Optimal Scheduling for Refueling Multiple Autonomous Aerial Vehicles

Zhipu Jin, Tal Shima, and Corey J. Schumacher

***Abstract*—The scheduling, for autonomous refueling, of multiple unmanned aerial vehicles (UAVs) is posed as a combinatorial optimization problem. An efficient dynamic programming (DP) algorithm is introduced for finding the optimal initial refueling sequence. The optimal sequence needs to be recalculated when conditions change, such as when UAVs join or leave the queue unexpectedly. We develop a systematic shuffle scheme to reconfigure the UAV sequence using the least amount of shuffle steps. A similarity metric over UAV sequences is introduced to quantify the reconfiguration effort which is treated as an additional cost and is integrated into the DP algorithm. Feasibility and limitations of this novel approach are also discussed.**

***Index Terms*—Aerial vehicles, autonomous refueling, dynamic programming (DP), formation reconfiguration, scheduling.**

I. Introduction

UNMANNED aerial vehicles (UAVs) are commonly used in military and civilian applications. Their usefulness stems not only from the ability to avoid placing human life in harm's way, but also from improved persistence due to the removal of crew endurance limitations. Military missions include reconnaissance gathering, providing a reliable communication platform, and even carrying out precision strikes. Civilian missions include geological surveying, pipeline monitoring, and environmental surveillance.

One of the limitations of many current UAVs is the restriction in flight duration due to the limited fuel capacity. Having autonomous aerial refueling (AAR) capability will allow UAVs to remain airborne longer, thus extending their operational range and reducing the logistical trail needed for in-theater operation. It will also enable taking off with a larger payload, opening up new operational capabilities.

Aerial refueling of manned aircraft was first envisioned in 1917 by Alexander P. de Seversky, a pilot in the Imperial Russian Navy. Flight tests first took place in the early 1920s; however, World War II brought about a hiatus in aerial refueling technology development as combatants sought to develop extremely long-range aircraft with large internal fuel capacity. It became a major operational need when after World War II, jet-powered bombers, consuming far more fuel than piston-engine planes, entered service and also needed to fly farther—from the United States to targets deep in the Soviet Union and back [1]. Today, almost all manned military aircraft are equipped with aerial refueling capability. More examples and data regarding current aerial refueling systems can be found in [2] and [3].

Recently, attention has been given to the need to refuel autonomous aerial vehicles. The main challenges in single-tanker-single-UAV refueling are the need to obtain accurate measurements of the relative tanker-UAV position and to maintain stable docking in the presence of wake effects from the tanker. A vision based navigation system for autonomous refueling was studied in [4] where the UAV model is based on the AV-8B Harrier and precise docking movements in the presence of turbulence was simulated. However, temporary loss of visibility of the tanker markers may occur due to the weather and relative positions. A machine vision (MV) estimation algorithm with the capability of handling temporary visibility loss is proposed in [5]. Fravolini *et al.* designed the docking control scheme by proposing a UAV trajectory generator and a linear quadratic regulation (LQR)-based controller for smooth and reliable docking maneuvers where a fuzzy sensor fusion strategy featuring GPS and MV data was employed [6].

In this paper, we address the scheduling problem associated with the AAR of multiple UAVs by a single tanker. Considering the limited waiting time, finding the optimal refueling sequence for UAVs is similar to the scheduling problem for a single machine with "nonresumable" operations [7]. If we assume that each UAV can only be refueled once during the entire refueling process then, from the combinatorial point of view, the problem is equivalent to driving the tanker to visit each UAV in some optimal order. Thus, the problem resembles in some aspects the restricted traveling salesman problem with time windows [8], [9] and the vehicle routing problem with time windows [10].

Many efficient methods have been developed to solve such stationary NP-hard problems, including linear programming [11], [12] branch-and-bound [13], and genetic algorithm [14]. We apply the dynamic programming (DP) method based on [15], [16] to develop an efficient recursive algorithm to find the optimal initial sequence for the AAR scheduling problem. By using a prior examination and feasibility tests during the execution, the proposed algorithm efficiently reduces the search space in cases where the waiting time constraints are active.

One uniqueness of the AAR scheduling problem is that the conditions may change prior to the end of the entire refueling

Manuscript received December 5, 2005; revised April 24, 2006. This paper was recommended for publication by Associate Editor D. Sun and Editor L. Parker upon evaluation of the reviewers' comments. The work of Z. Jin was supported by the Control Science Center of Excellence, Air Force Research Labs, Wright-Patterson AFB. This work was performed while T. Shima held a National Research Council Research Associateship award at the Control Science Center of Excellence, Air Force Research Labs, Wright-Patterson AFB. This paper was presented in part at the American Control Conference, 2006.

Z. Jin is with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: jzp@caltech.edu).

T. Shima is with the Department of Aerospace Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel (e-mail: tal.shima@technion.ac.il).

C. J. Schumacher is with the Control Design and Analysis Branch, Air Vehicles Directorate, Air Force Research Laboratory, Wright-Patterson AFB, OH 45433 USA (e-mail: Corey.Schumacher@wpafb.af.mil).

Color versions of Figs. 2–7 are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2006.878793

process. The changes may occur as UAVs join the queue or leave it unexpectedly. Also, the UAVs' parameters affecting the optimization may change. These changes require reconfiguration of the UAV formation as it is dependent on the ordering of the queue. Thus, one challenge of the AAR scheduling problem is how to efficiently resolve the scheduling problem under new conditions while paying special consideration to the UAV formation reconfiguration.

Due to safety considerations, there are strict restrictions on the maneuvers that the UAVs can perform near the tanker. For simplicity, we assume that the UAV sequence can be reconfigured by shuffling the UAVs' positions. This reconfiguration is far from shuffling cards, which is relatively well studied in [17]–[19]. Shuffling a deck of cards is nothing more than subjecting it to a random permutation. Two random card shuffling algorithms were popularized by Knuth [20] and have wide applications in gambling, group theory, and computer science. However, in the AAR problem, we are interested in the minimum number of shuffling movements of UAVs to form a new sequence. Three different algorithms are developed in the paper to calculate the number of shuffle steps needed for reconfiguration given the initial and final UAV sequence. Since these movements consume fuel/time and introduce disturbances, we consider them as a reconfiguration effort and integrate it into the DP algorithm, when searching for the new optimal sequence.

The problem of maintaining and reconfiguring a stable autonomous formation has been studied in the last decade. Stability analysis of multiple agents formation control via interaction topology has been studied in [21]–[24]. Beard *et al.* [25] dealt with a satellite formation initialization problem as a crossover between formation control and target assignment. A consensus problem in multiple vehicle systems with dynamically changing topologies was addressed in [26]. Some researchers concentrated on the scalability of disturbance resistance performance [27], [28]. Other works include using Lie group for abstraction and control for robot teams [29] and vision-based mobile robots formation control [30].

Other issues related to the AAR scheduling problem are path planning, trajectory generation, and task assignment for cooperative UAVs. A fundamental structure in UAV path planning is the Voronoi diagram [31]. Zhu *et al.* [32] proposed the Delaunay triangulation method to generate way points for cooperative UAVs with Dubin's car model. Single UAV real-time motion planning and trajectory generation are discussed in [33]–[35]. Cooperative path planning and task assignment were recently investigated in [14] and [36].

In this paper, in order to focus on the combinatorial part of the AAR problem, we omit the UAV dynamics. We propose a shuffle scheme in which only one UAV will be shuffled at each time, and the cost for each movement is identical. The reconfiguration cost is related to the number of the necessary shuffle movements. However, it is impractical to directly add this cost into the DP algorithm as it will require exhaustive searching. Thus, a similarity metric over UAV sequences is introduced. The relationship between the metric and the shuffle movements is investigated. We quantify the reconfiguration cost using this metric and integrate it into the DP algorithm such that optimality may be achieved while considering the reconfiguration effort and without using exhaustive searching. Real-time trajectory planning and collision avoidance during the formation reconfiguration could be a natural extension to the proposed algorithms.

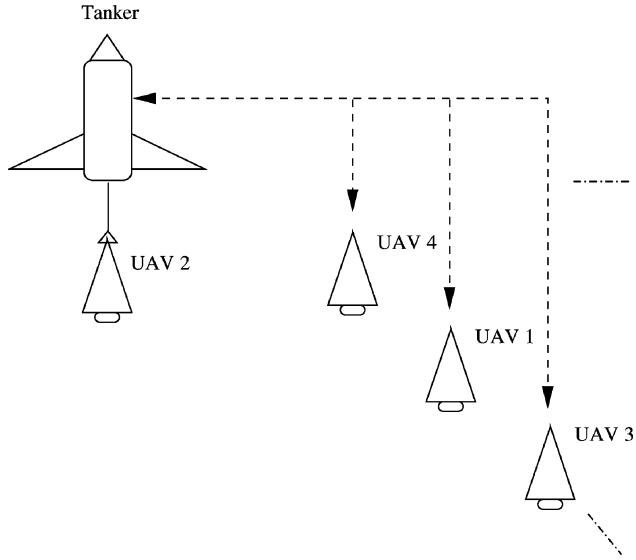


Fig. 1. Echelon formation of UAVs for AAR.

