Flying Swarm of Drones Over Circulant Digraph

ZBIGNIEW R. BOGDANOWICZ
Armament Research, Development and Engineering Center, Picatinny Arsenal, NJ USA

Drones are critical to military collaborative engagement, surveillance, and monitoring of areas of interest. They also play an increasingly influential role in commercial applications. We describe a new innovative method for collisionless, survivable, and persistent flying of given k drones over $2n$ waypoints that results in dispersed coverage of a given area of interest, where $n > k \geq 3$ and k divides n. Our method is fundamentally different than any known to date since it does not need sensors to avoid collisions between drones. In addition, the flight of drones does not have to be coordinated.

Manuscript received January 5, 2016; revised February 15, 2017; released for publication May 20, 2017. Date of publication May 29, 2017; date of current version December 5, 2017.

DOI. No. 10.1109/TAES.2017.2709858

Refereeing of this contribution was handled by S. Karaman.

Author's address: Z. R. Bogdanowicz is with Armament Research, Development and Engineering Center, Picatinny Arsenal, NJ 07806-5000 USA E-mail: (zbigniew.bogdanowicz.civ@mail.mil).

I. INTRODUCTION

Unmanned aerial vehicles (UAVs), also known as drones, are becoming truly sophisticated and reliable nowadays for various military operations, e.g., [2], [5], [21]. They are also becoming smaller and less expensive [14], [22], [27]. While many missions were already carried out that involved drones, they usually were focused on a single drone at a time, e.g., [12], [16]. The affordability of deploying tens, hundreds, or even thousands of drones can soon become reality, in which case efficient collaborative operation will be a critical factor and value-added differentiator on battlefields. One such obvious collaborative operation of many drones will be focused on collisionless, persistent flying with space-dispersed coverage of a given area of interest. It would have significant application in surveillance, search, and rescue missions, as well as in collaborative engagement of targets. Many other commercial applications are possible as well. In this paper, we describe a new innovative method for doing that for given k drones and $2n$ waypoints, where $n > k \geq 3$ and k divides n. That is, we provide a method to deploy an arbitrary number of drones for surveillance of an area of interest such that zero coordination between drones is required, collision-free movements are guaranteed, and it is relatively robust to failure of single drones in the sense that the area visited by the failed drone is regularly visited by another drone.

There are a number of papers devoted to collision avoidance control in multiagent networks and formation flights (e.g., [9], [10], [25]) as well as to flying drones, or equivalently to flying UAVs [2], [20], [23], [24]. Many of such papers consider the flying of multiple drones in various formations [1], [3], [11], [13], [15], [17]–[24], meaning in general that the relative distance between the drones remains constant. However, none of them considers flying drones without specific sensors detecting possible collisions with other drones. Our solution is fundamentally different than any other one to date because it does not require the sensors to avoid possible collisions between drones, nor does it require the coordination of flying drones. In particular, our solution is based on flying drones over pairwise disjoint trajectories (i.e., so called three-dimensional (3-D)-deconfliction that we describe in Section V), which assure that the drones will not collide with each other in the air. Otherwise, midair collision avoidance can be avoided by scheduling the flights of drones resulting in 4-D-deconfliction that we also describe in Section V but that would not be as robust and scalable as our presented method. Our solution relies on constructing a special directed graph (see Fig. 1) whose decomposition results in space-dispersed flying of drones in a swarmlike fashion.

A *circulant* digraph $G = G_n(a_1, a_2, \ldots, a_k)$ on n vertices with k pairwise distinct jumps $a_1, a_2, \ldots, a_k$ has vertices $i + a_1, i + a_2, \ldots, i + a_k \pmod n$ adjacent to each vertex i, where each $a_i < n$ [4]. In this paper, we focus on flying multiple drones implied by a circulant digraph on two jumps (i.e., $G = G_n(a_1, a_2)$). Prior work on Hamilton cycles in circulant digraphs [8], [26] as well as on Hamiltonian

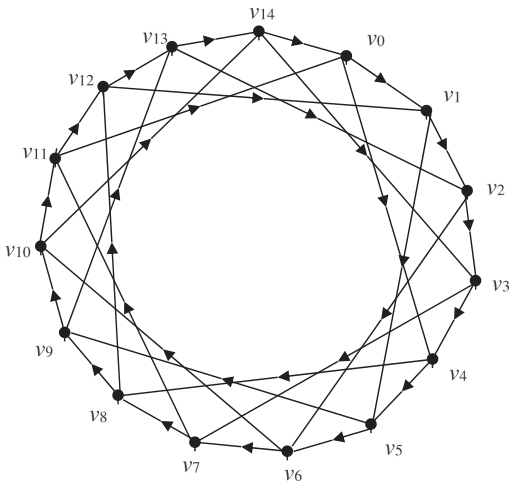

Fig. 1. Example of circulant digraph $G_n(a_1, a_2) = G_{15}(1, 4)$.

decomposition of circulant digraphs [7] has been helpful in solving this problem. In particular, we will show that if each vertex of G is split into two waypoints then it will facilitate a 3-D swarmlike flying of multiple drones without the need for drone-detecting sensors (e.g., sonar, radar, LIDAR, etc.) to avoid collisions with each other. Specifically, the projection of directed routes of k flying drones from 3-D space along z-axis onto xy-plane will induce directed circulant $G_n(1, k-1)$.

The solution presented here has many benefits that we list in Section VI as well as the quite unique following characteristics: We design 3-D disjoint orbits, also known as 3-D-deconfliction that we describe in Section V, for drones around an area of interest. Hence they do not collide even without communication. Our drone-flying solution on the one hand tends to disperse the drones over a given area of interest based on the coverage of $G_n(1, k-1)$, while on the other hand it allows the flying of two drones over each vertex $G_n(1, k-1)$. Hence, for the military applications that use unarmed as well as armed drones, it might mean that our drone flying tends to maximize the coverage/surveillance of the area of interest, while at the same time it can maximize the lethality of the armed drones in respect to the targets that are in that area of interest.

