AIAA AVIATION Forum
5-9 June 2017, Denver, Colorado
17th AIAA Aviation Technology, Integration, and Operations Conference

Check for updates

Applying Graph Theory to Problems in Air Traffic Management

Amir H. Farrahi[1]
Universities Space Research Association, Moffett Field, CA, 94035

Alan T. Goldberg[2]
Kestrel Institute, Palo Alto CA, 94304

Leonard N. Bagasol[3]
Universities Space Research Association, Moffett Field, CA, 94035

and

Jaewoo Jung[4]
NASA Ames Research Center, Moffett Field, CA 94035

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

Graph theory is used to investigate three different problems arising in air traffic management. First, using a polynomial reduction from a graph partitioning problem, it is shown that both the airspace sectorization problem and its incremental counterpart, the sector combination problem are $\mathcal{NP}$-hard, in general, under several simple workload models. Second, using a polynomial time reduction from maximum independent set in graphs, it is shown that for any fixed ε, the problem of finding a solution to the minimum delay scheduling problem in traffic flow management that is guaranteed to be within $n^{1-\varepsilon}$ of the optimal, where n is the number of aircraft in the problem instance, is $\mathcal{NP}$-hard. Finally, a problem arising in precision arrival scheduling is formulated and solved using graph reachability. These results demonstrate that graph theory provides a powerful framework for modeling, reasoning about, and devising algorithmic solutions to diverse problems arising in air traffic management.

Nomenclature

$\varnothing$	=	The empty set
φ_i	=	A division of the airspace in the constructed ASP_D instance for a given PLANAR-P3(6) instance
A	=	The airspace, which is typically a closed subset of 2- or 3-dimensional Euclidean space
A_A	=	Arrival airport in the constructed SMDS problem instance
A_D	=	Departure airport in the constructed SMDS problem instance
ALG	=	A polynomial time algorithm for SMDS problem
ALG(I)	=	The delay associated with the solution resulting from algorithm ALG on instance I of SMDS
ASP	=	The Airspace Sectorization Problem (optimization version) in DAC
ASP_D	=	The decision version of ASP
ASP_MS	=	Min-sector version of the ASP optimization problem
ASP_MW	=	Min-workload version of the ASP optimization problem
ATC	=	Air Traffic Control
ATM	=	Air Traffic Management
$c_{down}(t)$	=	The capacity of sector $down_j$ in the constructed SMDS instance

[1] Senior Research Scientist, Universities Space Research Association, NASA Ames Research Center, Moffett Field, CA 94035, AIAA Member.
[2] Research Staff, Kestrel Institute, 3260 Hillview Avenue, Palo Alto, CA 94304.
[3] Senior Software Engineer, Universities Space Research Association, NASA Ames Research Center, Moffett Field, CA 94035.
[4] Research Engineer, NASA Ames Research Center, Moffett Field, CA 94035, Senior AIAA Member.

Copyright © 2017 by the American Institute of Aeronautics and Astronautics, Inc. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes.

$c_i(t)$	=	The capacity of sector s_i at time t in the SMDS instance
c_s	=	The upper bound on number of sectors in a feasible solution for an ASP_D instance
$c_{up_j}(t)$	=	The capacity of sector up_j in the constructed SMDS instance
c_w	=	The upper bound on the sector workload in a feasible solution for an ASP_D instance
d_{ij}	=	The delay assigned to the j^{th} sector along flight f_i's route in the constructed SMDS instance
$\boldsymbol{d}_i$	=	The sequence of delays assigned to flight f_i in different sectors along its flight path in the constructed SMDS instance
D	=	The set of delay sequences imposed on the set of flights in the constructed SMDS instance
DAC	=	Dynamic Airspace Configuration
$down_i$	=	The lower sector in a sector pair in the constructed instance of SMDS
E	=	The edge set of a graph
$\|E\|$	=	The size of E (number of edges in E)
F	=	Set of flights or flight trajectories in ASP, SCP, MDS, SMDS and MSDA problems
f_i	=	The i^{th} flight in ASP, SCP, MDS or MSDA problems
G	=	A graph
MDS	=	The Minimum Delay Scheduling problem in traffic flow management
MILP	=	Mixed Integer Linear Programming
$m_j(F, s_i)$	=	Workload model m_j, accounting for the workload in sector s_i due to the set of flights in F
MSDA	=	Maximum Set of Dependent Aircraft problem in air traffic management
n	=	Problem size, e.g., the number of flights in the problem instance
OPT(I)	=	Optimal (smallest) delay of a feasible solution to instance I of SMDS
P	=	Sector aggregation (partition) function in SCP
p_i	=	The i^{th} partition in P
PNANAR-3DM	=	Three dimensional matching problem in planar graphs
PLANAR-P3	=	The problem of partitioning a planar graph into triangles
PLANAR-P3(6)	=	A restricted version of PLANAR-P3 in which each vertex is incident on 6 or fewer edges
$\mathbf{R}^n$	=	n-dimensional Euclidean space
S	=	The set of sectors in the ASP, SCP, MDS and SMDS
SCP	=	Sector Combining Problem in DAC
s_i	=	The i^{th} sector in ASP, SCP and SMDS problems
SMDS	=	The simplified version of the MDS problem
$spread(D)$	=	The difference between the *max* and *min* total delay imposed on different flights in a feasible solution to the constructed SMDS instance
t	=	Time dimension in ASP, SCP, MDS and SMDS
T	=	The scheduling function in MSDA
TFM	=	Traffic Flow Management
up_i	=	The upper sector in a sector pair in the constructed instance of SMDS
V	=	Vertex set of a graph
$\|V\|$	=	The size of V (number of vertices in V)
x, y, z	=	Spatial dimensions

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

I. Introduction

Modern air transportation systems involve some of the most complex and challenging multi-disciplinary technical challenges of the age. In recognition of these challenges, a multi-faceted research and development effort is underway to realize the Next-Generation Air Transportation System (NextGen) [1], intended to accommodate the projected growth in the demand for air transportation and to achieve increased efficiency in the effective use of the airspace and Air Traffic Control (ATC) resources. As part of its research and development efforts to help address NextGen, the National Aeronautics and Space Administration (NASA) is involved in carrying out foundational research and technology development to extend the state of the art in Air Traffic Management (ATM) using aeronautics engineering, computer science, software engineering, applied physics, mathematics, human factors, and automation design[2].

Given a nontrivial computational problem that we would like to solve, it is often useful to understand its various characteristics. One of the fundamental characteristics of a problem is its computational complexity. Understanding a problem's computational complexity is an important step in the quest for devising efficient and effective algorithmic solutions for the problem. Computational complexity is the study of the inherent difficulty of solving problems, and

provides techniques for relating new problems to the pool of problems that have been characterized before. Graph theory provides a versatile and powerful mathematical abstraction for expressing, maintaining, testing, qualifying, quantifying, and reasoning about various characteristics, relationships, hypotheses, and queries among interacting objects or components comprising a complex system. The versatility of graph theory has enabled its successful application to a wide variety of technical problems in a multitude of fields in science and technology, from physics to chemistry to biology, and from manufacturing to transportation to marketing, and beyond.

In this work, we advance the idea that Air Traffic Management (ATM) is ripe with a multitude of complex technical problems that can be formulated as graph problems. Thus, we argue that the ATM research community can benefit greatly from the wealth of knowledge and techniques developed in (a) graph theory to solve various graph theoretic problems, and (b) the theory of computational complexity that is devoted to studying and classifying computational problems according to their inherent difficulty.

To illustrate this, we use graph theory to investigate three different problems arising in ATM. First, it is shown that the Airspace Sectorization Problem and the related Sector Combination Problem to optimize the airspace and the ATC resources are $\mathcal{NP}$-hard, in general under several simple workload models. This is done by establishing a polynomial reduction from an NP-hard graph partitioning problem. Second, it is shown that for any fixed $\varepsilon > 0$, the problem of finding a feasible scheduling solution to the traffic flow management problem with n aircraft in the problem instance, that is guaranteed to be within $n^{1-\varepsilon}$ of the optimal solution is $\mathcal{NP}$-hard. This is done by establishing a polynomial time reduction from the maximum independent set problem in graphs that is also known not to be approximable. Finally, a problem arising in precision arrival scheduling is formulated and solved using graph reachability. While the results of our investigation in each of these problems is useful on its own merits, the unifying theme and the underlying purpose of presenting them in this work has been *i*) to invite the ATM research and development community to familiarize itself better with graph theoretic ideas and the theory of computational complexity, and *ii*) to provide examples illustrating how this can lead to new insights and/or efficient algorithmic solutions for problems arising in ATM.

Basic familiarity with key concepts and notation in graph theory[3, 4] and the theory of computational complexity[5, 6] is assumed. A refresher on these concepts, sufficient for understanding the material in this paper, is provided in the appendix. The reader is encouraged to consult references for more detailed and thorough discussion on these subjects. The rest of this paper is organized as follows. Sections II, III, and IV each focuses on one of the three ATM problems that were investigated and present the corresponding results. The paper continues in section V with additional remarks and discussion on some of the underlying purposes and unifying themes of this work, and some suggestions on why, when, and how to make better use of graph theory and the theory of computational complexity in tackling ATM problems. The paper concludes with section VI which provides a summary of the technical contributions and lists some related open problems.

II. Airspace Sectorization Problem

Dynamic Airspace Configuration (DAC)[7,8] is a NextGen technical area primarily concerned with optimal strategic organization and dynamic tactical reallocation of the airspace and ATC resources. The Airspace Sectorization Problem (ASP) [9-17] is the problem of partitioning the airspace into sectors in such a way as to minimize or balance the *controller workload* in each sector by allowing sufficient room in the airspace for the controllers to handle and accommodate planned activities and unplanned disturbances (such as flight schedule changes, bad weather conditions, etc.) in a safe and timely fashion. Researchers and practitioners have introduced metrics such as sector capacity[15], monitor alert parameter[18–19], dynamic density[20–22], simplified dynamic density[16], and have used these to monitor, measure and analyze the workload and/or to compare the result of different airspace sectorization algorithms. Basu, Mitchell and Sabhnany[15] showed that under a specific workload model, ASP is $\mathcal{NP}$-hard if the sector boundaries are constrained to axis-aligned rectangles. We extend their result to arbitrary shape sectors and under additional workload models, even if the flight trajectories or their projection onto the two-dimensional plane is a planar graph with maximum vertex degree six.

A. Problem Formulation

In general, ASP can be formulated in the n-dimensional Euclidean space $\mathbf{R}^n$. In practice, we are typically working either in $\mathbf{R}^3$ (x, y, t) or $\mathbf{R}^4$ (x, y, z, t), where x, y, z represent the spatial dimensions and t represents time. In the rest of the paper, we will use $\mathbf{R}^3$ for simplicity and will generalize the results to higher dimensions as appropriate. The problem instance includes the airspace A, which is a closed sub-space of $\mathbf{R}^2$ (x, y), and a set F of flight trajectories inside A, where each trajectory is a curve in $\mathbf{R}^3$ (x, y, t) signifying the prescribed path traveled by an aircraft in the space-time. Generally, two flavors of the problems are of interest: *min-workload* (ASP_{MW}) and *min-sector* (ASP_{MS}).

