Virrankoski and Savvides [14] proposed a topology adaptive spatial clustering (TASC) for sensor networks. TASC is a distributed algorithm to partition the network into subgroups (clusters) without the knowledge of the number of clusters, cluster size and node coordinates. Karger and Stein [15] presented an approach to find the minimum cuts in undirected graphs. This approach is based on the fundamental principle that the edges in a graph's minimum cut form an extremely small fraction of the graph's edges. To do that they gave a randomized, strongly polynomial algorithm that finds the minimum cut in an arbitrarily weighted undirected graph with high probability.Derbel and Mosbah [16] proposed a linear time distributed algorithm for decomposing a graph into a disjointed set of clusters. This algorithm is parallel in its nature. In [17], [18], Goebels *et al*. presented a neighborhood-based strategy, a border switch strategy, and an exchange target strategy for the partitioning of large sets of agents onto multiple groups. In summary, the previous works solved the graph partitioning problem in both centralized and decentralized fashions, but in the decentralized way they are usually based on the density of node's distribution. Hence the number of nodes in each sub-group is different.

The rest of this paper is organized as follows. In Section II we present the flocking control algorithm for single target tracking and observing. Section III presents the dynamic multiple targets tracking and observing algorithm. Section IV presents the experimental test results. Finally, Section V concludes this paper.

II. Flocking control for single target tracking and observing

To describe a dynamic topology of flocks or swarms we consider a dynamic graph $G(\vartheta, E)$ consisting of a vertex set $\vartheta = \{1, 2..., n\}$ and an edge set $E \subseteq \{(i, j) : i, j \in \vartheta, i \neq j\}$. In this topology each vertex denotes one member of the flock, and each edge denotes the communication link between two members.

Let $q_i, p_i \in R^m$ $(m = 2, 3)$ be the position and velocity of node i, respectively. We know that during the motion of sensors, the relative distance between them may change, hence the neighbors of each sensor also change. Therefore, we can define a set of neighbors of sensor i at time t as follows:

$$N_i(t) = \left\{ j \in \vartheta : \|q_j - q_i\| \leq r,\ \vartheta = \{1, 2, ..., n\}, i \neq j \right\} \quad (1)$$

here r is an interaction range (radius of neighborhood circle in the case of two dimensions, $m = 2$, or radius of neighborhood sphere in the case of three dimensions, $m = 3$), and $\|.\|$ is the Euclidean distance.

Now, we consider n sensors moving in an m dimensional Euclidean space. We address the motion control problem for a group of sensors with the objective of dynamic target tracking. In this problem we assume that each sensor has a limited communication range to allow it to communicate with others and a large enough sensing range to make it sense the target. We also assume that each sensor is equipped with sonar or laser sensor that allows it to estimate the position and velocity of the target.

The dynamic equation of each sensor is described as follows:

$$\begin{cases} \dot{q}_i = p_i \\ \dot{p}_i = u_i, \quad i = 1, 2, ..., n. \end{cases} \quad (2)$$

The geometry of flocks is modeled by an α-lattice [2] that has the following condition:

$$\|q_i - q_j\| = d, j \in N_i \quad (3)$$

here d is a positive constant indicating the distance between sensor i and its neighbor j.

The configuration which approximately satisfies the condition (3) is called a quasi α-lattice, i.e. $(\|q_i - q_j\| - d)^2 < \delta^2$, with $\delta << d$.

Firstly, based on Olfati-Saber's flocking algorithm with obstacle avoidance [2] we design a flocking control algorithm with a dynamic γ-agent. In this scenario, the dynamic γ-agent is considered as a moving target.

$$\begin{aligned} u_i \quad = \quad & c_1^{\alpha} \sum_{j \in N_i^{\alpha}} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i^{\alpha}} a_{ij}(q)(p_j - p_i) \\ & + c_1^{\beta} \sum_{k \in N_i^{\beta}} \phi_{\beta}(\|\hat{q}_{i,k} - q_i\|_{\sigma}) \hat{n}_{i,k} + c_2^{\beta} \sum_{k \in N_i^{\beta}} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ & - c_1^{mt}(q_i - q_{mt}) - c_2^{mt}(p_i - p_{mt}). \end{aligned} \quad (4)$$

