage scenario, which will provide a basis to introduce the new *PatrolGRAPH** algorithm. In particular, it is assumed in the following that the M robots execute, in parallel, a particular

[4]The intuitive notion of "simple motion" can vary, depending on the robot kinematics, localization skills, etc.

instance of Algorithm 1, which describes a class of navigation strategies, which includes *random walk*, *edge counting*, as well as *PatrolGRAPH**.

Algorithm 1 itself is straightforward. Line 1 chooses an arbitrary start vertex s_{start}, which can be different for different robots. When the robot is in vertex s_c, the operator $\text{choose}(s_c, \text{Alg})$ in line 3 returns one of the directed edges $a_{cl} \in A_c$, according to a strategy that depends on the algorithm Alg, which identifies a specific navigation strategy. Line 4 summarizes all procedures that are requested for the robot to move to the next vertex (including motion control, obstacle avoidance, localization, etc.). As discussed in the previous section, it is assumed that each robot is capable to move along edge a_{cl} and to reach the next *smart node* correctly, i.e., it is equipped with proper hardware and software subsystems to achieve this. The operator $\text{succ}(s_c, a_{cl})$ in line 5 returns the successor vertex that results from the traversal of edge $a_{cl} \in A_c$ starting from vertex $s_c \in \hat{G}_N$. In practice, the output of this operator is the effect of the actual robot motion.

Algorithm 1 Navigation Algorithm

1: $s_c = s_{start}$
2: **while** TRUE **do**
3: $a_{cl} := choose(s_c, Alg)$
4: Move along edge a_{cl}
5: $s_c := succ(s_c, a_{cl})$
6: **end while**

Remark 8: In a real-world implementation, the operator $\text{choose}(s_c, \text{Alg})$ requires the robot to communicate with the *smart node*, which stores information about vertex s_c, and to retrieve navigation directions that provide guidance to move toward the next vertex.

The behavior of *random walk* and *edge counting* can be obtained by properly implementing the operator $\text{choose}(s_c, \text{Alg})$, as defined by the Algorithms 2 and 3.

Algorithm 2 Operator $choose(s_c$, *Random Walk*$)$

1: $a_{cl} = get(A_c, random)$
2: **return** a_{cl}

Algorithm 3 Operator $choose(s_c$, *Edge Counting*$)$

1: $a_{cl} = get(A_c, switch_c)$
2: $switch_c = (switch_c + 1) \mod |A_c|$
3: **return** a_{cl}

Random walk selects one of the edges departing from s_c randomly with equal probability so that the relative frequency of selections tends to $1/|A_c|$ as the number of visits to node s_c tends to infinity. Line 1 of Algorithm 2 picks an edge a_{cl} in the set A_c with discrete uniform distribution.

Edge counting selects edges departing from s_c in circular order, in order to guarantee that, every $|A_c|$ visits, all edges are chosen with the same relative frequency $1/|A_c|$. Line 1 of Algorithm 3 picks an edge a_{cl} in the set A_c, depending on the current value of switch_c, and line 2 updates switch_c to point to the next edge.

In order to demonstrate the behavior of *random walk* and *edge counting* with M robots executing Algorithm 1 in parallel, it is convenient to model the system as a closed queueing network (CQN), which is a dynamical model usually found in application domains, like process automation, communication networks, etc. [62], with the purpose of describing and analyzing how *service centers* are allocated to *customers* in time. CQNs can be modeled using continuous-time Markov chains (CTMCs), with the final purpose of determining the stationary distribution that describes the probability of being in a given state at a given time. However, since the MRCFC problem focuses on the frequency distribution of visits over vertices, in the following, it will be sufficient to consider flow-balance equations for every vertex of the graph.

In general, CQNs are specified by

1) the number N of service centers s_i (corresponding to the vertices of $\hat{G}_N$);
2) the number M of customers (corresponding to robots);
3) the average arrival/departure rate λ_i to/from node s_i (i.e., expressed as *number of arriving customers* divided by *time*), and the arrival and departure rate in s_i are the same in a CQN (and, obviously, correspond to the visiting rate), since customers cannot exit or enter the network from outside;
4) the number of servers m_i running in parallel at s_i (i.e., *how many free places* there are for robots to perform a task in s_i at the same time), the time t_i requested to complete a task in s_i, and the maximum service rate $\mu_i = 1/t_i$ (i.e., *how many robots can complete the task* per *time unit*);
5) a routing policy, which is expressed by a $N \times N$ transition matrix P.

The transition matrix P deserves attention. In case of *random walk*, the generic element $p_{ij} \in P$ describes the probability that a robot in s_i heads toward s_j after finishing its task; in the case of *edge counting*, $p_{ij} \in P$ describes the ratio of robots that, after visiting s_i, must head toward s_j.

By recalling that $|A_i|$ is the number of edges departing from s_i, it can be easily verified that the routing policy of *random walk* and *edge counting* can be modeled by a transition matrix P whose elements p_{ij} are set as follows: If s_i and s_j are adjacent, then

$$p_{ij} = \frac{1}{|A_i|} \tag{1}$$

whereas if s_i and s_j are not adjacent, then

$$p_{ij} = 0. \tag{2}$$

Equations (1) and (2) state that the probability (in the case of *random walk*) or the relative frequency (in the case of *edge counting*) according to which a_{ij} is chosen is the same for all edges departing from s_i. Since P will be used in the following to write flow-balance equations describing average visiting rates at

steady state, the probability and the relative frequency assume the same meaning.

Starting from (1) and (2), which define the routing policy of *random walk* and *edge counting*, the corresponding transition matrix P turns out to be a nonsymmetrical matrix with *zeros* in the diagonal since no vertex is connected to itself, i.e.,

$$P = \begin{bmatrix} 0 & \cdots & p_{1j} & \cdots & p_{1N} \\ \vdots & \ddots & & & \vdots \\ p_{i1} & \cdots & 0 & \cdots & p_{iN} \\ \vdots & & & \ddots & \vdots \\ p_{N1} & \cdots & p_{Nj} & \cdots & 0 \end{bmatrix}. \tag{3}$$

P is a *stochastic matrix* [63], i.e., it is subject to the following constraints:

$$\sum_{j=1}^{N} p_{ij} = 1, \qquad i = 1, \ldots, N \tag{4}$$

$$0 \leq p_{ij} \leq 1, \qquad i, j = 1, \ldots, N. \tag{5}$$

In addition, P is irreducible, since $\hat{G}_N$ is strongly connected (i.e., it is possible to reach every vertex from every other vertex in a finite number of transitions).

Under these conditions, it is a known result from Markov process theory that the average rate of visits λ_i that each vertex s_i of $\hat{G}_N$ receives is proportional to the number of incident edges $|A_i|$ of s_i (i.e., the return time to s_i is inversely proportional to $|A_i|$; see [63]). To verify this, consider that in a CQN the average arrival rate at steady state equals the average departure rate for every vertex. Then, it is possible to write the following flow-balance equations:

$$\lambda_i = \sum_{j=1}^{N} p_{ji} \lambda_j, \qquad i = 1, \ldots, N. \tag{6}$$

Let $\mathbf{1} = [1 \ldots 1]^T$ be the unitary vector. The flow-balance equations in matrix form, by considering the additional requirement that the sum of average visiting rates, computed over all vertices of $\hat{G}_N$, is constant, can be written as

$$\begin{bmatrix} P^T - I \\ \mathbf{1}^T \end{bmatrix} \lambda = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ C \end{bmatrix}. \tag{7}$$

System (7) has exactly one solution whenever P is stochastic and irreducible. In order to prove this statement, it is sufficient to observe that, according to the Perron–Frobenius theorem [63], the flow-balance equations admits solutions with all positive components (in the following referred to as *positive solutions*), which correspond to the left eigenvector of P with eigenvalue 1. The last equation in (7) allows one to determine the unique solution whose components sum up to C. In particular, by setting $C = 1$, the resulting visiting rate corresponds exactly to the frequency distribution of visits over vertices.

Solutions have the form

$$\lambda_i = C \frac{|A_i|}{\sum_k^N |A_k|}, \qquad i = 1, \ldots, N. \tag{8}$$

This can be verified by substituting (8) into (6) and (7) and by recalling that, for a generic vertex s_i on the left side of (6), there are exactly $|A_i|$ elements $p_{ji} = 1/|A_j|$ on the right side of the equation corresponding to adjacent vertices s_j.

Remark 9: The parameters m_i (i.e., the number of robots allowed to perform the assigned task in s_i at the same time), t_i (i.e., the time requested to complete a task in s_i, as well as the distance (or the navigation time) between vertices), the interference between robots due to traffic, etc., do not appear in the flow-balance equations (6), i.e., they are not required to compute the mutual relationships between the components of λ at steady state, corresponding to the frequency distribution of visits. It will be shown in the following that instead they play a role in computing the actual value of the visiting rate λ by determining the appropriate value of the parameter C.

Even if the frequency distribution of visits, when averaged over a time interval tending to infinite, is identical in the case of *random walk* and *edge counting*, the latter distributes visits better when considering shorter time intervals. This is due to the fact that *edge counting* chooses all edges departing from s_c in a circular order, thereby ensuring that, every time instant, the ratio of robots that have chosen a_{cj} up to that time approximates p_{cj} within a bounded limit. The behavior of *random walk* and *edge counting* can be better compared through experiments [64].