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

In both the min-workload and the min-sector formulations, the exact definition of the workload function depends on the *workload model* or *workload metric* being used. Some simple examples of workload model include:

$$m_1(F, s_i) = \text{Total \# of flights that intersect } s_i \quad (1)$$
$$m_2(F, s_i) = \text{\# of flights simultaneously present in } s_i \quad (2)$$
$$m_3(F, s_i) = \text{\# of flights present in any time interval of fixed duration } d \text{ in } s_i \quad (3)$$
$$m_4(F, s_i) = \text{Total \# of intersections between the boundary of sector } s_i \text{ and the flight trajectories in } F \quad (4)$$
$$m_5(F, s_i) = a.m_1(F, s_i) + b.m_4(F, s_i) \quad \text{(linear combinations of workload models } m_1 \text{ and } m_4) \quad (5)$$

More complicated workload models can be formed. As an example, the workload model $m_5(F, s_i)$ above, is defined as a linear combination of $m_1(F, s_i)$ and $m_4(F, s_i)$ to account for both the workload associated with safe separation of flights inside the sector ($m_1(F, s_i)$) as well as the coordination workload associated with flights crossing the sector boundary ($m_4(F, s_i)$). Other examples of more sophisticated characteristics used to define workload can be found in the literature for dynamic density metric[20–22]. We can combine the min-workload and min-sector formulation into one decision problem, denoted $ASP_D(m)$, with an instance (A, F, c_s, c_w), where m represents the workload model. The problem seeks to determine whether there exists a sectorization of the airspace A into c_s or fewer sectors such that the workload for each sector (under the workload model m) is no more than c_w. An airspace sectorization is called *feasible* (or c_w-*feasible*) if it meets the maximum workload constraint. Therefore, in the $ASP_D(m)$ problem, we seek to determine if there exists a c_w-feasible sectorization of the airspace into c_s or fewer sectors. This problem relaxes into its min-sector or min-workload versions if the maximum workload or the maximum sector count constraining factor is missing (e.g., set to infinity), respectively, while the objective is to minimize the remaining factor.

B. Previously Known Results on the Computational Complexity of ASP

Basu, Mitchell and Sabhnany[15], used sector capacity as workload metric, which they defined as the number of aircraft present simultaneously in a sector (equivalent to $m_2(F, s_i)$ defined in Eq. (1).) They showed the following:

[15]*Theorem* 1. *The 2D version of* $ASP_{MS}(m_2)$ *can be solved exactly in worst-case (deterministic) time* $O(kn \log^2 n)$*, where* k *is the optimal number of sectors in the problem instance and* n *is the number of flight trajectories in the problem instance. The problem can be solved in* $O(kn \log n)$ *expected time using a randomized algorithm.*

[15]*Theorem* 2. *The 2D version of* $ASP_{MW}(m_2)$ *can be solved exactly in worst-case (deterministic) time* $O(kn \log^3 n)$*. The problem can be solved in* $O(kn \log^2 n)$ *expected time using a randomized algorithm.*

[15]*Theorem* 3. *The 3D version of* $ASP_{MS}(m_2)$ *and* $ASP_{MW}(m_2)$ *are* $\mathcal{NP}$*-hard if we restrict the solution space to those sectorizations in which each sector is an axis-aligned rectangle.*

In general, restricting the solution space of a problem can have different impacts. It can turn an easy problem into one that is intractable, or it can turn an intractable problem into one that can be solved optimally in deterministic polynomial time, or it might not change the computational complexity class of the problem. This might be somewhat counter-intuitive. It may seem that a restriction resulting in a drastic reduction on the size of the solution space would make the problem easier to solve. This s not necessarily the case, because the inherent difficulty of a problem is generally determined by the interaction of several factors, including the size, form and the structure of the solution space, as well as whether and how the problem characteristics may be used to navigate the solution space to home-in on an optimal solution. Since it is not clear how the axis-aligned restriction on the shape of the sectors would impact the computational complexity of the airspace sectorization problem, one cannot readily assume that the sectorization problem under workload model m_2 remains $\mathcal{NP}$-hard in three or more dimensions, in general, if the restricted version is $\mathcal{NP}$-hard.

C. Our Results on the Computational Complexity of ASP

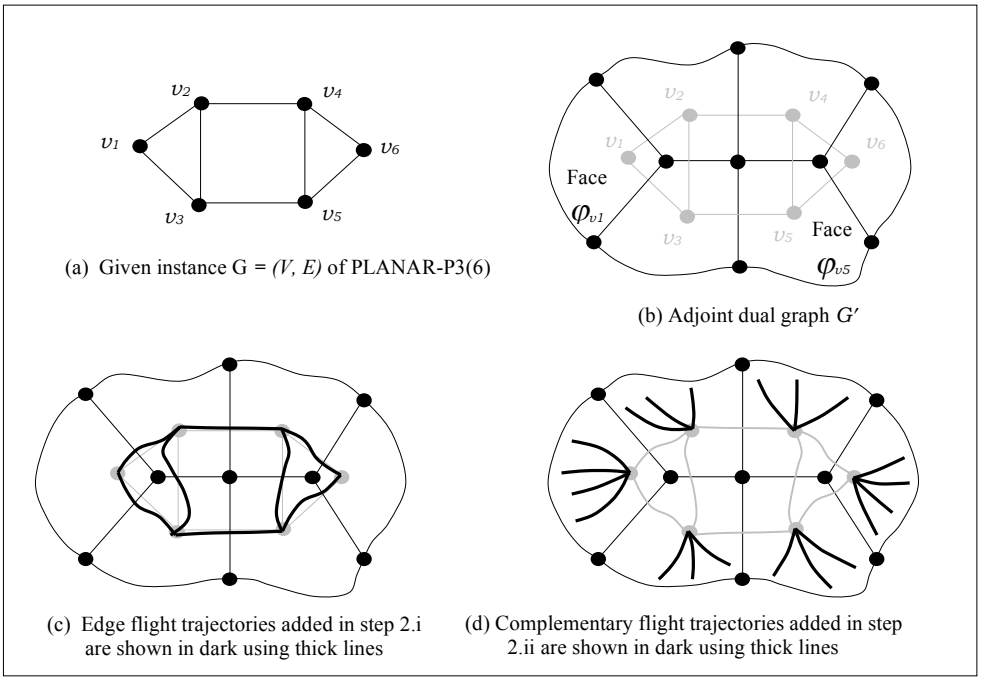
In this section, we investigate the computational complexity of ASP_D, in general, for the workload models defined in Eqs. (1) through (5), when the sectors are allowed to take arbitrary shapes and orientations and thus are not required to be axis-aligned rectangles. We show that under most of these workload models, the problem is $\mathcal{NP}$-complete.

To prove ASP_D is $\mathcal{NP}$-complete under a given workload model, we establish a polynomial time reduction from PLANAR-P3(6), which is a restricted form of *Planar 3-Dimensional Matching*[23] (PLANAR-3DM). Under this restriction, each vertex in the planar graph constituting the problem instance is connected to no more than six other

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

vertices. Establishing this reduction essentially proves that ASP_D has immense expressive power. In essence, this proves that ASP_D can express an arbitrary problem in $\mathcal{NP}$, even the hardest ones among them. This shows solving ASP_D in polynomial time is tantamount to solving an arbitrary problem in $\mathcal{NP}$ in polynomial time, thus proving that under the given workload model, ASP_D is $\mathcal{NP}$-complete, and hence the optimization versions of the problem (ASP_{MS} and ASP_{MW}) are $\mathcal{NP}$-hard.

In PLANAR-P3(6), given a planar graph $G = (V, E)$ with maximum edge degree six, we seek to determine if there exists a proper partition of V into subsets $V_1, V_2, \ldots, V_n$ such that $\forall i \in \{1, 2, \ldots, n\}$, G_{Vi}, the subgraph induced by V_i, forms a *triangle*. That is, G_{Vi} is a graph with three vertices, where each vertex is connected to the other two. Dyer and Frieze[23] showed among other things that PLANAR-P3(6) is $\mathcal{NP}$-complete. It is clear that an instance $G=(V, E)$ of PLANAR-P3(6) is trivially solvable with a NO answer, if the number of vertices in V is not a multiple of 3. So, the hard instances of PLANAR-P3(6) are among those wherein the number of vertices is a multiple of 3, that is $|V|=3n$ for some positive integer n.



(a) Given instance G = *(V, E)* of PLANAR-P3(6)

(b) Adjoint dual graph G'

(c) Edge flight trajectories added in step 2.i are shown in dark using thick lines

(d) Complementary flight trajectories added in step 2.ii are shown in dark using thick lines

Figure 1. Construction of the $ASP_D(m_1)$ instance from given PLANAR-P3(6) instance.

Let $G = (V, E)$ be an arbitrary instance of PLANAR-P3(6) wherein $n = |V| / 3$ is an integer. Construct an instance $I = (A, F, c_s, c_w)$ of $ASP_D(m_1)$ as follows (see Fig. 1):

1) A: Since G is planar, it can be embedded in the plane such that there are no edge intersections (except at the vertices incident on the edges). Consider a planar embedding of $G = (V, E)$ as shown in Fig. 1(a). Create another planar graph G', which we will call the *adjoint graph* of G. The new planar graph G' is obtained from G by expanding each vertex $v \in V$ into a face φ_v in G' such that two faces φ_u, φ_v share an edge in G' if and only if their corresponding vertices in V share and edge, that is, $(u, v) \in E$. An example of going from graph G to the adjoint graph G' is shown in Fig. 1. The area captured by the faces of the adjoint graph G' forms the airspace A in our $ASP_D(m_1)$ instance.

2) F: To begin with, let $F=\varnothing$. Then proceed as follows:
   i. For each edge $e=(u, v)$ in E, create a flight trajectory f_e that is completely contained inside the faces φ_u and φ_v, originates at u, and terminates at v. Let $F = F \cup f_e$. The flight trajectories created in this step are referred to as the *edge-flight trajectories*. Fig. 1(c) demonstrates this step, given the PLANAR-P3(6) instance of Fig. 1(a).

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

ii. Note that, by definition, each vertex $v \in V$ in the given PLANAR-P3(6) instance is connected to no more than 6 edges. This means that the number of flight trajectories created in step (i) above that intersect (originate or terminate at) a vertex $v \in V$ is at most 6. However, some vertices might connect to fewer than 6 edges. We would like to ensure additional flight trajectories are created per vertex (as needed), so that we have exactly 6 flight trajectories (including those generated in step i) originating or terminating at each of the vertices $v \in V$. Therefore, for each vertex $v \in V$ that is connected to j edges with $j < 6$, we will create an additional $(6 - j)$ flight trajectories. These flights originate at vertex v and terminate at some location inside the corresponding face φ_v. Let's call these the *complementary flight trajectories* for vertex v, and denote them as $f_{c1}(v), \ldots, f_{c6-j}(v)$. Extend F by adding these flight trajectories to it. That is, let: $F = F \cup \{ f_{c1}(v). \ldots, f_{c6-j}(v)\}$. Fig. 1(d) demonstrates the complementary flight trajectories created in this step for the PLANAR-P3(6) instance shown in Fig. 1(a).

3) Let $c_s = n$ and $c_w = 15$. We have thus completed constructing the $\mathrm{ASP_D}(m_1)$ instance $I = (A, F, c_s, c_w)$.