In this control protocol, the pair $(q_{mt},\ p_{mt})$ is the position and velocity of the moving target respectively. The constants are chosen as $c_1^{\alpha} < c_1^{mt} < c_1^{\beta}$, and $c_2^{\nu} = 2\sqrt{c_1^{\nu}}$. Here c_{η}^{ν} are positive constants for $\forall \eta = 1, 2$ and $\nu = \alpha, \beta, mt$. The $\sigma - norm$, $\|.\|_{\sigma}$, of a vector is a map $R^m \Longrightarrow R_+$ defined as $\|z\|_{\sigma} = 1/\varepsilon[\sqrt{1 + \varepsilon\|z\|^2} - 1]$. $\phi_{\alpha}(z)$ and $\phi_{\beta}(z)$ are the action functions to control the attractive or repulsive forces between sensor i and its neighbor j, and the repulsive force between sensor i and its obstacle k, respectively. n_{ij} and $\hat{n}_{i,k}$ are the vectors along the line to connect the pair $(q_i,\ q_j)$, and the pair $(\hat{q}_{i,k},\ q_i)$, respectively. $a_{ij}(q)$ and $b_{i,k}(q)$ are adjacency matrices. $\hat{q}_{i,k}, \hat{p}_{i,k}$ are the position and velocity of sensor i projecting on the obstacle k, respectively. The set of α neighbors at time t, $N_i^{\alpha}(t)$, is defined the same as $N_i(t)$ in (1), and the set of β neighbors (virtual neighbors) of sensor i at time t with k obstacles is $N_i^{\beta}(t) = \left\{ k \in \vartheta_{\beta} : \|\hat{q}_{i,k} - q_i\| \leq r', \vartheta_{\beta} = \{1, 2, ..., k\} \right\}$ with r' being selected to be bigger than r, in our simulations $r' = 1.2 * r$. More details of the these terms, please see [2].

The dynamic target is defined as follows:

$$\begin{cases} \dot{q}_{mt} = p_{mt} \\ \dot{p}_{mt} = f_t(q_{mt}, p_{mt}). \end{cases} \quad (5)$$

In the control protocol (4), the first two terms are used to control the formation (collision avoidance and velocity matching among sensors). The third and fourth terms are used to allow sensors to avoid obstacles. The last term (negative feedback) is used for target tracking. If it is absent the control will lead to the fragmentation of the sensor network [2].

III. Dynamic multiple targets tracking and observing

In many surveillance applications the sensor networks have to deal with the dynamic situation of targets appearing and disappearing in the field. In the following subsections we first address the problem of sensor network partitioning and then discuss multiple dynamic targets tracking.

A. Sensor network partitioning

To deal with a new emerging target, the sensor network should automatically decompose into equal sub-groups and then each sub-group will be assigned to track one target. For example, consider M targets existing at time t and M sensor groups $(G_1, G_2, ..., G_M)$ which are tracking these targets (each group has about n/M sensors). If the $(M+1)$th target appears then $\frac{n}{M+1}$ sensors should split off from M existing groups to form a new group to track the new target. On the other hand to deal with a disappearing target, the sensors which are tracking this target should split and merge with the existing groups.

As discussed in Section II, the mobile sensor network can be considered as a dynamic graph (dynamic topology). Hence we can apply some graph partitioning algorithms to decompose the graph into sub-graphs (sub-groups). However, some existing methods for graph partitioning are centralized methods, which means that each sensor need global knowledge of the whole network's state to split from the network. There are also some distributed graph partitioning or distributed graph clustering methods, but they are usually based on the density of node's distribution (see *Literature review section*). Hence the size of sub-groups is not predetermined, or the number of sensors in each sub-group is different.

Based the above analysis, this paper proposes a seed growing graph partition (SGGP) algorithm to decide which sensor in the network should track new targets. The main idea of this algorithm is based on seed growing. This means that the mobile sensor which is closest to the new target will initiate the growth of the sensors into a new group by broadcasting the message to its sons in a recursive fashion until the number of sensors in the subgroup is equal to a predetermined threshold (Θ_S). By growing the number of sensors in each generation from the seed sensor (the sensor closest to the new target), the formation of each sub-group is maintained during splitting. This leads to minimized total energy and time consumption.

Assume all mobile sensors already formed a network with an α-lattice configuration (see Figure 1). In this configuration if the sensor has 5 or 6 neighbors (6 is the maximum number of neighbors in this configuration) this sensor will be inside the network. If the sensor has less than or equal to 4 neighbors it will be on the border of the network. This sensor is called a border sensor. Based on this fact, the SGGP algorithm is summarized as follows:

Step 1. Each sensor checks to find how many neighbors it has and decides if it is a border sensor.

