# Path-Planning-Enabled Semi-Flocking Control for Multi-target Monitoring in Mobile Sensor Networks

Wanmai Yuan, Nuwan Ganganath, *Member, IEEE,* Chi-Tsun Cheng, *Senior Member, IEEE,* Guo Qing, *Member, IEEE,* Francis C.M. Lau, *Senior Member, IEEE,* Yanjie Zhao

***Abstract*—Mobile sensor networks (MSNs) are good candidates for large-scale unattended surveillance applications. However, it is challenging to track moving targets due to their complex dynamic behaviors. Semi-flocking algorithms have been proven to be efficient in controlling MSNs in both area coverage and target tracking applications. While many existing literatures on the study of semi-flocking algorithms often assume an area of interest (AoI) to be regular and with unified traversal cost, the uneven and rough landscapes in real-life applications have imposed extra challenges and raised demands for new management strategies. In this paper, a mobility map is used to incorporate different costs associated with irregular terrains which results in different maximum allowed speeds on nodes in different regions. In order to reduce target detection time and node energy consumption, a heuristic search algorithm is developed to find time-efficient and feasible paths between nodes and sensing targets. Under the proposed algorithm, nodes can effectively select a target to track or search for new targets in the AoI. Results of extensive experiments show that semi-flocking-controlled nodes together with path planning can reach their targets faster with lower energy consumption compared to three exiting flocking-based algorithms.**

***Index Terms*—Semi-flocking, path planning, mobility maps, mobile sensor networks, area coverage, target tracking**

## I. Introduction

WIRELESS sensor networks (WSNs) can gather vast volume of data using their sensing modules with minimum human intervention. A typical node in a WSN consists of sensing circuitries for target detection, power-aware processing units for data conditioning and pre-processing, an energy-efficient transceiver for data transmission, and a limited power source. For static WSNs, to improve their sensing capabilities, nodes are strategically deployed within an area of interest (AoI) in which prior knowledge of target motion attributes is given. However, the computational overheads due to sensor placement and their relatively low scalability have reduced their applicability in monitoring moving targets in large AoIs. Moreover, it is challenging for static WSNs to monitor multiple moving targets and to provide the required sensing coverage to each of them.

Mobile sensor networks (MSNs) are recognized as one of the promising approaches in wide-area target monitoring [1]. Compared with ordinary WSNs, MSNs are capable of repositioning and reorganizing themselves to provide better sensing coverage. MSNs have been applied in many real-life monitoring scenarios, such as pursuit-evasion [2], search and rescue [3], intrusion detection [4], and border patrol [5], due to their low operating cost, high operating flexibility, and monitoring capabilities.

In most MSN monitoring applications, movements of nodes consume most of the nodes' limited energy. Many existing target-tracking algorithms in MSNs assume a flat operating environment. With such a design limitation, these algorithms may lead nodes to pass through high-cost regions, and will ultimately lead to an even higher energy consumption. Therefore, it is necessary to incorporate traverse cost in MSN management strategy designs.

In this paper, a semi-flocking-controlled algorithm for MSNs with path planning is proposed to tackle the multi-target monitoring problem in an uneven environment. The incorporation of the path-planning algorithm allows nodes to pick time-efficient and feasible paths to reach their targets and deliver better sensing performance.

### A. Related Work

Researchers have extensively examined different approaches for reducing resources involved in monitoring moving targets and for improving tracking performances of sensory systems. A Voronoi-based clustered target-tracking scheme is presented in [6], where barrier coverage and moving-target tracking are studied together to improve target detectability. Bocca *et al.* [7] proposed a real-time radio tomographic imaging method to track multiple targets using received signal strength indicator (RSSI) measurements. Milan *et al.* [8] proposed an unified model of data association and trajectory estimation to carry out multi-target tracking through the minimization of a consistent discrete-continuous energy function.

In [9], a decentralized motion coordination algorithm is proposed based on Kalman filter to control MSNs in static and dynamic target tracking applications. The problem of collecting local information and generating the information report is explored in [10], where a tree-based technique is utilized to increase sensing coverage and lower energy consumption. A prediction-based method for deriving target trajectories,

This work is supported by the China Academy of Electronics and Information Technology, China. (Projects 41411030501).

W. Yuan and Y. Zhao are with the China Academy of Electronics and Information Technology, Beijing, P.R.China.

N. Ganganath is with the School of Electrical, Electronic and Computer Engineering, the University of Western Australia, WA, Australia.

C.-T. Cheng is with the Department of Manufacturing, Materials and Mechatronics, RMIT University, Melbourne, Australia.

Q. Guo is with the School of Electronics and Information Engineering, the Harbin Institute of Technology, Harbin, P.R.China.

F.C.M. Lau is with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong.

velocities, and residual energy is designed in [11]. In [12], Wang and Gu have presented a cooperative target tracking strategy to estimate target position and to maneuver nodes toward the estimated position under flocking control.

In [13], a geographic routing protocol in MSNs is proposed to steer nodes to bypass obstacles in given areas. The work focuses on selecting cluster heads and forming energy-efficient paths between cluster heads and sink nodes. An energy-efficient target tracking algorithm is proposed in [14]. The algorithm has translated the minimum energy problem into a constrained shortest path problem.

Semi-flocking algorithms had been proposed in [15] and [16] which combine flocking [17], [18] and anti-flocking [19], [20] controls via mode switching. Initially, the nodes search the AoI to detect targets. Once a node finds a target, it will attract nearby nodes to move toward the target via straight lines. Such a design performs well in ideal scenarios where an AoI consists of regular regions with unified motion cost. However, for real-life monitoring applications, the AoI often imposes speed constraints to mobile platforms. To the best knowledge of the authors, there is no formal investigation on multiple target tracking using MSNs on terrains with non-uniform speed limits.

### B. Contributions

The main contributions of our work are as follows:

1) This paper proposes a path-planning-enabled semi-flocking algorithm for multi-target monitoring in terrains with irregular costs. Semi-flocking-controlled nodes can simultaneously sense the AoI efficiently and track multiple targets rapidly. These nodes are capable of determining whether they should be involved in target tracking or they should continue to search the AoI.
2) A mobility map is used to represent different maximum allowed speeds that a node can achieve when passing through patches in the AoI. A novel heuristic search algorithm is developed to perform path planning on the mobility map. Nodes can travel along time-efficient paths to reach a target and cooperate with other nodes that are tracking the same target.
3) Performances of the proposed algorithm are evaluated under two different scenarios, i.e., a $200 \times 200$ m$^2$ AoI with obstacles and a $200 \times 200$ m$^2$ AoI without obstacles. Our work validates that path-planning-enabled semi-flocking-controlled MSNs can efficiently and effectively monitor multiple targets in terrains with non-uniform maximum speed limits.

The rest of the paper is organized as follows. Section II reviews the formulations of MSNs and the concepts of mobility maps. In Section III, a novel A*-based path planning algorithm is introduced. The proposed path-planning-enabled semi-flocking algorithm is presented and elaborated in Section IV. Test results are presented and analyzed in Section V. In Section VI, discussions on the computational complexity of the proposed algorithm are provided. Finally, conclusions are given in Section VII.