We will show that the answer to the given instance $G=(V, E)$ of PLANAR-P3(6) instance is YES if and only if the answer to the constructed instance $I = (A, F, c_s, c_w)$ of $\mathrm{ASP_D}(m_1)$ is YES. In other words:

Lemma 4. *In the PLANAR-P3(6) instance* $\mathrm{G}=(\mathrm{V}, \mathrm{E})$*, the vertex set* V *can be partitioned into* $\mathrm{V}_1, \mathrm{V}_2, \ldots, \mathrm{V}_n$ *such that* $\mathrm{G_{Vi}}$*, the subgraph induced by* $\mathrm{V_i}$*, forms a triangle if and only if in the constructed instance* $\mathrm{I} = (\mathrm{A}, \mathrm{F}, \mathrm{c_s}, \mathrm{c_w})$ *of* $ASP_D(\mathrm{m}_1)$*, the airspace* A *can be partitioned into* $\mathrm{c_s}=\mathrm{n}$ *sectors each with workload no more than* $\mathrm{c_w} = 15$.

Proof: **(IF)**: Assume there exists a partition $V_1, V_2, \ldots, V_n$ of the vertex set V such that $i \in \{1, 2, \ldots, n\}$, $|V_i| = 3$, and G_{Vi}, the subgraph induced by vertices in V_i, forms a triangle. Let $S = \{s_1, s_2, ..., s_n\}$, with $s_i = \varphi_{i1} \cup \varphi_{i2} \cup \varphi_{i3}$, where φ_{ij} is the face in the adjoint graph G' containing vertex v_{ij}. Note that at this point, the union of the sectors in S covers all of A. Note that $s_i = \varphi_{i1} \cup \varphi_{i2} \cup \varphi_{i3}$ is a contiguous sector in the airspace A since G and G' are both planar graphs, and G_{Vi} forms a triangle. Furthermore, for $j \in \{1, 2, 3\}$, each of the faces φ_{ij} intersects exactly 6 flight trajectories. These are the flight trajectories originating or terminating at vertex v_{ij}. However, since v_{i1}, v_{i2}, v_{i3} form a triangle, there is one flight that is shared between each pair of these vertices. In other words, each of the sectors s_i defined above intersects no more than $(6-1)\times 3 = 15$ flight trajectories. Thus, $S = \{s_1, s_2, ..., s_n\}$ is a feasible sectorization of the airspace with exactly $c_s = n$ sectors, where each sector has workload $c_w = 15$ under workload model m_1. Thus, the answer to the constructed instance $I = (A, F, c_s, c_w)$ of $\mathrm{ASP_D}(m_1)$ is YES.

(**ONLY IF**): Conversely, assume there exists a contiguous partitioning $S=\{s_1, s_2, ...\}$ of the airspace A into n or fewer sectors such that each sector intersects no more than 15 flight trajectories in F. It is easy to see that the only way this can happen is for each sector to contain exactly three vertices that are pairwise connected in G, thus collectively contributing a total of exactly 15 to the workload of the sector. Take any sector $s_i \in S$. Let v_{i1}, v_{i2}, v_{i3} denote the three vertices in s_i, and let $V_i = \{ v_{i1}, v_{i2}, v_{i3}\}$. It is clear that G_{Vi} is a triangle, meaning each pair of the three vertices in V_i is connected by an edge in E. In other words, the answer to the PLANAR-P3(6) on instance $G=(V,E)$ is YES. ■

Lemma 4 proves $\mathrm{ASP_D}(m_1)$ is $\mathcal{NP}$-complete, in general in three dimensions. To extend the result to higher dimensions, we note that the 3-dimensional (3D) version of the airspace sectorization problem is a restricted case of the 4D and above, in the sense that each 3D instance can also be considered a 4D, 5D, … instance with the values of the parameters along all entities at dimensions four or above are set to zero. Therefore, the $\mathrm{ASP_D}(m_1)$ problem is $\mathcal{NP}$-complete in any dimension higher than three as well. In other words:

Corollary 5. $ASP_D(\mathrm{m}_1)$ *is* $\mathcal{NP}$*-complete, in general in three or more dimensions.*

Note that at no time in the proof of Lemma 4 did we use the flight time spans or durations. It is straightforward to see that by setting the flight time spans appropriately, we can extend the proof of Lemma 4 to workload models m_2, m_3, defined in Eqs. (2), (3). Also, note that the construction presented in the proof can be done in such a way that the flight trajectories in the problem instance do not intersect one another at any point other than at their origin and termination points. This is because the vertices in the adjoint graph could be considered area objects as opposed to point objects, without posing any problems in the proof; thus, we have the following:

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

Corollary 6. *The $ASP_D(\mathrm{m})$ remains $\mathcal{NP}$-complete under the workload models* m_1, m_2, m_3, *defined in Eqs. (1), (2), (3), even when the flight trajectories (or their projection onto the two-dimensional plane) form a plane graph of maximum vertex degree six.*

Earlier, we noted that more complicated workload models can be formed by combining simpler workload models, and provided as an example m_5, defined in Eq. (5), as a linear combination of m_1 and m_4, where constant factors a and b were used for setting the relative contribution of m_1 and m_4 (corresponding to the workload internal to the sector and the workload due to coordination effort of aircraft transitioning from one sector to another) to the overall workload m_5. Note that if b is set to zero, then the workload model m_5 would be effectively equivalent to m_1. This, together with Corollary 4 shows the following:

Theorem 7. *The problems $ASP_D(\mathrm{m}_1)$, $ASP_D(\mathrm{m}_2)$, $ASP_D(\mathrm{m}_3)$ and $ASP_D(\mathrm{m}_5)$ (wherein* m_1, m_2, m_3 *and* m_5 *are the workload models defined in Eqs. (1), (2), (3), and (5) respectively) in three or more dimensions are $\mathcal{NP}$-complete even when the flight trajectories (or their projection onto the two-dimensional plane) form a planar graph of maximum vertex degree six.*

Note that Theorem 7 does not say anything about the difficulty of workload model m_4, defined in Eq. 4. Under this model, the workload for a sector is defined as the total number of intersections between flight trajectories and the boundary of the sector. This quantity can be thought of as the coordination workload between a sector and the adjacent sectors when aircraft are exiting one sector and entering an adjacent one. However, it is easy to see that a trivial solution to the airspace sectorization problem under this workload model is YES wherein the sectorization consists of a single sector encompassing the whole airspace with zero workload for the sector, unless of course $c_s<1$ or $c_w<0$, in which case there exists no sectorization meeting the constraints, and the answer would be NO. In other words:

Observation 8. *$ASP_D(\mathrm{m}_4)$ is in $\mathcal{P}$ and is trivially solvable.*

Theorem 7 shows that unless $\mathcal{P}=\mathcal{NP}$, finding optimal solutions to ASP under workload models m_1, m_2, m_3, and m_5 is prohibitively difficult, even if the flight trajectories or their projections onto the 2-dimensional plane seem reasonably simple (form a planar graph of maximum vertex degree six).

D. Sector Combination Problem

ASP can be thought of as a clean-sheet approach to the DAC, in the sense that it disposes of the current sectorization in favor of another one that is being computed from scratch. In addition to the fact that finding an optimal sectorization seems prohibitively expensive, there is the problem of safe transition from the current sectorization to another one that may be computed when solving ASP, which could be drastically different from the current sectorization result. Such an abrupt change in the sectorization solution may have unexpected consequences and pose safety risks. In the absence of a robust strategy that guarantees safe and smooth transition between the current sectorization and one resulting from solving ASP, a more incremental approach is desirable. One such strategy is the *Sector Combining Problem* (SCP) proposed and studied by Bloem and Kopardekar[23], Bloem, Gupta and Kopardekar[24], and Drew[26]. They proposed heuristic and greedy algorithms for the problem, without providing guarantees on the quality of the results obtained. In the SCP, the current sectorization undergoes local adjustments consisting of combining neighboring sectors to find another sectorization that makes better use of the ATC resources. The goal is to do this intelligently so as to minimize the number of remaining sectors, while ensuring that the workload in the resulting sector does not violate the maximum workload constraint. These earlier research, however, left open the discussion about the computational complexity of the problem, a question that we will address in this section.

Again, both optimization and decision versions of the problem can be formulated. Here, we formulate the decision version to study its computational complexity. Formally, an instance $\mathrm{SCP_D}(m)$, where m is the workload model, is a tuple (A, S, F, c_s, c_w), where A is the airspace partitioned into n sectors $s_1, s_2, \ldots, s_n$ forming $S = \{s_1, s_2, \ldots, s_n\}$, F is the set of flight trajectories, and c_s and c_w are constants. A *sector aggregation* is a partition $P = \{p_1, p_2, \ldots, p_k\}$ of S such that if sectors s_i, s_j are in partition p_k then there is a path (or a set of neighboring sectors) from s_i to s_j within p_k. The workload model m associates a workload $m(F, p_i)$ with partition p_i. We seek to determine whether there exists an aggregation function $P = \{p_1, p_2, \ldots, p_k\}$ with $k \le c_s$ such that for all i, we have $m(F, p_i) \le c_w$. We can show the following:

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

Theorem 9. *The problems* $SCP_D(m_1)$, $SCP_D(m_2)$, $SCP_D(m_3)$ *and* $SCP_D(m_5)$ *(wherein* m_1 , m_2, m_3 *and* m_5 *are the workload models defined in Eqs. (1), (2), (3), and (5) respectively) in three or more dimensions are* $\mathcal{NP}$*-complete even when the flight trajectories (or their projection onto the two-dimensional plane) form a planar graph of maximum vertex degree six.*

The proof is very similar to that presented for the $\mathcal{NP}$-completeness of ASP, and is based on reduction from PLANAR-P3(6), hence we simply provide the main idea of the proof. Consider a planar graph $G=(V, E)$, as the PLANAR-P3(6) instance given, and assume without loss of generality that $|V| = 3$. Consider an airspace A consisting of the unit square. Let R be an arbitrary partition of A that is *consistent* with G. That is, A is subdivided into $|V|$ regions $r_1, r_2, \ldots, r_n$ such that r_i is adjacent to r_j if and only if $(v_i, v_j) \in E$. The correspondence between graph G and airspace A here is just like what was discussed for ASP and illustrated in Fig. 1(a), 1(b). The rest of the proof parallels what was presented for ASP.

III. Minimum Delay Scheduling in Traffic Flow Management

Traffic Flow Management[27–33] (TFM) is concerned with choreographing the flow of air traffic across the *national airspace system* (NAS) based on demand and available capacity. It consists of a number of strategic programs and practices conducted by the Federal Aviation Administration (FAA) to ensure safety, while at the same time trying to minimize the costs associated with delays incurred. Fig. 2 provides a notional illustration of aircraft flying through the NAS, passing through different centers, each of which comprising of a number of sectors along the designated flight paths of the aircraft. The *Minimum Delay Scheduling* (MDS) problem in TFM involves introducing strategic delays on the ground or en-route in order to meet *i*) the airport arrival and departure rates and *ii*) the capacity constraints in the sectors across the airspace, at all times. The MDS problem studied here is modeled after what was presented by Betrsimas and Stock-Patterson[28]. They presented a Mixed Integer Linear Programming (MILP) formulation for the problem and showed that the problem is $\mathcal{NP}$-hard. Landry el al.[30] developed a system comprised of a distributed network of loosely coupled schedulers sharing capacity information, thus creating increased tolerance against uncertainties involved in estimating arrival times over long distances, while still allowing the construction of short-term schedules. Tandale et al.[31] developed a simplex-based Dantzig-Wolfe decomposition based on the MILP formulation of the problem, thus allowing a massively parallelizable solution strategy for the problem. Barnhart et al.[32] developed a fairness metric to balance equity and efficiency and developed an integer programming formulation targeting the minimization of this fairness metric. Zhang el at.[33] used an integer quadratic programming formulation that was first relaxed and solved as a quadratic programming problem by a distributed approach, followed by a heuristic forward-backward propagation phase to discretize the solution and obtain a solution to the original integer quadratic programming problem.