The properties of *random walk* and *edge counting* do not make them suitable to solve the MRCFC problem as the frequency distribution of visits is proportional to the degree of the vertices. In the following section, the PatrolGRAPH* algorithm is presented, which allows robots to solve MRCFC in its general formulation, while exhibiting a good transient behavior.

V. PatrolGRAPH*—Description of the Algorithm

*PatrolGRAPH**, which is the navigation strategy considered in this study to solve the MRCFC problem, is composed of following two phases:

1) an offline setup phase that is executed only once for every given MRCFC problem;
2) an online phase that is fully distributed and executed in parallel by the M robots during coverage.

The offline phase has the main purpose of finding a solution to the so-called inverse problem, i.e., the problem of choosing P to guarantee that the solution of (7), with $C = 1$, corresponds to the prescribed frequency distribution. The offline phase does not involve the individual robots.

The online phase assumes that the M robots execute in parallel a particular instance of Algorithm 1, by implementing a specific strategy to choose the next vertex to be visited. It is worth noting that in this phase, robots need only to get local information required to their navigation, from the closest smart node. This mechanism is important also since it makes possible to insert new robots into the working team without an *a priori* configuration phase of the robots themselves.

Remark 10: In a practical implementation, the matrix P can be stored in distributed form in the *smart nodes* forming the navigation graph. In particular, every *smart node* stores a different row of the matrix P (actually only its non zero elements).

Remark 11: In principle, the transition matrix P computed in the offline phase could be used for implementing a *nonuniform random walk* algorithm solving MRCFC. The PatrolGRAPH* online phase is designed to guarantee better performance, over short time intervals, with respect to a *nonuniform random walk*, as will be shown in Section VII.

A. Offline Phase

In [10], it has been shown that it is always possible to find a stochastic matrix P with a prescribed stationary distribution (this is also known in Markov-chain theory as the inverse problem). In particular, for $C = 1$, and for any given frequency distribution λ^*, it is straightforward to determine a proper assignment of elements p_{ij} such that (4), (5), and (7) are satisfied. In fact, by substituting λ^* into the flow-balance equations (6), a possible solution for p_{ji}, for a fully connected graph, is the following:

$$p_{ji} = \lambda_i^*, \qquad i, j = 1, \ldots, N. \tag{9}$$

In the special case that λ^* has a uniform distribution, (6) becomes

$$\sum_{j=1}^{N} p_{ji} = 1, \qquad i = 1, \ldots, N. \tag{10}$$

Equations (4), (5), and (10) correspond to the definition of a doubly stochastic transition matrix, whose stationary vector is known to be the uniform distribution.

The inverse problem is sometimes solved by adopting the so-called metropolis rule [9]. Let us assume that the transition matrix P is a nonsymmetrical matrix with *zeros* in the diagonal since no vertex is connected to itself, as in (3). The metropolis rules state that, by letting elements in the diagonal be nonzero (i.e., by adding self-loops departing from/arriving to the same vertex) and by modifying transitions probabilities to neighboring vertices accordingly, it is possible to obtain a new transition matrix P with the desired steady-state distribution. In practice, adding self-loops in the navigation graph $\hat{G}_N$ can make sense in some applications but not in others. In cleaning, for instance, it really does not make sense to clean the same room twice. In surveillance applications, e.g., if the focus is on periodically checking the status of doors, windows, and alarms, that there are no lost objects around, etc., self-loops appear to be a waste of time and must be consequently avoided.

In the definition of the MRCFC problem, it has been assumed that the topology of the navigation graph $\hat{G}_N$ is given *a priori*, and therefore, it is not possible to arbitrary assign all entries of P. In particular, p_{ij} can be assigned a nonzero value only if the couple of vertices s_i, s_j in $\hat{G}_N$ are adjacent. When an edge from s_i to s_j does not exist, the corresponding entry in matrix P is constrained to be zero, and a solution to MRCFC cannot be found by adopting standard techniques, e.g., the metropolis rule.

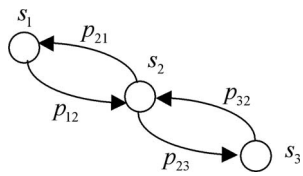

Fig. 2. Example of navigation graph $\hat{G}_N$ for which it is not possible to find an exact solution to the MRUFC problem.

Let us consider the case in–Fig. 2: only p_{12}, p_{21}, p_{23}, and p_{32} can be arbitrarily assigned, whereas all the remaining elements are constrained to be zero. Assume also, for simplicity, that λ^* is the uniform distribution. Equations (4) and (9) require that, for P to be bistochastic, its nonzero elements are subject to

$$\begin{aligned} p_{12} = 1, \quad p_{21} = 1 \\ p_{21} + p_{23} = 1, \quad p_{12} + p_{32} = 1 \\ p_{32} = 1, \quad p_{23} = 1. \end{aligned} \tag{11}$$

The above system has no admissible solutions. In fact, it is impossible that vertex 2 receives the same number of visits as vertices 1 and 3, since robots must necessary visit 2 whenever moving from 1 to 3 and *vice versa*.

Since an exact solution to the MRCFC problem is not guaranteed to exist, the offline phase of *PatrolGRAPH** searches for a solution in the sense of the least squares.

This requires some additional definitions. First of all, letting $\mathbf{b} = [b_1 \cdots b_{N^2}]^T \in \Re^{N^2}$ be defined as a vector that parameterize the elements of the transition matrix P so that $P = P(\mathbf{b}) \in \Re^{N \times N}$. This is done by ideally subdividing $\mathbf{b}$ into N vectors with length N and by stacking them onto each other to build up the rows of P. More formally

$$p_{ij} = b_{N \times (i-1)+j}, \qquad i, j = 1, \ldots, N \tag{12}$$

and

$$P(\mathbf{b}) = \begin{bmatrix} b_1 & \cdots & b_N \\ \cdots & b_{N\times(i-1)+j} & \cdots \\ b_{N\times(N-1)+1} & \cdots & b_{N^2} \end{bmatrix}. \tag{13}$$

Second, as long as $P = P(\mathbf{b})$ is stochastic and strongly connected, the vector of average visiting rates at steady state can be expressed as a function $\lambda = \lambda(\mathbf{b}) : \Re^{N^2} \rightarrow \Re^N$ and computed by solving the following system:

$$\begin{bmatrix} P^T(\mathbf{b}) - I \\ \mathbf{1}^T \end{bmatrix} \lambda(\mathbf{b}) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ C \end{bmatrix}. \tag{14}$$

The first N rows in (14) correspond, for every $\mathbf{b}$, to the flow-balance equations at steady state (6). The last row states that the sum of average visiting rates, which is computed over all vertices of $\hat{G}_N$, is constant. Since the offline phase aims at guaranteeing a prescribed frequency distribution of visits, in the following, it can be assumed $C = 1$. System (14) has exactly one solution

$\lambda = \lambda(\mathbf{b})$ for every $\mathbf{b}$, ensuring that $P(\mathbf{b})$ is stochastic and fully connected.

Finally, letting λ^* represent the ideal desired solution of the MRCFC problem. Then, $\lambda^* - \lambda(\mathbf{b})$ is a representation of the mismatch between the actual frequency distribution and the desired one as a function of the structure of the navigation graph (which is expressed by the entries of matrix P).

Following the discussion above, the offline phase of the *PatrolGRAPH** algorithm is the search for a solution to the following minimization problem with respect to the unknown variable $\mathbf{b}$.

Problem 1: This is given by

$$\mathbf{b}^* = \arg\min_{\mathbf{b}} (\lambda^* - \lambda(\mathbf{b}))^T (\lambda^* - \lambda(\mathbf{b})) \tag{15}$$

subject to

$$\sum_{j=1}^{N} b_{N\times(i-1)+j} = 1, \qquad i = 1, \ldots, N \tag{16}$$

$$b_{N\times(i-1)+j} \geq |\epsilon| \qquad \forall i, j \qquad \text{s.t.} \qquad a_{ij} \in A_i \tag{17}$$

$$b_{N\times(i-1)+j} = 0 \qquad \forall i, j \qquad \text{s.t.} \qquad a_{ij} \notin A_i. \tag{18}$$

Equations (16)–(18) guarantee that $P(\mathbf{b})$ is a strongly connected stochastic matrix, as long as $\hat{G}_N$ is a strongly connected graph. In particular, (16) simply corresponds to (4) rewritten for $\mathbf{b}$. Equations (17) and (18) correspond to (5) but with the additional constraints that two vertices that are adjacent in $\hat{G}_N$ must be necessarily assigned a positive value greater than $|\epsilon|$, whereas two vertices that are not adjacent in $\hat{G}_N$ must be necessarily assigned a zero value. This is sufficient (even if not necessary) to guarantee that, after the minimization process, the resulting $P(\mathbf{b})$ still will be strongly connected. Summarizing, (17) and (18), taken together, ensure that the topology of $\hat{G}_N$ is preserved after the minimization process.

Remark 12: Depending on the graph topology, the solution of the minimization problem above can lead to a matrix $P(\mathbf{b})$ whose stationary vector might be very far from the desired frequency distribution λ^*. Considering once again the graph in Fig. 2, and assuming now that $\lambda_1^* = \lambda_3^* = n\lambda_2^*$, with $n > 1$, meaning that every n visits to s_1 and s_3, vertex s_2 should be visited only once. The problem has not an exact solution, since the topology of the graph forces the number of visits received by s_2 to be half the number of total visits (as it happened in the MRUFC case discussed before). Moreover, the optimal solution in the least-square sense turns out to be a *factor* $2n$ *approximation* of the desired solution, in that λ_2^* is visited $2n$ times as often than required.