The rest of this paper is organized as follows. In Section II, we give basic notation related to graphs that will be used in this paper. In Section III, we cover the assignment of drones into a circulant digraph and prove that there exists an assignment that allows drones to fly over the pairwise disjoint cycles of identical lengths. In Section IV, we discuss the flying of k drones over circulant digraph $G_n(1, k-1)$ and give the flying rules over digraph H_{n+1} constructed from $G_n(1, k-1)$. In Section V, we prove that flying drones according to the rules introduced in Section IV results in collisionless persistent flying. An algorithm describing our framework of flying and controlling the drones from a single operator control unit (OCU) is discussed in Section VI. Finally, in Section VII, we summarize the accomplishments and importance of this paper.

II. NOTATION RELATED TO GRAPHS

In this paper, we use the following basic notation and terminology related to graphs. Directed graph (i.e., digraph) G on n vertices is denoted by G_n and consists of set of vertices $V(G)$ and set of arcs $A(G)$. In particular circulant digraph G of order n and jumps $a_1, a_2, \ldots, a_k$ is denoted by $G_n(a_1, a_2, \ldots, a_k)$. Notation $a_i a_j \cdots$ denotes a directed path (or cycle) represented by a sequence of arcs in $A(G)$ identified by corresponding jumps $a_i, a_j, \ldots$ in a circulant digraph. By *pairwise disjoint paths (respectively cycles)* in this paper we mean paths (respectively cycles) that do not share any arc with any other path (respectively cycle) in $A(G)$. If there are two waypoints associated with vertex $x \in V(G)$, then, we denote them by x^1, x^2. Closed directed walk over the waypoints starting from x^i is denoted by $x^i y^j z^k \cdots x^i$, where $x, y, z \in V(G), i, j, k \in \{1, 2\}$, and each waypoint is allowed to be visited multiple times. Finally, gcd() represents greatest common denominator of its arguments.

III. ASSIGNMENT OF DRONES TO CIRCULANT DIGRAPH

In this section, we will show that having k drones and a circulant digraph G of certain order there is a way to assign all k drones to k pairwise disjoint cycles of G (i.e., no two cycles would share an arc) that exhaust all the cycles in G. The following results will be useful in proving that. First, the following theorem generalizes the decomposition of a circulant digraph from two Hamilton cycles [7] to the cycles of equal lengths.

THEOREM 3.1 (SEE [6]) A connected circulant digraph $G_n(a_1, a_2)$ can be decomposed into directed cycles of equal lengths of form $a_1 a_2 a_1 a_2 \cdots a_1 a_2$ if $\gcd(n, a_1 + a_2) \geq 2$.

Second, due to Boesch and Tindell [4], we have the following:

THEOREM 3.2 A circulant digraph $G_n(a_1, a_2, \ldots, a_k)$ is connected if and only if $\gcd(n, a_1, a_2, \ldots, a_k) \geq 1$.

It is a direct and straightforward consequence of Theorem 3.2 that we have the following:

COROLLARY 3.3 *A circulant digraph $G_n(a_1)$ has 2-factor of* $\gcd(n, a_1)$ *cycles of equal lengths.*

Based on Theorem 3.1 and Corollary 3.3, we can now state the following:

THEOREM 3.4 A connected circulant digraph $G_n(a_1, a_2)$ can be decomposed into k, $k \geq 2$, directed cycles of the equal lengths of form $a_1 a_2 a_1 a_2 \cdots a_1 a_2$ if $\gcd(n, a_1 + a_2) = k$.

PROOF Suppose $\gcd(n, a_1 + a_2) = k \geq 2$. By Theorem 3.1, $G_n(a_1, a_2)$ can be decomposed into l, $l \geq 2$, cycles of the equal lengths of form $a_1 a_2 a_1 a_2 \cdots a_1 a_2$. Suppose, $l \neq k$. Consider corresponding circulant digraph $H = H_n(a_1 + a_2)$. By Corollary 3.3, H has 2-factor of k cycles, each of length $\frac{n}{k}$. This implies that $G_n(a_1, a_2)$

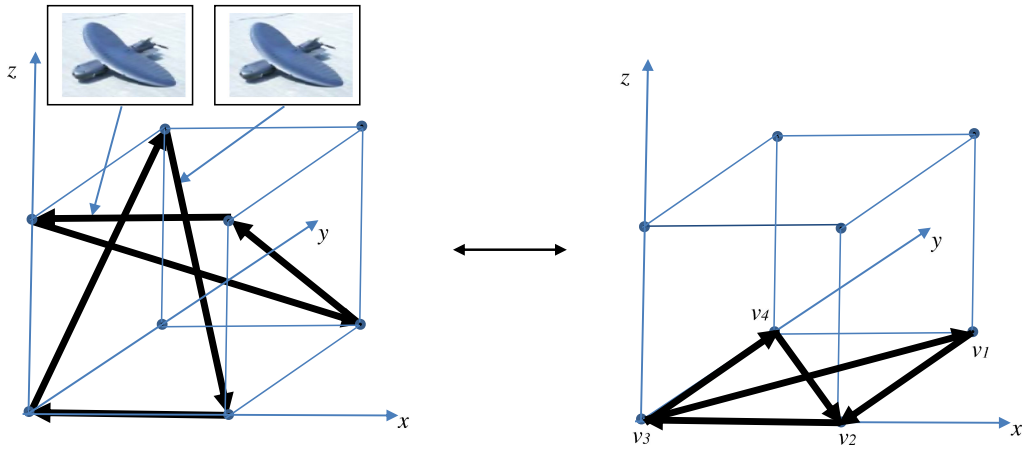

Fig. 2. Example of projecting two drone paths on xy-plane resulting in digraph G.

can be decomposed into k cycles of equal lengths of form $a_1a_2a_1a_2\cdots a_1a_2$—a contradiction, which completes the proof. ■

In particular, if we set $k = a_1 + a_2$, then, we obtain:

COROLLARY 3.5 A connected circulant digraph $G_n(a_1, a_2)$ can be decomposed into $a_1 + a_2$ directed cycles of the equal lengths of form $a_1a_2a_1a_2\cdots a_1a_2$ if $\gcd(n, a_1 + a_2) = a_1 + a_2$. ■

Let q be an integer and $q \geq 2$. Having a fleet of k drones, Corollary 3.5 implies that by choosing a circulant $G_n(a_1, a_2)$ with $n = qk$ and $a_1 + a_2 = k$, we can assign all drones to k pairwise disjoint cycles of equal lengths that decompose $G_n(a_1, a_2)$. Furthermore, each such cycle is of the same form $a_1a_2a_1a_2\cdots a_1a_2$.