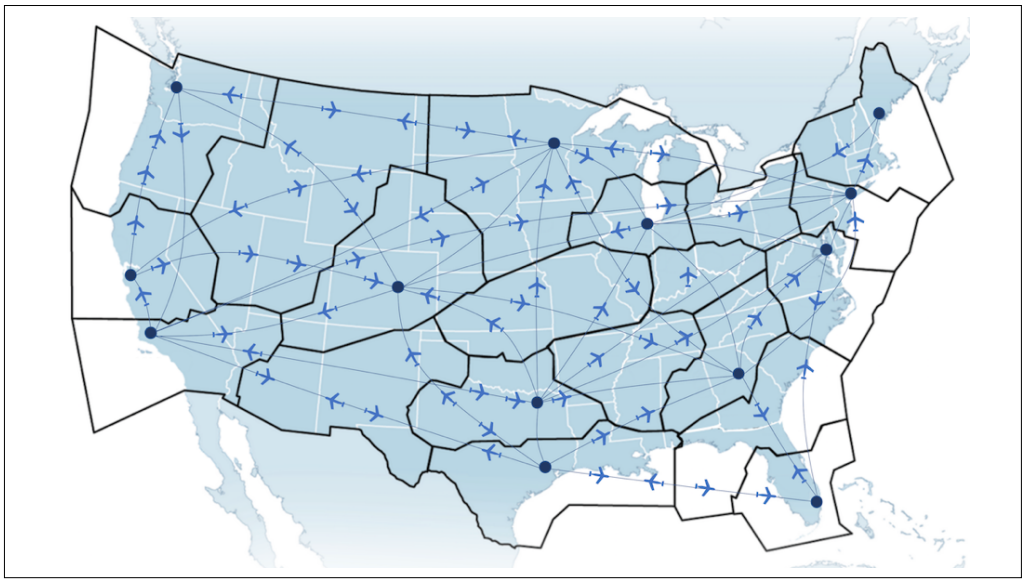


Figure 2. Notional illustration of aircraft flying along their designated paths across the NAS.

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

While the MDS problem is known to be $\mathcal{NP}$-hard, it is not clear whether on may be able to devise a worst-case polynomial time algorithms for the problem that while not guarantee finding an optimal solution, will find solutions that are guaranteed to be within reasonable approximation of the optimal. This is a problem that we address in this paper, showing that it is not possible to devise such an algorithm that guarantees a reasonable approximation of the optimal solution, unless $\mathcal{P}=\mathcal{NP}$.

A. Problem Formulation

Informally, the MDS problem in TFM consists of a set F of flights, where flight must fly a designated route through the airspace. S is the collection of sectors sub-dividing the airspace A, and each flight must spend a designated *minimum transit time* in each sector along its path. Thus, each flight departs its designated airport, traverses a sequence of sectors, spending a minimum *transit time* in each sector, and then arrives at its designated arrival airport. Each airport has a time-varying arrival- and departure-rate constraint. Each sector has a time-varying capacity that constrains the number of flights simultaneously present in the sector. At no time t may the number of flights in the sector exceed its capacity. Flights may be delayed at the departure airport or in a sector as a means of avoiding the violation of the airport arrival or departure rates or sector capacity constraints. The objective of the problem is to find the minimal total delay required so that each flight flies its designated route without violating any airport arrival or departure rate or any sector capacity constraint.

Our focus in this paper is on a simplified version of the MDS problem denoted as Simplified Minimum Delay Scheduling (SMDS). This simplified version is sufficient for establishing our computational complexity result, and yet our result generalizes immediately to the more general problem. In this simplified version, all flights depart from the same departure airport and all flights arrive at the same arrival airport. In addition, there are no arrival and departure rate constraints at the arrival and departure airport.

More formally, the SMDS problem consists of a departure airport A_D, an arrival airport A_A, and a finite set $S = \{s_1, s_2, \ldots, s_n\}$ of sectors. Time values are integral (integer values) and in the range $[0, T]$. For each sector s_i a capacity schedule $c_i : [0, T] \rightarrow \mathbf{N}$ is given. Each flight is ready to depart from A_D at time 0, and must arrive at A_A prior to T. $F = \{f_1, f_2, \ldots, f_n\}$ is the set of flights to be scheduled, and for each flight f_i, its schedule $\sigma_i = [(s_{i1}, t_1), (s_{i2}, t_2), \ldots, (s_{in}, t_n)]$ is a sequence of pairs, where the first element in each pair (s_{ij}) denotes a sector and the second element (t_j) denotes the transit time of the flight in the sector. Let $usage_i(t) = |\{f \in F : f \text{ is in sector } s_i \text{ at time } t\}$. Then, the capacity constraint can be expressed as $usage_i(t) \leq c_i(t)$. A solution to an SMDS instance is a set $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n\}$, wherein $\boldsymbol{d}_i = [d_{0i}, d_{1i}, \ldots, d_{ki}]$ denotes the delays assigned to flight f_i along its path, where d_{0i} is the delay at the departure airport and d_{ji} is the delay incurred in the j^{th} sector along the path of the flight. A solution is *feasible* if no sector capacity constraint is violated and each flight arrives at the arrival airport A_A no later than T. That is, for each flight the sum of the delays and transit times along its path should not exceed T. The total delay of a solution $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n\}$ is defined as $delay(D) = \sum_{\boldsymbol{d}_i \in D} delay(\boldsymbol{d}_i)$, where $delay(\boldsymbol{d}_i) = \sum_{j=\{1, 2, \ldots, ki\}} d_{ij}$ is the total delay assigned to flight f_i. including its delay at the departure and all the sectors along its path. The objective in the SMDS problem is to find the feasible scheduling solution that has the minimal $delay(D)$. An additional concern may be the notion of *equity* in treating different aircraft in the problem instance. Towards that end, we may wish to limit the maximum difference between total delay imposed on different aircraft. That is, to ensure $spread(D) \leq K$, where $spread(D) = max_{\boldsymbol{d}_i \in D}(delay(\boldsymbol{d}_i)) - min_{\boldsymbol{d}_i \in D}(delay(\boldsymbol{d}_i))$, and K is a given quantity.

B. Our Result on the Computational Complexity of MDS

Since MDS is known to be $\mathcal{NP}$-hard[28], developing an efficient (worst-case polynomial time) algorithm for MDS that finds the optimal solution is out of the question, unless $\mathcal{P}=\mathcal{NP}$. There are $\mathcal{NP}$-hard problems for which one can find reasonable approximations to the optimal solution in worst-case polynomial time. In this section, we prove that unless $\mathcal{P}=\mathcal{NP}$, there is no good approximation algorithm for MDS. We prove this result for the simplified version of the MDS, defined in the previous section (SMDS). The non-approximability result generalizes immediately to the more general MDS problem.

Let I be an SMDS instance and OPT(I) be the delay of an optimal feasible solution. Let ALG be any polynomial time algorithm for SMDS and ALG(I) the delay associated with the solution computed by ALG(I). One might hope to find an algorithm ALG such that ALG(I) $\leq k \times$ OPT(I) for some constant k. Such approximation algorithms exist for other $\mathcal{NP}$-hard problems such as the knapsack problem[34], Traveling Salesman problem with triangle inquality[35], and the Euclidean Traveling Salesman problem[36]. We prove, however, that such an algorithm does not exist for SMDS, even for arbitrarily large values of k, unless $\mathcal{P}=\mathcal{NP}$. More precisely, we say ALG is an $f(n)$ approximation

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

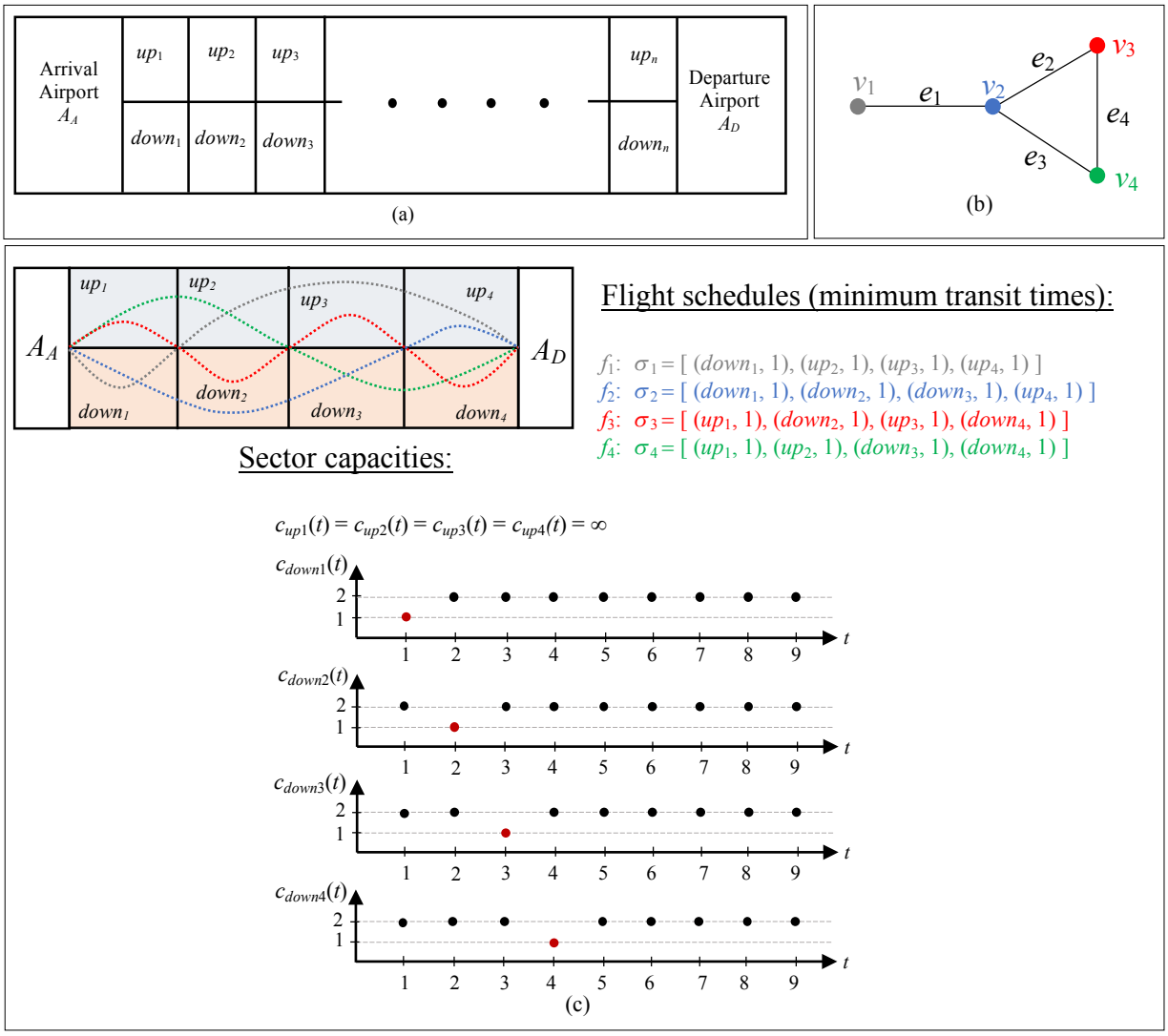


Figure 3. (a) Airspace geometry in the constructed SMDS instance, (b) Given MIS instance, (c) Constructed SMDS instance for given MIS instance.

algorithm for SMDS if, for an arbitrary instance of SMDS with n flights, we have ALG(I) ≤ $f(n)$ × OPT(I). In this section, we prove that no polynomial time $f(n)$ approximation algorithms exist for SMDS, where $f(n)$ is of the form $f(n) = n^{1-\varepsilon}$.