Remark 13: Even if this study focuses on stationary properties, the system has obviously a transient behavior. The rate of convergence to stationarity (which is also referred to as the mixing rate of the Markov chain) is known to depend on the second largest (in absolute value) eigenvalue of the transition matrix P [63]. For every stationary distribution λ, it has been demonstrated in [10] that the set of transition probability matrices $P(\mathbf{b})$ such that $\lambda = P^T(\mathbf{b})\lambda$ is a compact and convex set. Then, it is, in principle, possible to search for the transition matrix with the fastest mixing rate while looking for a solution to MRCFC. This problem has been addressed in [65] but only in case of a symmetric transition matrix (i.e., when the equilibrium distribution is uniform). A detailed analysis on the convergence rate is outside the scope of this paper.

Remark 14: If one wants to control the actual value of the average visiting rates $\lambda = C\lambda^*$, it is necessary to control the scaling factor C, which can be done by choosing the number M of patrolling robots depending on m_i, t_i, the navigation time and other relevant quantities (for details, see CQN theory [62]). An approximate estimate of C can be obtained by observing that λ_i^* measures the ratio of navigation paths starting from vertex s_i during the online phase. By considering the average service time t_i in s_i and the average navigation time τ_{ij} to move along a_{ij} (where t_i and τ_{ij} include possible delays due to traffic and other random factors), it is possible to compute

$$\bar{\tau} = \sum_{i=1}^{N} \lambda_i^* \sum_{j=1}^{N} p_{ij}\ \tau_{ij} \tag{19}$$

where $\bar{\tau}$ is the mean navigation time (which is averaged over all edges chosen for navigation), and

$$\bar{t} = \sum_{i=1}^{N} \lambda_i^*\ t_i \tag{20}$$

where $\bar{t}$ is the mean service time (which is averaged over all vertices visited). Finally, by selecting the number of patrolling robots M, it is possible to control the scaling factor C

$$C = \sum_{i}^{N} \lambda_i = M \frac{1}{\bar{t} + \bar{\tau}}. \tag{21}$$

which measures how many times per second it is possible to observe one of the M robots arriving to one of the N vertices. Note that, if one would be allowed to alter the topology of the graph, C could also be increased by connecting two nonadjacent vertices s_i and s_j whose average navigation time τ_{ij} is such that $\bar{\tau}$ in (19) decreases. A self-loop in s_i with $\tau_{ii} = 0$ is a special case of this possibility.

B. Online Phase

After having properly set P, the online phase of *PatrolGRAPH** implements the particular instance of the operator $\mathrm{choose}(s_c, \mathrm{Alg})$ described in Algorithm 4.

Algorithm 4 Operator *choose*(s_c, *PatrolGRAPH**)

1: $v_c = v_c + 1$
2: **for all** j such that $a_{cj} \in A_c$ **do**
3: $\quad \Delta p_{cj} = (k_{cj} - |\eta_j|)/v_c - p_{cj}$
4: **end for**
5: $l = \arg\min_j(\Delta p_{cj})$
6: $k_{cl} = k_{cl} + 1$
7: **return** a_{cl}

In order to introduce the rationale beyond *PatrolGRAPH**, let us assume that a robot arrives to vertex s_c. Then, the operator

$\text{choose}(s_c, \text{Alg})$ must take a decision about which edge to follow in order to proceed. A *nonuniform random walk* strategy would simply select the departing edge according to the probability distribution computed in the offline phase. In this way, the ratio of robots that follow edge a_{cj} (i.e., the relative frequency of the choice) tends to p_{cj} as time tends to infinity. In the case of *PatrolGRAPH**, the routing policy has a similar rationale, but it is modified in order to ensure that, every time instant, the ratio of robots that have chosen a_{cj} up to that time approximates p_{cj} within a bounded limit. This guarantees that the frequency distribution of visits, when averaged over a short period, approximates the stationary vector better than in a *nonuniform random walks*.

More in detail, Algorithm 4 works as follows. Let us define the following variables.

1) v_i is an integer variable initialized to 0, which counts the overall number of visits to vertex s_i.
2) k_{ij} is an integer variable initialized to 0, which counts the number of times that robots have chosen to proceed to s_j after leaving s_i.
3) η_j is a zero-mean random variable with standard deviation σ, i.e., $\eta_j = N(0, \sigma)$.

Let us assume for the moment that the $|A_c|$ random variables η_j are equal to zero. Line 1 updates the number of visits v_c received by s_c; line 3 computes, for every adjacent vertex, the error Δp_{cj} between the ratio k_{cj}/v_c and the desired relative frequency p_{cj}; line 5 picks the edge a_{cl} for which Δp_{cj} is minimum; line 6 updates k_{cl}.

Let us consider now, in line 3, the random variables $\eta_j = N(0, \sigma)$: By subtracting $|\eta_j|$ from k_{cj} in line 3, there is a nonzero probability that an edge for which Δp_{cj} is not minimum (i.e., which has been already selected) is selected again. If requested by the application, this mechanism can be used to purposely introduce a factor of unpredictability.

In Section VI-B, it is demonstrated that the routing policy adopted by *PatrolGRAPH** in Algorithm 4 to distribute robots along edges a_{ij} departing from s_i converges to the desired relative frequencies p_{ij}, and therefore, it can be modeled as a Markov chain with transition matrix P.

Finally, and similar to a *nonuniform random walk*, the online algorithm exhibits the desirable properties defined in Section III. In particular, we have the following.

1) *Low computational cost.* The computational requirements are minimal, as *PatrolGRAPH** only requires updating v_c and k_{cl} and computing Δp_{cj}. The latter requires to consider all $|A_c|$ edges departing from s_c and, hence, has a linear computational complexity $O(|A_c|)$.
2) *Local memory.* Memory requirements are very limited, as *PatrolGRAPH** requires storing, in the *smart node* associated with vertex s_c, the variable v_c, which counts the number of visits received, a set of $|A_c|$ variables k_{cj}, which counts the number of times that each edge has been selected, and, finally, a set of $|A_c|$ variables p_{cj}, which store the desired relative frequency for every departing edge. Summarizing, memory complexity is $O(|A_c|)$ for every *smart node*.
3) *Local communication.* Robots are only required to write to/read from the *smart node* that stores information about the vertex of the graph in which they currently are.
4) *Unpredictability. PatrolGRAPH** can be made as unpredictable as desired by properly increasing the variance σ of the random variable η (which can be simply set to zero if unpredictability is not required by the application). However, when the desired frequency distribution of visits is not uniform, an intruder could ideally observe which areas are visited less frequently and potentially use this information.

VI. PatrolGRAPH*—Properties of the Algorithm

The goal of this section is to discuss some properties of the offline and the online phases of *PatrolGRAPH**.

A. Offline Phase

The offline phase of *PatrolGRAPH** requires finding a solution to Problem 1, which is a constrained optimization problem admitting a minimum point $\mathbf{b}^*$, since (15) is a lower bounded function, and the constraints in (16)–(18) define a bounded and closed set.

The nonlinear dependence of $\lambda(\mathbf{b})$ on $\mathbf{b}$ makes a convexity analysis of Problem 1 not straightforward. Therefore, a conservative approach would require to adopt a global optimization algorithm. Even though minimization is performed offline, and therefore, the time required to solve Problem 1 should not be considered as a critical issue, computational complexity of a global method could become unacceptable as the size N of the graph increases [66]. Simulative experiments (among those discussed in Section VII) have shown that the solutions of Problem 1, computed using global-[5] and local-[6] search algorithms, always correspond to comparable values of the minimization residual.

When solving Problem 1 for a given λ^*, it is interesting to check if MRCFC has an exact solution, i.e., if $\mathbf{b}^*$ exists such that $\lambda^* = \lambda(\mathbf{b}^*) = P^T(\mathbf{b}^*)\lambda(\mathbf{b}^*)$. A test can be performed by solving the following problem.

Problem 2: This is given by

$$\mathbf{b}^* = \arg\min_{\mathbf{b}} (\lambda^* - P^T(\mathbf{b})\lambda^*)^T (\lambda^* - P^T(\mathbf{b})\lambda^*) \qquad (22)$$

subject to (16)–(18).

Even if Problems 1 and 2 are different, they have the same compact and convex set of global minima whenever an exact solution to MRCFC exists [10]. Therefore, if an exact solution to MRCFC exists, one is guaranteed to find it by solving Problem 2. The advantage of Problem 2 over Problem 1 is that the former can be formulated as a quadratic optimization problem [66] subject to linear constraints.

[5]The branch-and-bound-based global search and the Adaptive multistart global random search implemented in the TOMLAB optimization toolbox; see `http://tomopt.com`.

[6]Implemented in the basic MATLAB `fmincon(...)` function; see `http://www.mathworks.com`.

Theorem 1: Problem 2 is a quadratic optimization problem. From (13), $P^T(\mathbf{b})$ is linear with respect to $\mathbf{b}$; then, it follows that (22) can be rewritten as

$$\mathbf{b}^* = \arg\min_{\mathbf{b}} (\lambda^* - \Gamma\mathbf{b})^T (\lambda^* - \Gamma\mathbf{b}) \tag{23}$$

where $\Gamma \in \Re^{N \times N^2}$ is defined as

$$\Gamma = [\lambda_1^* I \mid \lambda_2^* I \mid \cdots \mid \lambda_N^* I] \tag{24}$$

with $I \in \Re^{N \times N}$ the identity matrix.