IV. FLYING DRONES

By *flying drones over graph* G, we mean flying drones in 3-D space (x, y, z) in such a way that projecting all flight paths along z-axis on xy-plane results in G. Fig. 2 illustrates an example of such a projection for two flight paths of two drones resulting in

$$G(V, E) = G(\{v_1, v_2, v_3, v_4\}, \{(v_1, v_2), (v_2, v_3), (v_3, v_1), (v_3, v_4), (v_4, v_2)\}).$$

We now show how all k drones can fly over a directed graph H_{n+1} on $n + 1$ vertices constructed from circulant digraph $G_n(1, k - 1)$ and an additional vertex corresponding to a maintenance site. The objective can be stated as follows. For given k drones and given circulant digraph $G = G_n(a_1, a_2)$, we want to assign all drones to induced directed cycles by G in such a way that each arc in G will be assigned to a single unique drone and each vertex in G (corresponding to two waypoints) will be visited by unique pair of drones implied by its two incoming arcs. This feature will assure that if any single drone is lost, the coverage of all vertices in G will remain unchanged. So, in terms of the coverage of vertices of G our assignment will result in the survivable (i.e., resilient to any single drone failure) flying of drones.

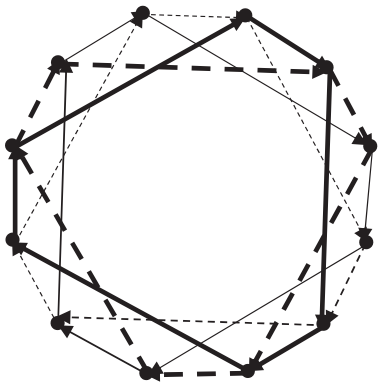

Fig. 3. Independent flying cycles for four drones in $G_{12}(1, 3)$.

We select $G = G_n(a_1, a_2)$ based on a given fleet of $k \geq 3$ drones. Specifically, we choose $G_n(a_1, a_2)$ that satisfies $n > k$ and $\gcd(n, k) = k$. Based on Corollary 3.5 $G_n(a_1, a_2)$ can be decomposed into $a_1 + a_2$ pairwise arc-disjoint directed cycles of equal lengths. In addition, each such cycle in $G_n(a_1, a_2)$ is of the form $a_1a_2a_1a_2\cdots a_1a_2$. This implies that our k drones can be assigned into k pairwise arc-disjoint (and equal in terms of the number of arcs) directed cycles in $G_n(1, k - 1)$ that cover all arcs in $G_n(1, k - 1)$ as illustrated in Fig. 3 for $n = 12$ and $k = 4$. In particular, four drones are assigned to four cycles in $G_{12}(1, 3)$; two drones are assigned to two thick cycles designated by solid and dashed lines, and two drones are assigned to two thin cycles designated by solid and dashed lines.

Based on k drones and $G_n(1, k - 1)$, we construct H_{n+1} and assign drones to edges of H_{n+1} in three steps. First, we assign unique IDs into G and maintenance site as follows: 1) for vertices in $G_n(1, k - 1)$: $v_0 = 0$, $v_1 = 1, \ldots, v_{n-1} = n - 1$, and 2) for maintenance site $v_n = n$. Second, we build a digraph H_{n+1} as illustrated in Fig. 4. Third, we assign unique IDs $0, 1, \ldots, k - 1$ into k drones that are assigned to pairs of arcs in opposite directions $u_0, u_1, \ldots, u_{k-1}$ in H_{n+1}, respectively.

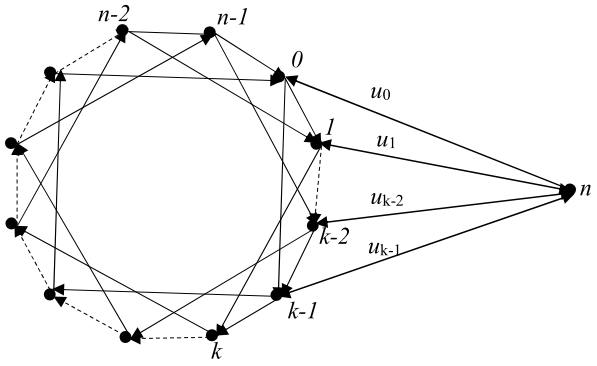

Fig. 4. Illustration of H_{n+1}.

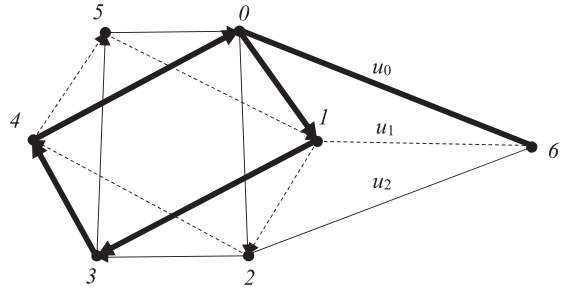

Fig. 5. Flying three drones over H_7 generated from $G_6(1, 2)$.

Let drone with ID $= i$ be denoted by drone(i). Each drone(i) flies over H_{n+1} from the maintenance node v_n with the same pattern as follows:

$v_n u_i$
$a_1 a_2 a_1 a_2 \cdots a_1 a_2 v_i$
$a_1 a_2 a_1 a_2 \cdots a_1 a_2 v_i$
$\cdots$
$a_1 a_2 a_1 a_2 \cdots a_1 a_2 v_i$
$u_i v_n$

as illustrated in Fig. 5. Note that in this example $v_n u_i$ denotes outgoing arc e_i from v_n and $u_i v_n$ denotes incoming arc $\bar{e}_i$ to v_n, where $v_n \in V(H_{n+1})$ and $e_i, \bar{e}_i \in A(H_{n+1})$ represent two opposite arcs u_i. Let Z be altitude of our maintenance site, L be low altitude, and H be high altitude (i.e., a predetermined fixed altitude that is higher than predetermined fixed "low" altitude and that can be achieved by drones) to fly by the drones. Each drone(i) flies from Z to L when it flies from the maintenance site, it flies from L to H when it flies according to a_1 between two waypoints, it flies from H to L when it flies according to a_2, and it flies from L to Z when it flies back to maintenance site.

