February 2008

Dynamic Assignment in Distributed Motion Planning With Local Coordination

Michael Zavlanos
University of Pennsylvania

George J. Pappas
University of Pennsylvania, pappasg@seas.upenn.edu

Dynamic Assignment in Distributed Motion Planning With Local Coordination

Abstract

Distributed motion planning of multiple agents raises fundamental and novel problems in control theory and robotics. In particular, in applications such as coverage by mobile sensor networks or multiple target tracking, a great new challenge is the development of motion planning algorithms that dynamically assign targets or destinations to multiple homogeneous agents, not relying on any *a priori* assignment of agents to destinations. In this paper, we address this challenge using two novel ideas. First, distributed multi-destination potential fields are developed that are able to drive every agent to any available destination. Second, nearest neighbor coordination protocols are developed ensuring that distinct agents are assigned to distinct destinations. Integration of the overall system results in a distributed, multiagent, hybrid system for which we show that the mutual exclusion property of the final assignment is guaranteed for almost all initial conditions. Furthermore, we show that our dynamic assignment algorithm will converge after exploring at most a polynomial number of assignments, dramatically reducing the combinatorial nature of purely discrete assignment problems. Our scalable approach is illustrated with nontrivial computer simulations.

Keywords

distributed control, hybrid systems, motion planning, multiagent assignment problems

Dynamic Assignment in Distributed Motion Planning With Local Coordination

Michael M. Zavlanos, *Student Member, IEEE*, and George J. Pappas, *Senior Member, IEEE*

***Abstract*—Distributed motion planning of multiple agents raises fundamental and novel problems in control theory and robotics. In particular, in applications such as coverage by mobile sensor networks or multiple target tracking, a great new challenge is the development of motion planning algorithms that dynamically assign targets or destinations to multiple homogeneous agents, not relying on any *a priori* assignment of agents to destinations. In this paper, we address this challenge using two novel ideas. First, distributed multidestination potential fields are developed that are able to drive every agent to any available destination. Second, nearest neighbor coordination protocols are developed ensuring that distinct agents are assigned to distinct destinations. Integration of the overall system results in a distributed, multiagent, hybrid system for which we show that the mutual exclusion property of the final assignment is guaranteed for almost all initial conditions. Furthermore, we show that our dynamic assignment algorithm will converge after exploring at most a polynomial number of assignments, dramatically reducing the combinatorial nature of purely discrete assignment problems. Our scalable approach is illustrated with nontrivial computer simulations.**

***Index Terms*—Distributed control, hybrid systems, motion planning, multiagent assignment problems.**

I. Introduction

RECENT advances in communication and computation have given rise to distributed control of multiagent systems, which, compared to conventional centralized control, provides increased efficiency, performance, scalability, and robustness. Motivated by these appealing properties of distributed control, we investigate the multiagent assignment problem and propose a distributed and online solution in the absence of any *a priori* assignment information to the system.

Assignment problems are fundamental in combinatorial optimization and, roughly, consist of finding a minimum weight matching in a weighted bipartite graph. They arise frequently in operations research, computer vision, as well as distributed robotics, where graphs are recently emerging as a natural mathematical description for capturing interconnection topology [1]–[11]. Depending on the form of the cost function, assignment problems can be classified as linear or quadratic. Optimal solutions to the linear assignment problem can be computed in polynomial time using the Hungarian algorithm [12]. The quadratic assignment problem, however, is NP-hard [13], and suboptimal solutions are achieved by means of various relaxations. Approaches are either purely discrete [14], [15] or continuous [16], based on the solution of differential equations that always converge to a discrete assignment.

In distributed robotics, the assignment problem naturally arises in settings involving destination or target allocation. Depending on whether the discrete assignment is addressed simultaneously with the continuous navigation strategies or is solved independently in advance, approaches can be either online or offline. An online approach is proposed in [17], where the space of permutation-invariant multirobot formations is represented using complex polynomials whose roots correspond to the configurations of the robots in the formation. The proposed approach is open loop and centralized, since it requires global knowledge of the environment. On the other hand, in [18], a polynomial-time algorithm is developed that computes offline a suboptimal assignment between agents and destinations based on a "minimum distance to the goal" policy.

Under the assumption that the agents have knowledge of all destinations, in this paper, we *simultaneously* address the discrete assignment of destinations to agents as well as the continuous control strategies for driving the individual agents to the destinations. The resulting hybrid controller for each agent consists of both local coordination protocols guaranteeing that distinct destinations are assigned to distinct agents, and multidestination potential fields ensuring convergence of every agent to an available destination, while significantly reducing the complexity of our model. Composition of the hybrid controllers for all agents results in a highly efficient overall system that is illustrated in nontrivial multiagent motion planning tasks. The assignment of destinations to agents is determined dynamically by means of exploration of available destinations and nearest neighbor communication regarding explored destinations [20], while a sensor-based approach [19] is also discussed that solves the problem in the absence of interagent communication. The overall system is shown to almost always converge to an assignment that has the mutual exclusion property and to have at most polynomial complexity, despite the exponential growth of the number of assignments with respect to the number of agents.

The rest of this paper is organized as follows. In Section II, we define the dynamic assignment problem. In Section III, we define the multidestination potential fields and discuss their

Manuscript received November 28, 2006; revised July 6, 2007. This paper was recommended for publication by Associate Editor J. Wen and Editor L. Parker upon evaluation of the reviewers' comments. This work was supported in part by the Autonomous Robots and Mobile Sensors Multidisciplinary University Research Initiative (ARO MURI) under SWARMS Grant W911NF-05-1-0219 and in part by the National Science Foundation (NSF) under Information Technology Research (ITR) Grant 0324977. Preliminary versions of this work can be found in [19] and [20].

The authors are with the General Robotics and Active Sensory Perception (GRASP) Laboratory, Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: zavlanos@grasp.upenn.edu; pappasg@grasp.upenn.edu).

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.2007.913992

convergence properties, while in Section IV we develop the local coordination protocols that consist the hybrid automata that model the agents. Properties of the overall system are discussed in Section V, while in Section VI we illustrate our scalable approach with nontrivial computer simulations.

II. Problem Formulation

Consider n identical agents in $\mathbb{R}^2$ and denote by $x_i(t) \in \mathbb{R}^2$ the coordinates of agent i at time t. We assume fully actuated kinematic agents described by

$$\dot{x}_i(t) = u_i(t) \quad \forall\, i = 1, \ldots, n \tag{1}$$

where $u_i(t)$ is the control vector taking values in $\mathbb{R}^2$. Consider, further, $m \geq n$ destinations and denote by $d_k \in \mathbb{R}^2$ the coordinates of destination k. For any destination k, let

$$\mathcal{B}_r(d_k) \triangleq \{x \in \mathbb{R}^2 \mid \|x - d_k\|_2 < r\} \tag{2}$$

denote an open ball of radius $r > 0$ centered at d_k and define its closure by $[\mathcal{B}_r](d_k) \triangleq \{x \in \mathbb{R}^2 \mid \|x - d_k\|_2 \leq r\}$.

We say that agent i has *reached* destination k if for any given constant $\delta > 0$ there exists a time instant $T_i > 0$ such that $x_i(t) \in [\mathcal{B}_\delta](d_k)$ for all $t > t_0 + T_i$. Let $T = \max_i\{T_i\}$ denote the time instant that every agent has reached a distinct destination. Then, the time instant $t_0 + T$ corresponds to the termination of the motion planning task and indicates a *final assignment* between agents and destinations.

Unlike *centralized* and *offline* approaches that decouple the assignment and navigation problems and focus on designing control laws that drive every agent to a preassigned destination [21], we propose a *dynamic* and fully *distributed* solution to the multiagent assignment problem. In particular, we assume that every agent has knowledge of the positions of all available destinations, while the assignment decision is embedded in its control law. We, therefore, address the following problem.

Problem 1 (Dynamic Assignment): Given n identical agents, $m \geq n$ destinations, and no *a priori* assignment information, derive distributed control laws for every agent i such that, for any $\delta > 0$ and any initial configuration $x_i(t_0)$, there exists a $T > 0$ such that $x_i(t) \in [\mathcal{B}_\delta](d_k)$ for all time $t > t_0 + T$, all agents i, and distinct destinations k.

Implicit in Problem 1 is the *mutual exclusion* property of the final assignment, i.e., that no two agents may be assigned to the same destination. Moreover, since the agents are considered *identical*, any assignment, among the $\binom{m}{n}n!$ possible assignments between agents and destinations, is equally desirable.

The main idea behind our approach is to let every agent explore destinations that it considers *available* and use nearest neighbor communication to propagate information about *taken* destinations in the underlying network.[1] Eventually, a sequence of destinations will be explored by every agent and an assignment will be established with the first available destination to be explored (Fig. 1). In the spirit of analog solutions to combinatorial problems [16], in this paper, we propose novel *multidestination* potential fields that dynamically determine the sequence of destinations to be explored, while driving the agents to their destinations. This approach eliminates any computational complexity that could be introduced in the model by employing deterministic algorithms to determine such a sequence of destinations.

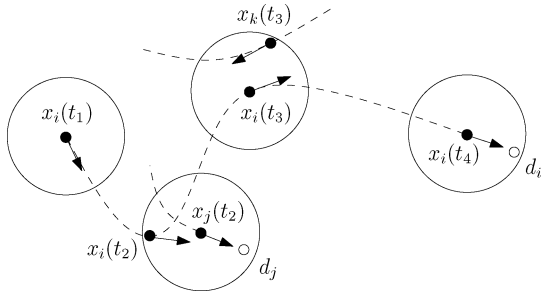

Fig. 1. The following scenario describes the main idea behind our approach. The large circles indicate the communication ranges of the agents. Agent i, initially located at $x_i(t_1)$, explores destination j at time t_2. Since destination j is taken by its current neighbor j, agent i proceeds to exploring other destinations. At time t_3, a neighbor k informs agent i about new taken destinations and agent i proceeds to exploring destination i at time t_4, which is available, and hence, is assigned to agent i.