The function to be minimized is a semidefinite positive quadratic form with respect to $\mathbf{b}$ with linear constraints. ■

B. Online Phase

A demonstration is given for the following theorem.

Theorem 2: Algorithm 4 guarantees that the expected value $E(k_{ij}/v_i - p_{ij})$ is bounded and tends to zero as the number of visits $v_i \to \infty, (i, j = 1, \ldots, N)$. •

Proof: It is important to note that, according to Algorithm 4, only edges $a_{ij} \in A_i$ can be selected, and therefore, the assumptions are necessarily valid whenever $p_{ij} = 0$. Let us now consider edges $a_{ij} \in A_i$, and define k_{ij}^- as the value of k_{ij} at visit v_i before line 6, i.e., before selecting an edge and, eventually, incrementing the corresponding counter.

Let us assume that a robot has just arrived in s_i, i.e., v_i has been just increased by one according to line 1, and Δp_{ij} has been computed for all edges $a_{ij} \in A_i$ according to line 3 of the algorithm

$$\Delta p_{ij} = \frac{k_{ij}^- - |\eta_j|}{v_i} - p_{ij}, \qquad j \text{s.t.} a_{ij} \in A_i. \tag{25}$$

The following must hold for the generic edge, say a_{il}, that is selected at visit v_i

$$\Delta p_{il} = \min_j \Delta p_{ij}. \tag{26}$$

Let us consider now that, for all $v_i > 0$, the following conditions necessarily hold after a choice has been performed in line 6 from the definition of relative frequency k_{ij}/v_i:

$$\sum_{j \text{s.t.} a_{ij} \in A_i} \left(\frac{k_{ij}}{v_i} - p_{ij} \right) = 0. \tag{27}$$

By summing up (25) with respect to j, and by considering that $k_{ij} = k_{ij}^-$ for all $j \neq l$, whereas $k_{il} = k_{il}^- + 1$, and by using (27), yield

$$\sum_{j \text{s.t.} a_{ij} \in A_i} \Delta p_{ij} = -\frac{1}{v_i} \left(1 + \sum_{j \text{s.t.} a_{ij} \in A_i} |\eta_j| \right). \tag{28}$$

The above equation allows one to set the following upper bound:

$$\min_j \Delta p_{ij} \leq \frac{1}{|A_i|} \sum_{j \text{s.t.} a_{ij} \in A_i} \Delta p_{ij} \leq 0 \tag{29}$$

which, together with (25) and (26), yields the following condition associated with a_{il}:

$$k_{il}^- \leq p_{il} v_i + |\eta_l|. \tag{30}$$

The above equation allows one to state that the following upper bound must be valid for a_{il}, after the counter k_{il} has been incremented in line 6:

$$\frac{k_{il}}{v_i} - p_{il} = \frac{k_{il}^- + 1}{v_i} - p_{il} \leq \frac{|\eta_l| + 1}{v_i}. \tag{31}$$

Since $|\eta_j|$ is a random expression with the same expected value $E(|\eta|)$ for all j at every visit v_i, the following condition necessarily holds, whichever edge a_{il} is selected at visit v_i:

$$\lim_{v_i \to \infty} E\left(\frac{k_{il}}{v_i} - p_{il} \right) \leq \lim_{v_i \to \infty} \frac{E(|\eta|) + 1}{v_i} = 0. \tag{32}$$

Equation (32) refers only to the selected edge at visit v_i. For edges a_{ij}, which are not selected at visit v_i, i.e., for $j \neq l$, the counter k_{ij} is kept constant. This yields, as long as a_{ij} is never selected

$$\lim_{v_i \to \infty} \frac{k_{ij}}{v_i} - p_{ij} = -p_{ij}, \qquad j \neq l \text{ s.t. } a_{ij} \in A_i. \tag{33}$$

In general, there can be visits v_i such that a_{ij} is selected and $k_{ij}/v_i - p_{ij}$ increases up to the bound in (31), whose expected value monotonically tends to zero as $v_i \to \infty$, and visits v_i such that a_{ij} is not selected. $k_{ij}/v_i - p_{ij}$ monotonically decreases toward a limit lower than zero according to (33). This holds for all edges a_{ij}, and it allows the statement that

$$\lim_{v_i \to \infty} E\left(\frac{k_{ij}}{v_i} - p_{ij} \right) \leq 0, \qquad j \text{ s.t. } a_{ij} \in A_i. \tag{34}$$

Finally, (27) and (34) are valid for all vertices s_i; by recalling the initial consideration about edges $a_{ij} \notin A_i$, they are sufficient to guarantee that

$$\lim_{v_i \to \infty} E\left(\frac{k_{ij}}{v_i} - p_{ij} \right) = 0, \qquad i, j = 1, \ldots, N \tag{35}$$

which demonstrates the theorem. ■

The following is a corollary of the above theorem.

Theorem 3: If the random variable $\eta_j = 0, j = 1, \ldots, N$, Algorithm 4 guarantees that $k_{ij}/v_i - p_{ij}$ is bounded and tends to zero as the number of visits $v_i \to \infty, (i, j = 1, \ldots, N)$. •

Proof: The proof immediately follows by setting $\eta_j = 0$ and by rewriting (32), (34), and (35) for the actual value of $k_{ij}/v_i - p_{ij}$ instead of the expected value $E(k_{ij}/v_i - p_{ij})$. ■

VII. Experimental Validation

In this section, an assessment of the properties of *PatrolGRAPH** is presented. Since the focus of this paper is on the theoretical properties, simulated experiments have been considered.

*PatrolGRAPH** is compared with *node count* [2], *MSTC* [1], and a *nonuniform random walk* with the same transition matrix computed during the offline phase of *PatrolGRAPH** (see *Remark* 11). *Node count* is the simplest real-time-search algorithm for which formal proofs of complete coverage (in a statistical sense) have been given. *MSTC* is the only approach that explicitly guarantees a uniform frequency distribution of visits (i.e., it solves the MRUFC problem; for detail, see Section II). The

TABLE I
QUALITATIVE COMPARISON

	PatrolGRAPH*	Node Count	MSTC
Suited for MRCFC	good	**very bad**	**very bad**
Suited for MRUFC	good	good	very good
Domain of App.	very good	very good	**bad**
Low Comp. Cost	good	very good	good
Local Memory	very good	very good	very good
Local Comm.	very good	**very bad**	very good
Unpredictability	very good	**bad**	**very bad**

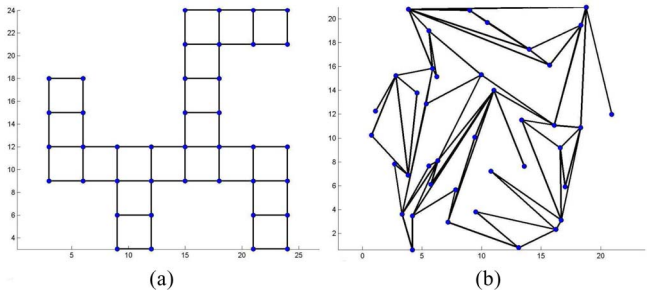

Fig. 3. Randomly generated nonoriented graphs used in experiments. (a) Grid-like topology. (b) Free topology.

nonuniform random walk allows verification of if the online phase of *PatrolGRAPH** improves the transient behavior.

First, a *qualitative* analysis of *node count* and *MSTC* is performed by outlining their computational, communication, and memory requirements, as well as their domain of applicability. Second, the ability of *node count*, i.e., *PatrolGRAPH**, and the *nonuniform random walk* to solve MRUFC is compared on a set of randomly generated navigation graphs. This quantitative analysis is not required for *MSTC*, since the approach guarantees a uniform distribution of visits whenever it can be applied (which turns out to be true in a limited subset of cases). Finally, the ability of *PatrolGRAPH** and the *nonuniform random walk* to solve the general MRCFC problem is tested by imposing a nonuniform distribution of visits. To the authors' knowledge, there are no other algorithms in the literature that deal explicitly with MRCFC.

A. Qualitative Analysis

Table I summarizes the results of the qualitative analysis.

MSTC gets a *bad* score in *domain of applicability*, since it puts very strong constraints on the topology of G_N, which must be a grid with no local-cut nodes (i.e., without nodes whose removal locally disconnects the graph induced by the grid). This necessarily follows from the fact that the spanning-tree is built starting from a coarse-grain grid (which can have local-cut nodes), but such a coarse-grain grid is never used for navigation. Instead, every cell of the coarse-grain grid is divided into four cells, and every cell of the resulting grid (which is used for navigation) is linked to neighboring cells in a way that eliminates the possibility of local-cut nodes. Navigation graphs with a free topology (i.e., whose vertices are not arranged on a grid) are not even considered.

Node count gets a *very bad* score in *local communication*, since it requires that a robot in a given vertex can communicate with *smart nodes* that store information about adjacent vertices in order to be informed about how many times they have been visited.

Finally, both algorithms get a *bad* score in *unpredictability*: The behavior of *node count* is deterministic and, hence, at least, in principle, predictable; *MSTC* completely fails to meet this requirement, since the Hamiltonian cycle followed is the same for all robots and does not vary in time.