We prove this result by establishing a polynomial time reduction from graph *Maximum Independent Set* (MIS) problem to SMDS. Given a graph $G=(V, E)$, the MIS problem seeks to find the largest subset W of V such that none of the vertices in W is connected by an edge in E. In other words: $\forall\, u, v \in W$: $(u, v) \notin E$. Since it is known that MIS is not approximable[37] to within $n^{1-\varepsilon}$ for $\varepsilon > 0$, our result would follow. The reduction given preserves approximability in the strong sense. If a graph with n nodes has a maximum independent set of size m, then the reduction constructs an instance of SMDS with n flights and minimum delay $n-m$.

Given an instance $G=(V, E)$ of MIS, we construct an instance of SMDS as follows (see Fig. 3):

- For each vertex $v_i \in V$, create a flight $f_i \in F$.
- For each edge $e_j \in E$, create two sectors up_j and $down_j$ in S and set their capacities as follows:
  - $c_{up_j}(t) = \infty$
  - $c_{down_j}(t) = 1$; if $t = j$
    $= 2$; otherwise

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

- For each flight $f_i \in F$ (corresponding to vertex $v_i \in V$) its schedule is $\sigma_i = [(s_1, 1), (s_2, 1), \ldots, (s_n, 1)]$, where:
  - $s_j = down_j$; if the edge e_j is incident on vertex v_i
    $= up_j$; otherwise

Intuitively, think of the airspace as a rectangular area decomposed into sectors as illustrated in Fig. 3(a). Note that each flight traverses the airspace from left to right, spending one unit of time in each vertical strip consisting of the union of two sectors up_j and $down_j$. The upper sector up_j of a strip has sufficient capacity so that all flights can be accommodated simultaneously. Recall that each strip corresponds to an edge in the graph and each vertex to a flight. The two flights f_u, f_v corresponding to vertices that are connected to the j^{th} edge $e_j = (u, v)$ must fly through the lower sector $down_j$ in the corresponding strip. This sector has capacity 1 at time j. Thus, if neither flight is delayed prior to entering $down_j$, there will be a capacity constraint violation. However, if either or both of the flights are delayed prior to entering the sector, there will not be a capacity constraint violation. To illustrate the full construction, a simple instance $G = (V, E)$ of MIS and the corresponding SMDS instance constructed for it are shown in Fig. 3(b), 3(c), respectively. The vertices in the MIS instance shown in Fig 3(b) are color-coded, as are the notional paths of their corresponding flights in the constructed SMDS instance shown with dotted lines at the top of Fig. 3(c) on the left-hand side; the minimum transit times for these flights are shown at the top of Fig. 3(c) on the right-hand side.

Since the capacity of the upper sectors ($up_1, up_2, \ldots, up_n$) is set to infinity, the only capacity constraints that we need to worry about are those involving the lower sectors ($down_1, down_2, \ldots, down_n$). As illustrated in Fig. 3 (c), for each sector $down_j$, corresponding to an edge $e_j = (u, v)$ of the given MIS instance, the only two flights entering the sector are f_u, f_v, which are those corresponding to vertices u, v incident on e_j. Since the vertical strips consisting of an upper sector and a lower sector are visited sequentially, with each strip taking one time unit to traverse, and the capacity of $down_j$ drops for one time unit from 2 to 1 at time $t = j$, it follows that if either or both of the flights f_u and f_v are delayed at least one unit of time before they pass through $down_j$, then their passage through $down_j$ will not violate the sector capacity, which would be 2, whether they enter $down_j$ simultaneously or not. If, on the other hand, neither f_u nor f_v is delayed prior to their arrival into sector $down_j$ (at time $t = j$), then their entrance into sector $down_j$ would violate the sector capacity $c_{down}(t)=1$. Moreover, we can show the following:

Lemma 10. *In any optimal scheduling solution, of the constructed SMDS instance, each flight is delayed by at most one unit of time.*

Proof: Consider a solution in which flight f is delayed by more than one time unit. We claim that the delay can be reduced to exactly one, without violating the sector capacity constraints. The reason is that for any flight, once it is delayed by one unit (or more) prior to its departure, the capacity for each of the sectors that it ever enters exceeds the number of flights flying through the sector. This is because once we delay a flight by one unit at its departure, all the down sectors that it ever enters would have capacity 2 during the flight's presence in the down sector, essentially imposing no constraints, since the remaining flight that traverses the down sector can also be there, without violating the down sector's capacity. Since a delay of one unit would suffice, there is no need to ever have a delay or more than one unit, and hence, in a delay optimal solution, each flight is delayed by at most one unit of time. ■

Theorem 11. *Let* G = (V, E) *be an instance of MIS and* I *be the constructed SMDS instance. Then the largest independent set of* G *has* m *vertices if and only if the minimal delay schedule for I has a delay* d=|V|−m.

Proof: Assume W is a maximal independent set of G with m vertices. We construct a feasible solution of I with delay $|V| - m$. The solution imposes a one unit of time delay before the departure on each flight that corresponds to a vertex in $V - W$. It is easy to see that this solution is feasible, because for any edge in e = $(u, v) \in E$, since the two vertices u, v are connected by an edge, they cannot both belong to the maximum independent set W, and hence at least one of the two vertices u, v must be a member of $V - W$. Since all flights that correspond to a vertex in $V - W$ are delayed by one time unit prior to their departure, it means that all the down sectors are going to have their capacity meet or exceed the demand. Hence the schedule is feasible. To prove optimality, suppose that there exists a schedule to the SMDS instance with delay d' with $d' < d$. From the optimality of the solution, and Lemma 10, it must be that exactly d' flights are delayed, each by one time unit, while the rest of the flights are not delayed. Let W' be the set of undelayed flights. It is easy to see that W' would be an independent set for G, with more vertices than W. But this cannot be the case, since we assumed W to be the maximal independent set.

Conversely, assume that the optimal feasible schedule for I has total delay d. We wish to show that the optimal solution to the MIS problem has size $|V| - d$. This must be the case, since otherwise, based on the previous argument,

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

if the MIS problem has a larger independent set, then we could construct a feasible schedule for the SMDS instance with smaller delay than d. ■

Since it is known[41] that MIS is not approximable to within $n^{1-\varepsilon}$ for $\varepsilon > 0$, unless $\mathcal{P}=\mathcal{NP}$, it immediately follows that neither can SMDS. In addition, as it is shown, the optimal schedule for the constructed instance I of SMDS can take the delay for each delayed flight prior to taking off, as a ground delay. Furthermore, the non-approximability generalizes immediately to the MDS problem, which included SMDS as a special case. Hence, we have the following:

Theorem 12. *The $\mathcal{NP}$-hard MDS problem cannot be approximated to within* $n^{1-\varepsilon}$ *for* $\varepsilon > 0$, *where* n *is the number of aircraft in the problem instance, even if all the delays are to be taken on the ground prior to the departures.*

Also note that according to Lemma 10, in an optimal scheduling solution D of the constructed SMDS instance, each flight is delayed by at most one unit of time, hence $spread(D) \leq 1$. In other words, the non-approximability result holds even when we are required to maintain equity among the aircraft. Note that in the construction of the SMDS instance, we have used time-varying sector capacities. It remains open whether the non-approximability result of Theorems 11, 12 hold if the sector capacities are fixed over time.

IV. Maximum Set of Dependent Aircraft in Precision Arrival Scheduling

A. Problem Formulation

Increasing the arrival throughput at busy airports and the ability of the air transportation system to accommodate and minimize the impact of unforeseen events such as missed approaches, off-nominal conditions, or accommodating emergency aircraft into arrival schedules is of increasing importance. NASA's recent initiative entitled Method to Enhance Scheduled Arrival Robustness[38] (MESAR) is an attempt to investigate the missed-approach problem in the Terminal Area Precision Scheduling and Spacing (TAPSS) system[39]. Off-nominal conditions in scheduled arrivals are defined as conditions that cause the actual landing sequence to be different from what was originally scheduled and was frozen earlier at the ATC facility. Missed approach is an off-nominal condition in which an aircraft that has missed or relinquished its originally scheduled slot in the arrival sequence is being inserted back into the arrival stream, thus altering the originally scheduled landing sequence.

To accommodate this change, the algorithm needs to identify and send for rescheduling a set of aircraft whose arrival schedules would have to be recomputed. The Maximum Set of Dependent Aircraft (MSDA) problem studied in this section is motivated by the need to minimize the disturbances on the arrival schedule in order to accommodate a change caused by off-nominal conditions. The idea is to find the minimum set of aircraft that need to be rescheduled in order to accommodate the change caused by an off-nominal condition.

Consider a set F of flights flying along their arrival routes to land at their designated runways, as shown in Fig 4(a). The arrival routes can be represented by a *directed acyclic graph* (DAG) $G = (V, E)$ where E is the set of edges representing the route segments, and V is the set of vertices representing the beginning or ending of a segment, which include the intersections (merging points) between two or more route segments, including the *meter fixes* and *arrival fixes*, constituting the sign posts that the arriving aircraft are scheduled to fly through on their way to land. A central concern in MSDA is the notion of *dependence* between flights.

Consider a vertex $v \in V$ along the arrival routes, and two aircraft corresponding to flights f_i, f_j scheduled to fly through it, such that f_j is scheduled to fly through v before f_i is. Note that if f_i is delayed for some reason, and hence cannot meet its original scheduled time of flying through v, in the absence of other constraints, f_j can still proceed with its original schedule with no disruption, as far as flying through v is concerned. On the other hand, if f_j is delayed and cannot meet its original schedule, then such a delay would impact f_i as well. Thus, we say *that f_i's schedule depends on f_j's* or simply that *f_i depends on f_j*, and denote it as: $f_i \rightarrow f_j$. Such dependence is a *direct dependence*, since both flights pass through the same vertex along the routing graph. In general, however, two flights that do not fly through the same vertex in the routing graph (and hence their flight paths do not even intersect) may have dependence, in which case their dependence would be *indirect*. Such indirect dependence is generally caused by a chain of direct dependencies. We say that the dependence relation is *transitive*. That is, for any three flights f_i , f_j, f_k, if f_i depends on f_j ($f_i \rightarrow f_j$) and f_j depends on f_k ($f_j \rightarrow f_k$), then f_i depends on f_k ($f_i \rightarrow f_k$). More generally, this can be stated as follows:

Observation 13. *Given two flights* f_i, f_j, *such that:* $f_i \rightarrow f_j$, *there must exist a sequence of flights (*f_{a0}, f_{a1}, …, f_{ak}*) such that*: $f_i = f_{a0}$, $f_j = f_{ak}$, *and* $\forall i \in \{0, 1, 2, \ldots, k-1\}$: $f_{ai} \rightarrow f_{ai+1}$.