## II. Preliminaries

### A. Formulation of MSNs

We consider a MSN consisting of $N$ nodes. The motion of node $i$ is governed by

$$\begin{cases} \dot{q}_i(t) = p_i(t), \\ \dot{p}_i(t) = u_i(t), \qquad i = 1, 2, \ldots, N, \end{cases} \tag{1}$$

where $q_i(t)$ and $p_i(t)$ are the position and velocity of the node $i$ at time $t$, respectively, and $u_i(t)$ is the control input of node $i$. For notational convenience, we define $q_i(t) = q_i$, $p_i(t) = p_i$, and $u_i(t) = u_i$ as in [17].

While moving in the AoI, a node is able to interact with other nodes within its communication range $r_c$. The set of neighbors of node $i$ within $r_c$ at time $t$ is denoted as

$$\mathcal{N}_i(t) = \{j : \|q_j - q_i\| < r_c, j = 1, 2, \ldots, N, j \neq i\}$$

where $q_j$ is the position of node $j$.

### B. Mobility Map Concepts

In real-life applications, an AoI can impose strong influences on the maximum speed of nodes. There are many approaches that can be used to construct a mobility map for an AoI and can be applied directly to this work. In [21], robots with incomplete differential-global-position-system (DGPS) information autonomously generate rough elevation maps of their terrains. An incremental terrain mapping algorithm is utilized to obtain the depth and elevation information of the AoI. Karnadi *et al.* in [22] developed a mobility model generator for vehicular networks. In their work, a real-world map is imported from publicly available databases. Then, a realistic mobility model is generated using a source micro-traffic simulator (SUMO). There are many factors that dominate the maximum allowed or achievable speed between any two points on a given AoI. Examples include [23]

- the availability of traction to overcome resistances resulting from terrain roughness, terrain slope, blocking obstacles, and vegetation density, etc; and
- the maneuverability of the node across obstacles.

Since this work focuses on incorporating path planning for navigation in the semi-flocking-controlled nodes, the construction of the mobility map is regarded as out of scope.

In this work, the speed limit of each sub-region is generated arbitrarily and the whole region is represented in form of a mobility map. As shown in Fig. 1, patches in the AoI with different colors correspond to different specific maximum allowed speeds. Anchor points are represented as black dots on the patches and they are the initial sites in the underlaid Voronoi diagram. As an example, a path connecting a start point and an end point is shown as a green line on the mobility map. From there, red stars denote the intersection points of the path and the boundaries of adjacent patches on the mobility map.

To speed up the path-planning process, the mobility map is first divided into a number of square cells. Then the map is transformed into a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is a set of nodes denoting the cell centers, and $\mathcal{E}$ represents

TABLE I
RESULTS OF PATH PLANNING ALGORITHMS UNDER TEST

| Path planning methods | Travel times (s) | Avg. online execution times (s) | Number of nodes expanded during path planning |
|---|---|---|---|
| Dijkstra's | 16.0712 | 235.6643 | 16157 |
| A* | 16.0712 | 221.0276 | 9726 |
| A*PL | 16.3147 | 8.3668 | 9783 |
| A*IL | 15.9477 | 0.0012 | 9783 |

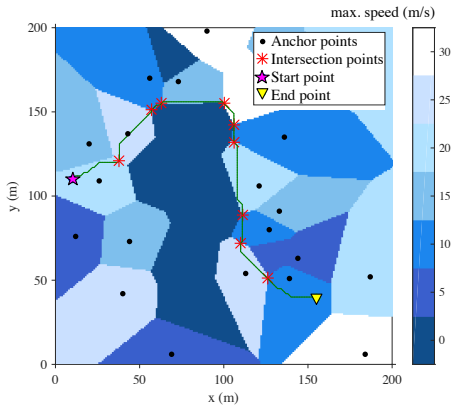



Fig. 1. An example of a mobility map.

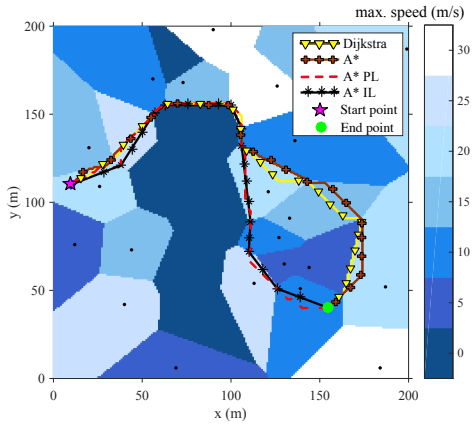



Fig. 2. Snapshots in executing 4 different search methods for finding time-efficient paths between a start point and an end point.

the set of connections between adjacent cell centers. In this paper, a time heuristic is proposed for finding the fastest path on mobility maps efficiently. The proposed time heuristic is proven to be both admissible and consistent [23]. When an A* search algorithm is guided by the proposed time heuristic, it guarantees an optimal solution can be found if there is such a solution for the problem. For every pair of adjacent cell centers $n$ and $n'$, their connection $(n, n') \in \mathcal{E}$ is associated with a time heuristic $c_{\mathrm{t}}(n, n')$ which can be obtained as

$$c_{\mathrm{t}}(n, n') = \frac{d(n, n')}{v_{\mathrm{avg}}}.$$

Here, $d(n, n')$ and $v_{\mathrm{avg}}$ are the Euclidean distance and average maximum allowed speed between $n$ and $n'$, respectively.

## III. PATH PLANNING ON MOBILITY MAPS

In this section, we report results of a preliminary study that aims to select a fast and scalable path-planning method for the proposed semi-flocking algorithm. Four different algorithms are put under test for path planning on mobility maps. They are (1) Dijkstra's algorithm [23], (2) A* algorithm [23], (3) A* algorithm with path lookup (A*PL), and (4) A* with intersection lookup (A*IL). Both the Dijkstra and A* algorithms are in their generic versions for generating time-efficient paths between the given start and end points on a mobility map. A*PL and A*IL are two variations of the A* algorithm that are proposed in this paper for time-efficient path planning on mobility maps. They are further elaborated in this section.

Both A*PL and A*IL consist of offline and online planning phases. In the offline planning phase, time-efficient paths between all anchor points are obtained using the A* algorithm which incorporates the costs in the mobility maps. In A*PL, the path segments between the first and the last intersection points in each time-efficient path are stored in a static routing table. In A*IL, only the intersection points on the time-efficient path are stored in a static routing table. Thus, the routing table built by A*IL is less memory-consuming compared to that built by A*PL.