B. Quantitative Analysis

*PatrolGRAPH**, *node count*, and the *nonuniform random walk* are compared by running them on a set of randomly generated graphs. Two types of graphs have been considered.

1) graphs with a grid-like topology and no local-cut nodes, which are built by deleting randomly chosen vertices and edges from a coarse-grain grid, and by using this grid to build a fine-grain grid with doubled resolution [see–Fig. 3(a)];
2) graphs with a free topology, which are built by randomly distributing vertices in a given area and by randomly connecting them through edges [see Fig. 3(b)].

Node count, *PatrolGRAPH**, and the *nonuniform random walk* are executed on graphs of types 1 and 2. *MSTC* can be executed and solves MRUFC only on graphs of type 1. However, as anticipated, experiments on *MSTC* are omitted since the result would be straightforward.

Experiments have been conducted by considering two sets of 50 randomly generated graphs corresponding to the above two topologies. First, for every graph G_N, the offline phase of *PatrolGRAPH** is executed, allowing computation of transition probabilities. Second, *node count*, *PatrolGRAPH**, and the *nonuniform random walk* are executed on each graph by assuming a team of M robots running concurrently for a given number of time steps (6000 time steps).

In particular, we have the following.

1) Starting positions are randomly chosen for all robots.
2) The navigation time between vertices depends on the traveled distance; a delay possibly due to traffic or navigation errors is randomly computed and added to the navigation time.
3) The time required to perform a task in a vertex is assumed to be constant; a delay is randomly computed and added to the task execution time.
4) The same experiment is executed with a varying number M of robots ($M = 5, 10, 20$).
5) The same experiment is performed twice, by imposing a uniform frequency distribution of visits (i.e., to solve MRUFC) and by imposing different visiting frequency to different vertices (i.e., to solve the more general MRCFC problem).

In order to compare the algorithms, the following quantities are recorded for every time step t:

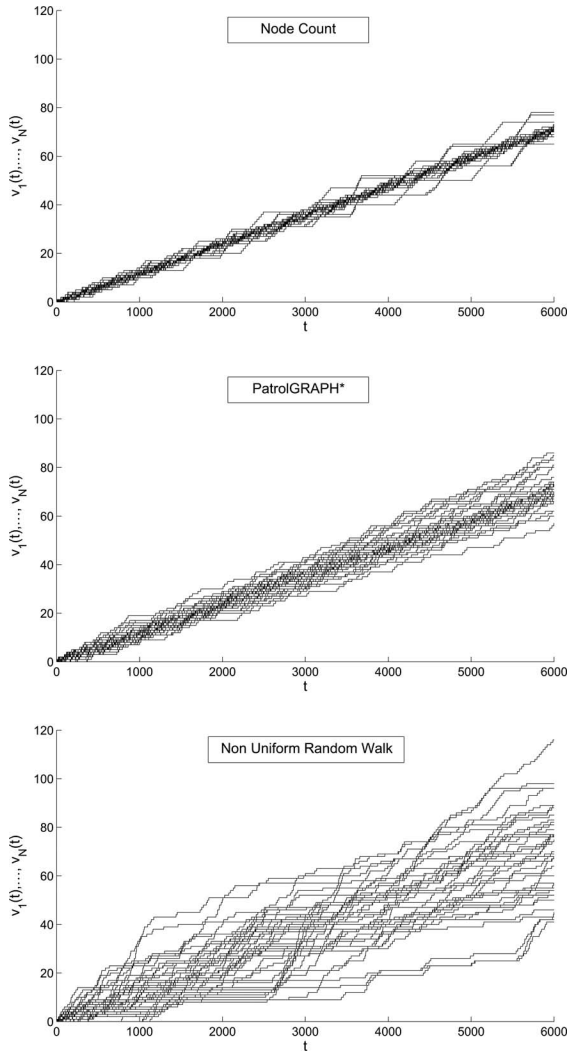

Fig. 4. Free-topology graph. Number of visits $v_1(t), \ldots, v_N(t)$ received by vertices in G_N versus time with *node count*, i.e., *PatrolGRAPH**, *nonuniform random walk* solving MRUFC. Every curve corresponds to a vertex s_i.

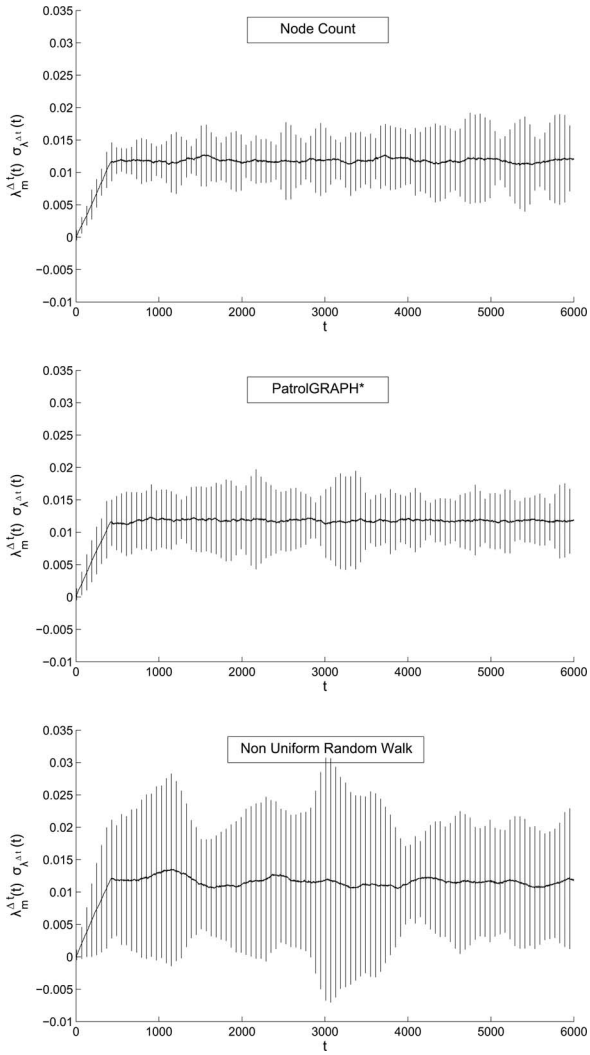

Fig. 5. Free-topology graph. Average visiting rate to vertices in G_N versus time with *node count*, i.e., *PatrolGRAPH**, *nonuniform random walk* solving MRUFC.

1) the number of visits $v_i(t)$ received by every vertex s_i up to time t;
2) the visiting rate $\lambda_i^{\Delta t}(t)$ to every vertex s_i, which is averaged over the last Δt time steps as follows:

$$\lambda_i^{\Delta t}(t) = \frac{v_i(t) - v_i(t - \Delta t)}{\Delta t}. \tag{36}$$

The typical result of a MRUFC simulation run is shown in Figs. 4 and 5, which illustrate an experiment with the randomly generated graph G_N in Fig. 3(b), with $N = 40$ vertices and $M = 5$ robots.

Fig. 4 shows, for every algorithm, the plot of $v_i(t)$ versus t; every curve corresponds to a different vertex s_i, and therefore, the ideal situation is when all curves are "almost straight" lines with the same slope. It can be observed that, in this case, *node count* provides slightly better performance, thanks to the fact that a robot in s_i unrealistically knows how many times neighboring vertices have been visited. In fact, if a vertex temporarily receives more (or fewer) visits, the error is corrected subsequent times. The same does not happen with *PatrolGRAPH**: Since, for this graph, MRUFC has not an exact solution, the prescribed uniform frequency distribution of visits can be only approximately achieved. Finally, it can be observed that in *PatrolGRAPH** the frequency distribution of visits, when averaged over a short period, approximates the stationary vector better than in the *nonuniform random walks*.

Fig. 5 shows, for every algorithm, the plot of the mean value $\lambda_m^{\Delta t}(t)$ of $\lambda_i^{\Delta t}(t)$, averaged over all vertices s_i, and the corresponding standard deviation $\sigma_{\lambda^{\Delta t}}(t)$. The ideal situation is when the *coefficient of variation* $\sigma_{\lambda^{\Delta t}}(t)/\lambda_m^{\Delta t}(t)$ is almost null every t, i.e., the average visiting rate is almost the same for all vertices.