Step 2. Each border sensor computes the distance to the new target and forwards this distance information to the other border sensors, and receives the distances from other border sensors.

Step 3. Each border sensor compares its distance with the received distances from other border sensors and finds the sensor with smallest distance to be set as the Seed Sensor (SS).

Step 4. The SS counts its sons and broadcasts the predetermined size of the new group to its sons. If the size of the new group is less than the predetermined size the sons will continue passing the message to their sons. This process is repeated until the size of the new group is equal to the predetermined size.

Remark. In the SGGP algorithm, the number of sons of sensor i is defined as:

$$|S_i| = |N_i| - |F_i| - |DB_i| \tag{6}$$

here $|S_i|$, $|N_i|$, $|F_i|$ and $|DB_i|$ are the number of sons, neighbors, fathers and the direct brothers of sensor i, respectively. For example in Figure 1, SS is the father of sensors 2, 3 and 4. Sensor 3 is the direct brother of sensor 2, hence the sons of sensor 2 are only sensors 5 and 6. Sensor 2 can know sensor 3 being its direct brother because its father (SS) sends a message $\{DB\}$ to tell which sensor is its direct brother. In addition, two or more sensors can have the same son, but if a sensor has the priority $\{P\}$ to count this same son first the remaining sensors will not count this son again. For an example of this situation, sensors 2 and 3 have the same son, sensor 5, but because of its smaller ID sensor 2 receives a message consisting of $\{P\}$ from its father (SS) hence it has priority to count sensor 5 as its son first then it sends the counting number (CN) to its direct brother sensor 3.

Figure 1 shows the message exchange when applying the SGGP algorithm. The slashed green arrows represent the counting number (CN) which is sent after counting, and the solid red arrows represent the message exchange. In this scenario assuming that we have 30 sensors (n=30), and they already formed a network with α-lattice configuration. This sensor network is tracking the current target. When a new target appears, by applying the SGGP algorithm 15 sensors (Θ_S = n/2) split from the network to track the new target with the total distance of all n/2 sensors to the new target being minimized.

B. Multiple dynamic targets tracking

In the multiple targets scenario, we assume that each sensor is integrated with the flocking control algorithm, which deals with each different target (q_{mt_l}, p_{mt_l}) with $l = 1, 2, ..., M$ described as below.

$$\begin{aligned} u_i \;=\; & c_1^{\alpha} \sum_{j \in N_i^{\alpha}} \phi_{\alpha}(\|q_j - q_i\|_{\sigma}) n_{ij} + c_2^{\alpha} \sum_{j \in N_i^{\alpha}} a_{ij}(q)(p_j - p_i) \\ & + c_1^{\beta} \sum_{k \in N_i^{\beta}} \phi_{\beta}(\|\hat{q}_{i,k} - q_i\|_{\sigma}) \hat{n}_{i,k} + c_2^{\beta} \sum_{k \in N_i^{\beta}} b_{i,k}(q)(\hat{p}_{i,k} - p_i) \\ & - c_1^t (q_i - q_{mt_l}) - c_2^t (p_i - p_{mt_l}). \end{aligned} \tag{7}$$

As discussed in Section II, the dynamic target (q_{mt_l}, p_{mt_l}) in (7) is exactly the navigation term to lead the flocks

