Optimization of Persistent Land Coverage
by Swarm of Drones

Zbigniew R. Bogdanowicz

Armament Research, Development and Engineering Center
Picatinny, NJ 07806, USA

Copyright © 2018 Zbigniew R. Bogdanowicz. This article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

Drones, or Unmanned Aerial Vehicles (UAVs), due to miniaturization have increasingly important role in the military operations as a swarm of drones. In this work we present an innovative optimization algorithm for the loitering pattern of the swarm of drones in order to maximize the persistent coverage of a given terrain. This algorithm, named $\text{OptLandCov}$, relies on three transformations on graph $G$ that preserve the assignments of drones to cycles in $G$. Computational results confirm the viability of $\text{OptLandCov}$. So, this work should have high relevance to persistent surveillance of a given area of interest as well as to collaborative engagement of multiple targets by swarm of weaponized drones.

Mathematics Subject Classification: 05C20, 05C38, 05C85

Keywords: Swarm of drones, persistent land coverage, UAV loitering pattern optimization, persistent surveillance

1 Introduction

Drones, also called Unmanned Aerial Vehicles (UAVs), are becoming smaller, more sophisticated and less expensive, which allows to deploy them as a swarm of UAVs for various military operations as well as for commercial applications [1],[7],[9]. Most missions that were already carried out, however, involved a
single UAV. The affordability of deploying hundreds of UAVs can soon become a reality, in which case a swarm-like operation will be an important/critical factor in the battlefields [8]. For consistency, in the rest of this paper we will use term drone instead of UAV.

One of the attractive objectives of collaborative operation of drones might be maximization of the land coverage because it could benefit the number of military missions, e.g., persistent surveillance, optimization of target engagement, search and rescue, etc. So, for a given fleet of drones with different mission payloads and characteristics, specified area(s) of interest we want to find an optimum set of loitering routes resulting in maximizing land coverage below.

The persistence requirement implies that each drone in a swarm will follow a closed directed cycle, and at some point of time it will come back to the place of origin – allowing maintenance, including recharging or refueling, of drones in the designated unique locations. Obviously the length of such cycles cannot exceed the range of flying drones assigned to them. Otherwise, the maintenance locations would have to be spread for some drones, which is not a desirable scenario in general. This is particularly critical in our case where the large swarm often implies deployment of small drones that cannot fly far or for a long time. Our optimization engine that we introduce in this paper relies on acting on the small cycles by transforming the assignments of drones between them. Hence, this approach is consistent with the constraints for the small drones described above.

In this paper we focus on maximizing the coverage of pairwise diverse visited points by swarm of drones for any given interval of time. That is, for any given time interval we want the drones to fly over as many unvisited points on the pre-assigned cycles as possible. We focus on this maximization of point coverage without explicit consideration of the schedule of flying drones due to the complexity/difficulty of this problem alone. However, the schedule consideration needs to be taken care of before the real world scenarios of flying drones being subject to wind gust and other relevant factors can be executed based on our optimized solution. Conversely, without careful scheduling consideration the drones could collide with each other or collide with the obstacles related to the surrounding terrain. Such scheduling can complement our optimized planning of the flights of drones in the post-processing module/activity, which we don’t cover in this work.

Planning and optimizing drones paths have been extensively covered in literature [6], [10-12], [15-16], [19-20], [23]. In particular, [4] introduced a framework for optimization of the persistent surveillance relevant to this work. Some work was also done for flying swarms of drones in the formation [3],[13-14],[17],[21-22], which means that the relative distances between the drones in a swarm remain fixed. However, there is not much other work that has been
done on the swarms of drones that don’t fly in the fixed formation, and hence this is the main focus of this work.

In this paper we introduce an optimization algorithm that is based on three graph transformations: (1) drones-cross, (2) 2drones-swap, and (3) 3drones-swap. The first two transformations are similar to the ones introduced in [4]. Transformation drones-cross differs from the one that was used in [4] by allowing drones-cross to be executed when just a single drone is assigned to one of the cycles as opposed to two fractions of drones being assigned to both cycles. Transformation 2drones-swap differs from the one that was introduced in [4] by considering either one cycle assigned to a drone or both cycles assigned to two distinct drones as opposed to two fractions of drones assigned to both cycles. Finally, transformation 3drones-swap is a new transformation that will allow to escape from the local minimum in some instances (not always) during the optimization of the land coverage. This novel algorithm we named Opt\_Land\_Cov, and it can be executed on the ground station in off-line mode for planning paths of individual drones incorporated into a swarm of drones, followed by actual flying them.