Note that, when MRUFC has an exact solution, the theoretical properties of *PatrolGRAPH** and the *nonuniform random walk* guarantee that the vector λ of the average visiting rates at steady state tends to the uniform distribution, but $\lambda_i^{\Delta t}(t)$ tends to λ_i for every s_i only when Δt is big enough. However, it is important to measure what happens for every t and for smaller Δt intervals. In

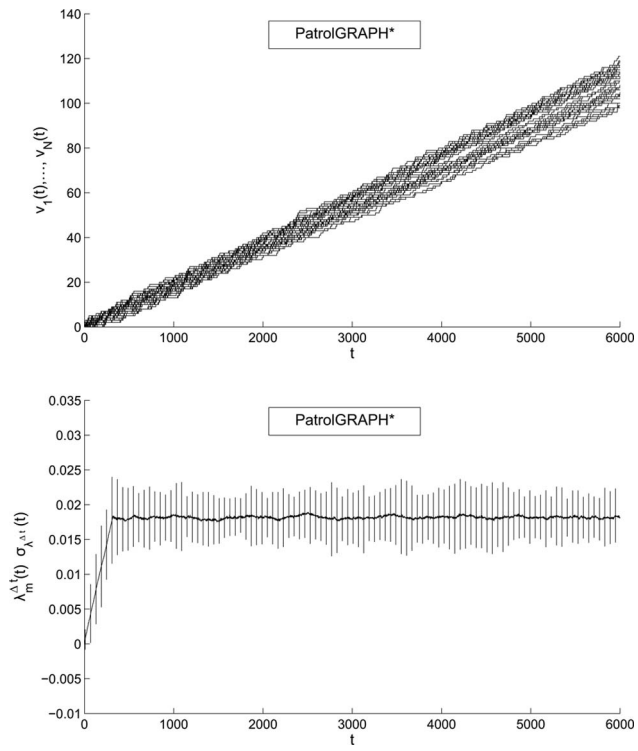

Fig. 6. *PatrolGRAPH** solving MRUFC on a grid-like graph.

the following tests, Δt is chosen in such a way that, on average, every vertex is visited five times in Δt. Informally speaking, if a test returns a coefficient of variation $\approx 1/5$, this means that the number of visits received by a vertex in Δt time steps is included in the range 5 ± 1 most of the time. It is possible to observe that, in this experiment, the *coefficient of variation* is ≈ 0.5 in case of the *node counting* and *PatrolGRAPH**, whereas it increases to ≈ 1 for the *nonuniform random walk*.

Fig. 6 shows the improved behavior of *PatrolGRAPH** in solving MRUFC with $M = 5$ robots when it is executed on a grid-like graph with no local cuts, like the one shown in Fig. 3(a).

Tables II and III summarize the results: *node count*, *PatrolGRAPH**, and the *nonuniform random walk* are executed on 50 randomly generated grid-like graphs and on 50 randomly generated free-topology graphs, and every test is repeated with five, ten, and 20 robots. Each cell contains aggregate information about a set of 50 tests executed with a given *algorithm/graph topology/number of robot*. In particular, the values written in the cells correspond to the 90th percentile of the *coefficient of variation* $\sigma_{\lambda^{\Delta t}}(t)/\lambda_m^{\Delta t}(t)$, i.e., meaning that the *coefficient of variation* has been below that value 90% of the time. It can be observed that

1) all algorithms perform better with grid-like graphs with no local cuts, i.e., the only graph typology for which *MSTC* could ideally be used;
2) *PatrolGRAPH** has only marginally better performance than *node count*; however, the former uses only local information, whereas *node count* requires information about neighboring vertices;

TABLE II
MRUFC ON GRID-LIKE GRAPHS

N. of robots	Node Count	PatrolGRAPH*	Non-Uniform Random Walk
5	0.29	0.27	1.03
10	0.33	0.22	0.86
20	0.28	0.20	0.74

TABLE III
MRUFC ON FREE-TOPOLOGY GRAPHS

N. of robots	Node Count	PatrolGRAPH*	Non-Uniform Random Walk
5	0.66	0.67	1.21
10	0.64	0.52	1.01
20	0.53	0.39	0.84

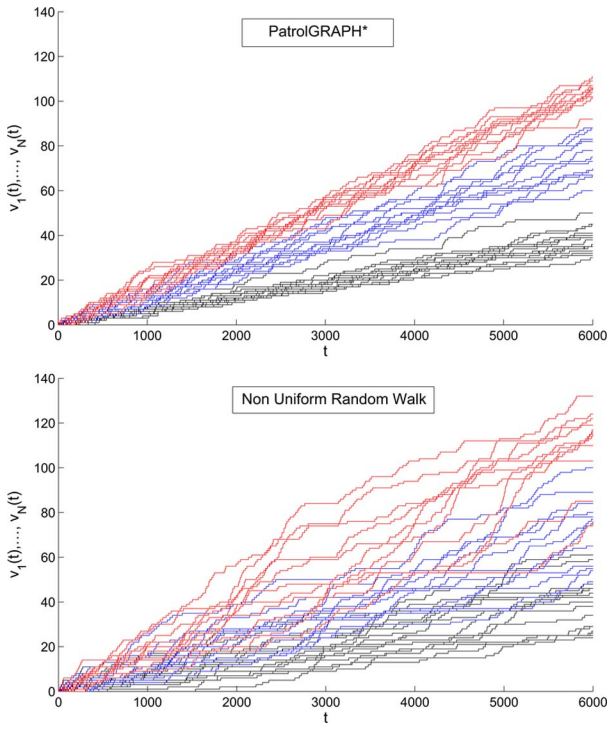

Fig. 7. Free-topology graph. Number of visits $v_1(t),\ldots,v_N(t)$ received by vertices in G_N versus time with *PatrolGRAPH** and the *nonuniform random walk* solving MRCFC. Every curve corresponds to a vertex s_i.

3) *PatrolGRAPH** performs much better than the *nonuniform random walk*.

The capabilities of *PatrolGRAPH** and the *nonuniform random walk* to solve MRCFC are tested by assigning a nonuniform frequency distribution of visits; *node count* is not considered since it cannot solve MRCFC. In particular, in order to better compare the results on different graphs, MRCFC is tackled by considering three "classes" of visiting frequencies: $\Lambda_1, \Lambda_2 = 2\Lambda_1$, and $\Lambda_3 = 3\Lambda_1$, and by randomly assigning every vertex s_i to one of the previous classes. The value $\lambda_m^{\Delta t}(t)$ versus t, as well as the standard deviation $\sigma_{\lambda^{\Delta t}}(t)$, are computed separately for each class. The ideal situation is when the coefficient of variation $\sigma_{\lambda^{\Delta t}}(t)/\lambda_m^{\Delta t}(t)$ is almost null every t for every class.

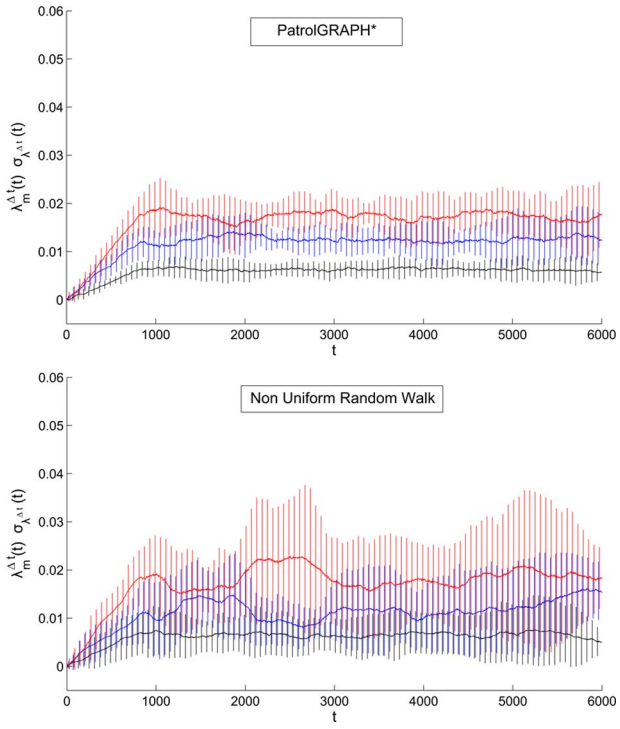

Fig. 8. Free-topology graph. Average visiting rate to vertices in G_N versus time with *PatrolGRAPH** and the *nonuniform random walk* solving MRCFC.

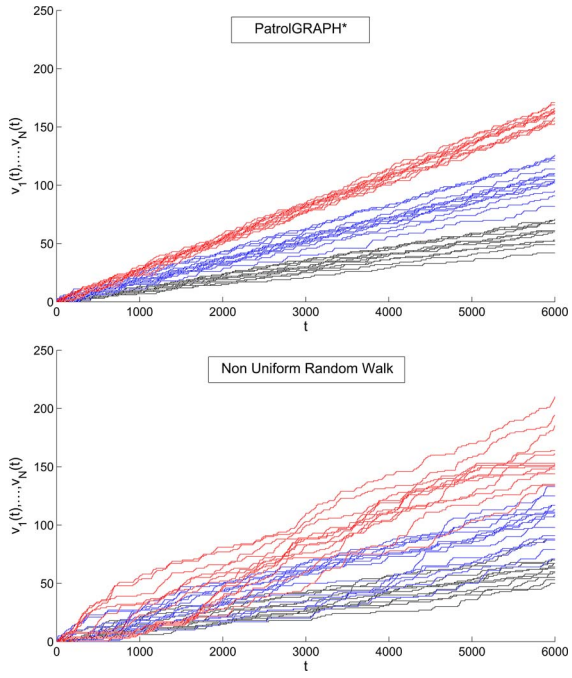

Fig. 9. Grid-like graph. Number of visits $v_1(t), \ldots, v_N(t)$ received by vertices in G_N versus time with *PatrolGRAPH** and the *nonuniform random walk* solving MRCFC. Every curve corresponds to a vertex s_i.

In Fig. 7, the behavior of *PatrolGRAPH** and the *nonuniform random walk*, with $M = 5$ robots, can be analyzed on a free-topology graph, like the one shown in Fig. 3(b). In the case of *PatrolGRAPH**, the three classes of curves with an approximate frequency $\Lambda_1, \Lambda_2 = 2\Lambda_1$, and $\Lambda_3 = 3\Lambda_1$ can be observed. In the case of the *nonuniform random walk,* the behavior is definitely worse.