In the online planning phase of A*PL, an A* algorithm is employed to plan time-efficient path segments (i) from the start point to the first intersection point and (ii) from the last intersection point to the end point. The time-efficient path between the first and the last intersection points can be retrieved from the static routing table. In the online planning phase of A*IL, the interconnection points on the time-efficient path between the first and the last intersection points are first retrieved from the static routing table. Then the time-efficient path between the first and the last intersection points is obtained by connecting these interconnection points with straight paths. The time-efficient path between the start point and the first intersection point within the same patch is again obtained by connecting them with a straight path. Similarly, the time-efficient path between the last intersection point and the end point on the same patch is obtained by connecting these points with another straight path.

Extensive tests have been conducted to evaluate the performance of the four path-planning algorithms. The paths obtained in one such test are illustrated in Fig. 2. All the tests were carried out in MATLAB on a computer with a 2.67 GHz Intel i5 processor, 8 GB memory, and Windows 10 operating system. It is obvious that all these 4 algorithms can find paths avoiding the obstructed areas, i.e., areas with maximum speed 0 m/s. The corresponding travel time and average execution

time are given in Table I. The average execution time is obtained by averaging the run time of each algorithm over 10 executions. Note that this excludes the time spent on building the static routing tables in the offline phase.

Referring to Table I, Dijkstra's algorithm has expanded 16157 nodes using 235.66 seconds in its search for the fastest path whereas A* with the proposed heuristic has found the exact same path by just expanding 14751 nodes using 221.03 seconds. With the help of heuristic functions to estimate the time-cost to the end point [23], the A*-based methods can find paths faster than the Dijkstra's method. While the generic A* algorithm improves only slightly over the Dijkstra's algorithm in terms of execution time, both A*PL and A*IL algorithms show significant improvements with the use of static routing tables. A*IL algorithm further outperforms A*PL because A*IL algorithm uses a much smaller routing tables which result in shorter execution times.

Note that in the Dijkstra's, A*, and A*PL algorithms, each move in the path is restricted between the current cell center and the center of a neighboring cell. Thus, each move can only point toward one of the 8 directions, e.g., $0°, 45°, 90°, \ldots, 315°$. As a result, the overall path may become longer due to unnecessary turnings. On the contrary, the path within each terrain is a straight line in the case of the A*IL algorithm. Therefore, provided that the intersection points stored in the static routing table are close to optimal, the A*IL algorithm is more likely to create the shortest travel time compared with the other three methods. The results in Table I concur with such a hypothesis.

It can also be observed that the travel time of the paths obtained by the A*PL algorithm is the longest. In the off-line planning phase of A* PL, the time-efficient path segments between the first and the last intersection points are planned and then stored in a static routing table. In the online planning phase of A*PL, an A* algorithm is employed to find time-efficient path segments (i) from the start point to the first intersection point and (ii) from the last intersection point to the end point. On the other hand, in Dijkstra's and the original A* algorithms, time-efficient paths are directly planned from the start point to the end point. Therefore, the paths generated by the A*PL algorithm that stored in the routing table can be less optimal compared with those generated by the Dijkstra's and A* algorithms.

Obviously, A*IL can eliminate this limitation and can find a faster path when compared with those obtained by A*PL which concur with the results provided in Table I. The travel time of the path obtained by A*IL is even smaller than those obtained by the Dijkstra's and A* algorithms because of using straight lines between each pair of neighboring intersections. Therefore, A*IL is selected as the path planning algorithm in the proposed semi-flocking algorithm.

## IV. Proposed Semi-Flocking Algorithm for Multi-target Monitoring

MSNs under the proposed algorithm can detect targets and inform other nodes to arrive quickly so as to accomplish the required number of monitoring nodes per target. The transition mechanism of the three operating states (namely *searching*, *traveling*, and *monitoring*) of a node with the proposed algorithm is introduced as follows.

**Algorithm 1** A state switching mechanism for node $i$.

Initialize $q_i$, $p_i$, $n_k$, $q_s$, $\theta_k$, and $T_{max}$;
$t = 0$;
**while** $t < T_{max}$ **do**
  **for** $k = 1$ to $M$ **do**
    Exchange information with other nodes nearby and update $n_k$;
    Access static routing table and acquire the corresponding traveling cost for reaching the target;
    Calculate the switching probability for tracking target $k$ based on the corresponding traveling cost;
  **end for**
  Select a target to track or remain in searching state;
  $f_i^g = \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij}$;
  $f_i^d = (p_j - p_i) a_{ij}(q_i, q_j)$;
  **if** node $i$ decides to track target $k$ **then**
    $n_k = n_k + 1$;
    **if** $\|q_k^\eta - q_i\| > \theta_k$ **then**
      Switch into the traveling state;
      Carry out A* IL-based path planning;
      Visit the pre-stored intersection points on the path one-after-another;
    **else**
      Switch into the monitoring state;
      $f_i^t = c_3(q_k^\eta - q_i) + c_4(p_k^\eta - p_i) + c_5 a_k^\eta$;
      $u_i = f_i^g + f_i^d + f_i^t$ ;
    **end if**
  **else**
    Remain in the searching state;
    Calculate $q_s$;
    $f_i^s = c_1(q_s - q_i)$;
    $u_i = f_i^g + f_i^d + f_i^s$ ;
  **end if**
  $q_i = p_i \cdot t + q_i$;
  $p_i = u_i \cdot t + p_i$;
  $t = t + 1$;
**end while**

### A. State Transition Mechanism

The state transition mechanism of node $i$ is shown in Algorithm 1. In the proposed algorithm, $n_k$ represents the number of nodes that are currently monitoring target $k$; $\theta_k$ is a predefined threshold; $T_{\max}$ denotes the operating time of the MSNs.

Initially, all nodes are in the *searching* state. They do not have any prior information about the targets. Therefore, they try to maximize the global sensing coverage by minimizing their overlapping sensing areas.

Once a target is detected by a node in *searching* state, other nearby nodes will be informed with the information of the target, including its location and velocity. The informed nodes can access their static routing table and acquire the

corresponding traveling cost for reaching the target. These nodes will make a decision to track the target only if the corresponding traveling cost is low. These nodes will return to the *searching* state if the target has already been tracked by a sufficient number of nodes while the nodes are approaching it. For nodes that have decided to track a target but are not close to it, they will be operating in the *traveling* state. A node in the *traveling* state will visit the pre-stored intersection points on the path one-after-another until they arrive at an area that is close to the target.

Once arrived, it will switch to the *monitoring* state and form a sensing cluster with other nearby nodes that are monitoring the same target.