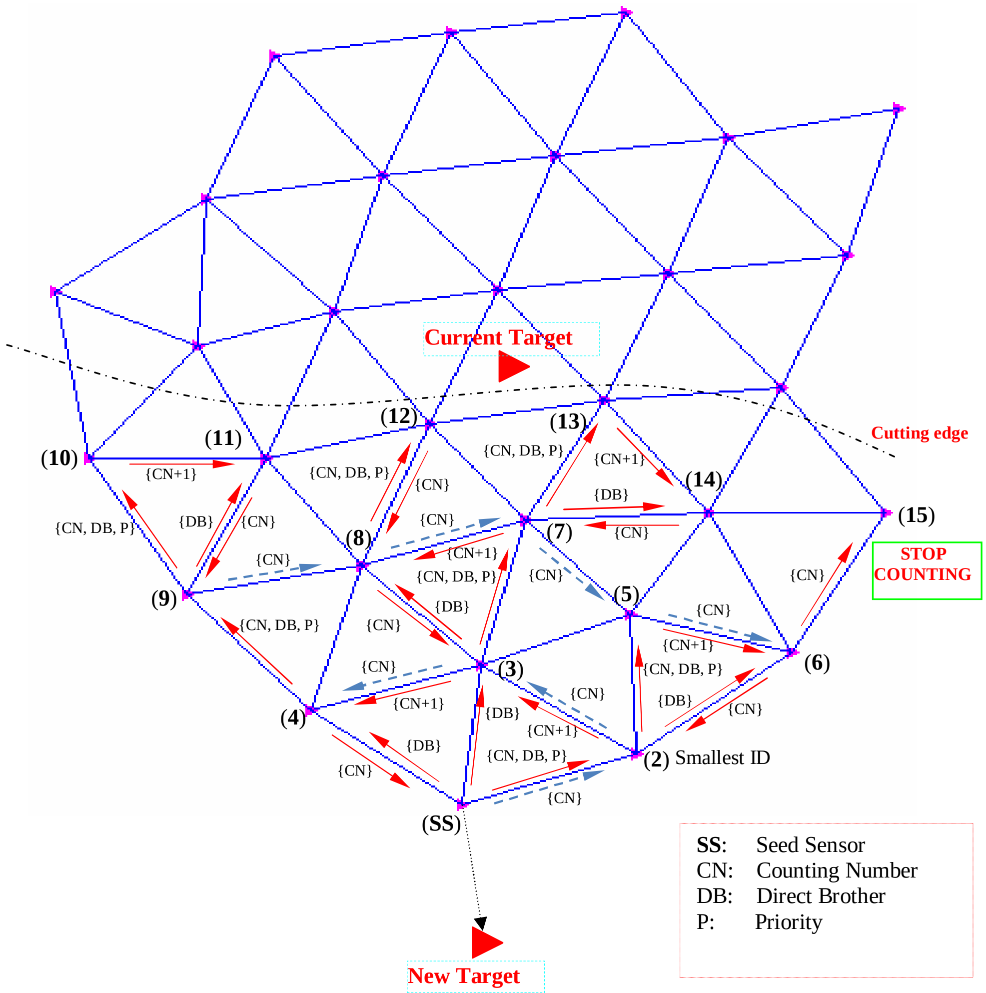

Fig. 1. Example of seed growing graph partition.

(mobile sensors) moving together. Without this term the sensor network leads to fragmentation. This means that if sensor i is assigned to track another target it only need switch to another navigation term. This also means that if the new target appears one by one the sensors which are selected by the SGGP algorithm will switch to another navigation term (another target).

On the other hand in the merging case, for example, three sensor subgroups are tracking three targets. If one of these targets disappears then this subgroup will decompose into two equal groups and each one will merge into one of the remaining subgroups to track the existing target by switching to another navigation term.

IV. Experimental tests

A. Test cases

In this section we will test our algorithm in two different cases of sensor splitting and merging. Parameters used in this simulation are specified as follows:

Case1. Two targets appear one by one and no target disappears.

- Parameters of flocking: Number of sensors = 120 (randomly distributed in the rectangular area with the size of 90x90), the communication range $r = 1.2 * d$ with $d = 7.5$,and $\varepsilon = 0.1$ for the σ-norm.

- Parameters of target movement: The targets move in the sine wave trajectory: For the target 1, $q_{mt_1} = [50+35t,\ 295-35sin(t)]^T$ with $0 \le t \le 8.5$, and for the target 2, $q_{mt_2} = [85+35t,\ 55-35sin(t)]^T$ with $1.26 \le t \le 8.5$, and $\Delta_t = 0.002$ is the step size.

In this case, the SGGP algorithm will be compared with a Random Selection (RS) algorithm. In the RS algorithm when the new target appears a half of the sensors in the network which are tracking the existing target are selected randomly to track the new target.

Case2. Two targets appear one by one and one target disappears.

- Parameters of flocking: these parameters are the same with the Case 1.

- Parameters of target movement: Parameters are set up the same as in Case 1, but the target 1 is set to run in the interval time $0 \le t \le 12.5$, and the target 2 appears at time $t = 1.26$ (at iteration 840) and disappears at time $t = 8.4$ (at iteration 4200).