The mean and the standard deviation of $\lambda_i^{\Delta t}(t)$, which are computed for the three classes Λ_1, Λ_2, and Λ_3 separately, are reported in Fig. 8, for every time step. Again, in the case of *PatrolGRAPH**, the three classes can be easily distinguished and have a lower coefficient a variation than in the case of the *nonuniform random walk*.

Figs. 9 and 10 show the improved behavior of *PatrolGRAPH** in solving MRCFC with five robots when it is executed on a grid-like graph with no local cuts, like the one shown in Fig. 3(a).

Remark 15: The value of $\lambda_m^{\Delta t}$ varies in different experiment, depending on the specific navigation graph considered. This is due to the different distances that the robots must travel between vertices of the graph.

Tables IV and V summarize the results: the maximum coefficient of variation for the three classes is computed every t. The values written in the cells correspond to the 90th percentile of the maximum coefficient of variation $\sigma_{\lambda^{\Delta t}}(t)/\lambda_m^{\Delta t}(t)$, i.e., meaning that the maximum coefficient of variation has been below that value 90% of the time. The same considerations as in the MRUFC case hold about the performance of different algorithms with different graph topologies.

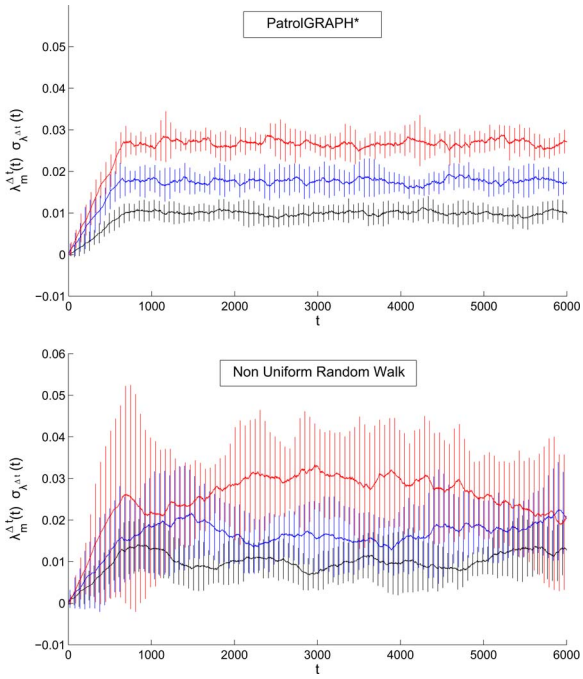

Fig. 10. Grid-like graph. Average visiting rate to vertices in G_N versus time with *PatrolGRAPH** and the *nonuniform random walk* solving MRCFC.

TABLE IV
MRCFC ON GRID-LIKE GRAPHS

N. of robots	Node Count	PatrolGRAPH*	Non-Uniform Random Walk
5	not applicable	0.38	1.09
10	not applicable	0.34	0.93
20	not applicable	0.33	0.84

TABLE V
MRCFC ON FREE-TOPOLOGY GRAPHS

N. of robots	Node Count	PatrolGRAPH*	Non-Uniform Random Walk
5	not applicable	0.78	1.30
10	not applicable	0.68	1.15
20	not applicable	0.54	0.99

VIII. CONCLUSION

This paper introduces the MRCFC problem, in which a team of robots are requested to repeatedly visit a set of predefined locations of the environment according to a specified *frequency distribution*. Furthermore, it describes the *PatrolGRAPH** algorithm, which is capable of solving MRCFC.

The theoretical properties of *PatrolGRAPH** have been demonstrated and validated through simulated experiments. To the authors' knowledge, there are no other algorithms in the literature that are able to solve the general formulation of MRCFC, whereas some algorithms can solve the special MRUFC problem, where all the vertices of the navigation graph must be visited with uniform frequency distribution. A comparison with other algorithms has been performed on a qualitative and quantitative basis, outlining the advantages of *PatrolGRAPH**.

The *PatrolGRAPH** algorithm can be implemented by distributing *smart nodes* with proper characteristics, such as passive (energetically self-sustaining) RFID tags with reduced memory-storage capabilities and communication range. This technology allows the algorithm to achieve desirable properties in terms of *low computational cost*, *local memory*, *local communication*, and *unpredictability*, which can play an important role in real-world implementations.

REFERENCES

[1] Y. Elmaliach, N. Agmon, and G. Kaminka, "Multi-robot area patrol under frequency constraints," in *Proc. IEEE Int. Conf. Robotics Automat.*, Apr. 2007, pp. 385–390.

[2] R. Korf, "Real-time heuristic search," *Artif. Intell.*, vol. 42, no. 2/3, pp. 189–211, 1990.

[3] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *IEEE Trans. Robotics Automat.*, vol. 7, no. 6, pp. 859–865, Dec. 1991.

[4] S. Koenig, B. Szymanski, and Y. Liu, "Efficient and inefficient ant coverage methods," *Ann. Math. Artif. Intell.*, vol. 31, no. 1–4, p. 4176, 2001.

[5] I. Wagner, M. Lindenbaum, and A. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Trans. Robotics Automat.*, vol. 15, no. 5, pp. 918–933, Oct. 1999.

[6] V. Ziparo, A. Kleiner, B. Nebel, and D. Nardi, "RFID-based exploration for large robot teams," in *Proc. IEEE Int. Conf. Robotics Automat.*, Apr. 2007, pp. 4606–4613.

[7] A. Kleiner, J. Prediger, and B. Nebel, "Rfid technology-based exploration and slam for search and rescue," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct., 2006, pp. 4054–4059.

[8] E. Ferranti and N. Trigoni, "Robot-assisted discovery of evacuation routes in emergency scenarios," in *Proc. IEEE Int. Conf. Robotics Automat.*, May, 2008, pp. 2824–2830.

[9] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times*. Providence, RI: Amer. Math. Soc., 2008.

[10] A. F. Karr, "Markov chains and processes with a prescribed invariant measure," *Stochastic Processes Appl.*, vol. 7, no. 3, pp. 277–290, 1978.

[11] S. Ntafos, "Watchman routes under limited visibility," *Comput. Geom. Theory Appl.*, vol. 1, no. 3, pp. 149–170, 1992.

[12] E. Lawler, J. Lenstra, A. R. Kan, and D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. New York: Wiley, 1985.

[13] E. Arkin, S. Fekete, and J. Mitchell. (1997). Approximation algorithms for lawn mowing and milling. Math. Inst., Univ. Köln, Köln, Germany, Tech. Rep. 255. [Online]. Available: ftp.zpr.uni-koeln.de/pub/paper/zpr97-255.ps.gz

[14] H. Choset, "Coverage path planning: The boustrophedon cellular decomposition," in *Proc. Int. Conf. Field Service Robotics*, 1997.

[15] I. Rekleitis, A. P. New, E. S. Rankin, and H. Choset, "Efficient boustrophedon multi-robot coverage: An algorithmic approach," *Ann. Math. Artif. Intell.*, vol. 52, no. 2–4, pp. 109–142, Apr. 2008.

[16] T. Min and H. Yin, "A decentralized approach for cooperative sweeping by multiple mobile robots," in *Proc. IEEE Int. Conf. Intell. Robotics Syst.*, 1998, pp. 380–385.

[17] W. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Proc. IEEE Int. Conf. Robotics Automat.*, vol. 1, 2001, pp. 27–32.

[18] Y. Guo, L. Parker, and R. Madhavan, "A mrobot system for continuous area sweeping tasks," in *Proc. Int. Symp. Collaborative Technol. Syst.*, 2004, pp. 235–240.

[19] M. Ahmadi and P. Stone, "A multi-robot system for continuous area sweeping tasks," in *Proc. IEEE Int. Conf. Robotics Automat.*, May, 2006, pp. 1724–1729.

[20] Z. Butler, A. Rizzi, and C. Butler, "Distributed coverage of rectilinear environments," in *Proc. Workshop Algorithmic Found. Rob.*, Hanover, NH, 2001.

[21] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida, "Cooperative sweeping by multiple mobile robots," in *Proc. IEEE Int. Conf. Robotics Automat.*, 1996, vol. 2, pp. 1744–1749.

[22] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot.," *Ann. Math. Artif. Intell.*, vol. 31, no. 1–4, pp. 77–98, 2001.

[23] Y. Gabriely and E. Rimon, "Competitive on-line coverage of grid environments by a mobile robot," *Comput. Geom. Theory Appl.*, vol. 24, no. 3, pp. 197–224, 2003.

[24] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *Proc. IEEE Int. Conf. Intell. Robotics Syst.*, 2005, pp. 3852–3857.

[25] N. Hazon, F. Mieli, and G. Kaminka, "Towards robust on-line multi-robot coverage," in *Proc. IEEE Int. Conf. Robotics Automat.*, 2006, pp. 1710–1715.

[26] C. Trevai, Y. Fukazawa, H. Yuasa, J. Ota, T. Arai, and H. Asama, "Exploration path generation for multiple mobile robots using reaction-diffusion equation on a graph," *Integr. Comput.-Aided Eng.*, vol. 11, no. 3, pp. 195–212, 2004.

[27] S. S. Ge and C. Fua, "Complete multi-robot coverage of unknown environments with minimum repeated coverage," in *Proc. IEEE Int. Conf. Rob. Autom.*, 2005, pp. 715–720.