### B. Nodes in the Searching State

Initially, all nodes operate in the *searching* state are aiming to maximize their area coverage by reducing their overlapping sensing area with their neighbors. In addition, each node in the *searching* state tries to visit regions that have not been visited for a long time. Under the proposed algorithm, node $i$ in the search state is steered by $u_i$, which is defined as

$$u_i = f_i^g + f_i^d + f_i^s, \tag{2}$$

where $f_i^g$ is the gradient-based term for regulating distances among nodes, $f_i^d$ is the velocity consensus term aims to match the velocities of nodes [17], and $f_i^s$ is the selfishness term which encourages nodes to regions where have not been visited for long time. The gradient-based term $f_i^g$ [17] is given as

$$f_i^g = \phi_\alpha(\|q_j - q_i\|_\sigma) n_{ij}, \tag{3}$$

where the $\sigma$-norm of a vector is defined as $\|y\|_\sigma = \left[\sqrt{1+\epsilon\|y\|^2} - 1\right]/\epsilon$, $n_{ij} = (q_j - q_i)/\sqrt{1+\epsilon\|q_j - q_i\|^2}$ and $\epsilon \in (0,1)$. The action function $\phi_\alpha(z)$ [17] is defined as

$$\phi_\alpha(z) = \phi(z - d_\alpha)\rho_h\left(\frac{z}{r_\alpha}\right), \tag{4}$$

where $d_\alpha$ is a constant, $r_\alpha = \|r_c\|_\sigma$, and

$$\phi(x) = \frac{1}{2}\left[(a+b)\sigma_1(x+c) + (a-b)\right], \tag{5}$$

with $\sigma_1(w) = w/\sqrt{1+w^2}$ and the parameters $a$, $b$, $c$ satisfy $0 < a \leq b$ and $c = |a-b|/\sqrt{4ab}$ [15]. Furthermore, the bump function $\rho_h(v)$ in (4) is expressed as [17]

$$\rho_h(v) = \begin{cases} 1, & \text{if } v \in [0,h), \\ \frac{1}{2}\left[1 + \cos\left(\frac{\pi(v-h)}{1-h}\right)\right], & \text{if } v \in [h,1], \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where $h \in (0,1)$. The velocity consensus term $f_i^d$ [17] is given as

$$f_i^d = (p_j - p_i) a_{ij}(q_i, q_j),$$

where $p_j$ is the velocity of node $j$, and the spatial adjacency matrix $a_{ij}(q_i, q_j)$ [17] is expressed as

$$a_{ij}(q_i, q_j) = \rho_h\left(\frac{\|q_j - q_i\|_\sigma}{r_\alpha}\right) \quad i \neq j.$$

The selfishness term $f_i^s$ is given as

$$f_i^s = c_1(q_s - q_i), \tag{7}$$

where $c_1$ is a positive constant, and $q_s$ is the next searching location of node $i$. Here, we apply the approach proposed in [20] to determine $q_s$ for node $i$. In that approach, each node records its last visiting time of cells in the AoI on its own information map. Apart from updating the information on cells that they have actually visited, nodes also can exchange information via local communications with nearby nodes to update their records. Then, the location with the longest time of not being visited in the database of node $i$ will be chosen as $q_s$.

### C. Nodes in the Traveling State

A node in the *traveling* state traverses along its planned path derived from its static routing table until its distance to the target lies within a predefined threshold $\theta_k$. The static routing table is obtained using the A*IL method based on the mobility map. When a target or node moves across terrain patches, the path needs to be updated accordingly.

In this work, only the intersection points are recorded in the static routing table. Once a node decided to approach a target, it consults its static routing table and visits the nearest intersection point. When a node enters a new terrain patch, it will select the next intersection point and proceed. The node will therefore visit the intersection points one-after-another until it meets the conditions mentioned above.

### D. Nodes in the Monitoring State

When a node is at proximity of its target, it will switch to the *monitoring* state. Nodes in the *monitoring* state perform coordinated motions and form small groups around targets. A node $i$ in its *monitoring* state is steered by $u_i$ according to

$$u_i = f_i^g + f_i^d + f_i^t,$$

where the navigational feedback term $f_i^t$ which steers node $i$ to follow the target $k$ is given as

$$f_i^t = \sum_{k=1}^{m} \frac{c_2(q_k^t - q_i) + c_3(p_k^t - p_i)}{n_k}, \tag{8}$$

with $c_2$ and $c_3$ being positive constants; $m$ being the total number of targets; $q_k^t$ and $p_k^t$ being the location and the velocity of target $k$, respectively.

## V. Evaluations

### A. Setup

Two $200 \times 200$ m$^2$ terrains with and without obstacles were used for evaluating the performances of the proposed path-planning-enabled semi-flocking algorithm, as shown in Figs. 3(a) and 3(b), respectively. Each mobility map consists of 30 terrain patches with 7 different maximum allowed speed values, i.e., 0 (or 1), $5, 10, \ldots, 30$ m/s. Fig. 4 show the snapshots taken at time zero, 10-th second, and 20-th second respectively after executing the proposed path-planning-enabled semi-flocking algorithm with 30 nodes and

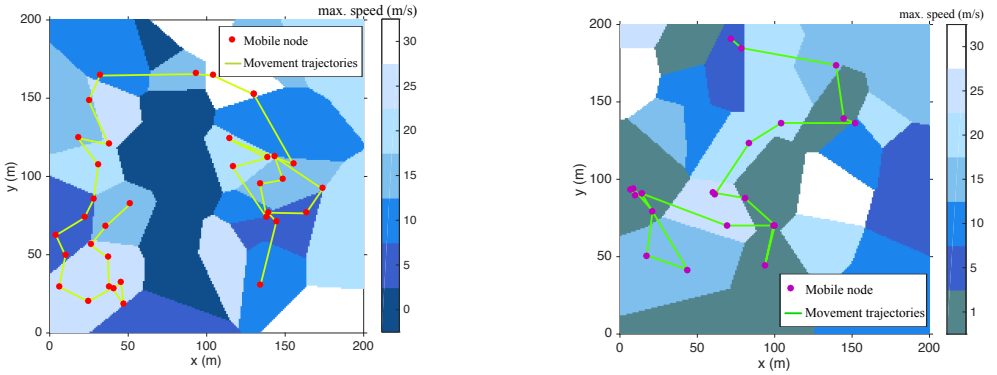



Fig. 3. Motion patterns of 2 targets in terrains (a) with obstacles and (b) without obstacle, respectively.

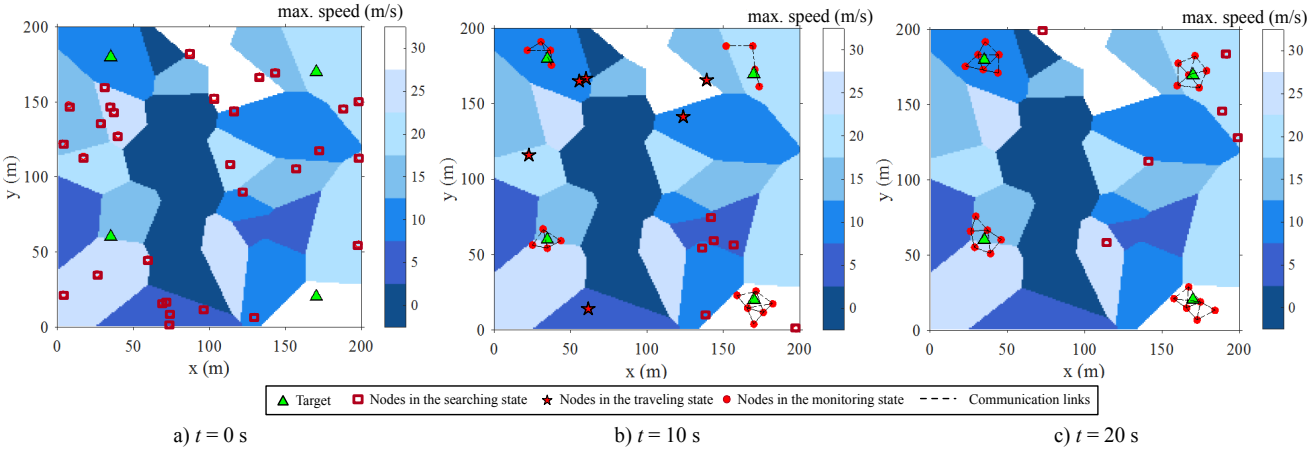



Fig. 4. Snapshots taken at (a) time zero, (b) the 10-th second and (c) the 20-th second, after executing the proposed path-planning-enabled semi-flocking algorithm with 30 nodes and having 4 targets in the terrain.