Figure 2 (a) displays the result of tracking of Case 1 where the targets appear one by one and move in a sine wave trajectory. Firstly, the whole group of 120 mobile sensors form an α-lattice configuration and track target 1. Then, at iteration 840 target 2 appears and the network decides which sensors will split and track this target. By applying the SGGP algorithm, the sensor network automatically decomposes into 2 equal sub-groups (60 sensors in each sub-group). The second sub-group which is closest to target 2 tracks target 2, and the first sub-group keep tracking target 1. The SGGP algorithm allows two sub-groups to maintain their formation when they split. Figure 2(b) represents the error between the average of positions in the whole network and target 1 (from iteration 1 to 839), and the error between the average of positions in sub-group 1 and target 1 (from iteration 840 to the end). Figure 2(c) represents the error between the average of positions in sub-group 2 and target 2. We see that at iteration 840, the average of positions of sensors slightly changes because at this time the average of sensors's positions in sub-group 1 will replace that of the whole network. In this figure we see that all tracking errors are very small in free space. This means that all sensors in the whole network or in each sub-group can surround the target closely to observe it easily. However in the presence of obstacles, the errors are significant because the repulsive forces generated from obstacles push the sensors away from them.

Figures 3 shows the results of tracking in Case 2 where the targets appear one by one and then one disappears. When target 2 appears at iteration 840 the results are similar with Figures 2. When target 2 disappears at iteration 4200 sub-group 2 which is tracking this target will rejoin sub-group 1 and continue to track target 1. The tracking result of the whole group after merging is good with small tracking error between the average of sensors's positions and target 1 in the free space as shown in Figure 3 (b) (from iteration 4200 to the end).

B. Comparison between the SGGP algorithm and the RS algorithm

In this subsection we will compare two algorithms, SGGP and RS, in term of tracking time, formation time, and total distance of all sensors in each sub-group to its target. These comparisons also imply the time consumption and power consumption in each sub-group.

Similar to Figures 2 (a, b, c), Figures 2 (a', b', c') also shows the results of tracking to Case 1 where the targets appear one by one and move in the sine wave trajectory. However, the difference here is that when target 2 appears half sensors in the whole network are split to track this target by using the RS algorithm. With this algorithm two sub-groups do not maintain their formation, and all sensors in each sub-group need certain time to reform a network. This is the main drawback of this algorithm, and some data are collected to compare the SGGP and the RS algorithms which is shown in Table *I*.

Parameters in the Table *I* are computed as follows:

TABLE I

COMPARISON BETWEEN TWO ALGORITHMS SGGP AND RS.

Algorithms	D_{tt} (units)	t_T (s)	t_F (s)
RS (G_1)	1184.7	1.000801	8.345623
RS (G_2)	14194	11.770489	11.125117
SGGP(G_1)	1185.6	1.203569	0.0
SGGP(G_2)	13126	9.007456	0.0

D_{tt} is the total travel distance between all sensors in each group and its target, and it is computed when the network is decomposed into sub-groups to when the average of positions of sensors in each sub-group reaches the target (this is evaluated based on the same condition as used to compute t_T below).

t_T is the tracking time which is computed based on the condition: $\|\frac{1}{n_{G_l}}\sum_{i=1}^{n_{G_l}} q_i - q_{t_l}\| \leq \Theta_T,\ l=1,2$; here n_{G_l} is number of sensors in each sub-group G_1 and G_2 respectively, and Θ_T is a given threshold.

t_F is the formation time representing the time that it costs all mobile sensors to form a network. This formation time is computed based on the following condition:

$Var(\|q_i - q_j\|) = \frac{1}{|E_l|}\sum(\|q_i - q_j\| - \frac{1}{n_{G_l}}\sum_{(i,j)\in E_l}\|q_i - q_j\|)^2 \leq \Theta_3$ with $i,j = 1,2,...,\ n_{G_l}$; $l=1,2$; here Θ_F is a given threshold, and $i \neq j$.

In the RS algorithm, the values of D_{tt}, t_T, and t_F are obtained based on the average value of 50 running times.

Comparison between RS and SGGP algorithms: The maximum of the tracking time and formation time in SGGP algorithm $t^{max}_{SGGP} = max(t_T, t_F)_{G_1} + max(t_T, t_F)_{G_2} = 10.211(s)$ while in RS algorithm $t^{max}_{RS} = 20.1161(s)$, or t^{max}_{SGGP} is 49.28 % less than t^{max}_{RS}. The total distance in SGGP algorithm $D^t_{SGGP} = D^{G_1}_{tt} + D^{G_2}_{tt} = 14311.6(units)$ while in the RS algorithm $D^t_{RS} = 15378.7(units)$, or D^t_{SGGP} is 7% shorter than D^t_{RS}.