III. Multidestination Potential Fields

Let $\mathcal{I}_0 = \{1, \ldots, m\}$ denote the index set corresponding to a fixed labeling of the destinations. We assume that every destination $k \in \mathcal{I}_0$ is uniquely associated to a coordinate vector $d_k \in \mathbb{R}^2$ through the injective map $dest : \mathcal{I}_0 \to \mathbb{R}^2$, which is such that,

$$dest(k) \triangleq d_k \quad \forall k \in \mathcal{I}_0. \tag{3}$$

Let $\mathcal{I}_i^a \subseteq \mathcal{I}_0$, with $|\mathcal{I}_i^a| = v \leq m$, denote the set of destinations that agent i considers available[2] and define the distance of agent i to destination $k \in \mathcal{I}_i^a$ by $\gamma_{dk}(x_i) \triangleq \|x_i - d_k\|_2^2$, where $x_i(t) \in \mathbb{R}^2$ denotes the coordinates of agent i at time t. Then, the function,

$$\gamma_v\left(x_i, \mathcal{I}_i^a\right) \triangleq \prod_{k \in \mathcal{I}_i^a} \gamma_{dk}(x_i) \tag{4}$$

is a measure of the distance of agent i to the set $\mathcal{I}_i^a$ consisting of the v destinations that are considered available, since $\gamma_v(x_i, \mathcal{I}_i^a) > 0$ for all $x_i \notin dest(\mathcal{I}_i^a)$ and $\gamma_v(x_i, \mathcal{I}_i^a) = 0$ only if $x_i \in dest(\mathcal{I}_i^a)$. Consider, further, the monotone increasing functions in $[0, \infty)$

$$\sigma(y) \triangleq \frac{y}{1+y} \quad \text{and} \quad \tau_\kappa(y) \triangleq y^\kappa \quad \text{with} \quad \kappa > 0$$

and define the v-destination potential function $\varphi_v \colon \mathbb{R}^2 \to [0, 1]$ by the composition (Fig. 2)

$$\varphi_v\left(x_i, \mathcal{I}_i^a\right) \triangleq \tau_{1/\kappa} \circ \sigma \circ \tau_\kappa \circ \gamma_v\left(x_i, \mathcal{I}_i^a\right). \tag{5}$$

[1] We call a destination *taken* if it is assigned to an agent and *available* otherwise. Note that, in this framework, a taken destination can be considered available by an agent until it is either explored or information to the contrary is provided by its neighbors.

[2] We denote by $|\mathcal{A}|$ the cardinality of the set $\mathcal{A}$.

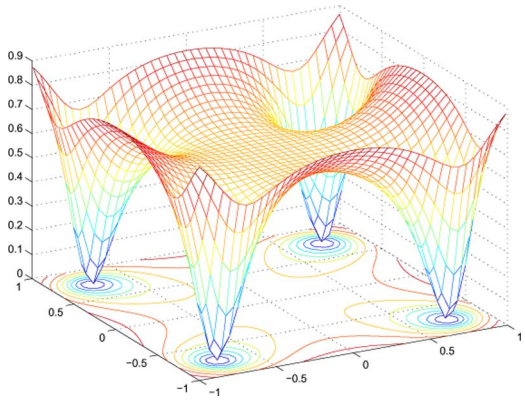

Fig. 2. Plot of the 4-destination potential function $\varphi_4(x_i, \mathcal{I}_i^a)$ for $dest(\mathcal{I}_i^a) = \{[.75\ .75], [-.75\ .75], [-.75\ -.75], [.75\ -.75]\}$.

We now show that $\varphi_v(x_i, \mathcal{I}_i^a)$ is free of local minima. The following proposition enables us to characterize the critical points of $\varphi_v(x_i, \mathcal{I}_i^a)$ by examining the simpler function $\gamma_v(x_i, \mathcal{I}_i^a)$.

Proposition 3.1 ([22]): Let $I_1, I_2 \subseteq \mathbb{R}$ be intervals, γ: $\mathcal{F} \to I_1$ and σ: $I_1 \to I_2$ be analytic. Define the composition φ: $\mathcal{F} \to I_2$ to be $\varphi = \sigma \circ \gamma$. If σ is monotonically increasing on I_1, then the sets of critical points of φ and γ coincide, i.e., $\mathcal{C}_\varphi = \mathcal{C}_\gamma$, and the index of each point is identical, i.e., $index(\varphi)|_{\mathcal{C}_\varphi} = index(\gamma)|_{\mathcal{C}_\gamma}$.

Proposition 3.1 implies that $\varphi_v(x_i, \mathcal{I}_i^a)$ and $\gamma_v(x_i, \mathcal{I}_i^a)$ share identical critical points. In order to characterize the critical points of $\gamma_v(x_i, \mathcal{I}_i^a)$, we make use of *harmonic functions* and their properties [23]. Harmonic functions are completely free of local minima and it is this property that we use to show global convergence of our potential field to the destination set $dest(\mathcal{I}_i^a)$.

Theorem 3.2: For any fixed destination set $\mathcal{I}_i^a$ with $|\mathcal{I}_i^a| = v$, the multidestination control system

$$\dot{x}_i = u_v\left(x_i, \mathcal{I}_i^a\right) \stackrel{\triangle}{=} -K\nabla_{x_i}\varphi_v\left(x_i, \mathcal{I}_i^a\right) \tag{6}$$

with $K > 0$ a positive constant, is globally asymptotically stable almost everywhere.

Proof: Let $\varphi_v(x_i, \mathcal{I}_i^a)$ be a Lyapunov function candidate for the system. Clearly,

$$\dot{\varphi}_v\left(x_i, \mathcal{I}_i^a\right) = -K\left\|\nabla_{x_i}\varphi_v\left(x_i, \mathcal{I}_i^a\right)\right\|_2^2 \leq 0$$

and so all initial conditions converge to the potential function's critical points. We now show that the only local minima are the destination points $dest(\mathcal{I}_i^a)$. By Proposition 3.1, we saw that $\varphi_v(x_i, \mathcal{I}_i^a)$ and $\gamma_v(x_i, \mathcal{I}_i^a)$ share identical critical points. Let $\hat{\gamma}_v(x_i, \mathcal{I}_i^a) \stackrel{\triangle}{=} \log(\gamma_v(x_i, \mathcal{I}_i^a))$. Since the function $\log(\cdot)$ is monotone increasing, applying again Proposition 3.1, we get that $\hat{\gamma}_v(x_i, \mathcal{I}_i^a)$ and $\gamma_v(x_i, \mathcal{I}_i^a)$ share identical critical points too. But the function $\hat{\gamma}_v(x_i, \mathcal{I}_i^a)$ is *harmonic* with respect to the variable x_i since it satisfies the Laplace equation. To see this, observe that, $\hat{\gamma}_v(x_i, \mathcal{I}_i^a) = \log(\prod_{k\in\mathcal{I}_i^a} \gamma_{dk}(x_i)) = \sum_{k\in\mathcal{I}_i^a} \log(\gamma_{dk}(x_i))$ which by Lemma 2.1 in Appendix II, implies that $\hat{\gamma}_v(x_i, \mathcal{I}_i^a)$ is harmonic as a sum of the harmonic functions $\gamma_{dk}(x_i)$. Hence, $\hat{\gamma}_v(x_i, \mathcal{I}_i^a)$ being harmonic implies that it satisfies both the *maximum* and *minimum principle* [23], and so its only critical points in the interior of the free space are nondegenerate saddle points. We conclude that system (6) is globally asymptotically stable except for a set of measure zero critical points (corresponding to the saddle points). ■

Equivalently, Theorem 3.2 implies that for any destination set $\mathcal{I}_i^a \subseteq \mathcal{I}_0$ and any given $\delta > 0$, there exists a destination $k \in \mathcal{I}_i^a$ and a time instant $T_i > 0$ such that $x_i(t) \in [\mathcal{B}_\delta](d_k)$ for all $t > t_0 + T_i$. Thus, the control law (6) guarantees the *necessary* condition $x_i(t) \in [\mathcal{B}_\delta](d_k)$ for an assignment between agent i and destination k, according to Problem 1. The *sufficient* condition for such an assignment is that destination k is *available* and is provided by a distributed coordination framework, based on interagent communication, that we develop in the following section.

IV. Distributed Coordination

Let $\mathcal{I}(t)$ and $\mathcal{I}^c(t)$ denote the index sets of available and taken destinations at time $t \geq t_0$, respectively, where, as before, we call a destination taken if it has been assigned to an agent and available otherwise. Clearly, $\mathcal{I}(t_0) = \mathcal{I}_0, \mathcal{I}^c(t_0) = \emptyset$ and $\mathcal{I}(t) \cap \mathcal{I}^c(t) = \emptyset$, $\mathcal{I}(t) \cup \mathcal{I}^c(t) = \mathcal{I}_0$ for all $t \geq t_0$. Since, in distributed control, the individual agents have no access to the system's global variables, we assume that every agent i is equipped with its own sets of available and taken destinations denoted by $\mathcal{I}_i^a(t)$ and $\mathcal{I}_i^t(t)$, respectively. The variables $\mathcal{I}_i^a(t)$ and $\mathcal{I}_i^t(t)$ are initialized such that every agent considers all destinations in $\mathcal{I}_0$ available, i.e., $\mathcal{I}_i^a(t_0) = \mathcal{I}_0$ and $\mathcal{I}_i^t(t_0) = \emptyset$, while an assignment between agent i and destination $k \in \mathcal{I}_0$ at time t is indicated by $\mathcal{I}_i^a(t) = \{k\}$. On the other hand, as long as agent i has not yet been assigned to a destination, i.e., as long as $|\mathcal{I}_i^a(t)| > 1$, no destination can be considered both available and taken, i.e., $\mathcal{I}_i^a(t) \cap \mathcal{I}_i^t(t) = \emptyset$, while any destination that is not available, has to be taken, i.e., $\mathcal{I}_i^a(t) \cup \mathcal{I}_i^t(t) = \mathcal{I}_0$.

To achieve local coordination among the agents, we further define the set of neighbors of agent i at time t by $\mathcal{N}_i^\epsilon(t) \stackrel{\triangle}{=} \{j \mid x_j(t) \in \mathcal{B}_\epsilon(x_i(t))\}$, where $\mathcal{B}_\epsilon(x_i(t))$ is defined as in (2), and call $\epsilon > 0$ the *coordination range* of agent i. We assume that every agent can exchange information regarding explored taken destinations only with its neighbors in $\mathcal{N}_i^\epsilon(t)$, for all $t \geq t_0$. With the above notation, we now state the assumptions for our model.

Assumptions 4.1: For every agent $i = 1, \ldots, n$, we assume that, for all time $t \geq t_0$,

1) it can be assigned to an available destination $k \in \mathcal{I}(t)$, if $k \in \mathcal{I}_i^a(t)$, $|\mathcal{I}_i^a(t)| > 1$ and $x_i(t) \in \mathcal{B}_\delta(d_k)$,
2) there is a controller $u_v(x_i(t), \mathcal{I}_i^a)$, that for any initial configuration $x_i(t_0)$, any fixed index set $\mathcal{I}_i^a = const.$ and any $\delta > 0$, guarantees that there exists a time instant $T_i > 0$ and a destination $k \in \mathcal{I}_i^a$ such that $x_i(t) \in \mathcal{B}_\delta(d_k)$ for all $t > t_0 + T_i$,
3) $\delta, \epsilon > 0$ are such that $\mathcal{B}_\delta(d_k) \cap \mathcal{B}_\delta(d_l) = \emptyset$ for all $k, l \in \mathcal{I}_0$ and $\epsilon > 2\delta$.