4 targets on the terrain with obstacles. The maximum number of monitoring nodes required for each target is 6. Simulations were conducted in MATLAB on a computer with a 2.67 GHz Intel i5 processor, 8 GB memory, and Windows 10 operating system. Fig. 4(a) shows that at time zero, all nodes are at the *searching* state and are seeking for targets. In Fig. 4(b), some nodes have detected the targets at the 10-th second and have informed nearby nodes to switch to the *traveling* state. Finally in Fig. 4(c), nodes form small groups around each target right after 20 seconds while other nodes continue to search the AoI for newly emerging targets.

Initial positions of nodes and targets were selected uniformly at random within the terrain except the obstructed areas (i.e., where speed limit equals 0 m/s), while the initial speeds of nodes and targets were selected uniformly at random in $[-10, 10]$ ms$^{-1}$ for both x- and y-directions. We adopt the Brownian motion as the movement pattern of each target. The speed of a node cannot be higher than the corresponding maximum allowed speed of the patch. In the following tests, the number of nodes is 24 and the number of targets in the terrain varies from 1 to 6. The following parameters remained fixed in all tests: $a = b = 5$ for $\phi(x)$, $h = 0.2$, $c_1 = 0.85$, $c_2 = 2$, $c_3 = 2$, $\theta_k = 15$ m, $\epsilon = 0.1$, $r_s = 10$ m, and $r_c = 18$ m [15], [16], [20]. The maximum and minimum numbers of monitoring nodes required for each target are both 3.

## B. Results

The first set of tests was conducted to compare the instantaneous area coverages of MSNs with different flocking-based algorithms, which are measured as the ratio of the union sensing coverage to the area of the AoI at a particular instance. Figs. 5(a) and 5(b) show the average instantaneous area coverage against the number of targets in the AoI with and without obstacles, respectively. It can be observed that the instantaneous coverage of MSNs with semi-flocking algorithms declines with the increasing number of targets while those with the anti-flocking algorithm in [20] remains unchanged. It is because anti-flocking-controlled MSNs only focus on providing coverage to the AoI. Without the need to track targets as in semi-flocking-controlled MSNs, anti-flocking-controlled MSNs can achieve a higher and more stable coverage even when the number of targets increases. In other words, MSNs with semi-flocking algorithms are trading-off their sensing coverage with higher target tracking capabilities.

Furthermore, it is observed that MSNs with the proposed algorithm can maintain a higher instantaneous area coverage compared than those with the semi-flocking algorithms in [15]

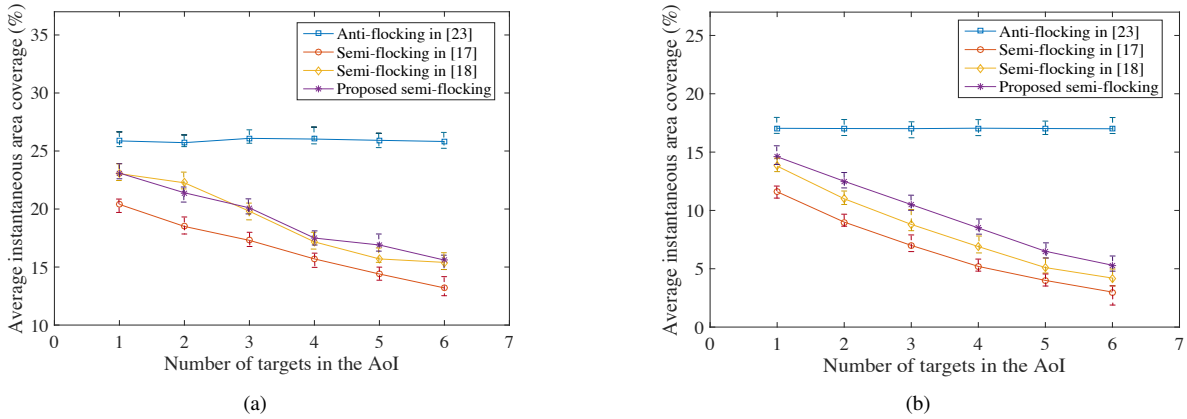



Fig. 5. Average instantaneous area coverage of MSNs operating on the terrain (a) with obstacles and (b) without obstacles. In the experiments, each MSN comprises 24 nodes. All the data points are the results of averaging over 50 executions.

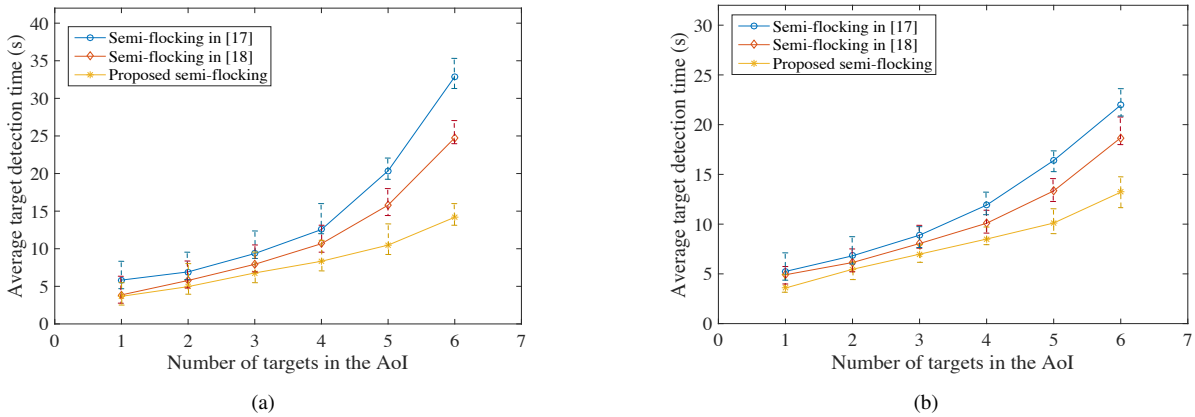



Fig. 6. Average multi-target detection time of MSNs operating on the terrain (a) with obstacles and (b) without obstacles. In the experiments, each MSN comprises 24 nodes. All the data points are the results of averaging over 50 executions.

and [16]. The reason is that each node under the algorithm in [15] is controlled based on the data of the 8 neighboring cells in the terrain. With such a strategy, nodes can easily get trapped in local regions. In contrast, nodes controlled by the algorithm in [16] or the proposed algorithm can exchange information with nearby nodes. Thus, nodes can acquire more knowledge about the overall operating environment and can decide to visit regions which have not been covered for the longest time. However, nodes controlled by the algorithm in [16] move along straight lines to reach their targets which can lead nodes into obstacles or areas with extremely low speed limit. In the proposed algorithm, nodes are capable of being guided along time-efficient paths toward their targets and maintain a relatively high instantaneous area coverage. According to Figs. 5(a) and 5(b), MSNs operating on the terrain with obstacles can achieve relatively higher instantaneous area coverage compared to those without obstacles, because the obstructed areas are excluded when calculating the ratio of visited areas to the total AoI.