In all the above simulation results, all sensors keep their formation (excepting in the case of the RS algorithm) and no collision occurs among them while tracking the moving target, and all sensors avoid obstacles successfully in a narrow space. For more details please see some video files which are available at our ASCC Lab's website.

$http://ascc.okstate.edu/projectshung.html$

V. CONCLUSIONS

This paper develops an approach to flocking control of a mobile sensor network to track and observe multiple dynamic targets. The SGGP algorithm is proposed to solve the problem of splitting/merging the sensor agents. To see the benefit of this algorithm we compared it with a random selection (RS) algorithm, and the results are promising. The maximum of the convergent distance and formation time in the SGGP algorithm is faster than that in the RS algorithm. In addition, the distance in the SGGP algorithm is shorter than that in the RS algorithm. The numerical experimental tests were done with two different cases of splitting and merging sensor agents to demonstrate our theoretical results.

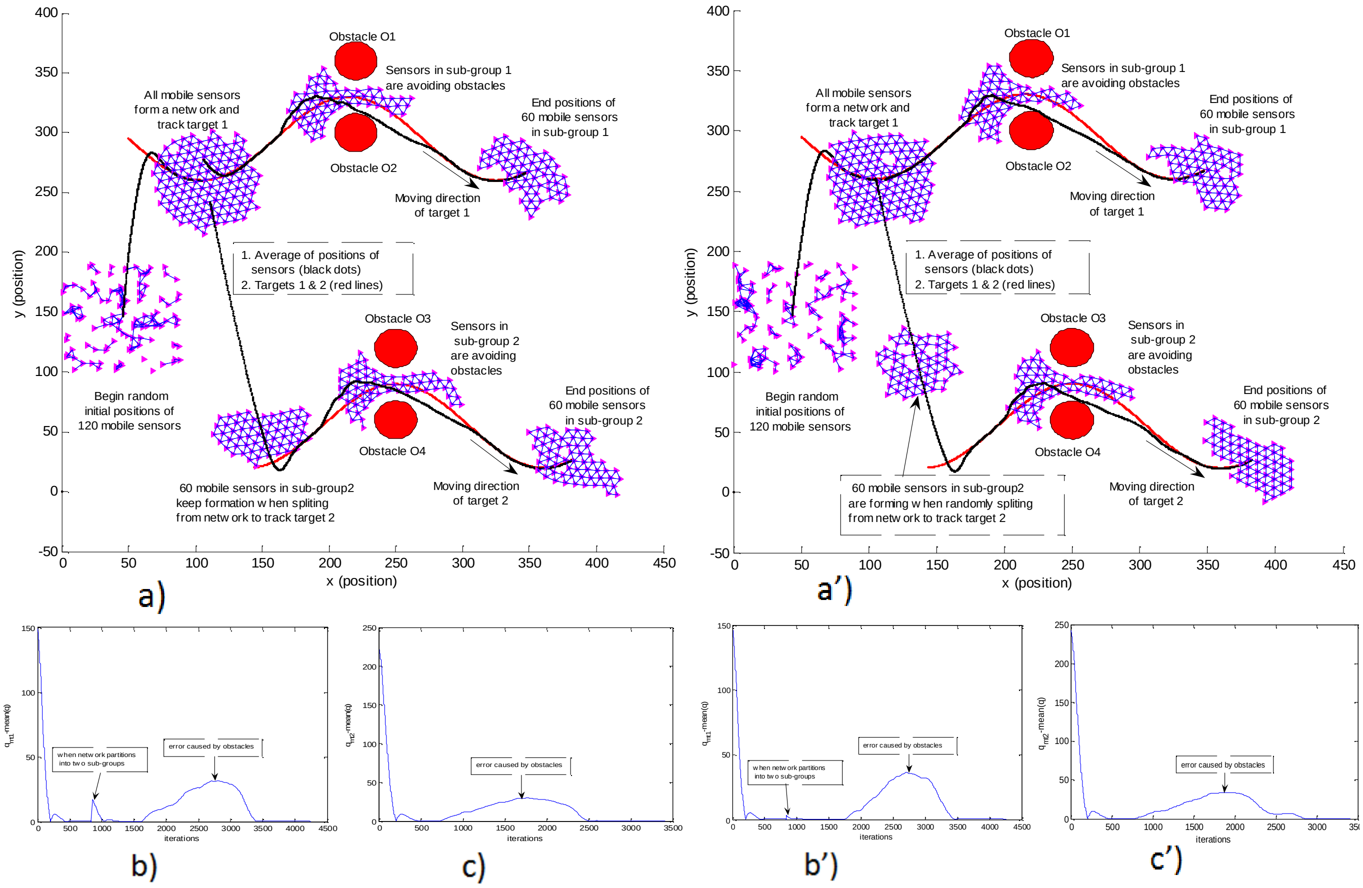

Fig. 2. (a, a')- Snapshots of the beginning initial position of whole group, splitting positions of sub-group 2 and the ending positions of two sub-groups which are tracking the targets moving in the sine wave trajectories, (b, b')- Error between the average of sensors's positions in the whole network and target 1 (iteration 1 to 839), and between the average of sensors's positions in sub-group 1 and target 1 (iteration 839 to the end), (c, c')- Error between the average of sensors's positions in sub-group 2 and target 2. These results are done by using the flocking control algorithm (7) with SGGP and RS algorithms, respectively

References

[1] S. Kamath, E. Meisner, and V. Isler. Triangulation based multi target tracking with mobile sensor networks. *IEEE International Conference on Robotics and Automation*, pages 3283–3288, 2007.

[2] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[3] C. Reynolds. Flocks, birds, and schools: A distributed behavioral model. *Computer Graphics,ACM SIGGRAPH '87 Conference Proceedings, Anaheim, California*, 21(4):25–34, 1987.

[4] H. Su, X. Wang, and Z. Lin. Flocking of multi–agents with a virtual leader part i: with a minority of informed agents. *in Proc. 46th IEEE Conf. Decision and Control*, pages 2937–2942, 2007.