Assumption 4.1 (point 1) implies that the condition $x_i(t) \in \mathcal{B}_\delta(d_k)$ is not sufficient for agent i to be assigned to destination k, since destination k must also be available. Assumption 4.1 (point 2), on the other hand, implies that every agent can

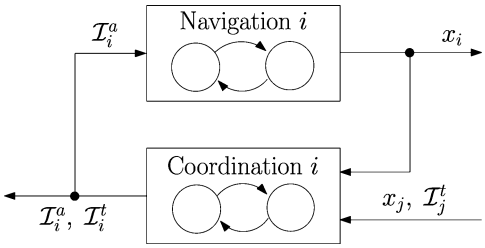

Fig. 3. Hybrid model for agent i.

navigate to any of its available destinations, unless it has already been assigned to one, whence it should always remain in a neighborhood of that destination. Note that the controller proposed in Theorem 3.2 satisfies this assumption. Finally, Assumption 4.1 (point 3) combined with Assumption 4.1 (point 1) implies that every agent can only claim one destination at a time, while combined with Assumption 4.1 (point 2) implies that any agent sufficiently close to a destination can know whether this destination is taken or not.[3]

To resolve *tie breaking* scenarios, where for any destination k, Assumption 4.1 (point 1) is simultaneously satisfied for multiple agents, we require that every agent i can identify the candidate agents, denoted by $\mathcal{C}_i(t)$, requesting to be assigned to destination $k \in \mathcal{I}_0$ at time t, and can also break the tie if necessary. To achieve this specification, we introduce the function $tb : 2^{\mathbb{N}} \backslash \{\emptyset\} \to \mathbb{N}$ such that,

$$tb(A) \stackrel{\triangle}{=} \alpha \in A \tag{7}$$

and assume that every agent is equipped with such a function.[4] Then, the action $tb(\mathcal{C}_i)$, taken by any one of the agents in $\mathcal{C}_i$, can break a tie for destination k, while the outcome can be transmitted to the other neighbors. Note that the set $\mathcal{C}_i$ is common for all agents $j \in \mathcal{C}_i$, by Assumptions 4.1 (point 1) and 4.1 (point 3). The rest of this section is devoted to defining and modeling the distributed coordination framework for the agents, that is according to Assumptions 4.1. Then, in Section V, the overall system is studied and is shown to satisfy Problem 1.

A. Modeling the Agents

To achieve distributed coordination, we propose a hybrid model for every agent [24] that consists of a *navigation* and a *coordination* automaton, as shown in Fig. 3. The navigation automaton receives as an *input* the set of available destinations $\mathcal{I}_i^a(t)$ of agent i and *updates* the state $x_i(t) \in \mathbb{R}^2$ of agent i, while the coordination automaton receives as an *input* the states $x_j(t)$ and sets of taken destinations $\mathcal{I}_j^t(t)$ of all agents, computes agent i's neighbors $\mathcal{N}_i^\epsilon(t)$, and coordinates only with them to *update* the sets $\mathcal{I}_i^a(t)$ and $\mathcal{I}_i^t(t)$.[5] In other words, the coordination automaton uses explicitly nearest neighbor information, hence the distributed nature of the approach. The two automata are synchronized and together consist the model of agent i. The following notion of a *predicate* enables us to formally define the aforementioned automata.

Defination 4.2 (Predicate): Let $X = \{x_1, \ldots, x_n\}$ be a finite set of variables. We define a predicate $\psi(X)$ over X to be a finite conjunction of strict or nonstrict inequalities over X. We denote the set of all predicates over X by $Pred(X)$.

In other words, a predicate is a logical formula. For instance, the predicate $\psi(X) \stackrel{\triangle}{=} (\|x - x_0\|_2 < r)$ over the set of variables $X \in \mathbb{R}^N$ for any $r > 0$, returns 1 if x belongs in the open ball $\|x - x_0\|_2 < r$ and 0 otherwise. Hence, the navigation automaton for agent i can be defined as follows.[6]

Definition 4.3 (Navigation Hybrid Automaton): We define the navigation hybrid automaton of agent i to be the tuple $N_i \stackrel{\triangle}{=} (X_{N_i}, V_{N_i}, E_{N_i}, \Sigma_{N_i}, sync, inv, init, guard, reset, flow)$, where,

1) $X_{N_i} \stackrel{\triangle}{=} \{x_i\}$ denotes the set of owned state variables with $x_i \in \mathbb{R}^2$.
2) $V_{N_i} \stackrel{\triangle}{=} \{1, \ldots, m, Init\}$ denotes the finite set of control modes.
3) $E_{N_i} \stackrel{\triangle}{=} \{(Init, m), (v, v-p),\ \forall\, 0 < p \leq v-1 \mid v \in V_{N_i} \backslash \{1, Init\}\}$ denotes the set of control switches.
4) $\Sigma_{N_i} \stackrel{\triangle}{=} \{update_i\}$ denotes the set of synchronization labels.
5) $sync$: $E_{N_i} \to \Sigma_{N_i}$ with $sync(e) \stackrel{\triangle}{=} update_i$ for all $e \in E_{N_i} \backslash \{(Init, m)\}$, denotes the synchronization map mapping each control switch to a synchronization label.
6) inv: $V_{N_i} \to Pred(X_{N_i})$ with $inv(v) \stackrel{\triangle}{=} true$ for all $v \in V_{N_i}$, denotes the invariant conditions of the hybrid automaton.
7) $init$: $V_{N_i} \to Pred(X_{N_i})$ with $init(v) \stackrel{\triangle}{=} true$ for $v = Init$ denotes the set of initial conditions.
8) $guard$: $E_{N_i} \to Pred(X_{N_i})$ with $guard((Init, m)) \stackrel{\triangle}{=} true$ and $guard\,((v,\ v-p)) \stackrel{\triangle}{=} (|\mathcal{I}_i^a| = v - p)$ for all $v \in V_{N_i} \backslash \{1, Init\}$ and all $0 < p \leq v - 1$, denotes the set of guards of the hybrid automaton.
9) $reset$: $E_{N_i} \to X_{N_i}$ with $x_i := reset(e) \stackrel{\triangle}{=} x_i$ for all $e \in E_{N_i}$, denotes the set of resets associated with the guards of the hybrid automaton.
10) $flow$: $V_{N_i} \to \dot{X}_{N_i}$ with $\dot{x}_i = flow(v) \stackrel{\triangle}{=} u_v(x_i, \mathcal{I}_i^a)$ for $v \in V_{N_i} \backslash \{Init\}$ and $\dot{x}_i = flow(Init) \stackrel{\triangle}{=} 0$, denotes the flow conditions of the hybrid automaton that constrain the first time derivatives of the system variables in mode $v \in V_{N_i}$.

By Definition 4.3, for any automaton N_i, we see that $|\mathcal{I}_i^a| = v$ for all $v \in V_{N_i}$. Hence, every mode of N_i corresponds to a distinct number v of available destinations for agent i. While automaton N_i is in mode $|\mathcal{I}_i^a| = v$, control law (6) "selects" a destination in $\mathcal{I}_i^a$ to drive agent i to, as discussed in Sections II and III. On the other hand, transitions in N_i are triggered whenever the

[3]The later property is due to $\epsilon > 2\delta$, which implies that the coordination range of the agents is larger than the diameter of the ball $\mathcal{B}_\delta(d_k)$ around any destination k.

[4]Note that any tie breaking policy can be used, deterministic or random.

[5]Technically, the model we propose for the agents does not correspond to an input/output hybrid automaton [25], but to a composition of synchronized automata. In our framework, the terms *input* and *output* are used for presentation purposes, exclusively.

[6]To simplify notation, we drop the dependence of the state variables on time. Moreover, in what follows, ":=" indicates a *transition reset* [24].

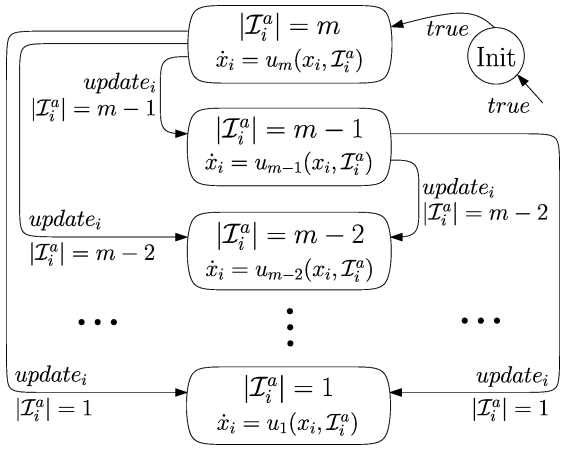

Fig. 4. Navigation automaton for agent i.

set of available destinations $\mathcal{I}_i^a$ is updated. Such updates can either take place because a *free* destination has been discovered or because information about *taken* destinations has been received from agent i's neighbors $\mathcal{N}_i^\epsilon$. Note, however, that every such transition $v \xrightarrow{e} v'$ results in $v' < v$, and so, eventually, $v = 1$, which indicates an assignment for agent i. Note also that these transitions are synchronized with transitions of the coordination automaton due to synchronization labels $sync(e) = update_i$. Fig. 4 shows the graph representation of hybrid automaton N_i.

In the following, we define the coordination automaton for agent i. The coordination automaton is designed to continuously compute agent i's neighbors $\mathcal{N}_i^\epsilon$, while the coordination mechanism uses nearest neighbor information and dictates how agent i should update its state variables $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$, when it is close to an available destination, when it is close to a taken destination, when it has been assigned to a destination, and when it is far from any destination.

Definition 4.4 (Coordination Hybrid Automaton): We define the coordination hybrid automaton of agent i to be the tuple $C_i \triangleq (X_{C_i}, V_{C_i}, E_{C_i}, \Sigma_{C_i}, sync, inv, init, guard, reset, flow)$, where,

1) $X_{C_i} \triangleq \{\mathcal{I}_i^a, \mathcal{I}_i^t, \mathcal{N}_i^\epsilon, \mathcal{C}_i\}$ denotes the set of owned state variables with $\mathcal{I}_i^a, \mathcal{I}_i^t \in 2^{\mathcal{I}_0}$ and $\mathcal{N}_i^\epsilon, \mathcal{C}_i \in 2^{\{1,\ldots,n\}}$.
2) $V_{C_i} \triangleq \{Init, N, I, U, O_k, A_k, T_k, B_k, R_k \mid k \in \mathcal{I}_0\}$ denotes the finite set of control modes.[7]
3) $E_{C_i} \triangleq \{(Init, N), (N, I), (I, N), (I, U), (U, N), (N, O_k), (O_k, N), (N, A_k), (A_k, T_k), (A_k, B_k), (T_k, N), (B_k, R_k), (R_k, N) \mid k \in \mathcal{I}_0\}$, denotes the set of control switches.
4) $\Sigma_{C_i} \triangleq \{update_i, tiebreak_k \mid k \in \mathcal{I}_0\}$ denotes the set of synchronization labels.
5) $sync$: $E_{C_i} \to \Sigma_{C_i}$ with,
   a) $sync((A_k, B_k)) \triangleq tiebreak_k$ for all $k \in \mathcal{I}_0$,
   b) $sync(e) \triangleq update_i$, for $e = (U, N)$, (T_k, N), (R_k, N).