Let $(\text{lat}_i, \text{lon}_i, \text{alt}_i)$ at vertex i denote a waypoint with latitude $= \text{lat}_i$, longitude $= \text{lon}_i$, and altitude $= \text{alt}_i$. Furthermore, let $(\text{lat}_i, \text{lon}_i, L) = v_i^1$ and $(\text{lat}_i, \text{lon}_i, H) = v_i^2$. Each drone flies according to the same five flying rules as follows:

1) If drone(i) is in waypoint v_n, then, it flies up from v_n to v_i^1 via u_i. This corresponds to flying from waypoint $(\text{lat}_n, \text{lon}_n, Z)$ to waypoint $(\text{lat}_i, \text{lon}_i, L) = v_i^1$.
2) If drone(i) is in waypoint $v_j^1, i \equiv j \pmod k$, and $i \neq j$, then, it flies up from vertex j to vertex $j' \equiv j + a_1 \pmod k$. This corresponds to flying from waypoint v_j^1 to waypoint $v_{j'}^2$.
3) If drone(i) is in waypoint $v_j^1, i = j$, and it has enough energy to fly another round over circulant $G_n(1, k-1)$; then, it flies up from vertex j to vertex $j' \equiv j + a_1 \pmod k$. This corresponds to flying from waypoint $(\text{lat}_j, \text{lon}_j, L) = v_j^1$ to waypoint $v_{j'}^2$.
4) If drone(i) is in waypoint $v_j^2, i \neq j \pmod k$, $j \neq n$; then, it flies down from vertex j to vertex $j' \equiv j + a_2 \pmod k$. This corresponds to flying from waypoint $(\text{lat}_j, \text{lon}_j, H) = v_j^2$ to waypoint $v_{j'}^1$.
5) If drone(i) is in waypoint $v_j^1, i = j$, and it has not enough energy to fly another round over circulant $G_n(1, k-1)$; then, it flies down from vertex j to vertex n. This corresponds to flying from waypoint $(\text{lat}_j, \text{lon}_j, L) = v_j^1$ to waypoint $(\text{lat}_n, \text{lon}_n, Z) = v_n$.

For example, for drone(0) in Fig. 5 the sequence of waypoints to fly (illustrated with thick arrows) is as follows:

$(\text{lat}_6, \text{lon}_6, Z)$
$v_0^1 v_1^2 v_3^1 v_4^2$
$v_0^1 v_1^2 v_3^1 v_4^2$
$\cdots$
$v_0^1 v_1^2 v_3^1 v_4^2$
$v_0^1 (\text{lat}_6, \text{lon}_6, Z)$.

V. DECONFLICTION OF FLYING DRONES

We assume that the drones fly between the waypoints in straight lines. *Deconfliction* is a well-known term in aviation. In this paper, if every waypoint is visited by at most one drone and no two straight lines corresponding to flying drones cross each other in 3-D space; then, we say that *3-D-deconfliction* is satisfied by such flying drones. Clearly, this means that the drones can fly without colliding into each other. In this section, we prove that k drones can fly with 3-D-deconfliction based on flight paths induced by $G_n(1, k-1)$, if $\gcd(n, k) = k$ and if vertices of $G_n(1, k-1)$ are placed on a circle equally separated over a given area of interest. Recall example from Fig. 2. The corners of a cube on the left-hand side of Fig. 2 represent the waypoints for both drones to fly. It is easy to see that two drones might collide between the waypoints in the center of cube at the left-hand side of Fig. 2 (corresponding to arcs (v_3, v_1) and (v_4, v_2) in G) if their timing of flight is "wrong." This conflict can be resolved in Fig. 2 by appropriately scheduling the timing of flights. If the collisions between the drones can be avoided by appropriate scheduling of their flights we say that such drone flights satisfy *4-D-deconfliction*. Hence, by definition every 3-D-deconfliction satisfies 4-D-deconfliction but the contrary is not always true. Finally, there are drone flights that cannot be fixed by re-scheduling (e.g., two drones flying over opposite circles), in which case they cannot satisfy 4-D-deconfliction.

The proof of the following theorem consists of two parts. In the first part, we prove that every waypoint induced by H_{n+1} is flown by exactly one drone, and every leg between two waypoints is flown by at most one drone. In the second

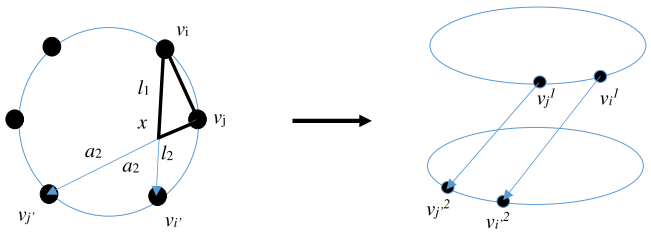

Fig. 6. Illustration of two legs induced by a_2 between the waypoints.

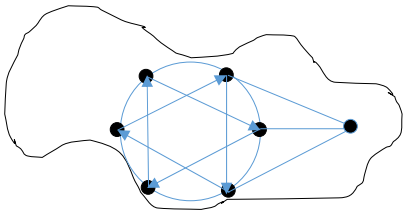

Fig. 7. Illustration of H_7 placed over a given area of interest.

part, we prove that two flown legs by drones never cross each other.

THEOREM 5.1 Let $n > k \geq 3$ and $\gcd(n, k) = k$. If k drones fly over circulant digraph $G_n(1, k-1)$ with consecutive vertices equally separated on a circle according to rules 1)–5), then, 3-D-deconfliction is satisfied.

PROOF Since rules 1)–5) are satisfied, then, the flight of each drone corresponds to a cycle $a_1a_2a_1a_2...a_1a_2$ of circulant digraph $G_n(1, k-1)$, where $a_1 = 1$ and $a_2 = k-1$. Because $\gcd(n, k) = k$, then, by Corollary 3.5 all k drones fly cycles of equal lengths corresponding to k pairwise disjoint cycles $a_1a_2a_1a_2...a_1a_2$ in $G_n(a_1, a_2)$. Furthermore, to each vertex v in $G_n(a_1, a_2)$ there correspond exactly two waypoints v^1 and v^2. Hence, by rules 1)–5) each waypoint v^i is visited by exactly one drone. This implies that a collision between two drones cannot happen at a waypoint. So, such collision can only occur between the waypoints.