This paper is organized as follows. In the next section, Section 2, we describe the assumptions and key inputs to our optimization framework. In Section 3, we formulate the objective of optimization and related constraints. In Section 4, we describe the first phase of Opt\_Land\_Cov, which is the initial assignment of drones to cycles of a graph. Then in Section 5 we describe individual components, which are applied in the second phase of Opt\_Land\_Cov covered in Section 6. In Section 7 we illustrate a core optimization engine of our infrastructure based on a simple example. In Section 8 we give the computational results for simple 12 scenarios. Finally, in Section 9 we briefly summarize the work covered in Sections 2 through 8.

## 2 Assumptions

We assume that the input to our optimization is a simple directed balanced digraph $G(V, E)$ with a set of vertices $V(G)$ corresponding to the given waypoints, and a set of arcs $E(G)$ corresponding to the directed straight lines between the waypoints for our drones to fly. We also assume that $G$ does not contain the opposite arcs (i.e., does not contain edges). These assumptions do not restrict neither generality nor practicality of our approach since having an area of interest and the waypoints for the drones we can derive our $G$ as follows.

Step 1: From some vertex $v_i \in V(G)$ draw a directed cycle that covers arcs corresponding to subset of either required or desired legs (i.e., directed
straight lines between the waypoints) and the remaining arcs corresponding to acceptable legs.

Step 2: Repeat Step 1 as long as not all required or desired legs have been matched with the generated arcs.

Step 3: While there are parallel arcs \(a_1, a_2, \ldots, a_r\) from \(v_i\) to \(v_j\) in \(G\) then for each \(a_i, i \geq 2\), create a vertex \(v_k\) corresponding to a distinct location of a new waypoint \(w_k\) and replace each \(a_i\) with arcs \((v_i, v_k), (v_k, v_j)\).

Step 4: While there are opposite arcs \(a_1 = (v_i, v_j), a_2 = (v_j, v_i)\) between \(v_i, v_j\) in \(G\) then create a vertex \(v_k\) corresponding to a distinct location of a new waypoint \(w_k\) and replace \(a_2\) with arcs \((v_i, v_k), (v_k, v_j)\).

By cycle in digraph \(G\) in this paper we mean a directed closed walk \([4]\) over \(G\), which visits each arc at most once. Hence, a cycle in this paper is a walk allowed to visit any vertex of \(G\) multiple times.

3 Formulation of optimization problem

Let \(S_i\) be an \(i\)'th unique and attainable assignment of drones to cycles in \(G\), and let \(\Omega\) be a collection of all such sets (i.e. \(S_i \in \Omega\) for \(i = 1, 2, \ldots, q\) and for some finite positive integer \(q\)). Let \(c_i^j\) denote a cycle assigned to drone \(d_j\) in assignment \(S_i\), and let \(l(c_i^j)\) denote the length of \(c_i^j\). Let \(\vartheta_i\) denote a nominal velocity of drone \(d_i\) and let \(r_i\) denote maximum range of \(d_i\) allowed to fly. Note, symbol \(\vartheta_i\) is different from symbol \(v_i\), which denotes a vertex \(i\) in \(G\).

We formulate our optimization problem based on an initial assignment of \(k\) drones to cycles \(S_1\) in \(G\) as follows:

\[
\max_{S_i \in \Omega} \sum_{j=1}^{k} (l(c_i^j) \cdot \vartheta_j) \quad \text{for } i \geq 1
\]  

subject to:

\[
r_j \geq l(c_i^j) > 0 \quad \text{for } k \geq j \geq 1, S_i \in \Omega.
\]  

Clearly, the objective of the stated optimization problem is to find a feasible assignment of \(k\) drones to cycles \(S_i\) in \(G\) for some \(i \geq 1\) that maximizes (1). Based on our objective (1) an absolute maximum in terms of visited points coverage would be attained if \(\frac{l(c_1^j)}{\vartheta_1} = \frac{l(c_2^j)}{\vartheta_2} = \cdots = \frac{l(c_k^j)}{\vartheta_k}\) and \(l(c_i^j) = r_1, l(c_2^j) = r_2, \ldots, l(c_k^j) = r_k\) for some assignment \(S_i\). This situation, however, in the real world will not likely be feasible. Hence, our optimization just tends to maximize (i.e., does not always maximize) the number of visited pairwise distinct points by \(k\) drones on the assigned cycles for any given time interval. The constraint (2) assures that any drone \(j\) can land for maintenance (e.g., to recharge or refuel) in the same maintenance location \(L(i, j)\) that corresponds to
some vertex $v$ on cycle $c_i$ in assignment $S_i$. In addition, (2) assures that every drone $j$ is assigned to some cycle of $G$. Furthermore, we can consider balancing the assignment of drones to cycles before, during, and after optimization by introducing an additional constraint for some positive $\Delta$ as follows:

$$\left|\frac{l(c^x_i)}{\vartheta_x} - \frac{l(c^y_i)}{\vartheta_y}\right| \leq \Delta \quad \text{for } k \geq x, y \geq 1, S_i \in \Omega.$$  \hspace{1cm} (3)

Hence, our optimization generically can be stated as maximization of objective (1) with constraints (2) and (3) because $\Delta$ can be set to an arbitrary large number if constraint (3) does not matter.