   denotes the synchronization map mapping each control switch to a synchronization label.
6) inv: $V_{C_i} \to Pred(X_{C_i})$ with $inv(v) \triangleq true$ for all $v \in V_{C_i}$, denotes the invariant conditions of the hybrid automaton.
7) $init$: $V_{C_i} \to Pred(X_{C_i})$ with $init(v) \triangleq true$ for $v = Init$, denotes the set of initial conditions.
8) $guard$: $E_{C_i} \to Pred(X_{C_i})$ with,
   a) $guard((N, I)) \triangleq (x_i \notin \cup_{l \in \mathcal{I}_0} \mathcal{B}_\delta(d_l)) \vee (x_i \in \mathcal{B}_\delta(d_k) \wedge \mathcal{I}_i^a \neq \{k\} \wedge k \in \cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t)$, for all $k \in \mathcal{I}_0$,
   b) $guard((I, N)) \triangleq (\mathcal{I}_i^a \cap (\cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t) = \emptyset)$,
   c) $guard((I, U)) \triangleq (\mathcal{I}_i^a \cap (\cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t) \neq \emptyset)$,
   d) $guard((N, O_k)) \triangleq (x_i \in \mathcal{B}_\delta(d_k) \wedge \mathcal{I}_i^a = \{k\})$ for all $k \in \mathcal{I}_0$,
   e) $guard((N, A_k)) \triangleq (x_i \in \mathcal{B}_\delta(d_k) \wedge \mathcal{I}_i^a \neq \{k\} \wedge k \notin \cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t)$ for all $k \in \mathcal{I}_0$,
   f) $guard((A_k, T_k)) \triangleq (\mathcal{C}_i = \emptyset)$ for all $k \in \mathcal{I}_0$,
   g) $guard((A_k, B_k)) \triangleq (\mathcal{C}_i \neq \emptyset)$ for all $k \in \mathcal{I}_0$,
   h) $guard(e) \triangleq true$, otherwise,

   denotes the set of guards of the hybrid automaton.
9) $reset$: $E_{C_i} \to X_{C_i}$ with, $[\mathcal{I}_i^a\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i] := reset(e)$ such that,
   a) $reset((Init, N)) \triangleq [\mathcal{I}_0\ \emptyset\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \emptyset]$,
   b) $reset((N, I)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i]$,
   c) $reset((I, N)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \mathcal{C}_i]$,
   d) $reset((I, U)) \triangleq [\mathcal{I}_i^a \backslash (\cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t)\ \mathcal{I}_i^t \cup (\cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t)\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i]$,
   e) $reset((U, N)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \mathcal{C}_i]$,
   f) $reset((N, O_k)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t \cup (\cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t)\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i]$,
   g) $reset((O_k, N)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \mathcal{C}_i]$,
   h) $reset((N, A_k)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t \cup \{k\} \cup (\cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t)\ \mathcal{N}_i^\epsilon\ \{j \in \mathcal{N}_i^\epsilon \mid x_j \in \mathcal{B}_\delta(d_k)\} \cup \{i\}]$,
   i) $reset((A_k, T_k)) \triangleq [\{k\}\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i]$,
   j) $reset((A_k, B_k)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ tb(\mathcal{C}_i)]$, if $i = \min\{\mathcal{C}_i\}$ and $reset((A_k, B_k)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i]$, if $i \neq \min\{\mathcal{C}_i\}$,
   k) $reset((T_k, N)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \mathcal{C}_i]$,
   l) $reset((B_k, R_k)) \triangleq [\{k\}\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i]$, if $i \in \mathcal{C}_i$ and $reset((B_k, R_k)) \triangleq [\mathcal{I}_i^a \backslash \mathcal{I}_i^t\ \mathcal{I}_i^t\ \mathcal{N}_i^\epsilon\ \mathcal{C}_i]$, if $i \notin \mathcal{C}_i$,
   m) $reset((R_k, N)) \triangleq [\mathcal{I}_i^a\ \mathcal{I}_i^t\ \{j \mid x_j \in \mathcal{B}_\epsilon(x_i)\}\ \mathcal{C}_i]$,

   for all $k \in \mathcal{I}_0$, denotes the set of resets associated with the guards of the hybrid automaton.
10) $flow$: $V_{C_i} \to \dot{X}_{C_i}$ with $[\dot{\mathcal{I}}_i^a\ \dot{\mathcal{I}}_i^t\ \dot{\mathcal{N}}_i^\epsilon\ \dot{\mathcal{C}}_i] = flow(v) \triangleq [0\ 0\ 0\ 0]$ for all $v \in V_{C_i}$, denotes the flow conditions of the hybrid automaton that constrain the first time derivatives of the system variables in mode $v \in V_{C_i}$.

[7]The shorthand notation stands for $I \triangleq$ *New Info*, $N \triangleq$ *Neighbors*, $U \triangleq$ *Update*, $O_k \triangleq$ *Dest k Owned*, $T_k \triangleq$ *Dest k Taken*, $A_k \triangleq$ *Dest k Available*, $B_k \triangleq$ *Tie Break k*, and $R_k \triangleq$ *Tie k Resolved*.

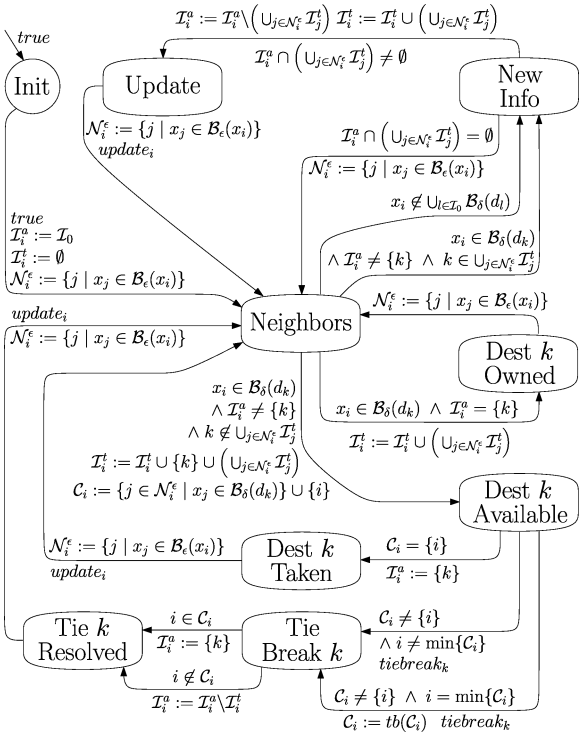

Fig. 5. Coordination automaton for agent i.

Observing Definition 4.4 we see that whenever agent i is sufficiently close to an available destination k, automaton C_i transitions to mode A_k and the set of taken destinations $\mathcal{I}_i^t$ is updated with new information from neighbors, according to $reset((N, A_k))$. If there is no need for *tie breaking* or if agent i wins the tie break, destination k is assigned to agent i, as indicated by the resets $reset((A_k, T_k))$ and $reset((B_k, R_k))$, respectively. On the other hand, if agent i loses the tie break, then $\mathcal{I}_i^a$ is updated by removing any new taken destinations, according to $reset((B_k, R_k))$. Now, if agent i is close to a taken destination or if it is far from any destination, automaton C_i transitions to mode I and exchanges information with its neighbors in order to update the sets of available and taken destinations $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$, according to the resets $reset((I, N))$ and $reset((U, N))$. Note that whenever the state variable $\mathcal{I}_i^a$ is updated with new information, a transition is automatically triggered in automaton N_i due to synchronization labels "$update_i$." This synchronization models the communication between automata C_i and N_i. Similarly, in a case of a *tie* for destination k, all the involved coordination automata are synchronized according to the synchronization labels "$tiebreak_k$" to participate in a tie break where the agent with the smallest label is responsible for breaking the tie, according to $reset((A_k, B_k))$. Fig. 5. shows the graph representation of hybrid automaton C_i.

Remark 4.5 (Coordination Radius): Clearly, the larger the coordination radius $\epsilon > 0$, the faster information about taken destinations propagates in the network. In particular, if $\epsilon > 2R$, where $R \stackrel{\triangle}{=} \inf_{r>0}\{\mathcal{W} \subseteq \mathcal{B}_r\}$ is the radius of the smallest ball containing the workspace $\mathcal{W} \subseteq \mathbb{R}^2$, then information in the network is *global*. On the other hand, if $\epsilon = 0$, then no information exchange among the agents can occur. To achieve coordination, in this case, we need to assume that the agents can sense the presence of other agents within a neighborhood of radius $\epsilon_s > 0$. We call $\epsilon_s > 0$ the *sensing range* of the agents and require that $\epsilon_s > 2\delta$. Fig. 6 shows the graph representation of the corresponding hybrid automaton. Note that the condition $\epsilon_s > 2\delta$ guarantees that if $x_i(t) \in \mathcal{B}_\delta(d_k)$ and there exists an agent $j \neq i$ such that $x_j(t) \in \mathcal{B}_\delta(d_k)$, then $j \in \mathcal{F}_i(t)$, where $\mathcal{F}_i(t) \stackrel{\triangle}{=} \{j \mid x_j(t) \in \mathcal{B}_\delta(d_k)\}\backslash\{i\}$ denotes the set of agents that are in a δ-neighborhood of destination k. If $\mathcal{F}_i(t) = \emptyset$, then the destination is *free*, while if $\mathcal{F}_i(t) \neq \emptyset$, then the destination is *taken*. Hence, agent i can sense whether a destination is taken or not. Note also that transitions in N_i can only be of the form $v \xrightarrow{e} v - 1$ or $v \xrightarrow{e} 1$, and that in the absence of communication, tie breaking scenarios cannot be dealt with. For more information on sensor-based dynamic assignment problems, we refer the reader to [19].

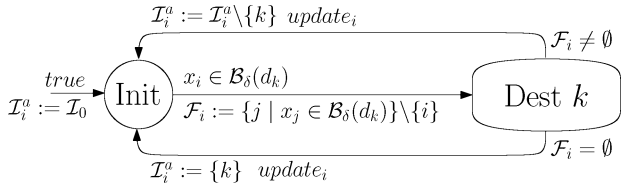

Fig. 6. Sensing automaton for agent i.

V. Integration of the Overall System

Having defined the models for the agents, we now proceed with their composition in a product system [24] and study the properties of the overall distributed coordination scheme.

Definition 5.1 (Product System): We define the product of the hybrid automata $N_1, \ldots, N_n, C_1, \ldots, C_n$ by the tuple $S \stackrel{\triangle}{=} (X_S, V_S, E_S, \Sigma_S, sync, inv, init, guard, reset, flow)$, where,

1) $X_S \stackrel{\triangle}{=} X_{N_1} \cup \cdots \cup X_{N_n} \cup X_{C_1} \cup \cdots \cup X_{C_n}$ denotes the set of state variables.
2) $V_S \stackrel{\triangle}{=} V_{N_1} \times \cdots \times V_{N_n} \times V_{C_1} \times \cdots \times V_{C_n}$ denotes the finite set of control modes.
3) $E_S \stackrel{\triangle}{=} \{e_S\}$ denotes the set of control switches such that,
   a) $e_S = e_{N_i} \| e_{C_i} \in E_S$ is defined as the set of control switches of S corresponding to control switches $e_{N_i} \in E_{N_i}$, and $e_{C_i} \in E_{C_i}$, with $sync(e_{N_i}) = sync(e_{C_i}) = update_i$,
   b) $e_S = \|_{i\in\mathcal{J}} e_{C_i} \in E_S$ is defined as the set of control switches of S corresponding to control switches $e_{C_i} \in E_{C_i}$, with $sync(e_{C_i}) = tiebreak_k$, for all $k \in \mathcal{I}_0$ and all $i \in \mathcal{J} \subseteq \{1, \ldots, n\}$, $\mathcal{J} \neq \emptyset$,
   c) $e_S = (v_S, v_S') \in E_S$ with $v_{C_i} = N$ and $v_{C_i}' = U, O_k, T_k, A_k, B_k$ for all $k \in \mathcal{I}_0$.