[5] H. Su, X. Wang, and Z. Lin. Flocking of multi–agents with a virtual leader part ii: with a virtual leader of varying velocity. *in Proc. 46th IEEE Conf. Decision and Control*, pages 1429–1434, 2007.

[6] H. Shi, L. Wang, T. Chu, F. Fu, and M. Xu. Flocking of multi–agents with a virtual leader. *in Proc of the 2007 IEEE Symposium on Artificial Life*, pages 287–29, 2007.

[7] Z. Wang and D. Gu. A survey on application of consensus protocol to flocking control. *in Proc of the 12th Chinese Automation and Computing Society Conference in the UK, England*, pages 1–8, 2006.

[8] H. G. Tanner, A. Jadbabai, and G. J. Pappas. Stable flocking of mobile agents, part i: fixed topology. *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 2010–2015, 2003.

[9] H. G. Tanner, A. Jadbabai, and G. J. Pappas. Stable flocking of mobile agents, part ii: dynamic topology. *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 2016–2021, 2003.

[10] A. Regmi, R. Sandoval, R. Byrne, H. Taner, and C. T. Abdallah. Experimental implementation of flocking algorithms in wheeled mobile robots. *American Control Conference*, pages 4917–4922, 2005.

[11] V. Gervasi and G. Prencipe. Coordination without communication: the case of the flocking problem. *Discrete Applied Mathematics*, pages 324–344, 2004.

[12] R. Olfati-Saber. Distributed tracking for mobile sensor networks with information driven mobility. *Proceedings of the 2007 American Control Conference,*, pages 4606–4612, 2007.

[13] C. Bettstetter. The cluster density of a distributed clustering algorithm in ad hoc networks. *IEEE Communications Society*, pages 4336–4340, 2004.

[14] R. Virrankoski and A. Savvides. Tasc: topology adaptive spatial clustering for sensor networks. *IEEE International Conference on Mobile Adhoc and Sensor Systems*, page 10 pp, 2005.

[15] D. R. Karger and C. Stein. A approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, 1996.

[16] B. Derbel and M. Mosbah. A fully distributed linear time algorithm for cluster network decomposition. *Proceeding of Parallel and Distributed Computing and Systems*, 2004.

[17] A. Goebels, H. K. Buning, S. Priesterjahn, and A. Weimer. Multi target partitioning of sets based on local information. *Proceedings of the fourth IEEE Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST).*

[18] A. Goebels, H. K. Buning, S. Priesterjahn, and A. Weimer. Towards online partitioning of agent sets based on local information. *Proceeding of Parallel and Distributed Computing and Networks*, 2005.