4 Initial Assignment of drones to cycles of $G$

Let $d_G(v^+)$ and $d_G(v^-)$ be outdegree and indegree of vertex $v$ in digraph $G$. Let $G$ be a simple balanced digraph (i.e., $d_G(v^+) = d_G(v^-)$ for any $v \in V(G)$) without opposite arcs (i.e., without edges). Let $\vartheta_k \geq \vartheta_{k-1} \geq \cdots \geq \vartheta_1 > 0$. We execute a single-source shortest path algorithm, preferably Dijkstra’s [5] algorithm due to its efficiency since there are no negative distances assigned to arcs in $G$ (but Bellman’s [2], Surballe’s [18], or other shortest path algorithms could be also used), to assign drones to cycles of $G$ in the following way. We execute a shortest path algorithm in the successive iterations generating one cycle at each iteration that is arc disjoint with all the previously generated cycles. If $i$'th cycles has been generated in $i$'th iteration then we assign drone $d_i$ to it. Hence, if $i > k$ then drone $d_i$ is considered to be a pseudo-drone with $\vartheta_i = 0$. These iterations decompose our $G$ into pairwise arc-disjoint directed cycles since $d_G(v^+) = d_G(v^-)$ for any $v \in V(G)$. The result of such assignment implies that the first generated cycles are assigned to drones with the lowest nominal velocities $\vartheta_1, \vartheta_2 \ldots$ out of $k$ first-generated cycles and the longest cycles are assigned to drones with the highest nominal velocities $\ldots \vartheta_{k-1}, \vartheta_k$ out of $k$ first-generated cycles. The remaining generated cycles $c_{k+1}, c_{k+2}, \ldots, c_r$ are assigned to pseudo-drones with velocities equal zero if the number of generated cycles $r$ in $G$ exceeds the number of drones $k$, which should be the case in most real-world scenarios.

Note that our approach based on the initial generation of the small cycles is practical for the swarms of drones because they might include tiny drones that cannot fly far distances. So, if our small cycles cannot satisfy the returns of these drones to their origins, which might happen, then other solutions are unlikely to satisfy that requirement either. On the other hand our approach tends to maximize probability of finding the feasible set of paths to fly by our swarm of drones.

The initial assignment of drones described in this section will be refined through Steps 1-4 of $Opt\_Land\_Cov$ algorithm in Section 6. Consequently,
the smallest cycles out of \( k \) smallest (instead of first-generated) cycles will be assigned to drones with the lowest minimal velocities \( \vartheta_1, \vartheta_2, \ldots \), and the largest cycles out of \( k \) smallest (instead of first-generated) cycles will be assigned to drones with the highest nominal velocities \( \ldots \vartheta_{k-1}, \vartheta_k \).

5 Operations on cycles

Once the drones are assigned to cycles of \( G \) the optimization of the coverage of area of interest can be executed by an optimization engine. Our optimization engine is based on three transformations of drones assignments to cycles. These transformations are called (1) \textit{drones-cross}, (2) \textit{2drones-swap}, and (3) \textit{3drones-swap}. They preserve the property that any cycle of \( G \) is either unassigned or assigned to a single unique drone. That is, no two cycles can be assigned to the same drone. Each transformation will be executed in our algorithm (in Section 6) only if it results in the increased coverage by considered swarm of drones. This is equivalent to assigning a drone with the highest speed to a longest derived cycle, and assigning a drone with the lowest speed to a shortest derived cycle resulting from our transformations. Note that if a cycle is unassigned that we assume that it is assigned to a pseudo-drone having speed equal zero. Let \( x^-(j), x^+(j) \) denote incoming and outgoing arcs at vertex \( v_i \) that are assigned by drone \( j \), respectively.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{drone-cross.png}
\caption{Illustration of drone-cross}
\end{figure}

5.1 Operation drones-cross

The first operation, drones-cross transformation, acts on two cycles at a time. One of these cycles has to be assigned and the other one has to be unassigned (i.e., assigned to pseudo-drone). These two directed cycles crossing each other in \( G \) produce a single assigned cycle (assigned to the same drone.
as was assigned to one of the original cycles) in $G$. Operation drones-cross is accomplished by the following steps:

1. Find vertex $v_1 \in V(G)$ that is visited by least two drone $d_i, d_j$, where $d_i \neq d_j$;
2. If $v_1$ does not exist then STOP – drones-cross cannot be executed;
3. At $v_1$, replace assignments $x_{i}^{-}(i) \rightarrow x_{i}^{+}(i), x_{j}^{-}(j) \rightarrow x_{j}^{+}(j)$ with $x_{i}^{-}(i) \rightarrow x_{i}^{+}(j), x_{j}^{-}(j) \rightarrow x_{j}^{+}(i)$. For example, in Fig. 1 we replace assignments $a_1 \rightarrow a_2, b_6 \rightarrow b_7$ with $a_1 \rightarrow b_7, b_6 \rightarrow a_2$.
4. Assign a drone with the speed greater than zero (i.e., not pseudo-drone) to a derived cycle.

![Figure 2: Illustration of 2drones-swap](image)

**5.2 Operation 2drones-swap**

The second operation, 2drones-swap transformation, like drones-cross acts on two cycles at a time. However, this time it allows either both cycles to be assigned to drones or one of the cycles to be assigned to pseudo-drone. This operation produces two different cycles, where at least one of them is assigned to a drone. Operation 2drones-swap can be thought of as the execution of two drones-cross operations applied at two distinct vertices of $G$ where our two cycles cross each other. Operation 2drones-swap is accomplished by the following steps:

1. Find two distinct vertices $v_1, v_2 \in V(G)$ that are visited by some drones $d_i, d_j$, where $d_i \neq d_j$;
2. If $v_1, v_2$ do not exist then STOP – 2drones-swap cannot be executed;
3. At $v_1$, replace assignments $x_{i}^{-}(i) \rightarrow x_{i}^{+}(i), x_{j}^{-}(j) \rightarrow x_{j}^{+}(j)$ with $x_{i}^{-}(i) \rightarrow x_{i}^{+}(j), x_{j}^{-}(j) \rightarrow x_{j}^{+}(i)$. For example, in Fig. 2 we replace assignments $a_1 \rightarrow a_2, b_1 \rightarrow b_2, b_1 \rightarrow a_2$;
(4) At $v_2$, replace assignments $x_1^-(i) \rightarrow x_1^+(i), x_1^-(j) \rightarrow x_1^+(j)$ with $x_1^-(i) \rightarrow x_1^+(j), x_1^-(j) \rightarrow x_1^+(i)$. For example, in Fig. 2 we replace assignments $a_5 \rightarrow a_6, b_1 \rightarrow b_2$ with assignments $a_5 \rightarrow b_2, b_1 \rightarrow a_6$.

(5) Assign a drone with the highest speed to a longest derived cycle and a drone with the lowest speed to a shortest derived cycle.

Figure 3: Illustration of 3drones-swap

5.3 Operation 3drones-swap

The third operation, 3drones-swap transformation, requires and acts on three assigned cycles to drones. Let these three cycles be denoted by $C_1, C_2, C_3$. Operation 3drones-swap can be executed only if there is an assigned walk (i.e., an assigned cycle that is allowed to visit any vertex multiple times but an arc only once) $C_i, i \leq 3$, that visits some vertex $v_1$ at least twice and it crosses $C_j$ in $v_1$ for $j \leq 3, j \neq i$. Furthermore, if a third cycle $C_k, k \leq 3$, crosses $C_i$ and $C_j$ in two distinct vertices $v_2, v_3$ (i.e., $v_1, v_2, v_3$ are pairwise distinct) then 3drones-swap can be executed. Figure 3 illustrates 3drones-swap, where vertex $v_1$ is on top (visited twice by an assigned cycle corresponding to thin solid line) and vertices $v_2, v_3$ in the bottom. Note that in this scenario neither drones-cross nor 2drones-swap can be executed. Operation 3drones-swap is accomplished by the following steps:

1. Find unfound vertex $v_1 \in V(G)$ that is visited at least twice by some drone $d_i$ and visited by some drone $d_j$, where $d_i \neq d_j$;
2. If $v_1$ does not exist then STOP (3drones-swap cannot be executed);
3. Find drone $d_k$ that crosses $d_i$ and $d_j$ in two distinct vertices $v_2$ and $v_3$, respectively;
4. If $v_2$ and $v_3$ do not exist then go back to step (1) because 3drones-swap cannot be executed;
5. At $v_1$, replace assignments $x_1^-(i_1) \rightarrow x_1^+(i_1), x_1^-(i_1) \rightarrow x_1^+(i_2), x_1^-(j) \rightarrow x_1^+(j)$ with $x_1^-(i_1) \rightarrow x_1^+(j), x_1^-(i_1) \rightarrow x_1^+(i_2), x_1^-(j) \rightarrow x_1^+(i_2)$, where $x_1^-(i_1) =$
$x_1(i)$ and it is identified by existence of a simple path $x_2^+(i),\ldots,x_r^-(i)$. For example, in Fig. 3 $x_1(i) = a_1$ is identified and assignments at $v_1$ are changed from $a_1 \rightarrow a_2,a_{15} \rightarrow a_{16},b_6 \rightarrow b_1$ to $a_1 \rightarrow b_1,a_{15} \rightarrow a_2,b_6 \rightarrow a_{16}$;