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

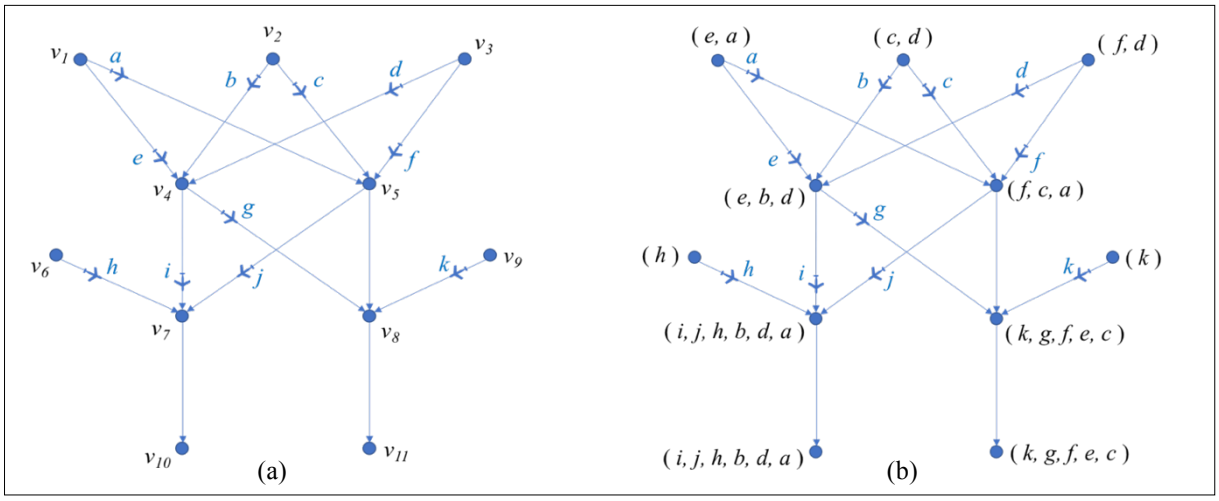


Figure 4. (a) Arrival routing graph and the flights, (b) Sequence of flights passing through each node in the graph.

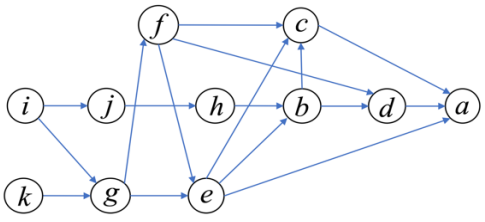
Given a flight $f \in F$, a subset of flights $H \subset F$ is called a *dependent set* of f if and only if $\forall h \in H$: $h \rightarrow f$. The *maximal dependent set* of f, denoted as $dep(f)$ is the set of all flights in F who depend on f (directly or indirectly). In the MSDA problem, we are given (G, F, T, f_d), where $G=(V, E)$ is the arrival routing graph, F is the set of arriving flights, T is the arrival schedule where $T(f_i, v)$ is the time at which flight f_i is scheduled to fly through vertex v, and $f_d \in F$ is the target flight that is about to miss its schedule. The objective is to find $dep(f_d)$, the maximal subset of flights in F that depend on f_d.

B. An Efficient Algorithm for Solving MSDA

Here, we present a simple and efficient algorithm for solving MSDA, using reachability in directed graphs. First, we define the dependence graph $G_D=(V_D, E_D)$, where $V_D = \{u_1, u_2, \ldots, u_n\}$ is the set of vertices, with vertex u_i corresponding to flight $f_i \in \{f_1, f_2, \ldots, f_n\}$. There is an edge in E_D from vertex u_i to vertex u_j if and only if flight f_i depends on flight f_j directly. That is, if f_i and f_j are scheduled to fly through the same node in the routing graph, with f_j scheduled to fly through the node before f_i. Figure 5 illustrates the construction of the dependence graph for the MSDA instance shown in Fig. 4. The direct dependencies corresponding to the set of flights going through each vertex are shown in Fig. 5(a), and the complete dependence graph is shown in Fig. 5(b).

Vertex	Flight arrival sequence at the vertex	Direct dependencies
v_1	(e, a)	$e \rightarrow a$
v_2	(b, c)	$b \rightarrow c$
v_3	(f, d)	$f \rightarrow d$
v_4	(i, g, e, b, d)	$i \rightarrow g \rightarrow e \rightarrow b \rightarrow d$
v_5	(f, c, d)	$f \rightarrow c \rightarrow a$
v_6	(h)	–
v_7	(i, j, h, b, d, a)	$i \rightarrow j \rightarrow h \rightarrow b \rightarrow d \rightarrow a$
v_8	(k, g, f, e, c)	$k \rightarrow g \rightarrow f \rightarrow e \rightarrow c$
v_9	(k)	–
v_{10}	(i, j, h, b, d, a)	$i \rightarrow j \rightarrow h \rightarrow b \rightarrow d \rightarrow a$
v_{11}	(k, g, f, e, c)	$k \rightarrow g \rightarrow f \rightarrow e \rightarrow c$

(a)



(b)

Figure 5. (a) Identifying direct dependences, (b) Constructing the dependency graph from direct dependencies.

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

The construction of the dependence graph has several benefits. It allows for a concise representation of the dependence among flights. In addition, it allows for the application of graph algorithms to discover dependence characteristics among flights. In particular, we show the following:

Theorem 14. *Consider MSDA instance (*G, F, T, f_d*) and let* G_D *be its corresponding dependence graph. Then, for any pair of aircraft* $f_i, f_j \in F$*, with corresponding vertices* $v_i, v_j \in V_D$*, we have* $f_i \rightarrow f_j$ *if and only if there is a directed path in* G_D *from* v_i *to* v_j.

Proof: (**IF**) Assume there is a direct path $P_{ij} = (u_0, u_1, u_2, \ldots, u_p)$ in the dependence graph from v_i to v_j, where $u_0 = v_i$, and $u_p = v_j$. Consider the vertices $(u_0, u_1, u_2, \ldots, u_p)$ and the edges (u_0, u_1), (u_1, u_2), $\ldots$, (u_{p-1}, u_p) along this path, and the corresponding flights $f_{u0}, f_{u1}, f_{u2}, \ldots, f_{up}$. Note that our construction of the dependence graph guarantees that each of the edges along P_{ij} corresponds to a direct dependence between the flights corresponding to the two vertices incident on each edge along the path P_{ij}. That is, we must have $f_i = f_{u0} \rightarrow f_{u1} \rightarrow f_{u2} \rightarrow \ldots \rightarrow f_{uk-1} \rightarrow f_{uk} = f_j$, which implies $f_i \rightarrow f_j$.

(**ONLY IF**): Assume $f_i \rightarrow f_j$, then according to observation 13, there must exist a sequence $F_{ij} = (f_{i0}, f_{i1}, \ldots, f_{ik})$ of flights in the MSDA problem instance with $f_i = f_{i0}$ and $f_j = f_{ik}$ such that for all $a \in \{0, 1, \ldots, k-1\}$, flight f_{ia} depends directly on flight f_{ia+1}. This means that there exists a direct path in G_D from vertex u_{ai} to vertex u_{aj} which passes through the vertices in V_D corresponding to the flights in F_{ij}. ■

C. An Efficient Algorithm for Solving MSDA

Theorem 14 suggests a simple approach for solving the MSDA. All we have to do is to construct the dependence graph G_D and then find all the vertices in V_D reachable from u_{id}, where u_{id} is the vertex in V_D corresponding to f_d, the target flight in the MSDA instance. The pseudocode for this algorithm is given in Fig. 6. To analyze the run-time of the algorithm, we first note that the dominant steps of the algorithm are steps 1 and 2, the creation of the dependence graph $G_D = (V_D, E_D)$, and the execution of BreadthFirstSearch on G_D.

```
Algorithm ComputeMaximalSetOfDependentAircraft (G, F, T, f_d)
BEGIN
    1.  u_d , G_D   = CreateDependenceGraph (G, F, T, f_d)
    2.  Reach(u_d) = BreadthFirstSearch(G_D, u_d)
    3.  dep(f_d) = {f_i ∈ F | u_i ∈ Reach(u_d)}
    4.  return dep( f_d )
END
```

Figure 6. Pseudocode of our algorithm for solving MSDA.

Construction of G_D : The amount of time required to construct the dependence graph G_D is proportional to the sum of its number of vertices and its number of edges, or $|V_D| + |E_D|$. Note from the definition of G_D that each vertex in the dependence graph corresponds to a flight in the MSDA problem instance. That is, $|V_D| = |F|$. Additionally, note that in the worst case, each flight can contribute to at most two direct dependencies for each of the vertices in the routing graph that it visits. Of course, it is possible (or perhaps even likely) that some of these direct dependencies involve the same pair of flights, as they may both cross different vertices in the routing graph consecutively. However, in our current analysis, we can ignore such a possibility as we are only concerned with calculating an upper bound for the number of edges in the dependence graph. Therefore, the total number of direct dependencies in an MSDA instance, as well as the number of edges $|E_D|$ in its corresponding dependence graph is upper bounded[#] by $|F| \times |V|$. In other words, we can construct G_D in time proportional to $|V_D| + |E_D|$, or O ($|A| \times |V|$) time.

Execution of BreadthFirstSearch : In its standard implementation, the breadth-first search algorithm has its worst-case running time proportional to the sum of the number of vertices and the number of edges, and thus this portion of the algorithm has running time $O(|V_D|+|E_D|) = O(|F| \times |V|)$. Therefore, we have the following result:

Theorem 15. *MSDA can be solved in time* O(mn) *using reachability on the dependence graph, where* m *is the number of vertices in the arrival routing graph, and* n *is the number of flights in the MDS problem instance.*

[#] Note that this is not necessarily a tight upper bound. As an example, the dependence graph in Fig. 5(b) has $|E_D|$=16 edges, while the upper bound provided by the expression $|F| \times |V|$ is 121. See Fig. 7(a) for more examples.

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

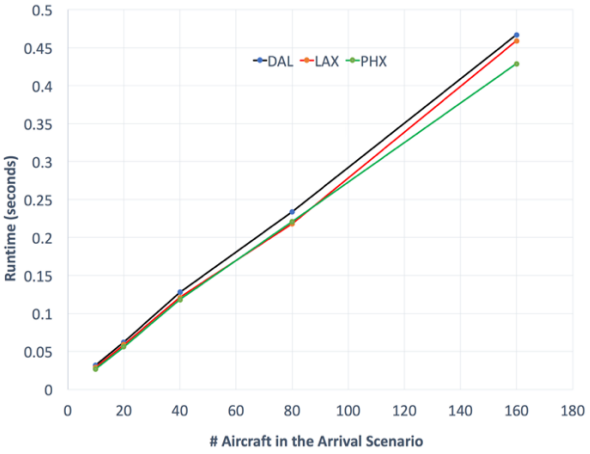
The algorithm of Fig. 6 was implemented in C/C++ and integrated into the simulation environment used in the real-time human-in-the-loop simulations reported by Jung, et al.[38] to minimize the extent of scheduling disturbances due to tactical updates to the arrival schedule.