The second set of tests was conducted to study the targets detection capabilities of MSNs with the proposed algorithms in AoI with/without obstacles. Similar to [15], it is considered that a target is 3-covered when it is tracked by 3 nodes simultaneously. Nodes under the anti-flocking algorithm in [20] are not able to track targets. Therefore, only the algorithms in [15] and [16] are compared with the proposed algorithm. In Figs. 6(a) and 6(b), the average targets-detection time is plotted versus the number of targets in terrains with and without obstacles. It is obvious that MSNs under the proposed path-planning-enabled semi-flocking algorithm requires a shorter time to successfully detect all the targets compared to MSNs with [15] and [16]. Under the algorithms in [15] and [16], nodes are instructed to navigate along straight paths toward the targets and therefore some nodes could be guided toward obstacles or areas with extremely low speed limits. In comparison, nodes under the proposed algorithm avoid obstacles and areas with extremely low speed limits, and can maneuver along time-efficient paths toward the targets. Thus nodes under the proposed algorithm can quickly reach the targets and then form small groups around them. Furthermore, it is obvious that MSNs under the semi-flocking algorithms in [15] and [16] in terrains with obstacles require longer target detection time compared to those in terrains without obstacles. The reason is same as above.

The third set of tests was conducted to analyze the average travel time of a node under the control of different algorithms. "Travel time" is defined as the time duration when a node is in the *traveling* state, i.e., traveling to reach a target. An

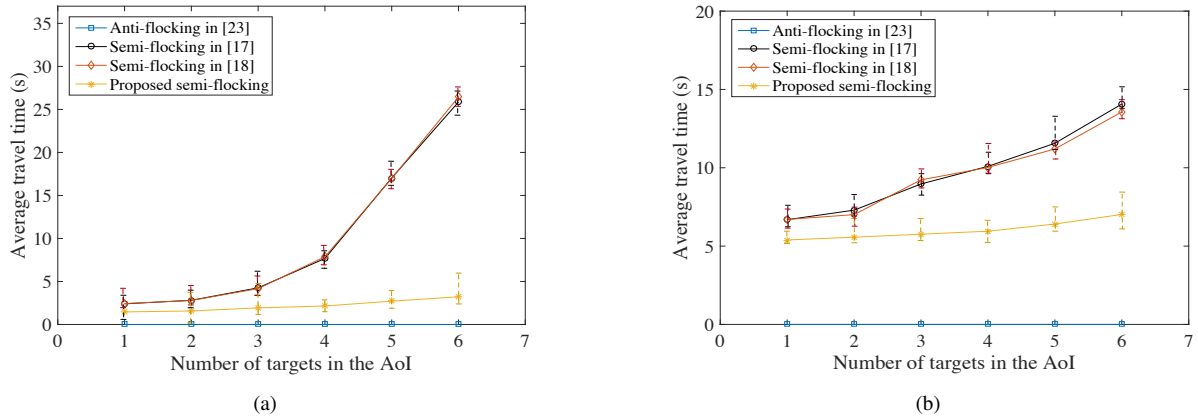



Fig. 7. Average travel time of nodes operating on the terrain (a) with obstacles and (b) without obstacles. In the experiments, each MSN comprises 24 nodes. All the data points are the results of averaging over 50 executions.

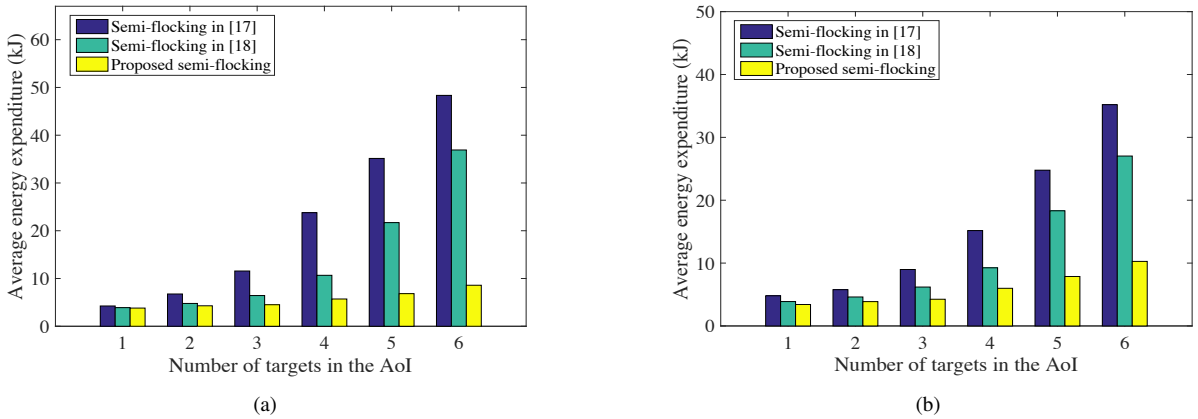



Fig. 8. Average energy expenditure of MSNs for detecting all targets on the terrain (a) with obstacles and (b) without obstacles. In the experiments, each MSN comprises 24 nodes. All the data points are the results of averaging over 50 executions.

anti-flocking-controlled node can only stay in the *searching* state and so its travel time is $\infty$. According to the test results presented in Figs. 7(a) and 7(b), the proposed algorithm can remarkably shorten the average travel time because a node can traverse to its target via a time-efficient path. Furthermore, in the proposed algorithm, via information exchanges with other nodes, a node can make better state-switching decisions based on its traveling costs to different targets compared with their peers. This design can avoid unnecessary traversals if a better candidate exists.

The last set of tests was conducted to compare the energy expenditure of MSNs in finding all targets in the AoI with/without obstacles. According to the results in Figs. 8(a) and 8(b), MSNs controlled by the algorithms in [15] and [16] need to spend significantly higher energy to detect all targets compared to the proposed algorithm. Due to the reasons mentioned above, nodes under the control of the algorithm in [15] or [16] need longer times to detect and reach the targets. However, nodes controlled by the proposed algorithm can detect all targets more quickly with the help of information exchanges. The nodes are also able to move along time-efficient paths toward their targets. Hence, MSNs with the proposed algorithm can consume less energy in finding all targets and ultimately prolong the network lifetime. Furthermore, MSNs with the proposed algorithm are able to avoid obstructed areas and reach their targets via time-efficient paths.