Suppose that a collision between two drones occurs on leg $v_i^x v_{i'}^y$ between waypoints v_i^x and $v_{i'}^y$. Because the consecutive vertices $v_0, v_1, \ldots, v_{k-1}$ of $G_n(a_1, a_2)$ are equally separated on a circle, as illustrated in Fig. 4, then by rules 1)–5), a collision between two drones can only happen on the intersection of leg $v_i^2 v_{i'}^1$ corresponding to arc $v_i v_{i'}$ induced by a_2 with some other leg corresponding to arc $v_j v_{j'}$ induced by a_2 (see left-hand side of Fig. 6). Equation $\gcd(n, k) = k$ for $G_n(1, k-1)$ implies that $k - 1 = a_2 < n/2$. Since $a_2 < n/2$, then, by symmetry of $G_n(a_1, a_2)$ and it is equally separated consecutive vertices on a cycle, it follows that $l_1 \neq l_2$ and the intersection of legs $v_i^2 v_{i'}^1$ and $v_j^2 v_{j'}^1$ does not happen in a 3-D space as illustrated on the right-hand side of Fig. 6—a contradiction that completes this proof. ■

Note, by symmetry of $G_n(a_1, a_2)$ and its equally separated consecutive vertices on a cycle, two legs $v_i^2 v_{i'}^1$ and $v_j^2 v_{j'}^1$ can cross each other in a 3-D space if and only if $l_1 = l_2$ on the left-hand side of Fig. 6. This implies that a_2 must equal $n/2$ to satisfy the intersections of $v_i^2 v_{i'}^1$ and $v_j^2 v_{j'}^1$.

VI. ALGORITHM FOR FLYING DRONES

In this section, we describe an algorithm that represents a framework for operating the swarm of drones from a single ground station (i.e., OCU) in order to fly them without colliding at each other in a persistent way. We assume that the following inputs are given:

1) fleet of k heterogeneous drones;
2) area of interest;
3) n vertices of circulant digraph $G_n(1, k-1)$ over area of interest such that $n > k \geq 3$ and $\gcd(n, k) = k$;
4) maintenance location for launching, refuel, or recharging of drones;
5) high and low altitude (i.e., upper bound "H" and lower bound "L") for flying drones;
6) single ground station (i.e., OCU) for managing flights of drones.

So, in addition to a single OCU, we also allow a single maintenance site for flying drones. Each drone must be equipped with the following sensors:

1) GPS;
2) camera or other situational awareness detection sensor (e.g., infrared sensor);
3) radio receiver.

No other sensors are required for drones. This will allow them to be very small. On the other hand, the ground station will be comprised of a radio transmitter to communicate dynamic change in the waypoints' locations to drones during their flight.

We assume that the fleet of drones will fly over straight lines in a 3-D space represented by directed legs between the given waypoints. First, based on the location of maintenance on the map and the area of interest, we place $G_n(1, n-1)$ with an appropriate radius (i.e., fitting the region of interest and contained in it) on the map that results in H_{n+1} from Fig. 4—see Fig. 7 for $n = 6$. Note that constructed H_{n+1} that is placed on the map results in space dispersion of $k = 3$ flying drones based on decomposition of $G_n(1, k-1) = G_n(1, 2)$ into three directed and equal cycles that we described in the previous section.

By Theorem 5.1, launching k drones from the location corresponding to $v_n \in V(H_{n+1})$ and flying them according to the rules 1)–5) (from Section IV) will result in collisionless and persistent flying over H_{n+1}. The fact that the flying of drones has to be persistent implies that each drone will fly over some directed cycle, and eventually return to its starting point, which corresponds to a maintenance site. After recharging or refueling, it will resume its flight over the same pattern designated to it. We allow a user to modify/edit H_{n+1} over the map through OCU as follows. Let L_i be a length of trajectory induced by a_2 from vertex v_i (see Fig. 8). When modifying the waypoint corresponding to v_i location two checks will be made as follows:

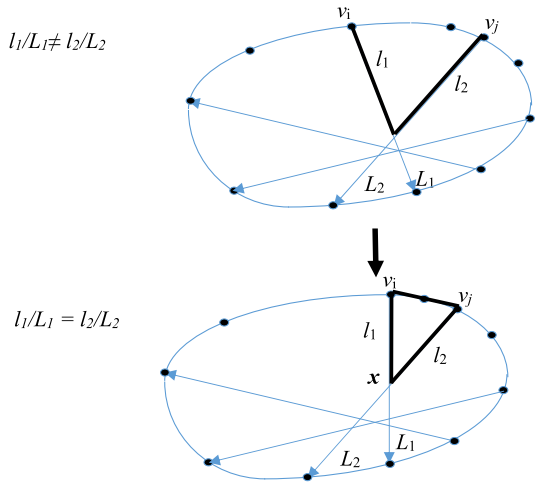

Fig. 8. Violation of check #2 (isosceles triangle with $l_1/L_1 = l_2/L_2$ created in 3-D space by intersection at some point x).

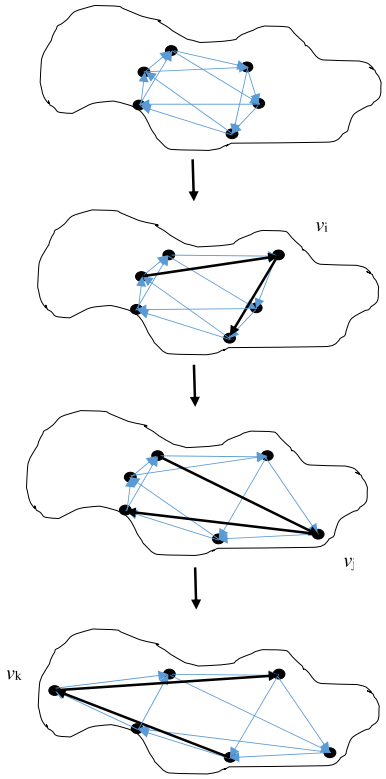

Fig. 9. Possible sequence of editing steps on $G_n(1, k-1)$.