(6) At $v_2$, replace assignments $x_2^-(i) \rightarrow x_2^+(i),x_2^-(k) \rightarrow x_2^+(k)$ with $x_2^-(i) \rightarrow x_2^+(k),x_2^-(k) \rightarrow x_2^+(i)$. For example, in Fig. 3 we replace assignments $a_{16} \rightarrow a_{17},c_5 \rightarrow c_6$ with $a_{16} \rightarrow c_6,c_5 \rightarrow a_{17}$;

(7) At $v_3$, replace assignments $x_3^-(j) \rightarrow x_3^+(j),x_3^-(k) \rightarrow x_3^+(k)$ with $x_3^-(j) \rightarrow x_3^+(k),x_3^-(k) \rightarrow x_3^+(j)$. For example, in Fig. 3 we replace assignments $b_5 \rightarrow b_6,c_6 \rightarrow c_1$ with $b_5 \rightarrow c_1,c_6 \rightarrow b_5$.

(8) Assign a drone with the highest speed to a longest derived cycle and a drone with the lowest speed to a shortest derived cycle.

6 Optimization of Land Coverage

In this section we present our main result - an optimization algorithm that maximizes coverage of the given area of interest based on our objective (1) with constraints (2),(3) from Section 3. Let $c_i^j$ denote an $i$'th cycle assigned to drone $d_j$, $j \geq 1$, and let $l(c_i^j)$ denote the length of $c_i^j$, which is the sum of lengths of arcs in $c_i^j$. In particular, let $c_0^j$ denote an $i$'th cycle that is either unassigned or it is assigned to a pseudo-drone of velocity assumed $\vartheta_i = 0$.

We assume that in most real-world scenarios the number of cycles $r$ initially generated in Step 1 will be significantly greater than the number of drones to be assigned (i.e., $r \gg k'$). However, if this is not the case and even if $r < k'$ our Algorithm 1 will still perform its function.

Step 1 assures that $G$ is completely decomposed into directed cycles $(c_{i_1}^0,c_{i_2}^0,\ldots,c_{i_r}^0)$ because digraph $G$ is balanced. In particular, $d_G(v^+) = d_G(v^-)$ for every $v \in V(G)$ assures that there is an Eulerian directed walk in $G$ [4], which implies our complete decomposition. Step 1 also enhances probability that $r$ generated "short" cycles can be assigned to $k$ drones, where $k$ is defined in the following Step 3.

Based on cycles $(c_{i_1}^0,c_{i_2}^0,\ldots,c_{i_r}^0)$, in Step 2 we obtain cycles $(c_1^0,c_2^0,\ldots,c_r^0)$ that satisfy $l(c_1) \geq l(c_2) \geq \cdots \geq l(c_r)$. Depending on the value of $r$ in respect to the number of available drones $k'$, in Step 3 we set integer $k$ expressing the number of drones that will be assigned by Algorithm 1. We also initialize integer $q$ in Step 3 that identifies the first cycle $c_q^0$ that will be assigned to our first drone $d_1$. So, $q$ is the smallest positive integer such that first drone $d_1$ (not pseudo-drone) will be assigned to it. Consequently in Step 4 we assign $k$ drones to cycles based on their velocities; the higher velocity of a drone the larger feasible cycle is assigned to it out of the $k$ shortest generated cycles. So, we obtain drone-cycle assignments $(c_1^0,c_2^0,\ldots,c_{q-1}^0,c_q^1,c_{q+1}^2,\ldots,c_{r-q+1}^r)$, where cycles $c_q^0,c_{q+1}^0,\ldots,c_{k+q-1}^0$ are assigned to drones $d_1,d_2,\ldots,d_k$ respectively, and...
Algorithm 1 Optimize land coverage

**Input:** Balanced digraph $G$; Drones $d_1, d_2, \ldots, d_{k'}$ that satisfy $\vartheta_1 \geq \vartheta_2 \geq \cdots \geq \vartheta_{k'}$;