## VI. Discussion

Under the control of semi-flocking algorithm in [15], nodes in *searching* state use a centralized mechanism to record the visiting information on cells in the AoI. Let $N_{\rm s}$ be the number of nodes in *searching* state. Let $N_{\rm c}$ and $N_{\rm a}$ be the number of cells of the AoI and the number of targets in the AoI, respectively. Thus, the total communication load for both area coverage and target tracking can be kept within $O(N_{\rm s})$ since each node needs to upload its own information to a control center and then receives updated information. The total computational load can be kept within $O[N_{\rm s}(N_{\rm c}+N_{\rm a})]$. For nodes in *monitoring* state, they only need to upload the target tracking information to the center and receive updated information. Let $N_{\rm m}$ be the number of nodes in *monitoring* state. Thus, the communication load is kept within $O(N_{\rm m})$ and the computational load is kept within $O(N_{\rm m}N_{\rm a})$.

Nodes under the control of the semi-flocking algorithm in [16] apply a distributed mechanism to make decisions on area searching and target tracking. Each node can independently process its information of visiting time on cells of the AoI and target information. Let $N_{\rm n}^i$ be the number of neighboring

nodes of node $i$. For nodes operating in *searching* state, their total communication load on visiting time information on each cell of the AoI is given as $\sum_{i=1} N_{\mathrm{n}}^{i}$. Nodes need to update the visiting time information on cells that they have actually visited, and they can exchange information via local communications with nearby nodes to update their records. Thus, the computational load of updating the visiting time information is calculated as $\sum_{i=1}(N_{\mathrm{c}} + N_{\mathrm{n}}^{i} N_{\mathrm{c}})$. Nodes in *searching* state also exchange information of targets. The computational load for sharing target information with nearby nodes is calculated as $\sum_{i=1}(N_{\mathrm{n}}^{i} N_{\mathrm{a}})$. The total computational load for nodes in *searching* state is therefore calculated as $\sum_{i=1}(N_c + N_n^i N_{\mathrm{c}} + N_{\mathrm{n}}^{i} N_{\mathrm{a}})$. The worst case is that all nodes in the AoI can connect with each other, which is calculated as $O[N_{\mathrm{s}}^{2}(N_{\mathrm{c}} + N_{\mathrm{a}})]$. For nodes in *monitoring* state, they only need to exchange target information with their peer nodes. Therefore, the communication load is calculated as $\sum_{i=1} N_{\mathrm{n}}^{i}$ and the computational load is given as $\sum_{i=1}(N_{\mathrm{n}}^{i} N_{\mathrm{a}})$. The complexity under the worst scenario is therefore equal to $O(N_{\mathrm{m}}^{2} N_{\mathrm{a}})$ when all nodes connect with each other.

Since nodes under the control of the proposed semi-flocking algorithm adopt the same mechanism to update their information as that in [16] and computational loads are calculated in the same way. However, there are nodes in *traveling* state which need to store their off-line routing tables. It is not necessary for these nodes to exchange information with other nodes. They only need to access their off-line routing tables. They need to read their off-line routing tables which record the information of intersection points on the planned path. Let $N_{\mathrm{t}}$ be the number of nodes in *traveling* state. Let $N_{\mathrm{b}}^{i}$ be the size of an off-line routing table for storing planned path of node $i$. Furthermore, let $N_{\mathrm{p}}$ denote the total number of patches of the mobility map. Therefore, computational load of reading tables can be identified as $\sum_{i=1} N_{\mathrm{b}}^{i}$. The worst-case computational load can be given as $O(N_{\mathrm{t}} N_{\mathrm{p}})$ which occurs when a planned path connects all the patches in the AoI.

Compared with the algorithms presented in [15] and [16], the main contributions of this paper are an A* IL path planning algorithm and a state transition model for mobile nodes. The proposed algorithm considers the cost of maneuvering nodes in the AoI and incorporates path planning in its mechanism. A path planning enabled semi-flocking approach for MSNs in monitoring multiple moving targets is proposed in this paper. An A*-based search algorithm is presented to rapidly generate time-efficient paths based on a mobility map that incorporates the speed constraints in the given AoI. Nodes are therefore able to navigate away from obstacles, avoid slow areas, reach targets via time-efficient paths, and form small groups around targets to provide the required sensing coverage. On the other hand, in order to improve coordination and cooperation among nodes, a state transition model is designed for nodes to switch among the searching state, traveling state, and monitoring state.

## VII. Conclusion

MSNs can monitor multiple moving targets in large sensing areas. Many existing MSNs control algorithms, however, do not consider the cost of maneuvering nodes in the AoI and do not incorporate path planning in their mechanisms. In this work, a path planning enabled semi-flocking approach for MSNs in monitoring multiple moving targets is proposed to tackle these challenges. An A*-based search algorithm is presented to rapidly generate time-efficient paths based on a mobility map that incorporates the speed constraints in the given AoI. Nodes are therefore able to navigate away from obstacles, avoid slow areas, reach targets via time-efficient paths, and form small groups around targets to provide the required sensing coverage. MSNs under the proposed algorithms can detect multiple targets faster with lower traveling costs and have a better practicability in real-life monitoring applications.

## References


[1] H. Mahboubi, K. Moezzi, A. G. Aghdam, K. Sayrafian-Pour, and V. Marbukh, "Distributed deployment algorithms for improved coverage in a network of wireless mobile sensors," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 163–174, 2014.

[2] Y. Zhang and Z. Wang, "Optimal RFID deployment in a multiple-stage production system under inventory inaccuracy and robust control policy," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3230–3242, 2018.

[3] Y. Toda and N. Kubota, "Self-localization based on multiresolution map for remote control of multiple mobile robots," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1772–1781, 2013.

[4] C.-F. Cheng and C.-W. Wang, "The target-barrier coverage problem in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1216–1232, 2018.

[5] P. Singh, P. Agrawal, H. Karki, A. Shukla, N. K. Verma, and L. Behera, "Vision-based guidance and switching-based sliding mode controller for a mobile robot in the cyber physical framework," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 1985–1997, 2018.

[6] J. Chen, M. B. Salim, and M. Matsumoto, "A single mobile target tracking in voronoi-based clustered wireless sensor network," *Journal of Information Processing Systems*, vol. 7, no. 1, pp. 17–28, 2011.

[7] M. Bocca, O. Kaltiokallio, N. Patwari, and S. Venkatasubramanian, "Multiple target tracking with rf sensor networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1787–1800, 2014.

[8] A. Milan, K. Schindler, and S. Roth, "Multi-target tracking by discrete-continuous energy minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2054–2068, 2016.

[9] S. Martínez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 4, pp. 661–668, 2006.