   Hence, the only variables that change with every transition $v_S \xrightarrow{e_S} v_S'$ are the ones involved in the control switch e_S through the corresponding automata.
4) $\Sigma_S \stackrel{\triangle}{=} \Sigma_{N_1} \cup \cdots \cup \Sigma_{N_n} \cup \Sigma_{C_1} \cup \cdots \cup \Sigma_{C_n}$ denotes the set of synchronization labels.
5) $sync$: $E_S \to \Sigma_S$ denotes the synchronization map mapping each control switch to a synchronization label.

6) inv: $V_S \to Pred(X_S)$ with $inv(v_S) \triangleq inv(v_{N_1}) \wedge \cdots \wedge inv(v_{N_n}) \wedge inv(v_{C_1}) \wedge \cdots \wedge inv(v_{C_n})$ for all $v_S \in V_S$, denotes the invariant conditions of the product automaton.
7) $init$: $V_S \to Pred(X_S)$ with $init(v_S) \triangleq init(v_{N_1}) \wedge \cdots \wedge init(v_{N_n}) \wedge init(v_{C_1}) \wedge \cdots \wedge init(v_{C_n})$ for all $v_S \in V_S$, denotes the set of initial conditions.
8) $guard$: $E_S \to Pred(X_S)$ such that for any $\mathcal{J} \subseteq \{1, \ldots, n\}$, $\mathcal{J} \neq \emptyset$,
   a) $guard(e_S) \triangleq guard(e_{N_i}) \wedge guard(e_{C_i})$, if $e_S = e_{N_i} \| e_{C_i} \in E_S$,
   b) $guard(e_S) \triangleq \wedge_{i \in \mathcal{J}} guard(e_{C_i})$ for all $k \in \mathcal{I}_0$ and all $i \in \mathcal{J}$, if $e_S = \|_{i \in \mathcal{J}} e_{C_i} \in E_S$,
   c) $guard(e_S) \triangleq guard((v_{C_i}, v'_{C_i}))$ for all $e_S = (v_S, v'_S) \in E_S$ with $v_{C_i} = N$ and $v'_{C_i} = U, O_k, T_k, A_k, B_k$,

   denotes the set of guards (or transitions) of the hybrid automaton.
9) $reset$: $E_S \to X_S$ such that for any $\mathcal{J} \subseteq \{1, \ldots, n\}$, $\mathcal{J} \neq \emptyset$,
   a) $reset(e_S) \triangleq reset(e_{N_i}) \wedge reset(e_{C_i})$, if $e_S = e_{N_i} \| e_{C_i} \in E_S$,
   b) $reset(e_S) \triangleq \wedge_{i \in \mathcal{J}} reset(e_{C_i})$ for all $k \in \mathcal{I}_0$ and all $i \in \mathcal{J}$, if $e_S = \|_{j \in \mathcal{J}} e_{C_i} \in E_S$,
   c) $reset(e_S) \triangleq reset((v_{C_i}, v'_{C_i}))$ for all $e_S = (v_S, v'_S) \in E_S$ with $v_{C_i} = N$ and $v'_{C_i} = U, O_k, T_k, A_k, B_k$,

   denotes the set of resets associated with the guards of the hybrid automaton.
10) $flow$: $V_S \to \dot{X}_S$ with $flow(v_S) \triangleq flow(v_{N_1}) \cup \cdots \cup flow(v_{N_n}) \cup flow(v_{C_1}) \cup \cdots \cup flow(v_{C_n})$ for all $v_S \in V_S$, denotes the flow conditions of the hybrid automaton that constrain the first time derivatives of the system variables in mode v_S.

Clearly, the product system S, being the composition of all elementary automata C_i and N_i, models the interconnection between them. Hence, studying S, we can identify the properties of the whole multiagent system. The following result characterizes the transition guards in S.[8]

Proposition 5.2: For any agent i and any destination $k \in \mathcal{I}_0$ such that $x_i \in \mathcal{B}_\delta(d_k)$ and $\mathcal{I}_i^a \neq \{k\}$, the product system S has the following properties:

a) $k \notin \cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t$ if and only if destination k is available.
b) $k \in \cup_{j \in \mathcal{N}_i^\epsilon} \mathcal{I}_j^t$ if and only if destination k is taken.

Proposition 5.2 implies that the product system S can always identify whether a destination is available or taken using information from its nearest neighbors. We now proceed to showing that under the product system S, agent i is always assigned to an available destination if it is sufficiently close to it, while it appropriately updates its sets of available and taken destinations, $\mathcal{I}_i^a$ and $\mathcal{I}_i^t$ respectively, otherwise.

Proposition 5.3: For any agent i, any destination $k \in \mathcal{I}_0$, and all time t, the product system S has the following properties:

a) If $x_i(t) \in \mathcal{B}_\delta(d_k)$ and destination k is available at time t, then $\mathcal{I}_i^a(t) := \{k\}$ and $\mathcal{I}_i^t(t) := \mathcal{I}_i^t(t) \cup \{k\} \cup (\cup_{j \in \mathcal{N}_i^\epsilon(t)} \mathcal{I}_j^t(t))$.
b) If destination k is available at time t and $x_i(t) \in \mathcal{B}_\delta(d_k)$ simultaneously for multiple agents i, then S is able to break the tie.
c) $\mathcal{I}_i^a(t) := \mathcal{I}_i^a(t) \backslash (\cup_{j \in \mathcal{N}_i^\epsilon(t)} \mathcal{I}_j^t(t))$ and $\mathcal{I}_i^t(t) := \mathcal{I}_i^t(t) \cup (\cup_{j \in \mathcal{N}_i^\epsilon(t)} \mathcal{I}_j^t(t))$ otherwise.

Note that Proposition 5.3 further implies that for all time $t \geq t_0$ such that $|\mathcal{I}_i^a(t)| > 1$, we have that $\mathcal{I}_i^a(t) \cap \mathcal{I}_i^t(t) = \emptyset$ and $\mathcal{I}_i^a(t) \cup \mathcal{I}_i^t(t) = \mathcal{I}_0$. Hence, the construction of our model is consistent with the system requirements in Section IV. The following proposition shows that every agent that has not yet been assigned to a destination, has always knowledge of at least all available destinations in $\mathcal{I}(t)$. This property of the product system S is necessary to show that every agent will eventually be assigned to a distinct destination in $\mathcal{I}_0$.

Proposition 5.4: The product system S guarantees that $\mathcal{I}(t) \subseteq \mathcal{I}_i^a(t)$ for all time t and all agents i with $|\mathcal{I}_i^a(t)| > 1$.

Our next result concerns the running time of the hybrid system S. In particular, we show that the product system S in the worst case can only take a finite number of transitions $v_S \xrightarrow{e_S} v'_S$ such that $sync(e_S) = update_i$, which is polynomial with respect to the number of agents n. Such transitions are triggered whenever $\mathcal{I}_i^a$ is updated for any agent i, and every time they result in $v_{N_i} = 1$, a destination is assigned to agent i, while every time they result in $v_{N_i} > 1$, information about taken destinations has been received by agent i and the set of available destinations $\mathcal{I}_i^a$ has been appropriately updated. Clearly, many other transitions may occur in the meanwhile, but as long as these transitions are polynomial with the number of agents n, it is guaranteed that the explored assignments are also polynomial with n. This result is important, given that the number of assignments, and hence, the space of control modes V_S of the product system S, grows exponentially with the number of agents.

Proposition 5.5: Let $v_S^\star \triangleq (v_{N_1}^\star, \ldots, v_{N_n}^\star, v_{C_1}, \ldots, v_{C_n})$ be such that $v_{N_i}^\star = 1$ and $\mathcal{I}_i^a \cap \mathcal{I}_j^a = \emptyset$ for all $j \neq i$. Then, initialized at v_S^0, the product system S can reach $v_S^\star$ in at most $n(n+1)/2$ transitions $v_S \xrightarrow{e_S} v'_S$ such that $sync(e_S) = update_i$, where n is the number of automata N_i.

Having shown that the product system S satisfies the problem specifications and has also reasonable complexity, we now show that it also has the desired *liveness* and *safety* properties. In other words, we show that every agent will eventually be assigned to a destination in the set $\mathcal{I}_0$ and that no two agents will be assigned to the same destination. We hence, have the following theorem.

Theorem 5.6: For almost all initial conditions $x_i(t_0)$, there exists a constant $T > 0$ such that for all time $t > t_0 + T$, the product system S is in mode $v_S^\star = (v_{N_1}^\star, \ldots, v_{N_n}^\star, v_{C_1}, \ldots, v_{C_n})$ with $v_{N_i}^\star = 1$ and $\mathcal{I}_i^a(t) \cap \mathcal{I}_j^a(t) = \emptyset$ for all $j \neq i$. We call $v_S^\star$ the equilibrium mode of the system.

VI. Simulation Results

We consider a navigation task where $n = 50$ agents, starting from randomly chosen initial configurations, have to reach the

[8]Proofs for this and the other results in this section can be found in Appendix I.

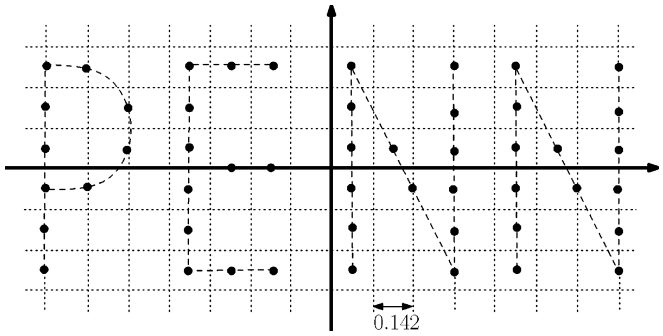

Fig. 7. Destination set $dest(\mathcal{I}_0)$.

destination set $dest(\mathcal{I}_0)$ in Fig. 7 consisting of $m = 50$ destinations. We apply both the communication-based and sensor-based coordination protocols that we have developed and observe that both approaches succeed in determining a valid assignment for the agents (Fig. 8).

Fig. 8 shows the evolution of the system at four consecutive time instants for the communication-based and sensor-based coordination protocols, respectively. The destinations are denoted with small circles and the δ-neighborhoods (with $\delta = 0.05$) around each destination with big circles. The agents, on the other hand, are denoted with dots and their ϵ-coordination ranges (with $\epsilon = 0.1$) with circles around each agent. Observe that in both cases the hybrid system S eventually drives every agent to a distinct destination. Note also the paths followed by the agents until they reach their final destinations (Fig. 8(g)–(i)). In the communication-based case, the agents switch to exploring new destinations as soon as they receive information about taken destinations from their neighbors (Fig. 8(g)). On the other hand, in the sensor-based case they explicitly visit destinations in order to determine whether they are taken or not (Fig. 8(h) and (i)). By comparing the communication-based and sensor-based protocols in each one of the pairs 8(a) and (b), 8(c) and (d), 8(e) and (f), and 8(g) and (h), we see that the communication-based coordination is much more efficient, as expected.