**Output:** Drones $d_1, d_2, \ldots, d_k$, $k \leq k'$, assigned to cycles $(c_{01}, c_{02}, \ldots, c_{0k})$ in $G$.

**Method:** $\text{OptLandCov}()$

Step 1. Decompose $G$ into directed short cycles by executing a single-source shortest path algorithm, one at a time, in successive iterations until complete decomposition $(c_{01}^0, c_{02}^0, \ldots, c_{0r}^0)$ of $G$ is achieved for some positive integer $r$;

Step 2. Sort cycles $(c_{01}^0, c_{02}^0, \ldots, c_{0r}^0)$ according to their lengths obtaining a permutation $(c_{01}^0, c_{02}^0, \ldots, c_{0r}^0)$ that satisfies:

$$l(c_{01}^0) \geq l(c_{02}^0) \geq \cdots \geq l(c_{0r}^0);$$

Step 3. If $(r \geq k')$ then $k := k'$ and $q := r - k' + 1$, else $k := r$ and $q := 1$;

Step 4. Assign drones $d_1, d_2, \ldots, d_k$ and pseudo-drones $d_{k+1}, d_{k+2}, \ldots, d_r$ to $(c_{01}^0, c_{02}^0, \ldots, c_{0r}^0)$ as follows:

$$(c_{01}^0, c_{02}^0, \ldots, c_{0q-1}^0, c_{0q}^1, c_{0q+1}^2, \ldots, c_{r-q+1}^r);$$

Step 5. While there exists 2drones-swap that increases objective (1) and satisfies constraints (2),(3) do

begin

execute 2drones-swap for some two cycles $c_{i1}^{j1}, c_{i2}^{j2}$ that increases (1) and satisfies (2),(3);

end

Step 6. If there exists 3drones-swap that increases objective (1) and satisfies constraints (2),(3) then

begin

execute 3drones-swap for some three cycles $c_{i1}^{j1}, c_{i2}^{j2}, c_{i3}^{j3}$ that increases (1) and satisfies (2),(3);

Go to Step 5;

end

Step 7. If there exists drones-cross that increases objective (1) and satisfies constraints (2),(3) then

begin

execute drones-cross for some two cycles $c_{i1}^{j1}, c_{i2}^{j2}$ that increases (1) and satisfies (2),(3);

Go to Step 5;

end

Step 8. Stop.
the remaining cycles $c_1^0, c_2^0, \ldots, c_{q-1}^0$ are assigned to pseudo-drones of velocity zero. In particular, if drone $d_1$ can be assigned to $c_1^0$ then we obtain assignment $c_1^1, c_2^2, \ldots, c_r^r$. Note that for some drones $d_i$ and corresponding cycles $c_{i+i-1}^0$, we might have violation of constraint (2), which means that such drones would need to be maintained (i.e., refuel/recharge) in at least two distinct locations if this constraint is not corrected by our optimization engine in Steps 5–7. In addition, constraint (3) might be violated by some drones assigned to cycle, but that would not make our initial assignment unfeasible. Note also that cycles $c_1^0, c_2^0, \ldots, c_{q-1}^0$ can be interpreted as each one being assigned to a special pseudo-drone $d_0$.

Steps 5 thorough 7 represent our optimization engine. As long as a transformation 2drones-swap is feasible (i.e., 2drones-swap exists that increases total score according to objective (1) and satisfies constraints (2) and (3)) on the currently assigned drones to cycles in $G$ we execute in Step 5 a feasible 2drones-swap. Note that even if there is just one 2drones-swap feasible at some point of execution of Algorithm 1, the loop in Step 5 might be executed multiple times. That is, an execution of 2drones-swap can create a new feasible 2drones-swap as illustrated in Fig. 4. When all feasible 2drones-swaps are exhausted, in Step 6 we execute a feasible 3drones-swap (i.e., 3drones-swap that increases total score according to objective (1) and satisfies constraints (2) and (3)) and go back to Step 5 since such a transformation could produce an assignment of drones to cycles in $G$ that allows a feasible 2drones-swap. Otherwise we execute a feasible drones-cross (i.e., drones-cross that increases total score according to objective (1) and satisfies constraints (2) and (3)) in Step 7 and go back to Step 5 since such drones-cross could produce an assignment of drones to cycles in $G$ that allows a feasible 2drones-swap and/or 3drones-swap. Finally, if no feasible drones-cross exists while we are in Step 7, our Algorithm 1 terminates in Step 8. The execution of $Opt_{Land\_Cov}()$ results in an optimized assignment of drones $d_1, d_2, \ldots, d_k$, $k \leq k'$, into cycles $(c_{i_1}^1, c_{i_2}^2, \ldots, c_{i_k}^k)$ of $G$ in respect to the coverage of the given area of interest, i.e., in respect to objective (1), with high probability that constraints (2),(3) are satisfied. In the event of violation constraint (2) by drone $d_i$ after optimization, $d_i$ would have to be refueled or recharged in at least two distinct locations on the designed flying/lending path.