[10] W. Zhang and G. Cao, "DCTC: dynamic convoy tree-based collaboration for target tracking in sensor networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1689–1701, 2004.

[11] P. M. Djuric, M. Vemula, and M. F. Bugallo, "Target tracking by particle filtering in binary sensor networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2229–2238, 2008.

[12] Z. Wang and D. Gu, "Cooperative target tracking control of multiple robots," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3232–3240, 2012.

[13] H. P. Gupta, S. Rao, A. K. Yadav, and T. Dutta, "Geographic routing in clustered wireless sensor networks among obstacles," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2984–2992, 2015.

[14] H. Mahboubi, W. Masoudimansour, A. G. Aghdam, and K. Sayrafian-Pour, "An energy-efficient target-tracking strategy for mobile sensor networks," *IEEE Transactions on Cybernetics*, vol. 47, no. 2, pp. 511–523, 2017.

[15] S. H. Semnani and O. A. Basir, "Semi-flocking algorithm for motion control of mobile sensors in large-scale surveillance systems," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 129–137, Jan 2015.

[16] W. Yuan, N. Ganganath, C.-T. Cheng, G. Qing, and F. C. Lau, "Semi-flocking-controlled mobile sensor networks for dynamic area coverage and multiple target tracking," *IEEE Sensors Journal*, vol. 18, no. 21, pp. 8883–8892, 2018.

[17] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006.

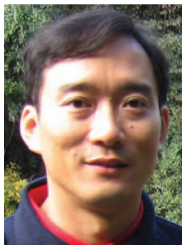
[18] H. Su, X. Wang, and Z. Lin, "Flocking of multi-agents with a virtual leader," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 293–307, 2009.
[19] Y.-Q. Miao, A. Khamis, and M. S. Kamel, "Applying anti-flocking model in mobile surveillance systems," in *International Conference on Autonomous and Intelligent Systems (AIS)*. IEEE, 2010, pp. 1–6.
[20] N. Ganganath, C.-T. Cheng, and C. K. Tse, "Distributed antiflocking algorithms for dynamic coverage of mobile sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1795–1805, 2016.
[21] L. E. Parker, K. Fregene, Y. Guo, and R. Madhavan, "Distributed heterogeneous sensing for outdoor multi-robot localization, mapping, and path planning," in *Multi-Robot Systems: From Swarms to Intelligent Automata*. Springer, 2002, pp. 21–30.
[22] F. K. Karnadi, Z. H. Mo, and K.-c. Lan, "Rapid generation of realistic mobility models for vanet," in *Wireless Communications and Networking Conference*. IEEE, 2007, pp. 2506–2511.
[23] W. Yuan, N. Ganganath, C.-T. Cheng, Q. Guo, and F. C. M. Lau, "A consistent heuristic for efficient path planning on mobility maps," in *International Symposium on A World of Wireless Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2017, pp. 1–5.

**Wanmai Yuan** received the B.Eng. degree in communication engineering from Xidian University, Xi'an, China, in June 2014, and the Ph.D. degree in electronics and information engineering from Harbin Institute of Technology, China in July 2019. He was a recipient of the National Scholarship in 2018 during his Ph.D. studies. He also received the Ph.D. degree in electronic and information engineering, the Hong Kong Polytechnic University, Hong Kong in Sep 2019. From 2018 to 2019, he was a Ph.D. visiting student in the Department of Electrical and Computer Engineering, University of Toronto. Since July 2019, he has been an engineer in the China Academy of Electronics and Information Technology, Beijing, China. His main research interests include flocking control and formation control for UAVs.

**Nuwan Ganganath** received the B.Sc. (Hons) degree with first class honors in electronics and telecommunication engineering from the University of Moratuwa, Sri Lanka, in 2010, the M.Sc. degree in electrical engineering from the University of Calgary, Canada in 2013, and the Ph.D. degree in electronic and information engineering from the Hong Kong Polytechnic University, Hong Kong in 2016. He is currently an Adjunct Research Fellow at the School of Engineering at the University of Western Australia, Australia.

**Chi-Tsun Cheng** received the B.Eng. and M.Sc. degrees from the University of Hong Kong in 2004 and 2005, respectively, and the Ph.D. degree from the Hong Kong Polytechnic University in 2009. From 2010 to 2011, he was a Post-Doctoral Fellow at the Department of Electrical and Computer Engineering, the University of Calgary. From 2012 to 2018, he was a Research Assistant Professor at the Department of Electronic and Information Engineering, the Hong Kong Polytechnic University. Since June 2018, he has been a Senior Lecturer at the Department of Manufacturing, Materials and Mechatronics, RMIT University, Melbourne, Australia. He serves as Associate Editors for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II, IEEE ACCESS, and IEICE NONLINEAR THEORY AND ITS APPLICATIONS. His research interests include Wireless Sensor Networks, Internet of Things, Industry 4.0 Technologies, Cloud Computing, and Additive Manufacturing.

**Guo Qing** received the B.Eng. degree in radio engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 1985, and the M.Eng. and Ph.D. degrees in information and communication engineering from Harbin Institute of Technology, Harbin, China, in 1990 and 1998. He is currently a professor and the dean of the School of Electronics and Information Engineering, Harbin Institute of Technology. His research interests include satellite communications, deep space communications,wireless transmission and broadband multimedia communication techniques.



**Francis C.M. Lau** received the BEng (Hons) degree in electrical and electronic engineering and the PhD degree from King's College London, University of London, UK. He is a Professor and Associate Head at the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. He is also a Fellow of IET and a Senior Member of IEEE. He is the co-author of two research monographs. He is also a co-holder of five US patents. He has published about 300 papers. His main research interests include channel coding, cooperative networks, wireless sensor networks, chaos-based digital communications, applications of complex-network theories, and wireless communications. He was the Chair of Technical Committee on Nonlinear Circuits and Systems, IEEE Circuits and Systems Society in 2012-13. He served as an associate editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II in 2004-2005 and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I in 2006-2007, and IEEE CIRCUITS AND SYSTEMS MAGAZINE in 2012-2015. He has been a guest associate editor of INTERNATIONAL JOURNAL AND BIFURCATION AND CHAOS since 2010 and an associate editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II since 2016. He served as a General co-chair of International Symposium on Turbo Codes and Iterative Information Processing 2018.

**Yanjie Zhao** received the B.Sc. degree from Tsinghua University, China in 2006, the M.Sc. degree from Colorado State University, United States in 2007, and the Ph.D. degree from Purdue University, United States in 2012, all in Physics. From 2012 to 2014, he was a Post-Doctoral Researcher with Purdue University. He is currently the director of Intelligent Systems Research Institute of China Academy of Electronics and Information Technology. He is also a member of advisory board of Chinese National Next-Generation AI Project, committee of Intelligent Unmanned Systems of Chinese Institute of Electronics, committee of Navigation Guidance and Control of Chinese Society of Aeronautics and Astronautics, committee of UAVs Autonomous Control of Chinese Association of Automation, and the board of directors of Chinese Institute of Command and Control.