[28] C. Luo, S. Yang, and D. Stacey, "Real-time path planning with deadlock avoidance of multiple cleaning robots," in *Proc. IEEE Int. Conf. Robotics Automat.*, vol. 3, 2003, pp. 4080–4085.

[29] A. Marino, L. E. Parker, G. Antonelli, and F. Caccavale, "Behavioral control for multi-robot perimeter patrol: A finite state automata approach," in *Proc. IEEE Int. Conf. Robotics Automat.*, 2009, pp. 831–836.

[30] T. D. Parsons, *Pursuit Evasion in a Graph*. New York: Springer-Verlag, 1976, pp. 426–441.

[31] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM J. Comput.*, vol. 21, no. 5, pp. 863–888, 1992.

[32] L. J. Guibas, J. claude Latombe, S. M. Lavalle, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *Int. J. Comput. Geom. Appl.*, vol. 9, pp. 471–494, 1996.

[33] S. M. Lavalle and J. E. Hinrichsen, "Visibility-based pursuit-evasion: The case of curved environments," , vol. 17, no. 2, pp. 196–201, 1999.

[34] S. Sachs, S. Rajko, and S. M. Lavalle, "Visibility-based pursuit-evasion in an unknown planar environment," *Int. J. Robotics Res.*, vol. 23, no. 1, pp. 3–26, 2004.

[35] N. Agmon, S. Kraus, and G. Kaminka, "Multi-robot perimeter patrol in adversarial settings," in *Proc. IEEE Int. Conf. Robotics Automat.*, May, 2008, pp. 2339–2345.

[36] T. Sak, J. Wainer, and S. Goldenstein, "Probabilistic multiagent patrolling," in *Proc. Adv. Artif. Intell. SBIA 2008, Lecture Notes Comput. Sci.*. vol. 5249, Berlin, Germany: Springer, 2008, pp. 124–133.

[37] F. Amigoni, N. Basilico, and N. Gatti, "Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments," in *Proc. IEEE Int. Conf. Robotics Automat.*, 2009, pp. 819–824.

[38] V. Isler, S. Kannan, and S. Khanna. (2005, Oct.)., "Randomized pursuit-evasion in a polygonal environment," *IEEE Trans. Rob.*, vol. 5, no. 21, pp. 864–875. [Online]. Available: http://www.cs.rpi.edu/∼isler/new/pub/pubs/randpe-tro.pdf.

[39] S. Koenig and R. Simmons, "Easy and hard testbeds for real-time search algorithms," in *Proc. Nat. Conf. Artif. Intell.*, 1996, pp. 279–285.
[40] A. Pirzadeh and W. Snyder, "A unified solution to coverage and search in explored and unexplored terrains using indirect control," in *Proc. IEEE Int. Conf. Robotics Automat.*, vol. 3, May, 1990, pp. 2113–2119.
[41] T. Balch, "Avoiding the past: A simple but effective strategy for reactive navigation," in *Proc. IEEE Int. Conf. Robotics Automat.*, May, 1993, vol. 1, pp. 678–685.
[42] M. Bender, A. Fernández, D. Ron, A. Sahai, and S. Vadhan, "The power of a pebble: Exploring and mapping directed graphs," in *STOC : Proc. 30th ACM Symp. Theory Comput.*. New York: ACM, 1998, pp. 269–278.
[43] D. Xiaotie and A. Mirzaian, "Competitive robot mapping with homogeneous markers," *IEEE Trans. Robotics Automat.*, vol. 12, no. 4, pp. 532–542, Aug. 1996.
[44] I. Wagner, M. Lindenbaum, and A. Bruckstein, "Efficiently searching a graph by a smell-oriented vertex process," *Ann. Math. Artif. Intell.*, vol. 24, no. 1–4, pp. 211–223, 1998.
[45] A. Machado, G. Ramalho, J. Zucker, and A. Drogoul, "Multi-agent patrolling: An empirical analysis of alternative architectures," in *Multi-Agent-Based Simulation II, Lecture Notes Computer Science*. vol. 2581, Berlin, Germany: Springer, 2003, pp. 81–97.
[46] J. Svennebring and S. Koenig, "Trail-laying robots for robust terrain coverage," in *Proc. IEEE Int. Conf. Robotics Automat.*, vol. 1, Sep., 2003, pp. 75–82.
[47] B. Grone and S. Pierce, "Extremal positive semidefinite doubly stochastic matrices," *Linear Algebra Appl.*, vol. 150, pp. 107–117, 1991.
[48] N. Shaked-Monderer and A. Berman, "More on extremal positive semidefinite doubly stochastic matrices," *Linear Algebra Appl.*, vol. 167, pp. 17–34, 1992.
[49] R. Loewy, D. R. Shier, and C. R. Johnson, "Perron eigenvectors and the symmetric transportation polytope," *Linear Algebra Appl.*, vol. 150, pp. 139–155, 1991.
[50] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, 1953.
[51] E. C. MacRae, "Estimation of time-varying Markov processes with aggregate data," *Econometrica*, vol. 45, no. 1, pp. 183–198.
[52] C. M. L. Kelton, "Estimation of time-independent Markov processes with aggregate data: A comparison of techniques," *Econometrica*, vol. 49, no. 2, pp. 517–518.
[53] M. Mamei and F. Zambonelli, "Pervasive pheromone-based interaction with rfid tags," *ACM Trans. Auton. Adapt. Syst.*, vol. 2, no. 2, p. 4, 2007.
[54] M. Kim and N. Chong, "Rfid-based mobile robot guidance to a stationary target," *Mechatronics*, vol. 17, no. 4/5, pp. 217–229, 2007.
[55] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with RFID technology," in *Proc. IEEE Int. Conf. Robotics Automat.*, vol. 1, Apr./May 2004, pp. 1015–1020.
[56] K. Tanaka, Y. Kimuro, K. Yamano, M. M. Hirayama, E. Kondo, and M. Matsumoto, "A Supervised learning approach to robot localization using a short-range RFID sensor," *IEICE Trans. Inf. Syst.*, vol. E90-D, no. 11, pp. 1762–1771, 2007.
[57] Vorwerk Group Website. (Mar. 2010). [Online]. Available: http://www.vorwerk-teppich.de/sc/vorwerk/rfid_en.html
[58] M. Batalin and G. Sukhatme, "The analysis of an efficient algorithm for robot coverage and exploration based on sensor network deployment," in *Proc. IEEE Int. Conf. Robotics Automat.*, pp. 3478–3485, Apr. 2005.
[59] A. Kemppainen, T. Mäkelä, J. Haverinen, and J. Röning, "An experimental environment for optimal spatial sampling in a multi-robot system," in *Proc. 10th Int. Conf. Intell. Autonom. Syst.*, W. Burgard, R. Dillmann, C. Plagemann, and N. Vahrenkamp, Eds., pp. 246–253, 2008.
[60] T. Tammet, J. Vain, A. Kuusik, and A. Puusepp, "Rfid-based communication for a self-organizing robot swarm.," in *Proc. 2nd IEEE Int. Conf. Self-Adapt. Self-Organizing Syst.*, Oct. 2008.
[61] Eu fp6 Roboswarm Project Website. (Mar. 2010). [Online]. Available: http://www.roboswarm.eu
[62] G. Bolch, S. Greiner, H. deMeer, and K. Trivedi, *Queueing Networks and Markov Chains*. Hoboken, NJ: Wiley, 1998.
[63] M. Kijima, *Markov Processes for Stochastic Modeling*. Boston, MA/Boca Raton, FL: Chapman & Hall/CRC, Jan. 1997.
[64] M. Baglietto, G. Cannata, F. Capezio, and A. Sgorbissa, "Multi-robot uniform frequency coverage of significant locations in the environment," in *Proc. Distrib. Auton. Rob. Syst.*, H. Asama, H. Kurokawa, J. Ota, and K. Sekiyama, Eds. Berlin, Germany: Springer-Verlag, 2009, pp. 3–14.
[65] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing Markov chain on a graph," *SIAM Rev.*, vol. 46, no. 4, pp. 667–689, 2004.
[66] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ. Press, Cambridge, U.K., Mar. 2004.

Giorgio Cannata (M'10) received the Laurea degree in electronic engineering from the University of Genova, Genova, Italy, in 1988.

He is currently an Associate Professor of automatic and digital control with the Faculty of Engineering, University of Genova. From 1989 to 1995, he has been a Research Scientist with the Naval Automation Institute, Italian National Research Council, engaged in the area of underwater robotics. From 1995 to 1998, he was an Assistant Professor with the Department of Communication, Computer, and System Sciences, University of Genova. His current research interests include humanoid robots, automatic control systems, control architectures for robotic and mechatronic systems, robotics and robot control theory, control of mechanical systems, and dynamic simulation.

Antonio Sgorbissa received the Laurea degree in electronic engineering in 1996 and the Ph.D. degree in robotics in 2000, both from the University of Genova, Genova, Italy.

From 2001 to 2004, he was a Postdoctoral Fellow with Georgia Tech, Atlanta; the University of Parma, Parma, Italy; and later with the University of Genova. Since 2005, he has been an Assistant Professor with the Department of Communication, Computer, and System Sciences, University of Genova, where he is engaged in teaching ambient intelligence and real-time operating systems with the Faculty of Engineering and geographic information systems and cognitive robotics with the Faculty of Humanities. His current research interests include mobile robotics, multirobot systems, ambient intelligence, and planning, knowledge representation, and machine consciousness.