Note that if $k$ drones are assigned to cycles of $G$ in Step 3 then our transformations 2drones-swap, 3drones-swap, and drones-cross in Steps 5-7 preserve $k$ drones being assigned to cycles of $G$. Hence, an alternative algorithm to Algorithm 1 can be easily derived by stopping it after either given amount of time (i.e., after predetermined time threshold) or after predetermined number of transformations on $G$.

In Fig. 4 above initially the highest speed drone is assigned to the largest
Figure 4: Propagation of 2drones-swap
cycle (i.e., thick solid line circle), the middle speed drone is assigned to the middle size dashed cycle, and the lowest speed drone is assigned to the smallest dotted cycle. These assignments are shown on the left-hand side of Fig. 4, and they could correspond to the situation after Step 4 execution of Algorithm 1. Note that in this case there is only one transformation 2drones-swap feasible, and it can be executed between middle speed and low speed drones. That is, 2drones-swap can be performed between middle size dashed circle and small dotted circle, which cross one another in two upper waypoints on the left-hand side of Fig. 4. The result of this 2drones-swap is a larger dashed cycle and smaller dotted cycle (i.e., not dashed/dotted circles any more) illustrated in the middle picture of Fig. 4. Furthermore, we observe that this 2drones-swap resulted in dashed cycle crossing solid line cycle in two lower distinct waypoints, allowing an addition 2drones-swap transformation. This second 2drones-swap transformation is now executed between high speed and middle speed drones. Specifically, the second 2drones-swap is now performed between solid circle and dashed cycle that results in the largest achievable solid cycle illustrated on the right-hand side of Fig. 4. Hence, Fig. 4 illustrates potential propagation/creation of 2drones-swap transformations as they are executed in Step 5 of Algorithm 1.

7 Example of optimization

In this section we describe a simple scenario for the optimization of land coverage based on persistent flying of three drones over digraph $G$ of order $|V(G)| = 9$ with $|E(G)| = 14$ arcs – Fig. 5. For the sake of simplicity of this example we discard constraint (3) from Section 3. We also assume that all arcs are of identical length of one (in some units of measurement), and that after initial assignment our three drones $d_1, d_2, d_3$ with speeds $\vartheta_1 = 10, \vartheta_2 = 5, \vartheta_3 = 3$ are assigned to cycles of lengths $l(c_1^1) = 5, l(c_2^2) = 3, l(c_3^3) = 3$ shown on the left-hand side of Fig. 5. Note that pseudo drone $d_4$ of speed $\vartheta_4 = 0$ is also assumed to be assigned to $c_4^4$ of length $l(c_4^4) = 3$ represented by dashed lines in Fig. 5. The total value of the coverage corresponding to the left-hand side of Fig. 5 based on (1) from
Section 3 equals initially 74.

Since neither 2drones-swap nor 3drones-swap can be executed based on left-hand side of Fig. 5, which is easy to verify, then based on Algorithm 1 drones-cross is executed that results in the middle picture of Fig. 5 of the total value of coverage increased to 89. Again, 2drones-swap cannot be executed at this stage, but 3drones-swap can be executed. Hence, based on Algorithm 1 3drones-swap is executed resulting in the right-hand side of Fig. 5 of the total value of coverage increased to 104 that represents the final optimization result.

8 Computational results

We evaluated the quality $Q$ of our optimization based on twelve simple scenarios in respect to the optimal solution, which we independently found and verified for these scenarios. The digraphs $G$ corresponding to these scenarios were all balanced with $d_G(v^+) = d_G(v^-)$ and they were derived from digraphs $H$ that were either the circulant digraph or Cartesian product of directed cycles. A circulant digraph $H_n(a_1, a_2, ..., a_k)$ on $n$ vertices with $k$ pairwise distinct jumps $a_1, a_2, ..., a_k$ has vertices $i + a_1, i + a_2, ..., i + a_k \ (\text{mod} \ n)$ adjacent to each vertex $i$, where for $k \geq j \geq 1$ each $a_j < n$. The following Fig. 6 illustrates circulant digraph $H_{12}(1, 3)$ that has been a basis for generating scenario 3 in Table 1 by substituting three of its arcs with directed paths consisting of two arcs each.