The remainder of this paper is organized as follows. In Section II we formulate the AAR scheduling problem of multiple UAVs, and an efficient recursive algorithm based on DP is developed. We examine the sequence reconfiguration in Section III where we introduce a similarity metric over UAV sequences and propose three reshuffle algorithms to implement the UAV formation reconfiguration. In Section IV we provide details on how to merge the reconfiguration cost into the recursive DP algorithm and present simulation results. Conclusions are offered in Section V. Proofs related to the DP algorithms are given in Appendix I, and stability of the UAV shuffling is studied in Appendix II.

II. Static AAR Scheduling Problem

In this section, we pose and solve the static AAR scheduling problem of multiple UAVs. In this problem, a single tanker needs to provide refueling service for multiple UAVs and the number of UAVs is known and does not change during the refueling process. Each UAV has different parameters such as current fuel level, refueling time, and return-to-field priority. We assume that communication between the tanker and UAVs is perfect; i.e., data is sent between the tanker and the UAVs without delays and errors. The tanker gathers information from all UAVs, computes the optimal refueling sequence, and sends the result back to the UAVs; thus, we deal with a centralized combinatorial optimization problem. The UAVs then form an echelon formation, as shown in Fig. 1, to follow the tanker and be refueled in an orderly manner.

A. Problem Formulation

Suppose that there are N UAVs, and each UAV is marked by an index i. The index set is $S = \{1, \ldots, N\}$. The parameters of each UAV are as follows.

- Maximum waiting time w_i—This parameter indicates how long UAV i can wait in the queue given its current fuel level. A threshold fuel level should be reserved that allows the UAV to split from the echelon formation, perform the docking maneuvers, and engage the refueling instrument. Since this threshold is known as *a priori*, w_i is calculated based on the current fuel level minus the threshold. As the UAV continuously consumes fuel, the value of w_i decreases as time progresses. We assume that $w_i > 0 \, \forall i \in S$ and w_i is sent, by each UAV, with a time stamp to enable synchronization.
- Refueling time τ_i—This is the time that the tanker needs to service UAV i. It includes the time of docking maneuvers, engagement and disengagement of refueling instrument, and fuel pumping time. According to [2] and [3], the normal service time for current probe-and-drogue refueling system is about five minutes with an actual fuel pumping duration of 1–2 min. Clearly, the actual refueling time is related to the lack of fuel. As it does not significantly change during the solution process of the assignment algorithm, we assume that τ_i is positive and time-invariant for any $i \in S$.
- Return-to-field priority p_i—This positive number is assigned to UAV i, possibly by a human operator. It indicates how important it is for the UAV to return for duty. The larger p_i is, the higher the priority is.
- Refueling sequence number k—After the optimal sequence is found, each UAV is assigned with a refueling sequence number $k \in S$. The tanker refuels UAVs according to this sequence from 1 to N.

The cost function for the static AAR scheduling problem is defined as

$$J = \sum_{i=1}^{N} \left(p_i \cdot \sum_{k=1}^{f(i)} \tau_{f^{-1}(k)} \right) \tag{1}$$

where $f(\cdot) : S \rightarrow S$ is a bijective function such that $k = f(i)$ for any UAV $i \in S$, $f(\cdot)^{-1}$ is the inverse function, and $\sum_{k=1}^{f(i)} \tau_{f^{-1}(k)}$ is the total time needed for refueling UAV i and the preceding UAVs in the queue. Suppose that the set of all possible bijective functions is F. The optimal scheduling problem is finding the function $f(\cdot) \in F$ which minimizes J. We can state this problem as

$$f(\cdot) = \arg \min_{f(\cdot) \in F} J \tag{2}$$

subject to

$$w_i \geq \sum_{k=1}^{f(i)-1} \tau_{f^{-1}(k)} \quad \forall i \in S. \tag{3}$$

Without the time constraints, there are totally $N!$ possible choices for $f(\cdot)$. However, the time constraints (3) may make some of them unfeasible. Choosing an unfeasible one means some UAVs will crash. We assume that the cost of crashing is infinity and enforce that only a feasible solution is chosen. Thus, the AAR scheduling problem is composed of two parts: (a) finding feasible sequences; and (b) obtaining the optimal one. According to the formulation, the solution of (2) may be not unique. For example, if two UAVs have the same parameters, then they can switch their position without affecting the cost. In that case, we just pick one heuristically.

Note that refueling all vehicles is one of the constraints of the optimization problem. Thus, for the sake of simplicity, we assume there always exists at least one feasible solution for the problem. In an actual implementation, if a feasible solution does not exist, then it is up to a human operator to decide which UAV can be sacrificed.

B. Recursive Algorithm Based on DP

In order to develop the search algorithm, a structure with $N+2$ layers of nodes is introduced. Each layer is marked by a number $j \in \{0, 1, \ldots, N+1\}$ which corresponds to one stage in DP. Nodes in each layer are the index numbers representing the UAVs that may be refueled at that stage. We use $i_j \in S$ to indicate these nodes, except on the initial layer ($j = 0$), where there is only one virtual starting node $i_0 = 0$, and the final layer ($j = N+1$), which only includes one virtual sink node $i_{N+1} = -1$. The node set in each layer is defined by $S_j \subseteq S$. The scheduling problem is to find an optimal path $\pi(0, -1)$ from the starting node to the sink node by connecting nodes in adjacent layers. For each layer, only one node can be visited. Also, each UAV index can be visited only once. When the path is found, the function $f(\cdot)$ is determined.

For each layer, the node set S_j is formed according to a prior examination. For node $i \in S$, if there exists a subset $K \subseteq S \backslash \{i\}$ and $|K| = j - 1$ such that

$$w_i \geq \sum_{n \in K} \tau_n \tag{4}$$

then $i \in S_j$. Note that $|K|$ is the size of the set K. This prior examination can reduce the search space when time constraints are tight. The worst-case running time for setting up the layer structure is $\Theta(N^2)$.

Following are two lemmas that are easy to prove according to the construction of the layer structure.

Lemma 2.1: If there exists a feasible path in the layer structure, then $|S_j| \geq N + 1 - j$ for any $j \in \{1, 2, \ldots, N\}$.

Lemma 2.2: For any $j \in \{1, \ldots, N-1\}$, $S_{j+1} \subseteq S_j$.

After constructing the layer structure, we break the problem into N stages, which correspond to the N layers except the initial and final layer, and transfer the problem into overlapping subproblems. We define $T(0, -1)$ as the cost of the optimal path. Before the path reaches the sink node, it must reach a node $i_N \in S_N$. Therefore

$$T(0, -1) = \min_{i_N \in S_N} \left(T(0, i_N) + d(i_N, -1) \right) \tag{5}$$

where $d(i_N, -1)$ is the cost from i_N to the sink node. For any other stage j and given the sequence $\{i_j, \ldots, i_N\}$, we have the similar recursion equation

$$T(0, i_j) = \min_{i_{j-1} \in S_{j-1} \backslash \{i_j, \ldots, i_N\}} \left(T(0, i_{j-1}) + d(i_{j-1}, i_j) \right) \tag{6}$$

where $d(i_{j-1}, i_j)$ can be calculated by

$$d(i_{j-1}, i_j) = \tau_{i_{j-1}} \sum_{n=j-1}^{N} p_{i_n}. \tag{7}$$

For the initial layer, we have

$$T(0, i_1) = d(0, i_1) = 0. \tag{8}$$

At each recursive stage, an additional feasibility test is needed. Let $\Gamma = \sum_{i=1}^{N} \tau_i$ be the total refueling time for all UAVs. At stage $(j-1)$, a feasibility test for node $i_{j-1} \in S_{j-1} \setminus \{i_j, \ldots, i_N\}$ is that if

$$w_{i_{j-1}} \geq \Gamma - \sum_{n=j-1}^{N} \tau_{i_n} \tag{9}$$

then i_{j-1} is feasible. If there is no node that passes this test, we let $T(0, i_j)$ be large enough such that it cannot be selected at the previous stage. One reasonable value is

$$T(0, i_j) = N \cdot \Gamma \cdot \max(p_i). \tag{10}$$

Given $\{i_j, \ldots, i_N\}$, nodes in layer $(j-1)$ that pass the test compose the feasible set Ω_{j-1}. When the algorithm finishes searching, if $T(0, -1) = N \cdot \Gamma \cdot \max(p_i)$, then there does not exist a feasible refueling sequence to meet the time constraints.

The computation complexity of this recursive algorithm is sensitive to the tightness of the time constraints. In the worst case, the time constraints are satisfied by any UAV sequence and the scheduling problem is solved in time $\Theta(N^2 2^N)$ [15]. The easiest case is that, when there exists only one feasible sequence, the optimal sequence is found as soon as the layer structure is determined.

The recursive DP algorithm can be used to solve general scheduling problems. Moreover, according to the structure of the cost function in (1), we find two rules that can greatly reduce the computation time.