1) convexity of polygon $v_0, v_1, \ldots, v_{n-1}$ of circulant $G_n(1, k-1)$ connected with jump a_1 is preserved on the map;
2) isosceles triangle (v_i, v_j, x) is not formed that is induced by two trajectories implied by any two a_2 s from v_i, v_j and crossing at some point x in 3-D space, where length of $(v_i, x)/L_1 = l_1/L_1$ equals length of $(v_j, x)/L_2 = l_2/L_2$—see Fig. 8.

Based on the above rules of editing H_{n+1} the following Fig. 9 illustrates a possible sequence of editing steps changing $G_n(1, k-1)$, that is a part of H_{n+1}.

Algorithm 1: Operation Framework for Persistent Flying of Drones Over a Circulant Digraph.

Input: Drones: given k heterogeneous drones with characteristics;
Map: area of interest;
Waypoints: $2n$ waypoints corresponding to vertices of
$G_n(1, k-1)$ and one corresponding to a maintenance site;
Altitudes: upper and lower bounds for flying altitudes;
Operator control units: single OCU;

Output:
Persistent flying of k drones over $G_n(1, k-1)$;

Step 1. Ground station assigns labels $0, 1, 2, \ldots, n-1$ to n vertices on a circle placed over area of interest as described in Section IV and illustrated in Fig. 4;
Step 2. Ground station assigns label n to the maintenance site;
Step 3. Ground station builds circulant digraph $G_n(a_1, a_2) = G_n(1, k-1)$ and corresponding digraph H on $n+1$ vertices placed over area of interest as described in Section IV and illustrated in Fig. 4.;
Step 4. Ground station optionally edits circulant digraph $G_n(1, k-1)$ preserving two rules from Section VI as illustrated in Fig. 9;
Step 5. Ground station launches drones (or a single drone) from the maintenance site n;
Step 6. All k drones fly on autopilot according to the rules specified in Section IV;
Step 7. If any drone(i) is lost and flying drones needs to be continued then ground station replaces every lost drone(i) with ID=i;
Step 8. If any drone(i) landed in the maintenance site n and flying drones needs to be continued then every landed drone(i) is refueled or recharged;
Step 9. If flying drones needs to be continued then go to Step 5. Otherwise, STOP.

This update might be done through the graphical user interface (GUI) on OCU. Every time a single vertex is moved (e.g., v_i moved first, followed by v_j move, and followed by v_k move in Fig. 9) through GUI on the map, two black arcs in Fig. 9 need to be checked (adjacent to v_i, v_j, and v_k, respectively) if any of them crosses some other arc corresponding to a_2 in 3-D in some point x that results in isosceles triangle with $l_1/L_1 = l_2/L_2$ as described in rule #2 above and illustrated in Fig. 8. This check can be easily made based on the basic laws of Cartesian geometry and it can be time efficient. It assures 3-D-deconfliction for our drones. Hence, our algorithm describing the main steps of flying drones over our graph can be stated as in Algorithm 1.

Note that there is an alternate approach of flying drones when instead of having high "H" and low "L" bounds on the altitudes (previously described in Section IV), we could enforce an equal fixed rate of ascend/descend for all drones. In this case, a simple check for $l_1 = l_2$ in Figs. 8 and 9 would have to be made instead of $l_1/L_1 = l_2/L_2$ to assure that an isosceles triangle is not created, which would imply a possible collision between two drones. This alternate approach, however, would not be as robust as ours since long trajectories could jeopardize reaching high altitudes for some drones.

Algorithm 1 results in the persistent flying of drones that tends to disperse them over a given area of interest based on given k drones and $2n$ waypoints, where k divides n and $n/k \geq 2$. Steps 1–4 initialize the flying of drones that will result in 3-D-deconfliction while they fly in Step 6. In particular, Step 4 optionally updates the waypoints of $G_n(1, k-1)$ assuring that the convexity of the polygon induced by $G_n(1, k-1)$ on the map is preserved and that a triangle is not created in 3-D space (i.e., preservation of rules #1 and #2 from Section VI is satisfied). Step 5 either launches all drones after initialization, it re-launches a replacement drone to the one that has been lost, or re-launches a drone after its refueling or recharging in the maintenance location. It is assumed that Step 6 continues as long as all k drones are flying. Steps 7 and 8 are executed only if the persistent flying of drones needs to be continued. That decision can be based on the elapsed/duration time of flying the drones, or on some other reason external to Algorithm 1. Finally, in Step 9 Algorithm 1 terminates if the flying of drones needs to be stopped. In particular, STOP in Step 9 does not mean that all the drones stop flying right away. Instead, the flying of the drones will gradually terminate by landing in the maintenance site one by one as the energy of the drones is exhausted. Otherwise, the loop consisting of Steps 5–9 continues resulting in persistent flying of drones. Note that an optional Step 4 could conceivably be incorporated into the loop consisting of Steps 4–9 instead by careful implementation that considers the dynamics of the flying drones.

A. OCU Editing Consideration

While moving vertices of $G_n(1, k-1)$, one at a time, by OCU in order to better cover a given area of interest over a map, we have to make sure that the drones continue collisionless flying between the waypoints. This might be violated due to the intermittent state of flying induced by the changing shape of $G_n(1, k-1)$ over the map. In particular, we have to make sure that a drone does not change direction while flying between two waypoints because our collisionless flying is based on the assumption that the drones fly over straight lines. Hence, careful systems engineering and implementation is required for this feature.

B. Maintenance Consideration

Even though our drones based on Algorithm 1 fly over $G_n(1, k-1)$ without collisions, this might not be the case in the maintenance vertex v_n of H_{n+1}. There are two basic solutions to this problem: 1) a time division solution, and 2) a space division solution—both implementation dependent.

The time-division approach achieves the synchronization of drones by pre-determined scheduling of sequential launching of drones from the maintenance location. That is, knowing the distance that the drones can fly based on their parameters and fuel/battery level during launch, one can estimate the time intervals in which they will land again due to well defined flying routes as specified in Section IV. Consequently, one can sequentially launch drones in an appropriate order that can assure that the landing of the drones in the maintenance vertex v_n will happen in pairwise separate time slots. This solution, however, requires careful pre-planning to assure that any two drones will not collide with each other in v_n.