A Cartesian product $H = C_{n_1} \square C_{n_2} \square \cdots \square C_{n_k}$ of $k$ directed cycles $C_{n_1}, C_{n_2}, \ldots, C_{n_k}$ is a digraph such that the vertex set $V(H)$ equals the Cartesian product $V(C_{n_1}) \times V(C_{n_2}) \times \cdots \times V(C_{n_k})$ and there is an arc in $H$ from vertex $u = (u_1, u_2, \ldots, u_k)$ to vertex $v = (v_1, v_2, \ldots, v_k)$ if and only if there exists $1 \leq r \leq k$ such that there is an arc $(u_r, v_r)$ in $C_{n_r}$ and $u_i = v_i$ for all $i \neq r$. The following Fig. 7 illustrates Cartesian product of directed cycles $C_4 \square C_3$ that has been a basis for generating scenarios 4, 5, 10 in Table 1.

Specifically, $G$ was derived from $H$ for our simple scenarios through subdivision by replacing some of $H$ arcs with directed paths consisting of two or more arcs each. The velocities of the fixed number of drones considered for assignment to $G$ were generated randomly in the range between 3 and 50 dis-
tance units per time unit. In addition, the lengths of arcs in $G$ were assumed one unit of distance each. In Table I, ”Init Score” – $S_{init}$ represents a score calculated after the drones are assigned to $G$ in Step 4 of Algorithm 1. Similarly, ”Optimized Score” – $S_{opt}$ represents an optimized score calculated in Step 8 of Algorithm 1 when optimization of $S_{init}$ is completed through Steps 5-7.
Our $Q$ represents \^"{}goodness\^{"} of our optimized solution score $S_{opt}$ in respect to the optimal solution score $S_{opt}^*$ for a given scenario. Hence, it is defined as

$$Q = 100 \cdot \left(1 - \frac{S_{opt}^* - S_{opt}}{S_{opt}^*}\right).$$

(4)

So, the larger and closer $Q$ is to 100\% the better result we achieve for a given scenario in Table I. In addition we define the improvement $I$ resulting from our score optimization according to objective (1) with constraints (2), (3) in respect to the initial score $S_{init}$ as follows:

$$I = 100 \cdot \frac{S_{opt} - S_{init}}{S_{opt}}.$$

(5)

Again, the larger $I$ is the better result we achieve in terms of optimization for a given scenario in Table I.

![Figure 8: Computational scores](image)

![Figure 9: Improvement and quality of scores after optimization](image)

The results of execution of twelve scenarios are included in Table 1. The first four left columns correspond to the inputs and the remaining five right columns
in Table 1 correspond to the results of these twelve scenarios. The scenarios are numbered according to no decreasing order of $G$ and they correspond to the rows in Table 1. We observed that for this small set of scenarios there is no apparent correlation between the improvement of the scores from the initial assignments and the input size in respect to either $|V(G)|$, or $|E(G)|$, or $k$. This is illustrated in Fig. 8 and it is also reflected by $I$ in Fig. 9, where $I$ is in the range between 10% and 31%. Similarly, we did not observe any correlation between the goodness of our optimization $Q$ and either $|V(G)|$, or $|E(G)|$, or $k$. This is also illustrated in Fig. 9, where $Q$ varies between 89% and 100%. However, these consistent high quality results for $Q$ are very encouraging.

9 Conclusion

In this paper we presented a novel algorithm, Algorithm 1 named Opt_Land_Cov, for optimization of persistent surveillance of the given area of interest by the swarm of drones. This algorithm is based on the initial assignment of given $k$ drones to small cycles of graph $G$ in Steps 1 through 4, followed by three graph transformations on cycles of $G$ in Steps 5 through 7: (1) drones-cross, (2) 2drones-swap, and (3) 3drones-swap. These three transformations have nice property of preserving the initial assignment of $k$ drones to cycles of $G$, which allows easy modification of Opt_Land_Cov to stop either after predetermined amount of time or after predetermined number of iterations corresponding to our transformations of $G$. Furthermore, by keeping all $k$ drones assigned to cycles Opt_Land_Cov indirectly supports lethality of the target(s) engagement by the potentially weaponized drones.

Our initial assignment of the drones to small cycles of $G$ is practical for the swarm of many drones because the larger the swarm is the smaller drones are implied, whose flying range might be severely limited. Hence, assignments of the drones to other larger set of cycles in $G$ might violate constraint (2) implying that some drones cannot be refueled/recharged in one designated location while our method might produce a solution without violation of constraint (2). Conversely, if our method does not produce an assignment without violation of (2) for the given swarm of drones then it is unlikely that other assignments are possible without violation of (2).

Computational results confirm viability of our approach. The results collected based on the execution of Opt_Land_Cov on twelve small scenarios indicate that the improvement percentage over the initial coverage was significant and ranged between 10% and 31%. At the same time the deviation of optimized solution score from the optimal score corresponding to our optimization objective (1) was less than 11% in all twelve cases.
References


Received: August 29, 2018; Published: October 3, 2018