Proposition 2.3: Suppose at stage $j-1$, Ω_{j-1} is the feasible set of layer $j-1$ for a given sequence $\{i_j, \ldots, i_N\}$. For any $m, n \in \Omega_{j-1}$, if $\tau_m = \tau_n$ and $p_m < p_n$, then

$$T(0, m) + d(m, i_j) < T(0, n) + d(n, i_j). \tag{11}$$

Proposition 2.4: Suppose at stage $j-1$, Ω_{j-1} is the feasible set of layer $j-1$ for a given sequence $\{i_j, \ldots, i_N\}$. For any $m, n \in \Omega_{j-1}$, if $p_m = p_n$ and $\tau_m < \tau_n$, then

$$T(0, m) + d(m, i_j) > T(0, n) + d(n, i_j) \tag{12}$$

The proofs can be found in Appendix I. According to the propositions, in each recursive step we pick the node with the least priority from those with the same refueling time, or the node with the largest refueling time from those with the same priority.

A simple example of the DP algorithm is illustrated through Tables I and II. Suppose there are four UAVs with the parameters listed in Table I. Table II represents the layer structure. After running the algorithm, we obtain the optimal sequence as [4,1,2,3] with the cost as 91.

TABLE I
PARAMETERS OF UAVS

UAV index i	1	2	3	4
Max. waiting time w_i	14	10	22	22
Refueling time τ_i	5	6	4	5
Priorities p_i	2	1	2	3

TABLE II
LAYER STRUCTURE OF AAR SCHEDULING PROBLEM

Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
0	1	**1**	1	**3**	**-1**
	2	2	**2**	4	
	3	3	3		
	4	4	4		

III. SEQUENCE RECONFIGURATION

Conditions of the AAR scheduling problem could be time-variant. First, the number of UAVs can change as new UAVs may join the refueling queue or some may leave due to emergency calls from other applications. Also, the parameters of the UAVs may be changed by human operators. In order to simplify this issue, we treat these changes as discrete time events.

When these changes occur we may need to reconfigure the UAV formation. Intuitively, the more similar the new optimal sequence is to the old one, the less reconfiguration is needed. In this section we first introduce a similarity metric over UAV sequences. Then, we propose three single-node shuffling algorithms to efficiently transform one sequence to another. The correspondence between the proposed metric and the number of single-node shuffle steps needed with the different algorithms is then studied. The efficiency of the algorithms is also examined.

A. Similarity Metric Over UAV Sequences

In order to define a metric to quantify the similarity over sequences with the same nodes, we need to introduce some concepts first. Suppose there is a node set M which has N nodes. A *permutation group* is a sequence group $G(N)$ whose elements are all possible sequences of M. Any element $x \in G$ is a sequence with N nodes. For each node $e_i \in M$ in a sequence $x = [e_1, e_2, \ldots, e_N]$ there exists two *adjacent nodes* $(e_i^l = e_{i-1}, e_i^r = e_{i+1})$ where e_i^l is the left neighbor and e_i^r is the right neighbor. For the first node e_1 and the last node e_N, the adjacent node pairs are $(e_1^l = \varnothing, e_1^r = e_2)$ and $(e_N^l = e_{N-1}, e_N^r = \varnothing)$, respectively, where $\varnothing$ means "None." With a little abuse of notations, we also use $(x(e_i)^l, x(e_i)^r)$ to present the adjacent nodes of node e_i in sequence x.

For any $x1, x2 \in G$, we assume set k_1 is composed of the nodes that keep identical neighbors in $x1$ and $x2$; set k_2 is composed of the nodes that only keep the same left neighbors; set k_3 is composed of the nodes that only keep the same right neighbors; and set k_4 is composed of the nodes that have different neighbors. It is clear that $|k_1| + |k_2| + |k_3| + |k_4| = N$.

For a permutation group $G(N)$, suppose $x1, x2 \in G$, a metric $\mathcal{D}(x1, x2)$ is defined by

$$\mathcal{D}(x1, x2) = \sum_{i=1}^{N} \mathcal{E}(e_i) \tag{13}$$

where $e_i \in M$ and

$$\mathcal{E}(e_i) = \begin{cases} 2, & \text{if } x1(e_i)^l \neq x2(e_i)^l \text{ and } x1(e_i)^r \neq x2(e_i)^r \\ 1, & \text{if } x1(e_i)^l \neq x2(e_i)^l \text{ and } x1(e_i)^r = x2(e_i)^r \\ 1, & \text{if } x1(e_i)^l = x2(e_i)^l \text{ and } x1(e_i)^r \neq x2(e_i)^r \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

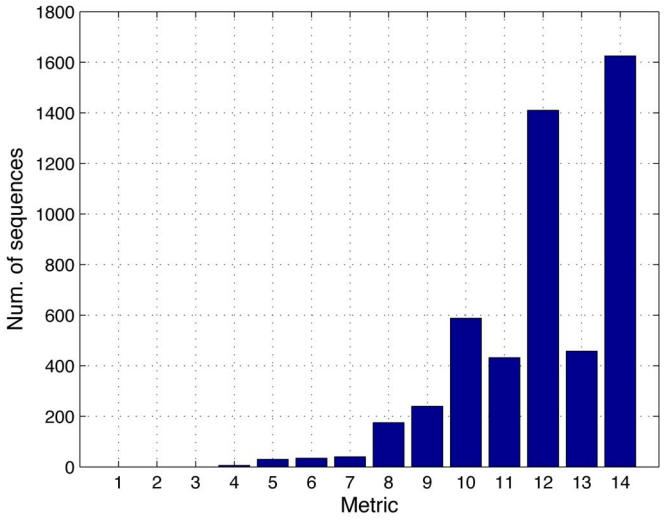


Fig. 2. Distribution of metric.

For example, suppose $M = \{1,2,3,4,5\}$; and thus G has $5! = 120$ sequence elements. Let $x1 = [1,3,4,5,2]$ and $x2 = [3,4,5,1,2]$. We obtain that $\mathcal{D}(x1,x2) = 5$ since both neighbors of node 1 are changed, the left neighbors of nodes 3 and 2 are changed, and the right neighbor of node 5 is changed.

Fig. 2 shows the distribution of the metric $\mathcal{D}$ for a permutation group of seven nodes. We pick the sequence [1, 2, 3, 4, 5, 6, 7] as the original sequence. The metric distances from this origin to all the other sequences in G are calculated and the Y axis represents the number of sequences which have the same metric distances from the origin. It is clear that most of the sequences are located far from the original. For other permutation groups, similar distributions are obtained.

According to the definition, $\mathcal{D}(x1,x2) = 0$ if and only if $x1 = x2$. Also, $\mathcal{D}(x1,x2) = \mathcal{D}(x2,x1)$. The next lemma shows that $\mathcal{D}$ satisfies the triangle inequality.

Lemma 3.1: For any $x1$, $x2$, and $x3 \in G$, $\mathcal{D}(x1,x2) + \mathcal{D}(x2,x3) \geq \mathcal{D}(x1,x3)$

Proof: Please refer to Appendix I. ■

Lemma 3.2: If $x1$, $x2 \in G$ and $x1 \neq x2$, then $4 \leq \mathcal{D}(x1,x2) \leq 2N$.

This lemma can be proved easily according to the metric definition.

B. Shuffling Algorithms

Due to safety considerations, especially for the refueling tanker, there are strict restrictions on the maneuvers that UAVs may perform while waiting in the queue. Also, the UAV echelon line must be maintained consistently to the refueling sequence. Because of these collision avoidance issues, we assume that when a refueling sequence needs to change, the reconfiguration is performed by moving one UAV at a time. We also assume that the initial sequence and the final sequence have the same size. Thus, if UAVs leave the queue, then the initial sequence is naturally formed by the ones that are left; and if new UAVs join the queue, then the initial sequence is formed by the old queue plus the new UAVs appending at the end.

We define the *single-node shuffle step* as:

Definition 3.3: A single-node shuffle step for a node sequence $x \in G$ is moving one node from its initial location in the sequence to a different one without changing the order of the other nodes.

For example, suppose we have a sequence with five nodes as $x1 = [1,2,3,4,5]$. Moving node 4 to the position between nodes 1 and 2 results in a new sequence $x2 = [1,4,2,3,5]$. Note that multiple consecutive single-node shuffle steps may be needed for a sequence transition.

The sequence transition is mathematically equal to sorting the initial sequence according to the final ordering. If we place the refueling sequence numbers of each UAV into data arrays in a computer, any sorting algorithm, such as Heapsort and Quicksort in [37], can do the transition. For sorting algorithm analysis, the running time of each instruction is counted and the algorithm is evaluated using the computational time. However, for AAR shuffling algorithms we are especially concerned with minimizing the number of single UAV moves. Moreover, we can't place UAVs in any extra spots outside the sequence, which is common in a random access machine. Thus, we judge a shuffling algorithm performance mainly based on the number of single-node moves it generates while reducing the computational time is of secondary importance.

Transferring a UAV sequence to another one, by using efficient single shuffle steps, is the main topic of this subsection. Next, we present three such algorithms that differ in their shuffling and computation efficiency. Stability of the single-node shuffle scheme is discussed in Appendix II.