Remark 6.1 (Collision Avoidance): Note that the proposed framework allows overlapping among the agents. Combining the multidestination potential fields in Section III with repulsive potentials that guarantee collision avoidance could make convergence analysis a challenging task. To avoid such complications, we can exploit our modeling framework, which naturally allows minor modifications to be made in order to incorporate various secondary objectives. In particular, lifting the selection of the sequence of destinations to be visited by an agent from the navigation automaton to the hierarchically higher coordination automaton, simplifies the continuous navigation controllers (which now become: "drive an agent to a single prespecified destination") and enables us to account for collision avoidance using controllers from the literature [26]. Alternatively, "traffic-based" rules for collision avoidance, such as yielding to the agent on the right, can be directly implemented in the proposed hybrid model for every agent.

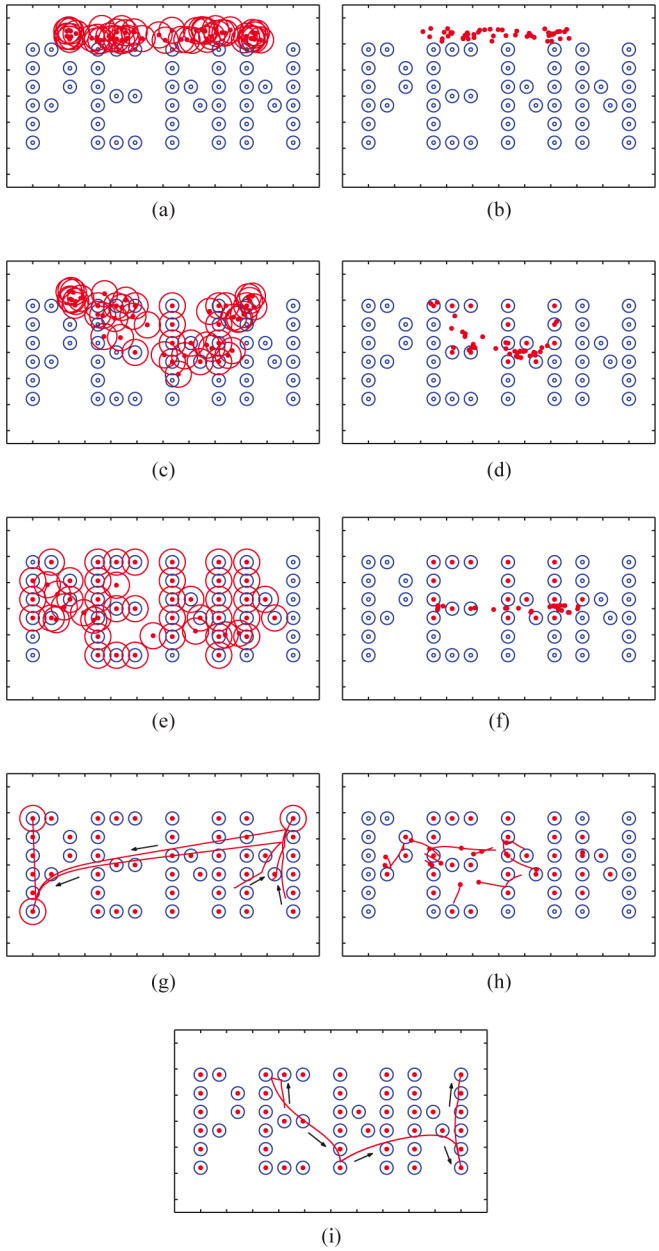

Fig. 8. Distributed assignment for $n = 50$ agents. Fig. 8(a), 8(c), 8(e), 8(g), and Fig. 8(b), 8(d), 8(f), 8(h), 8(i) depict consecutive stages of the communication-based and sensor-based protocols, respectively. Stages in each of the pairs 8(a) and 8(b), 8(c) and 8(d), 8(e) and 8(f) and 8(g) and 8(h) are taken at the same time instants. Note the faster convergence of the communication-based protocol.

VII. Conclusion

In this paper, we considered a distributed hybrid approach to the assignment problem in distributed motion planning that simultaneously addresses the discrete assignment of destinations to agents as well as the continuous control strategies for driving the individual agents to the destinations. The assignment was determined dynamically through distributed coordination protocols, while navigation of the agents to any of the available destinations was guaranteed for almost all initial conditions by novel multidestination potential fields that were also shown to

reduce significantly the complexity of the model. The overall hybrid system was shown to always guarantee the mutual exclusion property of the final assignment and have at most polynomial complexity, despite the exponential growth of the number of assignments with respect to the number of agents. Our scalable approach was verified through nontrivial computer simulations. Future work involves experimenting with the proposed framework and dealing with implementation issues such as collision avoidance and noisy measurements.

APPENDIX I

A. Proof of Proposition 5.2

To show property (a), note that if destination k is available, then, $k \notin \mathcal{I}_j^t$, and so $\mathcal{I}_j^a \neq \{k\}$ for all j, which shows the "only if" part of the claim. On the other hand, for any agent i such that $x_i \in \mathcal{B}_\delta(d_k)$ and $\mathcal{I}_i^a \neq \{k\}$, Assumption 4.1 (point 2) and the fact that $\epsilon > 2\delta$ guarantee that if there exists an agent $j \neq i$ such that $\mathcal{I}_j^a = \{k\}$, then $j \in \mathcal{N}_i^\epsilon$. Thus, if $k \notin \cup_{j\in\mathcal{N}_i^\epsilon} \mathcal{I}_j^t$, destination k has to be available, which shows the "if" part of the claim.

To show property (b), assume first that destination k is taken. Then, there exists an agent $j \neq i$ such that $\mathcal{I}_j^a = \{k\}$, and Assumption 4.1 (point 2) together with the fact that $\epsilon > 2\delta$ guarantee that $j \in \mathcal{N}_i^\epsilon$. Thus, $k \in \cup_{j\in\mathcal{N}_i^\epsilon} \mathcal{I}_j^t$, which shows the "only if" part of the claim. On the other hand, for any agent i, if $k \in \cup_{j\in\mathcal{N}_i^\epsilon} \mathcal{I}_j^t$ then, destination k clearly is taken, which shows the "if" part of the claim.

B. Proof of Proposition 5.3

Suppose that for $t = t_k$ automaton S is in mode v_S^k with $v_{C_i}^k = N$, for any agent i, and consider the following cases:

Case I: Assume that $x_i(t_k) \in \mathcal{B}_\delta(d_l)$ and $\mathcal{I}_i^a(t_k) \neq \{l\}$ for any $l \in \mathcal{I}_0$, and that destination l is available, i.e., $l \notin (\cup_{j\in\mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k))$. Then, at $t = t_{k+1}$, S transitions to mode v_S^{k+1} with $v_{C_i}^{k+1} = A_l$ and $\mathcal{I}_i^t(t_{k+1}) := reset(e_S^k) = \mathcal{I}_i^t(t_k) \cup \{l\} \cup (\cup_{j\in\mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k))$. Moreover, the reset $\mathcal{C}_i(t_{k+1}) := reset(e_S^k) = \{j \in \mathcal{N}_i^\epsilon(t_k) \mid x_j(t_k) \in \mathcal{B}_\delta(d_l)\} \cup \{i\}$ identifies other agents that simultaneously claim destination l. If i is the only agent claiming destination l, i.e., if $\mathcal{C}_i(t_{k+1}) = \{i\}$, then at $t = t_{k+2}$, S transitions to mode v_S^{k+2} with $v_{C_i}^{k+2} = T_l$ and $\mathcal{I}_i^a(t_{k+2}) := reset(e_S^{k+1}) = \{l\}$. Thus, property (a) is satisfied. Otherwise, if $\mathcal{C}_i(t_{k+1}) \neq \{i\}$, S transitions to mode v_S^{k+2} with $v_{C_j}^{k+2} = B_l$ for all $j \in \mathcal{C}_i(t_{k+1})$ instantaneously, due to the control switch $e_S^{k+1} = \|_{j\in\mathcal{C}_i(t_{k+1})} e_{C_j}^{k+1}$, which is such that $sync(e_{C_j}^{k+1}) = tiebreak_l$ for all $j \in \mathcal{C}_i$.[9] If i is the "leading" of the candidate agents in $\mathcal{C}_i(t_{k+1})$, i.e., if $i = \min\{\mathcal{C}_i(t_{k+1})\}$, then agent i "tosses a coin" to break the tie, i.e., $\mathcal{C}_i(t_{k+2}) := reset(e_S^{k+1}) = tb(\mathcal{C}_i(t_{k+1}))$, where the tie breaking function $tb(\cdot)$ is defined in (7). At time $t = t_{k+3}$, automaton S transitions to mode v_S^{k+3} with $v_{C_j}^{k+3} = R_l$ for all $j \in \mathcal{C}_i(t_{k+1})$ such that $\mathcal{I}_i^a(t_{k+3}) := reset(e_S^{k+2}) = \{l\}$ if $i \in \mathcal{C}_i(t_{k+2})$ and $\mathcal{I}_i^a(t_{k+3}) := reset(e_S^{k+2}) = \mathcal{I}_i^a(t_{k+2}) \backslash \mathcal{I}_i^t(t_{k+2})$ if $i \notin \mathcal{C}_i(t_{k+2})$, where $\mathcal{I}_i^t(t_{k+1}) = \mathcal{I}_i^t(t_{k+2})$. Hence, the tie is broken and S also satisfies property (b).

Observe that if at $t = t_{k+1}$ (or at $t = t_{k+2}$) there exists an agent $j \notin \mathcal{C}_i(t_{k+1})$ such that $x_j(t_{k+1}) \in \mathcal{B}_\delta(d_l)$, then $\mathcal{C}_i(t_{k+1}) \subseteq \mathcal{N}_j^\epsilon(t_{k+1})$, and so $l \in (\cup_{i\in\mathcal{N}_j^\epsilon(t_{k+1})} \mathcal{I}_i^t(t_{k+1}))$, i.e., destination l is considered taken for agent j. In other words, tie breaking occurs only among the agents in $\mathcal{C}_i(t_{k+1})$.

Case II: Assume that $x_i(t_k) \in \mathcal{B}_\delta(d_l)$ and $\mathcal{I}_i^a(t_k) \neq \{l\}$ for any $l \in \mathcal{I}_0$, and that destination l is taken, i.e., $l \in (\cup_{j\in\mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k))$. Then, at $t = t_{k+1}$, S transitions to mode v_S^{k+1} with $v_{C_i}^{k+1} = I$. If agent i has already knowledge of all taken destinations provided by its neighbors, i.e., if $\mathcal{I}_i^a(t_{k+1}) \cap (\cup_{j\in\mathcal{N}_i^\epsilon(t_{k+1})} \mathcal{I}_j^t(t_{k+1})) = \emptyset$, then at $t = t_{k+2}$, S transitions to mode v_S^{k+2} with $v_{C_i}^{k+2} = N$ and no update is done. Otherwise, if $\mathcal{I}_i^a(t_{k+1}) \cap (\cup_{j\in\mathcal{N}_i^\epsilon(t_{k+1})} \mathcal{I}_j^t(t_{k+1})) \neq \emptyset$, then S transitions to mode v_S^{k+2} with $v_{C_i}^{k+2} = U$ and $\mathcal{I}_i^a(t_{k+2}) := reset(e_S^{k+1}) = \mathcal{I}_i^a(t_{k+1}) \backslash (\cup_{j\in\mathcal{N}_i^\epsilon(t_{k+1})} \mathcal{I}_j^t(t_{k+1}))$ and $\mathcal{I}_i^t(t_{k+2}) := reset(e_S^{k+1}) = \mathcal{I}_i^t(t_{k+1}) \cup (\cup_{j\in\mathcal{N}_i^\epsilon(t_{k+1})} \mathcal{I}_j^t(t_{k+1}))$. So S satisfies property (c).