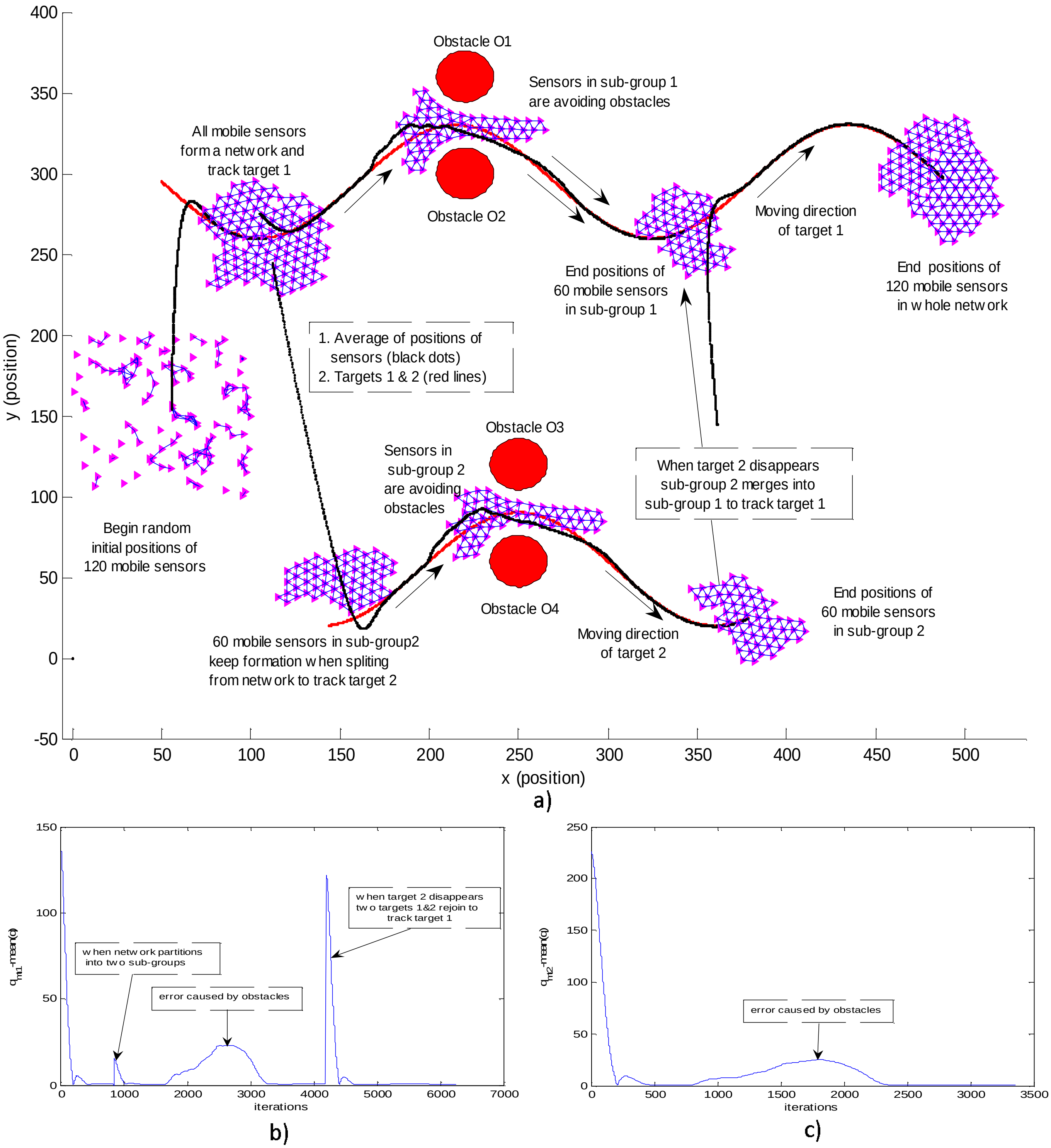

Fig. 3. (a)- Snapshots of the beginning initial position of whole group, splitting positions of sub-group 2 and the ending positions of two sub-groups which are tracking the targets moving in the sine wave trajectories, (b)- Error between the average of sensors's positions in the whole network and target 1 (iteration 1 to 839, and iteration 4200 to the end), and between the average of sensors's positions in sub-group 1 and target 1 (iteration 840 to 4200), (c)- Error between the average of sensors's positions in sub-group 2 and target 2. This result is done by using the flocking control algorithm (7) and SGGP algorithm.