1) Reshuffle Algorithm One: Suppose the initial sequence is $x1 = [a_1, a_2, \ldots, a_N]$ and the final sequence is $x2 = [b_1, b_2, \ldots, b_N]$. The algorithm is composed of the following steps:

- Let k = 1 and $\hat{x} = x1$.
- Start at the kth node in $\hat{x}$, from left to right, find the node a_i in $\hat{x}$ such that $a_i = b_k$. If $a_k = b_k$, keep $\hat{x}$ and directly jump to the next step. Otherwise, implement a single-node shuffle by moving a_i to the kth place and generate a new $\hat{x}$.
- Let k = k+1 and repeat the previous steps until $k = N-1$.

This algorithm is very similar to the "insertion sort" algorithm for sorting problems [37]. It is easy to show that the worst-case running time of the algorithm is $\Theta(N^2)$ and the upper bound on the number of single-node shuffle steps is $N-1$. The shortcoming of this algorithm is that it cannot guarantee finding the minimum number of single-node shuffle steps for a sequence transition. For example, suppose $x1 = [1,2,3,4,5]$ and $x2 = [2,3,4,5,1]$. The algorithm enables transformation using four shuffle steps. Obviously, the minimum number is one by moving node 1 to the right side of node 5.

2) Reshuffle Algorithm Two: In order to find a better reshuffle algorithm, we introduce the concept of *subsequence partition*. For two sequences $x1$, $x2$, there exists a subsequence partition such that each element δ is a non-empty subsequence for both $x1$ and $x2$ and the number of the elements in the partition is minimized. For example, suppose $x1 = [1,2,3,4,5]$ and $x2 = [3,4,5,1,2]$, $\{[1,2],[3,4,5]\}$ is the subsequence partition of $x1$ and $x2$. For any two sequences with the same nodes, the subsequence partition is unique. By switching the positions of

these subsequences, $x1$ can be transferred into $x2$. We define the *subsequence shuffle step* as:

Definition 3.4: A subsequence shuffle step of sequence x is moving a subsequence δ to a different location. The node order inside δ is not changed.

Using subsequence shuffle steps resembles shuffling a deck of cards by moving multiple cards together. Thus, a sequence transition can be treated in two levels: subsequence level and node level. Obviously, a subsequence shuffle step can be composed by $|\delta|$ single-node shuffles if we can move only one node each time.

For a sequence transition, it is important to find the subsequence partition. Similar to single node, we define the *left subsequence neighbor* of subsequence δ in x as $x(\delta)^l$ and the *right subsequence neighbor* as $x(\delta)^r$. All the elements of the subsequence partition can be put into three subsequence sets Λ_1, Λ_2, and Λ_3. Elements in Λ_1 only keep the same left subsequence neighbors in $x1$ and $x2$. Elements in Λ_2 only keep the same right subsequence neighbors. Elements in Λ_3 do not have any same subsequence neighbor in $x1$ and $x2$.

Finding Λ_1, Λ_2, and Λ_3 can be done in time $\Theta(N)$. Suppose we already have k_1, k_2, k_3, and k_4 which are defined in Section III-A. Let us start from the nodes of k_1. Suppose node $e \in k_1$, then subsequence $\delta = [x1(e)^l, e, x1(e)^r]$ does not change its node order in the transition from $x1$ to $x2$. If $x1(e)^l \in k_1$, then δ extends by adding $x1(e)^l$'s left neighbor on its left side until this left neighbor is $\varnothing$. If $x1(e)^l \in k_3$, then δ cannot extend itself on the left side. The same extending process can be done for $x1(e)^r$. Eventually, δ is extended to be the largest subsequence including e. During this process, the nodes used to form δ are eliminated from k_1, k_2, and k_3. When $k_1 = \emptyset$, we check any single node in k_2. If a node b belongs to k_2 and its left neighbor $x1(b)^l \neq \varnothing$, then $x1(b)^l$ must belong to k_3 since $k_1 = \emptyset$. Thus, $[x1(b)^l, b]$ is formed and belongs to Λ_3. A similar process can be performed for k_3. After the extending processes, we can find Λ_1, Λ_2, and Λ_3.

Lemma 3.5: $|\Lambda_1| \leq 1$ and $|\Lambda_2| \leq 1$.

Proof: The only possible element in Λ_1 is the subsequence that is located on the first position from the left in both $x1$ and $x2$. The same result holds for Λ_2. ■

Given the initial sequence $x1$ and final sequence $x2$, suppose we have Λ_1, Λ_2, and Λ_3, then reshuffle Algorithm Two is as follows.

- Find the smallest subsequence δ in Λ_3 with the condition that no node in δ has been moved before.
- According to $x2$, find the left subsequence neighbor $x2(\delta)^l$ and the right subsequence neighbor $x2(\delta)^r$.
- Implement a subsequence shuffle step such that
  - If $x2(\delta)^l \in \Lambda_1$ or if $x2(\delta)^l \notin \Lambda_1$ and $|x2(\delta)^l| \geq |x2(\delta)^r|$, then put δ on the right side of $x2(\delta)^l$ to form a new subsequence.
  - If $x2(\delta)^r \in \Lambda_2$ or if $x2(\delta)^r \notin \Lambda_2$ and $|x2(\delta)^l| < |x2(\delta)^r|$, then put δ on the left side of $x2(\delta)^r$ to form a new subsequence.
- Update Λ_1, Λ_2, and Λ_3 according to the new subsequences.
- Repeat the previous steps until Λ_3 is empty.

For this algorithm, the important parts are moving subsequences in Λ_3 to generate longer subsequences and to reduce the size of Λ_3. Whenever one subsequence δ is moved, the size of Λ_3 is decreased at least by one. This algorithm can run in polynomial time, but it cannot guarantee finding the minimum number of single-node steps.

3) Reshuffle Algorithm Three: Using the principle of optimality, we develop an algorithm to find the minimum number of single-node shuffle steps.

Suppose Λ_1, Λ_2, and Λ_3 are subsequences sets for $x1$ and $x2$. There are m elements in Λ_3. The minimum single-node shuffle steps are represented by $\mathcal{T}(m)$. We have

$$\mathcal{T}(m) = \min_{\delta_i \in \Lambda_3} \left(\min \left(\mathcal{T}(\hat{m})^l + |\delta_i|, \mathcal{T}(\hat{m})^r + |\delta_i| \right) \right) \quad (15)$$

where $\mathcal{T}(\hat{m})^l$ is the minimum number of single-node shuffle steps after δ_i is moved to the right side of its left subsequence neighbor, and $\mathcal{T}(\hat{m})^r$ is defined similarly. The subsequence set Λ_3 needs to be updated at each recursive step. Let $\hat{m}$ represent the size of Λ_3 after updating. Sometimes, moving δ_i to the right side of its left neighbor is also connecting it to its right neighbor. In that case, these three subsequences are formed into one larger subsequence and the elements number of Λ_3 is reduced by two. This recursive algorithm terminates when $\hat{m} = 0$.

$\mathcal{T}(m)$ is guaranteed to be the minimum number of single-node shuffles steps. The computation time of this recursive algorithm depends on m, i.e., how many elements exist in Λ_3. By using a DP method, the worst-case running time is $\Theta(m^2 2^m)$.

C. Comparison and Discussion

In order to verify the feasibility of these algorithms, we tested them on permutation groups with different numbers of nodes. The algorithms were coded in Matlab and run on a desktop computer with a Xeon CPU at 2.66 GHz and 1.00 GB of RAM. For each permutation group, we randomly picked a sequence as the initial one and calculated the shuffle steps that transfer this initial sequence to all the other sequences in the permutation group. The average values of shuffle steps and computation time are calculated over each permutation group.

We first examine the correspondence between the metric $\mathcal{D}$ and the number of single-node shuffle steps needed to transform from one sequence to another with the different algorithms. For $G(7)$, the average values and standard deviation of single-node shuffle steps with respect to the value of the metric $\mathcal{D}$ are shown in Figs. 3 and 4. These plots show that, on average, there exists a good correspondence between the metric value and the number of single-node shuffle steps. Similar results have been obtained for other permutation groups with different number of nodes. Also, there exist fluctuations corresponding to even and odd metric values. The reason lies in the metric definition in Section III-A. When the reconfiguration involves one boundary node, the expected number of nodes in k_4 is smaller and results in less shuffling steps.

Fig. 5 shows the average number of shuffle steps for the three different shuffling algorithms. For all algorithms the upper bound of the number of single-node shuffle steps is $N - 1$. Fig. 6 shows the average computation time for the different algorithms. According to the simulation results, Algorithm Three guarantees the least amount of single-node shuffle steps

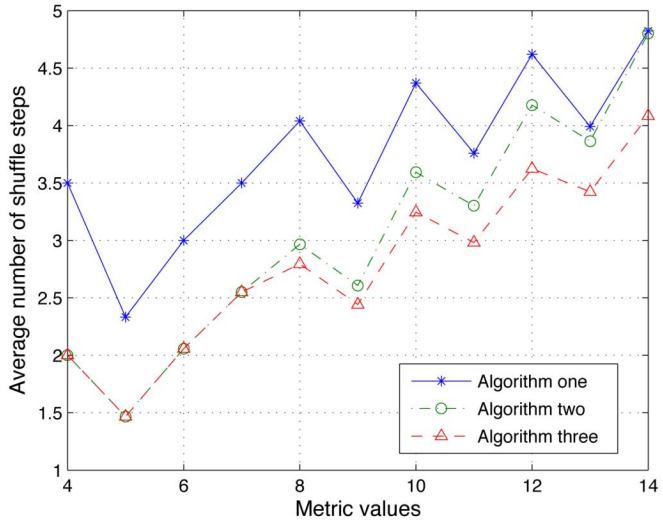


Fig. 3. Mean values of shuffle steps with respect to metric for $G(7)$.