Case III: Assume that agent i owns destination $l \in \mathcal{I}_0$, i.e., that $\mathcal{I}_i^a(t_k) = \{l\}$. Then, by Assumption 4.1 (point 2), $x_i(t_k) \in \mathcal{B}_\delta(d_l)$, and at $t = t_{k+1}$, S transitions to mode v_S^{k+1} with $v_{C_i}^{k+1} = O_l$. Clearly, $\mathcal{I}_i^t(t_{k+1}) := reset(e_S^k) = \mathcal{I}_i^t(t_k) \cup (\cup_{j\in\mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k))$, and so S satisfies property (c).

Case IV: Assume that agent i is far from any destination, i.e., that $x_i(t_k) \notin \cup_{l\in\mathcal{I}_0} \mathcal{B}_\delta(d_l)$. Then, at $t = t_{k+1}$, S transitions to mode v_S^{k+1} with $v_{C_i}^{k+1} = I$. If agent i has already knowledge of all taken destinations provided by its neighbors, i.e., if $\mathcal{I}_i^a(t_{k+1}) \cap (\cup_{j\in\mathcal{N}_i^\epsilon(t_{k+1})} \mathcal{I}_j^t(t_{k+1})) = \emptyset$, then at $t = t_{k+2}$, S transitions to mode v_S^{k+2} with $v_{C_i}^{k+2} = N$ and no update is done. Otherwise, if $\mathcal{I}_i^a(t_{k+1}) \cap (\cup_{j\in\mathcal{N}_i^\epsilon(t_{k+1})} \mathcal{I}_j^t(t_{k+1})) \neq \emptyset$, then S transitions to mode v_S^{k+2} with $v_{C_i}^{k+2} = U$ and $\mathcal{I}_i^a(t_{k+2}) := reset(e_S^{k+1}) = \mathcal{I}_i^a(t_{k+1}) \backslash (\cup_{j\in\mathcal{N}_i^\epsilon(t_{k+1})} \mathcal{I}_j^t(t_{k+1}))$ and $\mathcal{I}_i^t(t_{k+2}) := reset(e_S^{k+1}) = \mathcal{I}_i^t(t_{k+1}) \cup (\cup_{j\in\mathcal{N}_i^\epsilon(t_{k+1})} \mathcal{I}_j^t(t_{k+1}))$. So S satisfies property (c).

C. Proof of Proposition 5.4

Because of the assumption $|\mathcal{I}_i^a(t)| > 1$ on the system variables, we restrict our study to those agents i that are in a mode $v_{N_i} > 1$. Let t_k denote the time instant that the product system S takes its kth transition. Clearly, between transitions, the variables $\mathcal{I}_i^a(t)$ are constant, and so it is sufficient to show that $\mathcal{I}(t) \subseteq \mathcal{I}_i^a(t)$ for all i with $v_{N_i} > 1$, at the transition time instants t_k. To do so, we use induction on k. Clearly, for $k = 0$ we have that $\mathcal{I}_i^a(t_0) = \mathcal{I}(t_0) = \mathcal{I}_0$ for all i, by initialization of the problem, and so $\mathcal{I}(t_0) \subseteq \mathcal{I}_i^a(t_0)$ for all i. Assume that $\mathcal{I}(t_k) \subseteq \mathcal{I}_i^a(t_k)$ for any $k > 0$ and all i with $v_{N_i}^k > 1$, and consider the transition $v_S^k \to v_S^{k+1}$ with corresponding control switch $e_S^k = (v_S^k, v_S^{k+1})$. Then, for $t = t_{k+1}$, we have the following cases:

Case I: For all agents i such that $v_{C_i}^k = I$ and $v_{C_i}^{k+1} = U$, the reset becomes $\mathcal{I}_i^a(t_{k+1}) := reset(e_S^k) = \mathcal{I}_i^a(t_k) \backslash (\cup_{j\in\mathcal{N}_i^\epsilon(t_k)}$

[9] Note that the set $\mathcal{C}_i(t_{k+1})$ is common for all $j \in \mathcal{C}_i(t_{k+1})$ since condition $\epsilon > 2\delta$ guarantees that they are all neighbors of each other.

$\mathcal{I}_j^t(t_k)$). Since $v_{N_i}^k > 1$ we have that, $\mathcal{I}_i^a(t_k) \cup \mathcal{I}_i^t(t_k) = \mathcal{I}_0$ and $\mathcal{I}_i^a(t_k) \cap \mathcal{I}_i^t(t_k) = \emptyset$. Hence, by Lemma 2.2(a) in Appendix II, the induction hypothesis $\mathcal{I}(t_k) \subseteq \mathcal{I}_i^a(t_k)$ implies that $\mathcal{I}_i^t(t_k) \subseteq \mathcal{I}_0 \backslash \mathcal{I}(t_k)$, and so $\cup_{j \in \mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k) \subseteq \mathcal{I}_0 \backslash \mathcal{I}(t_k) = \mathcal{I}_0 \backslash \mathcal{I}(t_{k+1})$, since $\mathcal{I}(t_k) = \mathcal{I}(t_{k+1})$ if $v_{C_i}^k = I$ and $v_{C_i}^{k+1} = U$. By Lemma 2.2(b) in Appendix II, the induction hypothesis, and the fact that $\mathcal{I}(t_k) = \mathcal{I}(t_{k+1})$ we conclude that $\mathcal{I}_i^a(t_k) \backslash (\cup_{j \in \mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k)) \supseteq \mathcal{I}(t_{k+1})$, and so $\mathcal{I}_i^a(t_{k+1}) \supseteq \mathcal{I}(t_{k+1})$.

Case II: For all agents i such that $v_{C_i}^k = A_l$ and $v_{C_i}^{k+1} = T_l$, for any $l \in \mathcal{I}_0$, the reset $\mathcal{I}_i^a(t_{k+1}) := reset(e_S^k) = \{l\}$ gives $|\mathcal{I}_i^a(t_{k+1})| = 1$.

Case III: For any $l \in \mathcal{I}_0$ and $i \in \mathcal{C}_i(t_k)$ such that $v_{C_i}^k = B_l$ and $v_{C_i}^{k+1} = R_l$, the reset $\mathcal{I}_i^a(t_{k+1}) := reset(e_S^k) = \{l\}$ gives $|\mathcal{I}_i^a(t_{k+1})| = 1$.

Case IV: For any $l \in \mathcal{I}_0$ and all $i \notin \mathcal{C}_i(t_k)$ such that $v_{C_i}^k = B_l$ and $v_{C_i}^{k+1} = R_l$, the reset becomes $\mathcal{I}_i^a(t_{k+1}) := reset(e_S^k) = \mathcal{I}_i^a(t_k) \backslash (\{l\} \cup (\cup_{j \in \mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k)))$. Applying Lemma 2.2(a) in Appendix II, the induction hypothesis implies that $\mathcal{I}_i^t(t_k) \subseteq \mathcal{I}_0 \backslash \mathcal{I}(t_k)$, and so $\cup_{j \in \mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k) \subseteq \mathcal{I}_0 \backslash \mathcal{I}(t_k)$. Moreover,

$$\begin{aligned}(\mathcal{I}_0 \backslash \mathcal{I}(t_k)) \cup \{l\} &= (\mathcal{I}_0 \cap \mathcal{I}^c(t_k)) \cup \{l\} \\ &= (\mathcal{I}_0 \cup \{l\}) \cap (\mathcal{I}^c(t_k) \cup \{l\}) \\ &= \mathcal{I}_0 \cap (\mathcal{I}(t_k) \cap \{l\}^c)^c \\ &= \mathcal{I}_0 \cap (\mathcal{I}(t_k) \backslash \{l\})^c \\ &= \mathcal{I}_0 \backslash (\mathcal{I}(t_k) \backslash \{l\}) = \mathcal{I}_0 \backslash \mathcal{I}(t_{k+1})\end{aligned}$$

since $\mathcal{I}(t_k) \backslash \{l\} = \mathcal{I}(t_{k+1})$ if $v_{C_i}^k = B_l$ and $v_{C_i}^{k+1} = R_l$, and so

$$\{l\} \cup (\cup_{j \in \mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k)) \subseteq \mathcal{I}_0 \backslash \mathcal{I}(t_{k+1}).$$

By Lemma 2.2(b) in Appendix II, the induction hypothesis, and the fact that $\mathcal{I}(t_k) \backslash \{l\} = \mathcal{I}(t_{k+1})$, we conclude that $\mathcal{I}_i^a(t_k) \backslash (\{l\} \cup (\cup_{j \in \mathcal{N}_i^\epsilon(t_k)} \mathcal{I}_j^t(t_k))) \supseteq \mathcal{I}(t_{k+1})$, and so $\mathcal{I}_i^a(t_{k+1}) \supseteq \mathcal{I}(t_{k+1})$.

D. Proof of Proposition 5.5

Let, $v_S^\star = (v_{N_1}^\star, \ldots, v_{N_n}^\star, v_{C_1}, \ldots, v_{C_n})$ be such that $v_{N_i}^\star = 1$ for all i. Then, the condition that $\mathcal{I}_i^a \cap \mathcal{I}_j^a = \emptyset$ for all $j \neq i$ is guaranteed by Propositions 5.2–5.4. In particular, Propositions 5.2 and 5.3 imply that only available destinations can be claimed by any agent with $v_{N_i} > 1$, and Proposition 5.4 that there always exists an available destination in $\mathcal{I}_i^a$ if $v_{N_i} > 1$. Hence, a one-to-one correspondence is established between agents and destinations, which implies that mode $v_S^\star$ is a reachable mode of the system S. On the other hand, every transition $v_S \xrightarrow{e_S} v_S'$ such that $sync(e_S) = update_i$ decreases the value of v_{N_i}, and thus, results in progress toward reaching mode $v_S^\star$. Since $\mathcal{I}_i^a$ are finite sets, the number of these transitions can only be finite. To derive an upper bound on these transitions, we construct a worst case scenario and count the number of transitions that S takes in that scenario.[10]

Observe first that to maximize the total number of transitions, we require that $\mathcal{I}_i^a$ is always updated such that $|\mathcal{I}_i^a(t_{k+1})| := |\mathcal{I}_i^a(t_k)| - 1$, since larger updates result in faster progress toward mode $v_S^\star$. Without loss of generality, we also assume that the order of transitions is such that the last transition of v_{N_i} indicates the first transition of $v_{N_{i+1}}$. Reordering the transitions, or relabeling the automata N_i, we can get any desired transition scheme. With these observations, we construct a worst case scenario as follows.