A separate MATLAB implementation was also implemented to conduct a simple Monte Carlo study to test the performance and run time of the algorithm. Three different airports: Dallas Love Field (DAL), Los Angeles International Airport (LAX), and Phoenix Sky Harbor International Airport (PHX) were modeled, each with a subset of their corresponding arrival routing graphs. For each airport, the number of arrival aircraft were set to 10, 20, 40, 80, and 160, and for each of these problem sizes, 100 random arrival scenarios were generated, each with a randomly designated target aircraft. These scenarios were then run through the algorithm shown in Fig 6. The results of these experiments are summarized in Fig. 7, showing that the runtime of the algorithm is roughly linear in the number of aircraft in the problem instance, which is what is expected from the analysis presented earlier. The growth rate of the dependence graph is captured by $|E_D|$ shown in the last column of Fig. 7(a), next to its upper bound given by the expression $|F| \times |V|$. The average runtime for different problem sizes for each airport are shown in Fig. 7(b), confirming the efficiency of the algorithm and its suitability for solving MSDA in under a second for problem sizes up to a couple of hundred aircraft, making it suitable for real-time application.

An additional twist on the MSDA problem is that, at the moment the query comes, depending on where each arrival aircraft is situated along its arrival path, some of the aircraft have already passed a handful of the vertices along their arrival paths. This would allow for pruning the direct dependencies. Only those direct dependencies that lie in the future need to be taken into account. This allows for speeding up the algorithm, shrinking the size of the dependence graph. Such pruning was not implemented in the runs summarized in Fig. 7, so the reported run times represent a conservative view of the scalability of the algorithm.

Airport	$\|V\|$	$\|E\|$	$\|F\|$	$\|F\| \times \|V\|$	$\|E_D\|$ (averaged over 100 instances)
DAL	62	138	10	620	68
			20	1240	171
			40	2480	385
			80	4960	819
			160	9920	1688
LAX	55	92	10	550	79
			20	1100	185
			40	2200	396
			80	4400	815
			160	8800	1540
PHX	55	96	10	550	67
			20	1100	164
			40	2200	365
			80	4400	789
			160	8800	1593

(a)



(b)

Figure 7. (a) Number of edges in the dependence graph, on average, over 100 randomly generated MSDA problem instances for DAL, LAX and PHX airports. (b) The average runtime growth rate for the 100 random instances tested for each size for DAL, LAX, and PHX.

V. Discussion

A unifying theme and an underlying purpose of presenting results in this work has been to advocate more extensive use of graph theoretic ideas in the ATM domain, and to demonstrate that graph theory provides a suitable mathematical abstraction for formulating, reasoning about and solving many technical problems arising in ATM domain. The abstraction offered by graph theory has several notable advantages: 1) it provides a framework that is suitable for precise formulation of the problem that is free from unrelated details of the specific application domain, 2) this precise formulation allows for the application of graph-theoretic algorithms and ideas that may have been historically developed for very different application domains, to be readily applicable to new applications, and thus allow for cross fertilization and collaboration among researchers specializing in very different subject areas, 3) the graph theoretic mathematical formulation allows for different generalizations that may potentially lead to new, unforeseen insights and exciting developments, and 4) the rich and powerful libraries of customizable graph data

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

structures, graph analytics, and the associated algorithms and software toolkits[40–43] can be exploited to model, reason about, and eventually solve diverse problems in modern air transportation.

In addition, linking new (ATM) problems to (graph) problems that have been studied in the past, creates bridges among islands of knowledge, thus contributing to the expansion of our understanding of the problems and the relationships among them. Not only does this minimize the potentially wasted efforts to rediscover what is already known and the proverbial "reinventing the wheel", it also allows potential future discoveries in related domains to be more readily applicable to the ATM domain.

Gaining a better understanding of the inherent difficulty and the computational complexity of a problem allows the research effort to be better guided and focused on areas that are more likely to lead to promising solutions. Some of the approaches that can be taken include a) studying special classes of the problems that can be solved optimally or approximately, b) developing heuristics that may target special classes of the problem optimally or approximately, and then using those solutions as guidelines to develop solutions for practical problem instances, c) using strategies such as divide-and-conquer or dynamic programming to break the problem into smaller sup-problem instances that could each be solved optimally or approximately, and then combining the solutions to the sub-problems intelligently into reasonably good solutions for the original problem. Graph theoretic algorithms and libraries[40–43] can provide a suitable modeling, development, algorithmic framework to realize such strategies.

In formulating and presenting our results for the ASP, SCP, and MDS problems, we have mainly focused on, and used the terminology specific to, the underlying ATM applications. However, as an example for the 2nd advantage listed in the previous paragraph, note that due to the flexible nature of the way workload models and sector capacity may be defined, these problems can find applications that arise in diverse set of technical disciplines dealing with partitioning of a *geometric space* that may intersect a set of *interacting objects*. Depending on the application, the workload or the capacity models may be defined differently, and the objects may be static or moving entities modeled as points, line segments, curves, or other geometric shapes located in space-time. Some of these applications include geometric load balancing for parallel and distributed processing[44], geographic partitioning for workload balancing of municipal services (e.g., power districting[45], winter road maintenance[46], determination of police patrol coverage areas[47]), political redistricting[48], and sales force territory design[49].

VI. Conclusion

Graph theory is used to study different problems arising in air traffic management. First, using a polynomial reduction from an $\mathcal{NP}$-complete planar graph partitioning problem, it is shown that the airspace sectorization problem is $\mathcal{NP}$-hard, in general under several simple workload models, even if the projection of the flight trajectories in the two-dimensional plane forms a planar graph of maximum vertex-degree six. This result is extended to show that the seemingly simpler and more practical problem that tries to make local adjustment to a given sectorization (as opposed to computing one from scratch)—namely the Sector Combining Problem—is also $\mathcal{NP}$-hard. It remains open whether or not there exist polynomial time approximation algorithms for the airspace sectorization problem and the sector combining problem. Second, by establishing a polynomial time reduction from maximum independent set in graphs, it is shown that for any fixed ε, the problem of finding a solution to the minimum-delay scheduling problem in traffic flow management that is guaranteed to be within $n^{1-\varepsilon}$ of the delay-optimal solution, where n is the number of aircraft in the problem instance, is $\mathcal{NP}$-hard, even if the delays are all taken on the ground. Furthermore, this non-approximability result holds when we are required to maintain delay equity among different flights in the problem instance. The reduction, however, used a time-varying sector capacity schedule. It remains open whether the non-approximability result holds if the sector capacities are fixed and do not vary over time. Finally, a problem arising in precision arrival scheduling is formulated and solved using graph reachability. The algorithm was implemented and tested in a related real-time human-in-the-loop simulation, and experimental evidence is provided to demonstrate the scalability of the algorithm over randomly generated arrival scenarios for three different airports.

We believe effective exploration and application of graph theoretic ideas, in conjunction with proper use of data analysis and machine learning techniques could be a very attractive area for further investigation of manned and autonomous ATM that could help pave the path for more effective automation and automated discovery of optimized solutions.

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

Appendix

A. Refresher on Graph Theory

A very basic introduction and notation on the theory of graphs is presented below. A more thorough discussion can be found in standard texts on the subject [3, 4]. A *graph* G is defined as an ordered pair (V, E) of disjoint sets, where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ represents the set of *vertices* or *nodes*, and $E = \{e_1, e_2, \ldots, e_{|E|}\}$ is the set of *edges*. Each edge $e = (u, v) \in E$ is a pair of vertices in V, signifying that the two vertices u and v are *connected* or *joined* by the edge. The graph may be *directed* or *undirected*, in which case the edges are directed or undirected, and are represented by *ordered* or *unordered* pairs, respectively. An edge (u, v), which is also represented as uv, is said to *join*, *connect* or be *incident on* vertices u and v, causing the two vertices to become *adjacent to* (or *neighboring*) each other, and each being *incident* on the edge uv. For a vertex $v \in V$, the number of distinct edges incident on v, denoted as d_v, is known as its *degree*. Given graphs $G = (V , E)$ and $G' = (V', E')$, we say that G' is a *subgraph* of G if and only if $V' \subseteq V$ and $E' \subseteq E$. In subgraph $G' = (V', E')$, if E' includes all edges of G that join two vertices in V', then G' is the subgraph of G *induced by* V'.

It is often useful to draw a picture of a graph as its visual representation. In such representation, the vertices are drawn as distinct labeled points, and the edges are drawn as (potentially curved) lines connecting the points corresponding to the vertices it joins. If such a drawing can be accomplished on the plane in such a way that the edges do not intersect, except (potentially) at their endpoints corresponding to the vertices they connect, then the graph is a *plane* or *planar* graph and such a drawing is a *planar* drawing of the graph. A path P in the graph $G = (V , E)$ is a non-empty sequence $(p_1, p_2, \ldots, p_k)$ of vertices in V such that $\forall\, i \in \{1, 2, \ldots, k\}$: $(p_i, p_{i+1}) \in E$. Such a path is said to *connect* p_1 to p_k. A graph is *connected* if for every pair (u, v) of distinct vertices there is a path connecting u to v. A *cycle* is a path wherein the first and the last vertex in the sequence are the same. A graph that does not contain any cycles is a *forest* or an *acyclic graph*. A connected forest is a *tree*. A tree that visits all the vertices in the graph is called a *spanning tree*. Some of these notions are illustrated in Fig. 8.

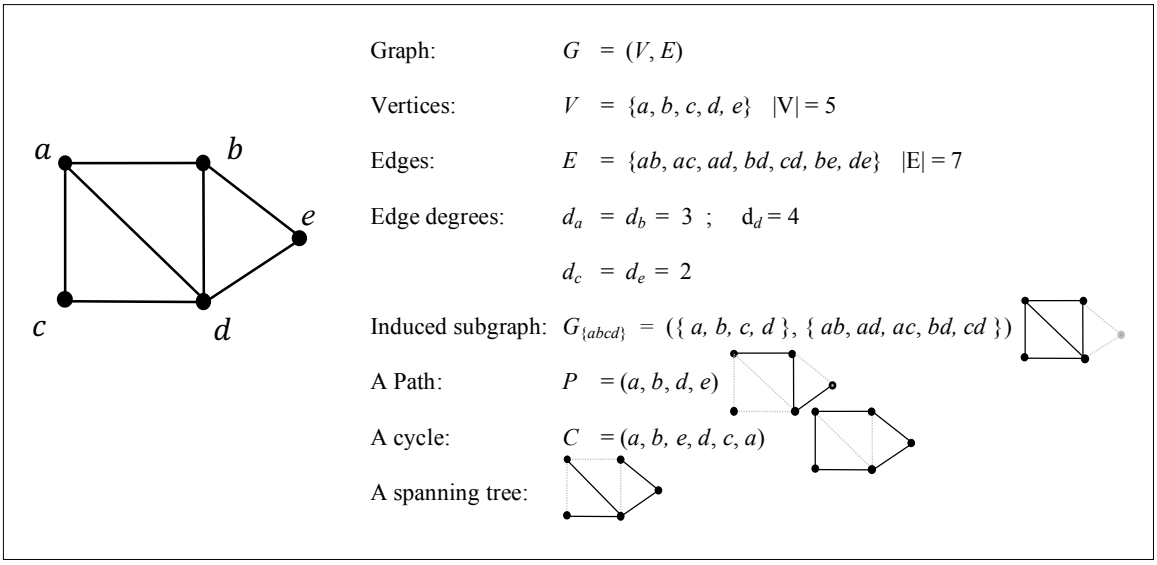


Figure 8. A planar undirected graph *G* and some related graph concepts.