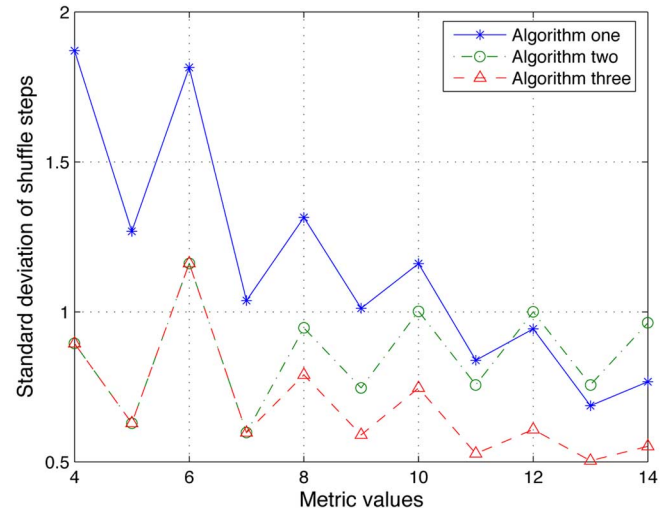


Fig. 4. Standard deviation of shuffle steps with respect to metric for $G(7)$.

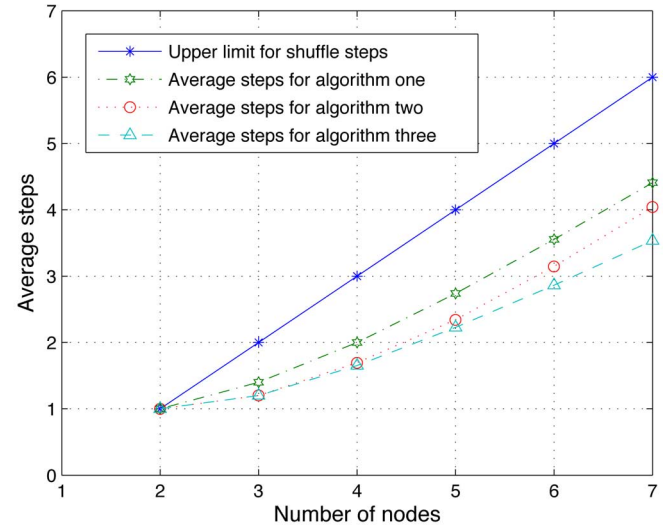


Fig. 5. Average number of shuffle steps for the different algorithms.

for a sequence transition, but it needs considerably more computational time.

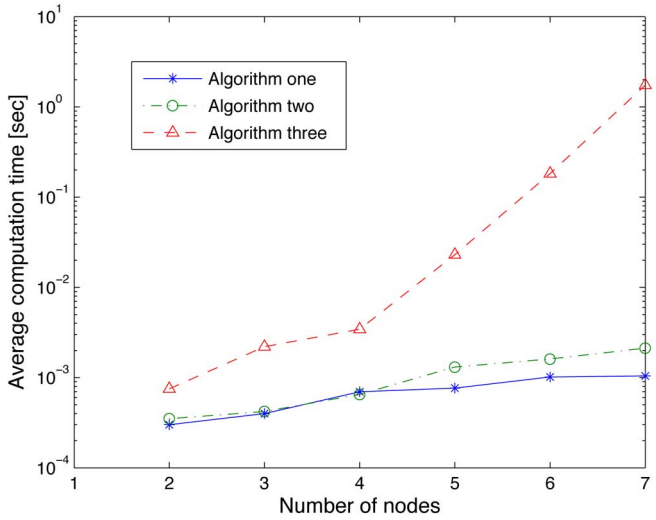


Fig. 6. Average computation time for the different algorithms.

IV. AAR Scheduling Problem With Time-Variant Conditions

This section deals with the time-variant AAR scheduling problem where the configuration needs to be resolved due to condition changes. For small groups of UAVs, the exhaustive searching method based on minimal shuffle steps could be used. We could generate the number of shuffle steps for each feasible refueling sequence, add it to the priority cost, and find the minimal one. The worst-case running time for exhaustive searching is $\Theta(N! \cdot N^2 2^N)$.

On the other hand, we want to use the DP method to get a solution quickly. Since the cost of reconfiguration depends on the number of single-node shuffle steps and this value can be determined only after the new sequence is known, we cannot directly consider the number of shuffle steps into the recursive algorithm discussed in Section II. Thus, we choose the similarity metric to estimate the reconfiguration cost and slightly modify the recursive algorithm.

A. Time-Variant Problem

Suppose the initial refueling sequence is $\pi_o = [i_1, \ldots, i_N]$. For each node i_j, the two adjacent nodes are $(i_j^l = i_{j-1}, i_j^r = i_{j+1})$. The new optimal sequence is indicated by π_n. We redefine the total cost function for refueling scheduling as

$$J = \sum_{i=1}^{N} \left(p_i \cdot \sum_{k=1}^{f_n(i)} \tau_{f_n^{-1}(k)} \right) + K \cdot \mathcal{D}(\pi_o, \pi_n) = \sum_{i=1}^{N} \left(p_i \cdot \sum_{k=1}^{f_n(i)} \tau_{f_n^{-1}(k)} \right) + K \sum_{j=1}^{N} \mathcal{E}(i_j) \tag{16}$$

where $f_n(\cdot)$ is the mapping function for π_n, the second term represents the metric distance, and K is the relative weight of the reconfiguration cost. Also, there are N time constraints as listed:

$$w_i \geq \sum_{k=1}^{f_n(i)-1} \tau_{f_n^{-1}(k)} \quad \forall i \in \{1, \ldots, N\}. \tag{17}$$

B. DP Algorithm

The additive property of the new cost function makes the DP algorithm in Section II still effective. The cost $d(i_{j-1}, i_j)$ in each recursive step is calculated by

$$d(i_{j-1}, i_j) = \tau_{i_{j-1}} \cdot \sum_{n=j-1}^{N} p_{i_n} + K \cdot S(i_{j-1}, i_j) \tag{18}$$

where $S(i_{j-1}, i_j)$ is

$$\mathcal{S}(i_{j-1}, i_j) = \begin{cases} 2, & \text{if } \pi_o(i_{j-1})^r \neq i_j \text{ and } \pi_o(i_j)^l \neq i^r_{j-1} \\ 1, & \text{if } \pi_o(i_{j-1})^r \neq i_j \text{ and } \pi_o(i_j)^l = i^r_{j-1} \\ 1, & \text{if } \pi_o(i_{j-1})^r = i_j \text{ and } \pi_o(i_j)^l \neq i^r_{j-1} \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

For the first layer, we have $d(0, i_1) = K \cdot S(0, i_1)$ where

$$S(0, i_1) = \begin{cases} 1, & \text{if } \pi_o(i_1)^l \neq \varnothing \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

For the final layer, we have

$$S(i_N, -1) = \begin{cases} 1, & \text{if } \pi_o(i_N)^r \neq \varnothing \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

Since we consider the metric as a part of the cost, *Propositions 2.3* and *2.4* are no longer valid. The worst-case running time for the algorithm is still $\Theta(N^2 2^N)$.

C. Example

We extend the example in Section II into a more complicated scenario. Suppose at the initial time ($t = 0$) the tanker is refueling one UAV and it will take five time units to finish. There are four UAVs waiting in line with the parameters given in Table I. According to Section II, the optimal sequence is [4, 1, 2, 3]. Let's consider the following three discrete events.

1) At $t = 2$, two UAVs (with index number 5 and 6) arrive for refueling.
2) At $t = 4$, UAV 5 leaves the echelon formation due to an emergency call.
3) At $t = 6$, another two UAVs (number 7 and 8) join the sequence and the priorities of UAV 1 and 2 are changed.

These events, the parameters for the corresponding UAVs, and the optimal sequences are listed in Table III. The coefficient of reconfiguration cost is set as $K = 10$. Note that the maximum waiting time w_i decreases as time progresses; and that between events 2 and 3, UAV 6 splits from the formation to refuel since the tanker is free at that time. The solution obtained using the algorithm is also presented in the tables along with the minimum number of shuffle movements. Obviously, when we consider the reconfiguration cost, the necessary number of shuffle steps is reduced.

We also deal with these events by using exhaustive search. Since, according to Fig. 3, the metric value approximates by a factor of three the number of shuffle steps, we choose the coefficient for the cost of reconfiguration as $K = 30$. Exhaustive searching generated the same results but consumed much more computation time. This is shown in Fig. 7. Note that at events 2 and 3, the time constraints become tight and reduce the computation time considerably.