In the space division approach, we consider v_n as a location, not as a point on the map, that contains pairwise distinct subvertices $v_{n,0}, v_{n,1}, \ldots, v_{n,k-1}$, corresponding to launching/landing locations of drone(0), drone(1), ..., drone(k − 1), respectively. Consider access subgraph of H_{n+1} induced by vertices $v_0, v_1, \ldots, v_{k-1}$ and a maintenance vertex v_n. Clearly, such a subgraph is a star $K_{1,k-1}$ that can also be represented as a bipartite graph

$$(V_1, V_2, E) = (\{v_0, v_1, \ldots, v_{k-1}\}, \{v_n\}, \{(v_0, v_n), (v_1, v_n), \ldots, (v_{k_1}, v_n)\}).$$

We convert $K_{1,k-1} \rightarrow K'_{1,k-1}$ by substituting maintenance vertex v_n in $K_{1,k-1}$ with subvertices $v_{n,0}, v_{n,1}, \ldots, v_{n,k-1}$ in $K'_{1,k-1}$. So, we obtain $K'_{1,k-1} = (V_1, V'_2, E') = (\{v_0, v_1, \ldots, v_{k-1}\}, \{v_{n,0}, v_{n,1}, \ldots, v_{n,k-1}\}, \{(v_0, v_{n,0}), (v_1, v_{n,1}), \ldots, (v_{k-1}, v_{n,k-1})\})$ that represents a new access subgraph of H'_{n+1}. Since by rule #1 of editing H_{n+1} convexity of polygon induced by $G_n(1, k-1)$ is satisfied, we can arrange the relative position of v_n in respect to $G_n(1, k-1)$ in H_{n+1} so the launching/landing trajectories of drones do not cross one another in H'_{n+1}. This in turn assures collisionless flying/launching/landing of our drones without the need for synchronization as was the case in the time division approach.

C. Benefits of Flying Drones According to Algorithm 1

There are two obvious key reasons for using Algorithm 1 for flying a swarm of k drones. The first reason is due to the collisionless flying of k drones without the need for collision-detecting sensors or flight synchronization. The second reason is the scalability of k. There are, however, a number of other important benefits resulting from Algorithm 1. We can summarize the benefits resulting from using Algorithm 1 as follows:

1) Algorithm 1 assures 3-D-deconfliction according to Theorem 5.1 and based on two rules from Section IV. This in turn assures collisionless flying of drones without the need for collision-detecting sensors.
2) Synchronization of flying the drones is not needed to avoid collisions due to 3-D-deconfliction.
3) Each drone has identical firmware. That is, any two drones i, j software-wise might differ from each other

only by their IDs: ID(i) $\neq$ ID(j). Yet, the decision on how to fly next from the waypoints is distributed and made by each drone independently (i.e., OCU is not involved in this decision making).

4) Algorithm 1 assures scalability to hundreds (and possibly more) of drones due to simplicity of design: symmetry of our infrastructure (i.e., circulant digraph $G_n(1, k-1)$), same firmware for all drones, distributed decision making (i.e., each drone independently decides on the next leg to fly in order to reach the next waypoint).
5) Flying of drones over circulant digraph $G_n(1, k-1)$) results in dispersed flying that tends to maximize coverage of the area of interest.
6) One maintenance location for all drones is sufficient. However, care has to be taken about possible collisions of drones in the maintenance site.
7) Our method of flying drones is relatively resilient and survivable: After a loss of any single drone all the vertices of G are covered by $k-1$ remaining drones.
8) Dynamic reconfiguration of drones flights from the OCU is possible (e.g., Fig. 9).
9) Easy and intuitive implementation/operation of flying drones.
10) Minimal requirement for the sensor payload of drones allows miniaturization of them.
11) Single OCU is sufficient to manage all the drones and their flights.
12) Radio bandwidth is not an issue since drones do not need to communicate with each other and OCU communication is not needed for navigation or localization of drones.

Note also that in our solution drones, do not need to fly in formation with fixed relative distances between them because the flying paths are pairwise disjoint.

VII. CONCLUSION

In this paper, we presented a novel framework for collisionless flying of many drones without the need for collision-detecting sensors or flight synchronization. Our method, presented in Algorithm 1, is scalable to hundreds of drones and allows to fly all of them from a single ground station (i.e., OCU). Algorithm 1 also allows intuitive dispersed coverage of the given area of interest in the persistent way for given k drones. Our framework is based on the collisionless flying of drones over a directed circulant digraph $G_n(1, k-1)$. As the result of our method of flying the drones, each vertex of $G_n(1, k-1)$ is visited by exactly two unique drones, implying survivability due to a single drone loss, and each arc of $G_n(1, k-1)$ is visited by a single unique drone. Our design assures that the drones fly over the pairwise arc-disjoint cycles consisting of the same number of arcs of $G_n(1, k-1)$ (i.e., cycles of equal lengths) and that these cycles decompose $G_n(1, k-1)$, which means that these cycles cover all the arcs of $G_n(1, k-1)$. Furthermore, to each vertex v_i in $G_n(1, k-1)$, there corresponds a pair of unique waypoints $(\text{lat}_i, \text{lon}_i, L)$, $(\text{lat}_i, \text{lon}_i, H)$, each visited by a unique drone. This allows a human in the loop to increase coverage of the area of interest by the collisionless flying drones (as illustrated in Fig. 9) through GUI in an intuitive way by simply moving one vertex of $G_n(1, k-1)$ at a time.

In the next step, we will perform modeling and simulation for the realistic detailed conditions of our collisionless flying drones (e.g., drag consideration, lift coefficient, longitudinal stability, lateral stability, GPS accuracy, etc.). In particular, we will estimate the minimum distances between the given drones that they are allowed to fly due to the specific sizes of drones or wind, which might require more restricted rules to be enforced. In the follow-on step, we plan to conduct live experiments in our US Army base in New Jersey. Initially, we will conduct the experiments with just a few drones, increasing their number gradually to tens and more. In future work, we will consider richer class of orbits, which are amenable to graph formalism described in this paper. Finally, this graph-theoretic framework for flying drones, we might perhaps be able to extend from convex polygons to irregular areas in the future.

ACKNOWLEDGMENT

The author would like to thank all of the referees for valuable comments, which resulted in greatly improved robustness, better organization, and better clarity of this paper.