Initially, v_S^0 is such that $v_{N_i}^0 = m$ for all i. Let, $(m,1)_{N_1} \| (T_1, N)_{C_1}$ be the first control switch to be enabled. Then, transition $v_S^0 \xrightarrow{e_S} v_S^1$ is such that $v_{N_1}^1 = 1$. Let $(m, m-1)_{N_2} \| (U, N)_{C_2}$ be the second control switch to be enabled. Then, transition $v_S^1 \xrightarrow{e_S} v_S^2$ is such that $v_{N_2}^2 = m-1$. The third control switch to be enabled is $(m-1,1)_{N_2} \| (T_2, N)_{C_2}$ and transition $v_S^2 \xrightarrow{e_S} v_S^3$ is such that $v_{N_2}^3 = 1$. In the same way the fourth, fifth, and sixth control switches to be enabled are $(m, m-1)_{N_3} \| (U, N)_{C_3}$, $(m-1, m-2)_{N_3} \| (U, N)_{C_3}$, and $(m-2,1)_{N_3} \| (T_3, N)_{C_3}$ respectively, and after the sixth transition, $v_{N_3}^6 = 1$. Proceeding in the same way and adding up these transitions, we get that S transitions $n(n+1)/2$ times in total, which completes the proof.

E. Proof of Theorem 5.2

Observe that we only need to show that there exists a constant $T > 0$ such that for all time $t > t_0 + T$, the product system S is in mode $v_S^\star = (v_{N_1}^\star, \ldots, v_{N_n}^\star, v_{C_1}, \ldots, v_{C_n})$ with $v_{N_i}^\star = 1$ since then, Propositions 5.2–5.4 guarantee that $v_S^\star$ is such that $\mathcal{I}_i^a(t) \cap \mathcal{I}_j^a(t) = \emptyset$ for all $j \neq i$. Thus, we only need to show that there does not exist an agent i such that $v_{N_i} > 1$ for ever. In other words, we need to show that transitions $v_S \xrightarrow{e_S} v_S'$ such that $sync(e_S) = update_i$ will eventually occur until $v_S' = v_S^\star$. Then, since by Proposition 5.5, the system S can only take a finite number of such transitions, we can get $T > 0$ simply by adding the time intervals between these transitions.

But, the flow conditions for any mode v_S of S are given by the system of differential equations

$$\dot{x}_i = u_v(x_i, \mathcal{I}_i^a) \quad \forall\, v \in V_{N_i} \quad \text{and} \quad i = 1, \ldots, n$$

which are decoupled and hence, applying Theorem 3.2, we have that for all i, almost all initial conditions $x_i(t_0)$, and any $\delta > 0$, there exist $T_i = T_i(\delta, t_0) > 0$ and $k \in \mathcal{I}_i^a(t_0)$ such that $x_i(t) \in \mathcal{B}_\delta(d_k)$, for all $t > t_0 + T_i$. Let $T = \min_i\{T_i\}$. Then $t = t_0 + T$ denotes the time of the transition $v_S \to v_S'$, where v_S' is such that $v_{N_j}' = 1$ for $j = \text{argmin}_i\{T_i\}$. Applying the same argument inductively until $v_S' = v_S^\star$ completes the proof.

APPENDIX II

Lemma 2.1: The function $f(x) = \log(\|x - y\|^2)$ with $x, y \in \mathbb{R}^2$ is harmonic.

Proof: Let $x = [x_1\ x_2]^T$ and $y = [y_1\ y_2]^T$. Then, the first derivative of $f(x)$ with respect to x_i for $i = 1, 2$

[10]Without loss of generality, we do not include in the worst case scenario transitions due to *tie breaking*, since they correspond to one single successful assignment and multiple failed assignments occurring simultaneously, and could hence be treated separately without affecting the total number of transitions.

is $(\partial/\partial x_i)f(x) = 2(x_i - y_i)/\|x-y\|^2$, and so the second derivative becomes,

$$\frac{\partial^2}{\partial x_i^2} f(x) = \frac{2\|x-y\|^2 - 4(x_i - y_i)^2}{\|x-y\|^4}.$$

So, the Laplace equation for the function $f(x)$ becomes,

$$\frac{\partial^2}{\partial x_1^2} f(x) + \frac{\partial^2}{\partial x_2^2} f(x) = 0$$

which clearly implies that $f(x)$ is harmonic. ■

Lemma 2.2: Let $A, B, C,$ and D be any sets. Then,

a) if $A \cup B = D$ and $A \cap B = \emptyset$, the inclusion $C \subseteq A$ implies, $B \subseteq D \backslash C$.
b) if $C \subseteq D$, the inclusions $B \subseteq D \backslash C$ and $C \subseteq A$ imply that $C \subseteq A \backslash B$.

Proof: To prove (a) observe that if $x \in B$ then $x \in A \cup B$ and $x \notin A$. Hence, $x \in D$ and $x \notin C$ which implies that $x \in D \backslash C$. To prove (b) observe that if $x \in C$ then $x \in A$ and $x \notin D \backslash C$. Hence, $x \in A$ and $x \notin B$ which implies that $x \in A \backslash B$. ■

REFERENCES

[1] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 988–1001, Jun. 2003.

[2] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[3] H. G. Tanner, A. Jadbabaie, and G. J. Pappas, "Flocking in fixed and switching networks," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 863–868, May 2007.

[4] J. Cortes, S. Martinez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *IEEE Trans. Autom. Control*, vol. 51, no. 8, pp. 1289–1298, Aug. 2006.

[5] L. Chaimowicz, N. Michael, and V. Kumar, "Controlling swarms of robots using interpolated implicit functions," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, pp. 2498–2503.

[6] R. Sepulchre, D. Paley, and N. E. Leonard, "Stabilization of planar collective motion: All-to-all communication," *IEEE Trans. Autom. Control*, vol. 52, no. 5, pp. 811–824, May 2007.

[7] G. Lafferriere, A. Williams, J. Caughman, and J. J. P. Veerman, "Decentralized control of vehicle formations," *Syst. Control Lett.*, vol. 54, no. 9, pp. 899–910, Sep. 2005.

[8] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 926–939, Dec. 1998.

[9] W. Ren and R. Beard, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661, May 2005.

[10] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, May 2004, pp. 165–172.

[11] J. Lin, A. S. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem," in *Proc. 42nd IEEE Conf. Decisi. Control*, Maui, HI, Dec. 2003, pp. 1508–1513.

[12] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Res. Logist.*, vol. 2, no. 1–2, pp. 83–97, Mar. 1955.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.

[14] H. A. Almohamad and S. O. Duffuaa, "A linear programming approach for the weighted graph matching problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 5, pp. 522–525, May 1993.

[15] H. Wolkowicz, "Semidefinite programming approaches to the quadratic assignment problem," in *Nonlinear Assignment Problems: Algorithms and Applications (Combinatorial Optimization)*. Norwell, MA: Kluwer, 2000, pp. 143–174.

[16] M. M. Zavlanos and G. J. Pappas, "A dynamical systems approach to weighted graph matching," in *Proc. 45th IEEE Conf. Decis. Control*, San Diego, CA, Dec. 2006, pp. 3492–3497.

[17] S. Kloder and S. Hutchinson, "Path planning for permutation-invariant multirobot formations," *IEEE Trans. Robot.*, vol. 22, no. 4, pp. 650–665, Aug. 2006.

[18] M. Ji, S. Azuma, and M. Egerstedt, "Role-assignment in multi-agent coordination," *Int. J. Assist. Robot. Mechatron.*, vol. 7, no. 1, pp. 32–40, Mar. 2006.

[19] M. M. Zavlanos and G. J. Pappas, "Sensor-based dynamic assignmnet in distributed motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, Apr. 2007, pp. 3333–3338.

[20] M. M. Zavlanos and G. J. Pappas, "Dynamic assignmnet in distributed motion planning with limited information," in *Proc. Am. Control Conf.*, New York, Jul. 2007, pp. 1173–1178.

[21] D. V. Dimarogonas, S. G. Loizou, K. J. Kyriakopoulos, and M. M. Zavlanos, "A feedback stabilization and collision avoidance scheme for multiple independent non-point agents," *Automatica*, vol. 42, no. 2, pp. 229–243, Feb. 2006.

[22] D. E. Koditschek and E. Rimon, "Robot navigation functions on manifolds with boundary," *Adv. Appl. Math.*, vol. 11, no. 4, pp. 412–442, Dec. 1990.

[23] J. O. Kim and P. K. Khosla, "Real-time obstacle avoidance using harmonic potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 3, pp. 338–349, Jun. 1992.

[24] T. A. Henzinger, "The theory of hybrid automata," in *Proc. 11th Annu. IEEE Symp. Logic Comput. Sci.*, New Brunswick, NJ, Jul. 1996, pp. 278–292.

[25] N. A. Lynch, R. Segala, and F. W. Vaandrager, "Hybrid I/O automata," *Inf. Comput.*, vol. 185, no. 1, pp. 105–157, Aug. 2003.

[26] D. V. Dimarogonas, K. J. Kyriakopoulos, and D. Theodorakatos, "Totally distributed motion control of sphere world multi-agent systems using decentralized navigation functions," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, May 2006, pp. 2430–2435.

Michael M. Zavlanos (S'05) received the Diploma in mechanical engineering from the National Technical University of Athens, Athens, Greece, and the M.S.E. degree in electrical and systems engineering from the University of Pennsylvania, Philadelphia, in 2002 and 2005, respectively. Currently, he is working toward the Ph.D. degree in the Department of Electrical and Systems Engineering, University of Pennsylvania.

His current research interests include distributed control of multiagent systems and hybrid dynamical systems with applications to assignment problems in robotics, topology control of dynamic networks, and formation control.

Mr. Zavlanos was a finalist of the Best Student Paper Award at the 45th IEEE Conference on Decision and Control, 2006.

George J. Pappas (S'91–M'98–SM'04) received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 1998.

He is currently a Professor at the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia. He also holds secondary appointments in the Departments of Computer and Information Sciences, and Mechanical Engineering and Applied Mechanics, and is a member and a former Director of the General Robotics and Active Sensory Perception (GRASP) Laboratory. He currently serves as the Deputy Dean in the School of Engineering and Applied Science. He is the Co-Editor of *Hybrid Systems: Computation and Control* (Springer-Verlag, 2004). His current research interests include the areas of hybrid and embedded systems, hierarchical control systems, distributed control systems, nonlinear control systems, and geometric control theory, with applications to robotics, unmanned aerial vehicles, and biomolecular networks.

Prof. Pappas was the recipient of numerous awards including the National Science Foundation (NSF) Career Award in 2002, the NSF Presidential Early Career Award for Scientists and Engineers (PECASE) in 2002, and the Eliahu Jury Award for Excellence in Systems Research from the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley in 1999. His and his students' papers were finalists for the Best Student Paper Award at the IEEE Conference on Decision and Control in 1998, 2001, 2004, and 2006, the American Control Conference in 2001 and 2004, and the IEEE Conference on Robotics and Automation in 2007.