TABLE III
PARAMETERS OF UAVS AND OPTIMAL SEQUENCES

Event 1 at time $t = 2$	**UAV parameters**					
UAV index i	1	2	3	4	5	6
Max. waiting time w_i	12	8	20	20	28	20
Refueling time τ_i	5	6	4	5	3	2
Priorities p_i	2	1	2	3	5	6
Initial sequence	4	1	2	3	5	6
New optimal UAV refueling sequence						
Without reconfiguration cost	6	5	2	1	3	4
Minimum number of shuffle movements						4
With reconfiguration cost	6	1	2	3	5	4
Minimum number of shuffle movements						2

Event 2 at time $t = 4$	**UAV parameters**				
UAV index i	1	2	3	4	6
Max. waiting time w_i	10	6	18	18	18
Refueling time τ_i	5	6	4	5	2
Priorities p_i	2	1	2	3	6
Initial sequence	6	1	2	3	4
New optimal UAV refueling sequence					
Without reconfiguration cost	6	2	1	4	3
Minimum number of shuffle movements					2
With reconfiguration cost	6	2	1	3	4
Minimum number of shuffle movements					1

Event 3 at time $t = 6$	**UAV parameters**					
UAV index i	1	2	3	4	7	8
Max. waiting time w_i	8	4	16	16	25	21
Refueling time τ_i	5	6	4	5	4	3
Priorities p_i	6	4	2	3	3	2
Initial sequence	2	1	3	4	7	8
New optimal UAV refueling sequence						
Without reconfiguration cost	2	1	4	3	8	7
Minimum number of shuffle movements						2
With reconfiguration cost	2	1	3	4	8	7
Minimum number of shuffle movements						1

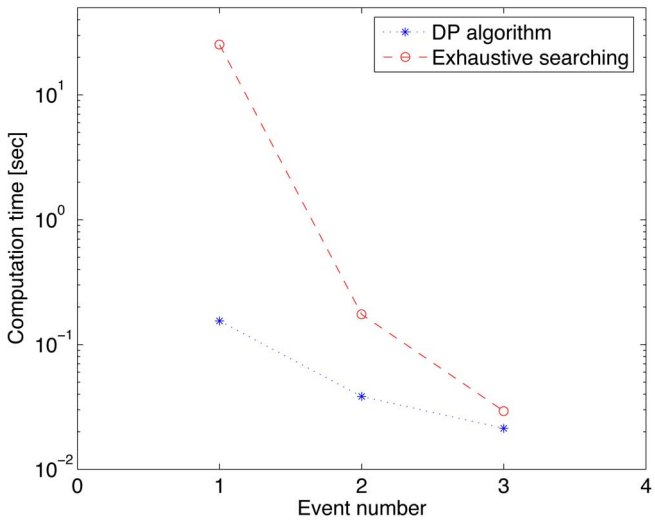


Fig. 7. Computation time for DP algorithm and exhaustive searching.

V. CONCLUSION

A recursive DP algorithm was developed for the AAR scheduling problem with static conditions. The optimal sequence is dependent on the UAVs' parameters, including time constraints. When the refueling time constraints are tight, a

prior examination and feasibility tests in each recursive step are necessary to reduce the search space and thus make the search more efficient.

The optimal refueling sequence needs to be recalculated when the conditions change. We introduced a new metric to indicate the similarity between sequences, and chose it as the reconfiguration cost. The additive property of the metric make it possible to add it to the DP algorithm as an additional cost term. Efficient reshuffle algorithms have also been proposed. It was shown that there exists a good correspondence between the new metric and the number of single-node shuffle steps needed for the reconfiguration using these shuffling algorithms.

Generally, the optimal scheduling of dynamic systems can be sensitive to the system states and constraints. Thus, the optimal solution could be totally different due to small changes in the conditions, resulting in high reconfiguration costs. The aforementioned AAR problem is a good example. The proposed framework naturally introduces this difference into the recursive DP algorithm. The novel approach can be used in general scheduling problems of dynamic systems, for taking into account the reconfiguration cost.

APPENDIX I
PROOFS

Proof: [*Proposition 2.3*] Since $m, n \in \Omega_{j-1}$, if there exists a feasible path $\pi(0,m)$ from the initial layer to layer $(j-1)$, then n must be on it since n must be visited before m. By switching the positions of m and n, we get another feasible path $\pi(0,n)$ because:

(a) if m is feasible in layer $j-1$, it is feasible in any other layer ahead of j;
(b) node n can be switched back since it is in Ω_{j-1};
(c) since $\tau_m = \tau_n$, the switching does not affect other nodes on the path.

The same result is true for any $\pi(0,n)$. Thus, there exists a bijective mapping between feasible path sets $\{\pi(0,m)\}$ and $\{\pi(0,n)\}$. Suppose the nodes on a path between n and m compose a node set $\mathcal{Q}$. Let $t(n,i_j)$ be the time interval from starting refueling n to finishing refueling i_j. For each pair $[\pi(0,m),\pi(0,n)]$, the difference of the cost between $\pi(n,m)$ and $\pi(m,n)$ is considered and we have

$$\begin{aligned} &T(0,m)+d(m,i_j)-T(0,n)-d(n,i_j)\\ &\quad=p_n\tau_n+\sum_{l\in\mathcal{Q}}p_l t(n,l)+p_m t(n,m)\\ &\qquad-p_m\tau_m-\sum_{l\in\mathcal{Q}}p_l t(n,l)-p_n t(m,n)\\ &\quad=(p_m-p_n)\left(t(n,m)-\tau_n\right)\\ &\quad<0 \end{aligned}$$

since $(t(n,m)-\tau_n)>0$. ∎

Proof: [*Proposition 2.4*] Using similar arguments and notations as in the previous proposition proof, there exists an injective mapping from feasible path set $\{\pi(0,m)\}$ to feasible path set $\{\pi(0,n)\}$. For each $\pi(0,m)$ and corresponding $\pi(0,n)$, we have

$$\begin{aligned} &T(0,m)+d(m,i_j)-T(0,n)-d(n,i_j)\\ &\quad=p_n\tau_n+\sum_{l\in\mathcal{Q}}p_l t(n,l)+p_m t(n,m)\\ &\qquad-p_m\tau_m-\sum_{l\in\mathcal{Q}}p_l t(m,l)-p_n t(m,n)\\ &\quad=p_m(\tau_n-\tau_m)+\sum_{l\in\mathcal{Q}}p_l(\tau_n-\tau_m)\\ &\quad>0. \end{aligned}$$

Thus, the minimum over a subset of $\{\pi(0,n)\}$ is smaller than the minimum over $\{\pi(0,m)\}$ and the result follows. ∎

Proof: [*Lemma 3.1*] Suppose that, for $x1$ and $x2$, there exist four node sets as k_1, k_2, k_3, and k_4 that are defined earlier. So

$$\mathcal{D}(x1,x2)=|k_2|+|k_3|+2|k_4|. \tag{22}$$

For $x2$ and $x3$, there exist similar node sets as $\hat{k}_1$, $\hat{k}_2$, $\hat{k}_3$, and $\hat{k}_4$. Thus

$$\mathcal{D}(x2,x3)=|\hat{k}_2|+|\hat{k}_3|+2|\hat{k}_4|. \tag{23}$$

Now suppose that the intersection of k_1 and $\hat{k}_1$ has r nodes, then between $x1$ and $x3$, there are at least r nodes that have identical neighbors. We rewrite k_1 and $\hat{k}_1$ as

$$\begin{cases} |k_1|=r+\alpha\\ |\hat{k}_1|=r+\beta. \end{cases} \tag{24}$$

Note that these α nodes must belong to $\hat{k}_2\bigcup\hat{k}_3\bigcup\hat{k}_4$. Since nodes in $\hat{k}_2$, $\hat{k}_3$, or $\hat{k}_4$ make different contributions to $\mathcal{D}$, we assume that there are $n1$ nodes in α that belong to $\hat{k}_2\bigcup\hat{k}_3$ and $n2$ nodes that belong to $\hat{k}_4$. Thus, we have

$$\begin{aligned} \alpha&=n1+n2\\ \mathcal{D}(x1,x2)&=n1+2\cdot n2+\Psi_1. \end{aligned} \tag{25}$$

For the same reasons, we have

$$\begin{aligned} \beta&=m1+m2\\ \mathcal{D}(x2,x3)&=m1+2\cdot m2+\Psi_2. \end{aligned} \tag{26}$$

For $x1$ and $x3$, we have

$$\begin{aligned} \mathcal{D}(x1,x3)&\le n1+2\cdot n2+m1+2\cdot m2+\Psi_1+\Psi_2\\ &=\mathcal{D}(x1,x2)+\mathcal{D}(x2,x3). \end{aligned} \tag{27}$$

∎

APPENDIX II
STABILITY ANALYSIS OF UAV SHUFFLE

In this section, we discuss the stability of a UAV echelon line for AAR when a UAV shuffle is performed.