REFERENCES

[1] R. W. Beard, J. Lawton, and F. Y. Hadaegh
A coordination architecture for spacecraft formation control
IEEE Trans. Control Syst. Technol., vol. 9, no. 6, pp. 777–790, Nov. 2001.

[2] R. W. Beard, T. W. McLain, D. B. Nelson, D. Kingston, and D. Johanson
Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs
Proc. IEEE, vol. 94, no. 7, pp. 1306–1324, Jul. 2006.

[3] C. Belta and V. Kumar
Abstraction and control of groups of robots
IEEE Trans. Robot. Autom., vol. 20 no. 5 pp. 865–875, Oct. 2004.

[4] F. T. Boesch and R. Tindell
Circulants and their connectivities
J. Graph Theory, vol. 8, pp. 129–138, 1984.

[5] S. Bhattacharya, R. Ghrist, and V. Kumar
Multi-robot coverage and exploration on Riemannian manifolds with boundaries
Int. J. Robot. Res., vol. 33, no. 1 pp. 113–137, 2014.

[6] Z. R. Bogdanowicz
Decomposition of circulant digraphs with two jumps into cycles of equal lengths
Discrete Appl. Math., vol. 180, pp. 45–51, 2015.

[7] Z. R. Bogdanowicz
Arc-disjoint and edge-disjoint Hamilton cycles in circulants with two jumps
Graphs Combinatorics, vol. 29, pp. 165–171, 2013.

[8] Z. R. Bogdanowicz
Hamilton cycles in circulant digraphs with prescribed number of distinct jumps
Discrete Math., vol. 309, pp. 2100–2107, 2009.

[9] D. E. Chang, S. C. Shaden, J. E. Marsden, and R. Olfati-Saber
Collision avoidance for multiple agent systems
In *Proc. IEEE Conf. Decis. Control*, 2003, pp. 539–543.

[10] M. Egerstedt and X. Hu
Formation constrained multi-agent control
IEEE Trans. Robot. Autom., vol. 17, no. 6, pp. 947–951, Dec. 2001.

[11] R. Fierro, A. Das, V. Kumar, and J. Ostrowski
Hybrid control of formation of robots
In *Proc. IEEE Int. Conf. Robot. Autom.*, 2001, pp. 2157–2162.

[12] L. P. Gewali, S. Ntafos, and I. G. Tollis
Path planning in the presence of vertical obstacles
IEEE Trans. Robot. Autom., vol. 6, no. 3, pp. 331–341, Jun. 1990.

[13] F. Giulietti, L. Pollini, and M. Innocenti
Autonomous formation flight
IEEE Control Syst. Mag., vol. 20, no. 6 pp. 34–44, Dec. 2000.

[14] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar
Towards a swarm of agile micro quadrotors
Auton. Robots, vol. 35, no. 4, pp. 287–300, 2013.

[15] J. Lawton, R. W. Beard, and B. J. Young
A decentralized approach to formation maneuvers
IEEE Trans. Robot. Autom., vol. 19, no. 6, pp. 933–941, Dec. 2003.

[16] R. C. Leishman, T. W. McLain, and R. W. Beard
Relative navigation approach for vision-based aerial GPS-denied navigation
J. Intell. Robot. Syst., vol. 74, no. 1/2, pp. 97–111, 2014.

[17] M. Mesbahi and F. Y. Hadaegh
Formation flying control of multiple spacecraft via graph, matrix inequalities, and switching
J. Guid. Control Dyn., vol. 24, no. 2, pp. 369–377, 2000.

[18] D. Panagou and V. Kumar
Cooperative visibility maintenance for leader-follower formations in obstacle environments
IEEE Trans. Robot., vol. 30, no. 4, pp. 831–844, Aug. 2014.

[19] W. Ren and R. Beard
Decentralized scheme for spacecraft formation flying via the virtual structure approach
J. Guid. Control Dyn., vol. 27, no. 1, pp. 73–82, 2004.

[20] P. B. Sujit, J. M. George, and R. W. Beard
Multiple UAV coalition formation
In *Proc. Amer. Control Conf.*, 2008, pp. 2010–2015.

[21] M. Turpin, N. Michael, and V. Kumar
Trajectory design and control for aggressive formation flight with quadrotors
Auton. Robots, vol. 33 no. 1/2, pp. 143–156, 2014.

[22] P. K. C. Wang and F. Y. Hadaegh
Coordination and control of multiple microspacecraft moving in formation
J. Astron. Sci., vol. 44, no. 3, pp. 315–355, 1996.

[23] X. Wang and S. N. Balakrishnan
Optimal and hierarchical formation control for UAVs
In *Proc. Amer. Control Conf.*, 2005, pp. 4685–4689.

[24] X. Wang, V. Yadav, and S. N. Balakrishnan
Cooperative UAV formation flying with obstacle/collision avoidance
IEEE Trans. Control Syst. Technol., vol. 15, no. 3, pp. 672–679, Jul. 2007.

[25] Y. Xia, X. Na, Z. Sun, and J. Chen
Formation control and collision avoidance for multi-agent systems based on position estimation
ISA Trans., vol. 61, pp. 287–296, 2016.

[26] Q. Yang, R. Burkard, E. Cela, and G. Woginger
Hamiltonian cycles in circulant digraphs with two stripes
Discrete Math., vol. 176, pp. 233–254, 1997.

[27] H. Yu, R. Sharma, R. W. Beard, and C. N. Taylor
Observability-based local path planning and collision avoidance for micro air vehicles using bearing-only measurements
In *Proc. Amer. Control Conf.*, 2011, pp. 4649–4654.

Zbigniew R. Bogdanowicz received the M.S. degree in electronics engineering from the Wroclaw University of Technology, Wroclaw, Poland, and the Ph.D. degree in electrical engineering from the Stevens Institute of Technology, Hoboken, NJ, USA, in 1976 and 1984, respectively.

He is currently with the US Army, Armament Research, Development, and Engineering Center, Picatinny Arsenal, NJ, USA. Prior to this, for many years he was a Distinguished Member of Technical Staff with Bell Laboratories (AT&T and Lucent), where he worked on survivable network design, network optimization, and traffic routing. He has published numerous journal/conference research papers, and has been a frequent speaker at academic and industrial events. He has significant contributions to graph theory in the mathematical journals. His research interests include scalable lethality, persistent surveillance, weapon-target pairing, and collateral damage estimation.