A. Information Flow and Formulation

We assume that each UAV can obtain the states of its immediate neighbors in the echelon line by perfect wireless communication channels. When a UAV changes it's location the topology

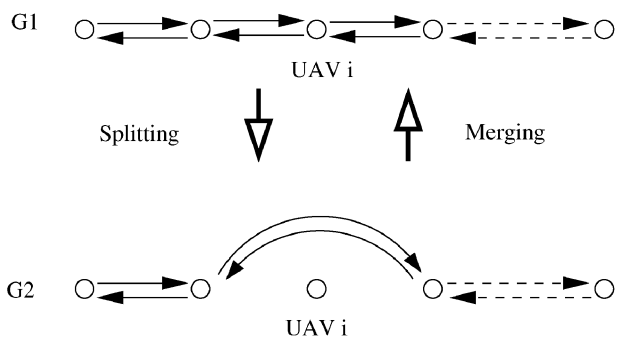


Fig. 8. Local information flow of single UAV shuffle step.

of local information flow[1] changes twice. We illustrate these changes in Fig. 8 and name them as follows.

- **Splitting**—A UAV splits out from the echelon line by decoupling its dynamics from its neighbors. Then, those two immediate neighbors set up connections and couple with each other. If the UAV is the first or the last of the platoon, it just decouples from its single immediate neighbor.
- **Merging**—When a UAV merges into a new position, it sets up connections with its new immediate neighbors. The original connections between these neighbors are terminated.

Suppose that each UAV is represented by an identical linear, time-invariant system model as

$$\dot{x}_i = Ax_i + B \cdot \sum_{j \in \mathcal{N}} x_j. \tag{28}$$

where A and B are n-by-n matrices, and $\mathcal{N}$ is the immediate neighbor set of UAV i. We assume that A is Hurwitz, i.e., each UAV is equipped with an on-board stabilizing controller. The entire echelon line with M UAVs can be presented by

$$\begin{aligned}\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{M-1} \\ \dot{x}_M \end{bmatrix} &= \begin{bmatrix} A & B & 0 & 0 & \cdots \\ B & A & B & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & B & A & B \\ \cdots & 0 & 0 & B & A \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{M-1} \\ x_M \end{bmatrix} \\ &= (I_M \otimes A + \mathcal{L}_M \otimes B) \cdot \mathcal{X}_M \\ &= \mathcal{A}_M \cdot \mathcal{X}_M \end{aligned} \tag{29}$$

where $\otimes$ represents the Kronecker product, I_M is a M-by-M identity matrix, $\mathcal{L}_M$ is an M-by-M matrix as

$$\mathcal{L}_M = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots \\ 1 & 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & 1 & 0 & 1 \\ \cdots & 0 & 0 & 1 & 0 \end{bmatrix}$$

[1]Local information flow only describes the states exchanged between immediate neighbors.

and $\mathcal{X}_M$ is the state vector. When UAV i splits away from the formation, its dynamics is decoupled from its neighbors, and (29) is switched to the following equation:

$$\begin{bmatrix} \dot{\mathcal{X}}_{M-1} \\ \dot{x}_i \end{bmatrix} = \begin{bmatrix} \mathcal{A}_{M-1} & 0 \\ 0 & A_i \end{bmatrix} \begin{bmatrix} \mathcal{X}_{M-1} \\ x_i \end{bmatrix}. \tag{30}$$

Similarly, suppose UAV k merges into the position between UAV $(i-1)$ and i. The dynamics before the merging is

$$\begin{bmatrix} \dot{\mathcal{X}}_{M-1} \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} \mathcal{A}_{M-1} & 0 \\ 0 & A_k \end{bmatrix} \cdot \begin{bmatrix} \mathcal{X}_{M-1} \\ x_k \end{bmatrix} \tag{31}$$

and, after merging, is identical to (29).

B. Stability for Topology Switches

For the stability of the entire formation, we have the following theorem.

Theorem 2.1: An echelon formation given by (29) is stable if and only if the following M subsystems

$$\dot{x} = (A + \lambda_i B)x$$

are stable simultaneously where λ_i are eigenvalues of $\mathcal{L}_M$.

Proof: This theorem is easy to proof using properties of the Kronecker product and Schur decomposition. ■

Since A is Hurwitz, there exists symmetric matrices $P > 0$ and $Q > 0$ such that

$$A^*P + PA = -Q.$$

Then we have

Lemma 2.2: A sufficient condition to stable the formation given by (29) is that

$$B^*P + PB < Q/2. \tag{32}$$

Proof: Since the eigenvalues λ_i of $\mathcal{L}_M$ are real and $|\lambda_i| \leq 2$, for any subsystem in *Theorem 2.1*, we choose the same Lyapunov function $v(x) = x^*Px$ and obtain

$$\begin{aligned}\dot{v}(x) &= x^* \left(A^*P + PA + \lambda_i(B^*P + PB)\right) x \\ &< -(1 - \lambda_i/2)x^*Qx \\ &< 0. \end{aligned}$$

and the result follows. ■

It is well known in algebraic graph theory [39] that $L = CC^*$ where L and C are Laplacian matrix and incidence matrix of a topology respectively. For topology switchings, we have the following theorem.

Theorem 2.3: When B satisfies

$$0 < B^*P + PB < Q/2 \tag{33}$$

the topology switchings of the echelon UAV formation are stable.

Proof: Suppose the echelon line in Fig. 8 has M UAVs. According to *Lemma 2.2*, the formation is stable before the splitting. In order to show that the switching is stable, a common Lyapunov function is needed before and after the splitting [38]. We define a Lyapunov function $V(\mathcal{X}_M)$ as

$$V(\mathcal{X}_M) = \sum_{i=1}^{M} x_i^* P x_i = \mathcal{X}_M^* (I_M \otimes P)\mathcal{X}_M = \mathcal{X}_M^* P_M \mathcal{X}_M.$$

Obviously, $P_M > 0$ and $V(\mathcal{X}_M) \geq 0$. We obtain

$$\begin{aligned}\dot{V}(\mathcal{X}_M) &= \mathcal{X}_M^* \left((I_M \otimes A + \mathcal{L}_M \otimes B)^* P_M \right. \\ &\qquad \left. + P_M (I_M \otimes A + \mathcal{L}_M \otimes B) \right) \mathcal{X}_M \\ &= \mathcal{X}_M^* W_M \mathcal{X}_M\end{aligned}$$

where W_M is

$$\begin{bmatrix} A^*P + PA & B^*P + PB & 0 & \cdots \\ B^*P + PB & A^*P + PA & B^*P + PB & \cdots \\ \vdots & \vdots & \vdots & \vdots \\ \cdots & 0 & B^*P + PB & A^*P + PA \end{bmatrix}.$$

We need to show that $\dot{V}(\mathcal{X}_{\mathcal{M}}) \leq 0$, i.e., W_M is negative definite.

Since $0 < B^*P + PB$, there exists a real nonsingular matrix R such that $B^*P + PB = RR^*$. We obtain that

$$\begin{aligned}W_M &< I_M \otimes (-RR^*) + \mathcal{L}_M \otimes (RR^*) \\ &= -(I_M - \mathcal{L}_M) \otimes (RR^*)\end{aligned}$$

Let L_M be the Laplacian matrix of graph G_1 shown in Fig. 8 and $L_M = C_M C_M^*$. Let $Z = \mathrm{diag}[0, 1, 1, \cdots, 1, 0]$ and we have

$$\begin{aligned}W_M &= -(L_M - Z) \otimes (RR^*) \\ &< -L_M \otimes (RR^*) \\ &= -CC^* \otimes (RR^*) \\ &= -(C \otimes R)(C \otimes R)^*.\end{aligned}$$

So $W_M < 0$.

When UAV i splits away from the formation, the function $V(\mathcal{X}_M)$ is still a Lyapunov function for the new echelon line plus the UAV i, and

$$\dot{V}(\mathcal{X}_M) = \mathcal{X}_M^* \hat{W} \mathcal{X}_M$$

where

$$\hat{W} = \begin{bmatrix} W_{M-1} & 0 \\ 0 & -Q \end{bmatrix} < 0.$$

So splitting is stable. Using the same approach, the stability of merging is easy to show. ■

Theorem 2.3 gives a sufficient condition for the stability of topology switchings. For AAR, it guarantees that the UAV formation is stable when a single-node shuffle step is performed. The analysis of stability is independent of the size of the formation, i.e., the result is scalable.

ACKNOWLEDGMENT

The authors would like to thank Prof. K. M. Passino from The Ohio State University and Prof. R. M. Murray from the California Institute of Technology, Pasadena, for helpful discussions.

REFERENCES

[1] D. A. Day, U.S. Centennial of Flight Commission, Aerial Refueling [Online]. Available: http://www.centennialofflight.gov/essay/Evolution_of_Technology/refueli%ng/Tech22.htm

[2] C. Bolkcom and J. D. Klaus, Air Force Aerial Refueling Methods: Flying Boom Versus Hose-and-Drogue, CRS Report for Congress 2005 [Online]. Available: http://www.fas.org/sgp/crs/weapons/RL32910.pdf

[3] W. R. Gates and M. J. McCarthy, "United States Marine Corps aerial refueling requirements analysis," in *Proc. Winter Simul. Conf.*, K. K. J. A. Joines, R. R. Barton, and P. A. Fishwick, Eds., 2000, pp. 1075–1081 [Online]. Available: http://www.informs-sim.org/wsc00papers/142.PDF

[4] J. Kimmett, J. Valasek, and J. L. Junkins, "Autonomous aerial refueling utilizing a vision based navigation system," in *Proc. AIAA Guid., Nav., Control Conf. Exhibit*, Monterey, CA, Aug. 2002, CD-ROM.

[5] G. Campa, M. L. Fravolini, A. Ficola, M. R. Napolitano, B. Seanor, and M. G. Perhinschi, "Autonomous aerial refueling for UAVs using a combined GPS-machine vision guidance," in *Proc. AIAA Guid., Nav., Control Conf. Exhibit*, Providence, RI, Aug. 2004, CD-ROM.

[6] M. L. Fravolini, A. Ficola, G. Campa, M. R. Napolitano, and B. Seanor, "Modeling and control issues for autonomous aerial refueling for UAVs using a probe-drogue refueling system," *Aerosp. Sci. Technol.*, vol. 8, pp. 611–618, 2004.

[7] P. Mauguiere, J. C. Billaut, and J. L. Bouquard, "New single machine and job-shop scheduling problems with availability constraints," *J. Schedul.*, vol. 8, no. 3, pp. 211–231, Jun. 2005.

[8] Y. Dumas, J. Desrosiers, E. Gelinas, and M. M. Solomon, "An optimal algorithm for the traveling salesman problem with time windows," *Oper. Res.*, vol. 43, no. 2, pp. 367–371, Mar.–Apr. 1995.

[9] E. Balas and N. Simonetti, "Linear time dynamic-programming algorithms for new classes of restricted TSPs: A computational study," *INFORMS J. Comput.*, vol. 13, no. 1, pp. 56–75, Winter, 2001.

[10] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Oper. Res.*, vol. 40, no. 2, pp. 342–354, Mar.–Apr. 1992.

[11] A. Richards, J. Bellingham, M. Tillerson, and J. P. How, "Coordination and control of multiple uavs," in *Proc. AIAA Guid., Nav., Control Conf. Exhibit*, Monterey, CA, 2002, CD-ROM.

[12] C. J. Schumacher, P. R. Chandler, M. Pachter, and L. Pachter, "Constrained optimization for UAV task assignment," in *Proc. AIAA Guid., Nav., Control Conf. Exhibit*, Providence, RI, 2004, CD-ROM.

[13] S. J. Rasmussen, T. Shima, J. W. Mitchell, A. Sparks, and P. R. Chandler, "State-space search for improved autonomous UAVs assignment algorithm," in *Proc. IEEE Conf. Decision Control*, Paradise Island, Bahamas, 2004, pp. 2911–2916.

[14] T. Shima, S. J. Rasmussen, A. G. Sparks, and K. M. Passino, "Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms," *Comput. Oper. Res.*, vol. 33, no. 11, pp. 3252–3269, 2005.

[15] R. Bellman, "Dynamic programming treatment of the traveling salesman problem," *J. ACM*, vol. 9, no. 1, pp. 61–63, Jan. 1962.

[16] D. K. Smith, *Dynamic Programming: A Practical Introduction*, ser. Mathematics and its Applications. London, U.K.: Ellis Horwood, 1991.

[17] D. Aldous and F. Diaconis, "Shuffling cards and stopping times," *Amer. Math. Month.*, vol. 93, pp. 333–348, 1986.

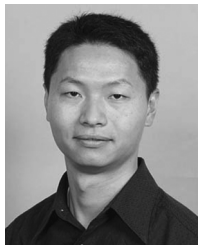
[18] D. Bayer and P. Diaconis, "Trailing the dovetail shuffle to its lair," *Ann. Appl. Probab.*, vol. 2, no. 2, pp. 294–313, 1992.

[19] L. N. Trefethen and L. M. Trefethen, "How many shuffles to randomize a deck of cards," in *Proc. Roy. Soc. London A*, 2000, vol. 456, pp. 2561–2568.

[20] D. E. Knuth, *The Art of Computer Programming*, ser. Computer Science and Information Processing. Reading, MA: Addison-Wesley, 1981, vol. 2.

[21] M. Mesbahi and F. Y. Hadaegh, "Formation flying control of multiple spacecraft via graphs, matrix, inequalities, and switching," *J. Guid. Control Dyn.*, vol. 24, no. 2, pp. 369–377, 2001.

[22] J. A. Fax, "Optimal and cooperative control of vehicle formations," Ph.D. dissertation, Calif. Inst. Technol., Pasadena, 2002.
[23] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1465–1476, Sep. 2004.
[24] R. Olfati-Saber and R. M. Murray, "Distributed cooperative control of multiple vehicle formations using structural potential functions," in *Proc. IFAC World Congr.*, Barcelona, Spain, Jul. 2002, CD-ROM.
[25] R. Beard, W. Stirling, and R. Frost, "A hierarchical coordination scheme for satellite formation initialization," in *Proc. AIAA Guid., Nav., Control Conf. Exhibit*, Boston, MA, 1998, pp. 677–685.
[26] W. Ren and R. W. Beard, "Consensus seeking in multi-agent systems using dynamically changing interaction topologies," *IEEE Trans. Autom. Control*, vol. 50, no. 5, pp. 655–661, May 2005.
[27] D. Swaroop and J. K. Hedrick, "String stability of interconnected systems," *IEEE Trans. Autom. Control*, vol. 41, no. 3, pp. 349–357, Mar. 1996.
[28] Z. Jin and R. M. Murray, "Double-graph control strategy of multi-vehicle formations," in *Proc. 43rd IEEE Conf. Decision Control*, Dec. 2004, vol. 2, pp. 1988–1994.
[29] C. Belta and V. Kumar, "Abstraction and control for groups of robots," *IEEE Trans. Robot.*, vol. 20, no. 5, pp. 865–875, Oct. 2004.
[30] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 813–825, Oct. 2002.
[31] O. Takahashi and R. J. Schilling, "Motion planning in a plane using generalized Voronoi diagrams," *IEEE Trans. Robot. Autom.*, vol. 5, no. 2, pp. 143–150, Apr. 1989.
[32] R. Zhu, D. Sun, and Z. Zhou, "Cooperation strategy of unmanned air vehicles for multitarget interception," *AIAA J. Guid. Control, Dyn.*, vol. 28, no. 5, pp. 1068–1072, Sep.–Oct. 2005.
[33] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA J. Guid. Control Dyn.*, vol. 25, no. 1, pp. 116–129, Jan.–Feb. 2002.
[34] N. Faiz, S. K. Agrawal, and R. M. Murray, "Trajectory planning of differentially flat systems with dynamics and inequalities," *J. Guid. Control Dyn.*, vol. 24, no. 2, pp. 219–227, 2001.
[35] T. Inanc, K. Misovec, and R. M. Murray, "Nonlinear trajectory generation for unmanned air vehicles with multiple radars," in *Proc. 43rd IEEE Conf. Decision Control*, pp. 3817–3822.
[36] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative-timing missions," *AIAA J. Guid. Control Dyn.*, vol. 28, no. 1, pp. 150–161, Jan.–Feb. 2005.
[37] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.
[38] J. P. Hespanha and A. S. Morse, "Switching between stabilizing controllers," *Automatica*, vol. 38, pp. 1905–1917, 2002.
[39] C. Godsil and G. Royle, *Algebraic Graphic Theory*, ser. Graduate Texts in Mathematics. New York: Springer, 2001, vol. 207.



Zhipu Jin received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1998 and 2001, respectively. He received another M.S. degree in electrical engineering in 2002 from the California Institute of Technology (Caltech), Pasadena, where he is currently working toward the Ph.D. degree.

His research interests include packet-based network control systems, cooperation/coordination between multiple vehicles with communication and computation constraints, and estimation over sensor networks.

Tal Shima received the Ph.D. degree in 2001 from the Technion—Israel Institute of Technology, Haifa, Israel.

Between 2000 and 2004, he was with RAFAEL Ltd. working in the Air-to-Air and Ground-to-Air Directorate. During 2004 and 2005, he was performing research on UAV cooperative decision and control at the U.S. Air Force Research Labs, Wright-Patterson AFB, OH. Since 2006, he has been with the Department of Aerospace Engineering of the Technion. His research interests include UAV cooperative decision and control, missile guidance and control, differential games, sensor networks, and genetic algorithms for optimization.

In 2004–2005, Dr. Shima was awarded a National Research Council scholarship for the work done at Wright-Patterson AFB, OH.

Corey J. Schumacher received the B.S. degree in aerospace engineering from the University of Michigan, Ann Arbor, in 1991, and the Ph.D. degree in 1997, also in aerospace engineering, with an emphasis on flight dynamics and control.

He then joined the U.S. Air Force Research Laboratory (AFRL), Munitions Directorate. He has also worked with the AFRL Air Vehicles Directorate, Wright-Patterson AFB, Dayton, OH. His research interests at AFRL include aircraft and missile autopilot design, multivariable robust control, adaptive control, nonlinear control, formation flight, and cooperative control of multiagent dynamical systems, with particular emphasis on unmanned aerial vehicles.

Dr. Schumacher is a member of the IEEE Committee on Aerospace Control, and Subject Editor for Robust Control Design for the *International Journal of Robust and Nonlinear Control*.