B. Refresher on the Theory of Computational Complexity

Basic background from the theory of computational complexity is provided here. For a more thorough discussion of the topic, the reader is referred to standard texts on the subject.[5, 6] Computational complexity is concerned with studying the amount of resources (such as time, space, power) required to solve different computational problems. At the most basic level, a problem is said to be in $\mathcal{P}$ if and only if it can be solved deterministically in polynomial time. That is a time, polynomial in the size of the problem instance, using a suitable unit of time. A problem is in $\mathcal{NP}$ if

and only if, given a candidate solution *c*, there is a deterministic algorithm that can verify in polynomial time that *c* is in fact a valid solution. A problem is $\mathcal{NP}$-hard if it is at least as hard as the hardest problems in $\mathcal{NP}$. In other words, a problem P is $\mathcal{NP}$-hard if and only if all problems in $\mathcal{NP}$ can be solved in deterministic polynomial time if P can be solved in deterministic polynomial time. A problem is $\mathcal{NP}$-*complete* if it is both in $\mathcal{NP}$ and is $\mathcal{NP}$-hard. Fig. 9 provides a notional illustration of how the complexity classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-Complete, and $\mathcal{NP}$-hard are related to one another.

When expressing the time complexity of an algorithm, we typically express the worst-case run-time of the algorithm as a function of the size of the problem instance. So, when we say that a problem has a deterministic polynomial time algorithm, we mean that there is a deterministic algorithm solving the problem whose worst-case running time is bound by a polynomial function of the input size (given a suitable unit of time). A problem that is shown to be $\mathcal{NP}$-complete or $\mathcal{NP}$-hard is generally considered *intractable*. In such cases, one turns to algorithms that do not guarantee finding the optimal solution. This is because there is widespread agreement (but no known proof, despite much dedicated effort) that no $\mathcal{NP}$-complete problem can be solved deterministically in polynomial time. In fact, the biggest open question in computer science and one of the greatest open problems in all of mathematics [50, 51] is whether the classes of problems that can be solved deterministically in polynomial time ($\mathcal{P}$) and those that can be solved non-deterministically in polynomial time ($\mathcal{NP}$) are the same. That is, whether $\mathcal{P} = \mathcal{NP}$.

The general technique used for showing that a given problem P is $\mathcal{NP}$-hard ($\mathcal{NP}$-complete, respectively) is to use what is typically called *polynomial time reduction*, sometimes referred to simply as *reduction*. That is, to find an $\mathcal{NP}$-hard ($\mathcal{NP}$-complete, respectively) problem Q, and to show that given an arbitrary instance I_Q of Q, it is possible to create an instance I_P of P in (deterministic) polynomial time, such that the solution of Q on instance I_Q can be readily translated into a solution of P on instance I_P, and vice versa. In the context of decision problems (problems with answers YES or NO), this means that the answer to the decision problem Q on instance I_Q is YES if and only if the answer to the decision problem P on instance I_P is YES. If we can show this, we have effectively proven that there is a way to solve an $\mathcal{NP}$-hard ($\mathcal{NP}$-complete, respectively) problem by first translating it into our problem P (using polynomial amount of effort), and then solving P on the constructed instance I_P.

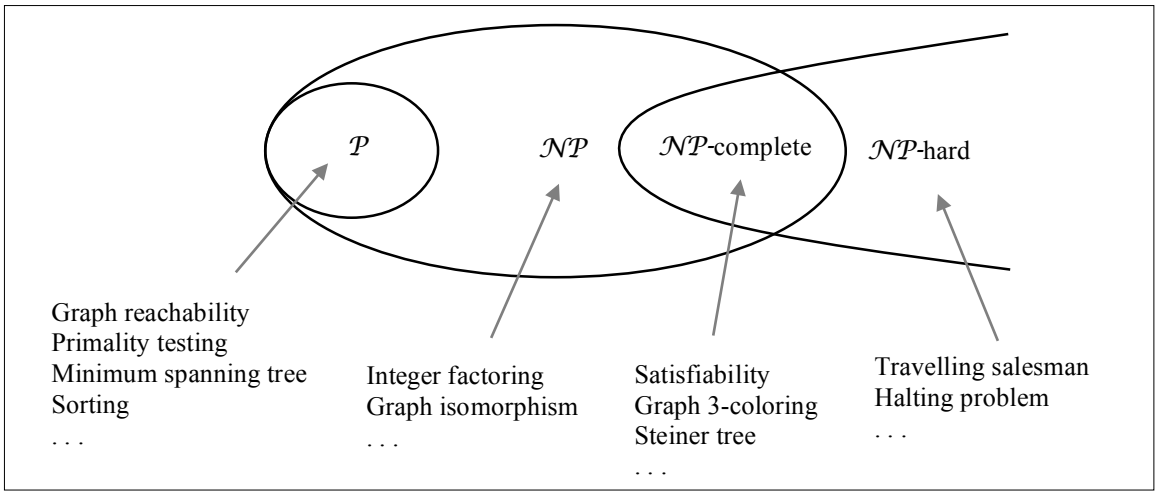


Figure 9. Classes $\mathcal{P}$, $\mathcal{NP}$, $\mathcal{NP}$-complete and $\mathcal{NP}$-hard of problems, and examples of each.

When a problem is shown to be $\mathcal{NP}$-hard, it is generally considered a problem whose optimal solution cannot be found except in special cases. In practice, several paths may be taken in such circumstances. One strategy is to find algorithms that do not guarantee to find the optimal solution, but are guaranteed to find solutions that are within provable bounds of the optimal. Such algorithms are called *approximation algorithms*[52] and are the subject of extensive studies in the theory of algorithms. Formally, an algorithm is called *r-approximate* for a minimization (maximization) problem if the algorithm finds a solution whose cost is at most a factor of *r* away from the optimal solution. An optimization problem is in $\mathcal{P}$ if it has a combinatorial structure that can be exploited as a foothold to efficiently home in on an optimal solution. While $\mathcal{NP}$-hard problems do not offer such a foothold, they may still possess structures that could be exploited to find solutions that are guaranteed not to be too far from the optimal. Yet,

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

there are optimization problems for which it is possible to prove that even the design of an *r*-approximate algorithm with small *r* is impossible, unless $\mathcal{P}=\mathcal{NP}$. An example of such a problem is Maximum Independent Set (MIS) in graphs, that involves finding the largest set of vertices none of which is connected by an edge to any other vertex in the set. In an influential paper, Håstad[37] proved that, for a graph with *n* vertices, MIS cannot be approximated to within $n^{1-\varepsilon}$ for any $\varepsilon > 0$, unless $\mathcal{P}=\mathcal{NP}$. More thorough discussion of the results on hardness of approximation can be found in the related literature for probabilistic checking of proofs[53, 54] and computational inapproximability[55, 56].

Acknowledgments

The authors would like to thank Mr. Todd Farley, Dr. Shon Grabbe, Mr. Saugata Guha, Dr. Sebastian Gutierrez Nolasco, and Mr. Zachary Wood for valuable discussions and feedback on earlier drafts of this manuscript. A. H. Farrahi would also like to thank Mr. Todd Farley for his support and encouragements during the course of this research.

References

[1]Next Generation Air Transportation System, Federal Aviation Administration, U.S. Department of Transportation, http://www.faa.gov.nextgen/.

[2]Kopardekar, P. H., Aquilina, R., Edwards, T.A., Crisp, V. K., and Covalowski, J. A., Airspace systems program: Next generation air transportation system concepts and technology development projects, 2011. FY2011-2015 Project Plan, National Aeronautical and Space Administration, April 5, 2011.

[3]Bollobas, B., *Modern Graph Theory*, Graduate Texts in Mathematics, book 184, corrected ed., Springer, October 2013.

[4]Distel, G., *Graph Theory*, Graduate Texts in Mathematics, book 179, 5th ed., Springer, October 2013.

[5]Garey, M. R., and Johnson, D. S., *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W.H. Freeman, January 1979.

[6]Arora, S., Boaz, B., *Computational Complexity: A Modern Approach*, 1st ed., Cambridge University Press, April 2009.

[7]Kopardekar, P., Bilimoria, K., Bridhar, B., "Initial Concepts for Dynamic Airspace Sectorization", *Proc. Of AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Belfast, Northern Ireland, September 18–20, 2007.

[8]Leiden, K., Kamienski, J., Kopardekar, P., "Initial Implications of Automation on Denamic Airspace Configuration", *Proc. Of AIAA Aviation Technology, Integration and Operations Conference (ATIO)*, Belfast, Northern Ireland, September 18–20, 2007.

[9]Trandac, H., Duong, V., "Optimized Sectorization of Airspace with Constraints", *Proceedings of the 5th Eurocontrol/FAA ATM R&D Seminar*, Budapest, Hungary, June 2003.

[10]Yousefi, A. Donahue, G.L., "Temporal and Spatial Distribution of Airspace Complexity for New Methodologies in Airspace Design, *Proceedings of the 4th AIAA Aviation Technology, Integration, and Operations Conference* (ATIO), Chicago, IL, USA, September 2004.

[11]Klein, A. "An Efficient Method for Airspace Analysis and Partitioning Based on Equalized Traffic Mass", *Proceedings of the 6th USA/Europe Seminar on Air Traffic Management Research and Development*, Baltimore, MD, USA, June 2005.

[12]Basu, A., Mitchell, J.S.B., Sabhnani, G., "Geometric Algorithms for Optimal Airspace Design and Air Traffic Controller Workload Balancing", 16th Fall Workshop on Computational Geometry, Northampton, MA, November 2006.

[13]Kopardekar, P., Bilimoria, K., Sridhar, B., "Initial Concepts for Dynamic Airspace Configuration", *Proceedings of AIAA Aviation Technology, Integration and Operations Conference*, September 2007.

[14]Martinez, S.A. Martinez, G.B. Chatterji, D. Sun, A.M. Bayen, "A Weighted-Graph Approach for Dynamic Airspace Configuration", *Proceedings of AIAA Guidance, Navigation and Control*, 2007.

[15]Basu, A., Mitchell, J.S.B., Sabhnani, G., "Geometric Algorithms for Optimal Airspace Design and Air Traffic Controller Workload Balancing", *Proceedings of SIAM Workshop on Algorithm Engineering and Experiments*, pp. 75–89, San Francisco, CA, January 2008.

[16]Klein, A., Rodgers, M.D., Kaing, H., "Dynamic FPAs: A New Method for Dynamic Airspace Configuration", *Integrated Communications Navigation and Surveillance* (ICNS), Bethesda, MD, May 2008.

[17]Xue, M., "Airspace Sector Redesign Based on Voronoi Diagrams", *Proceedings of AIAA Guidance, Navigation, and Control Conference*, Honolulu, HI, August 2008.

[18]FAA Order JO7210.3V. Facility Operation and Administration, Chapter 17, Section 7. Monitor Alert Parameter, 2007.

[19]Kopardekar, P., Schwartz, A., Magyarits, S., Rhodes, J. "Airspace Complexity Measurement: An Air Traffic Control Simulation Analysis", *Proceedings of the 7th USA/Europe R&D Seminar*, Barcelona, Spain, July 2007.

[20]Sridhar, B., Sheth, K.S., Grabbe, S., "Airspace Complexity and its Application in Air Traffic Management", *Proceedings of the 2nd USA/Europe Air Traffic Management R&D Seminar*, Orlando, FL, December 1998.

[21]Masalonis, A.J., Callaham, M.B., Wanke, C.R., "Dynamic Density and Complexity Metrics for Realtime Traffic Flow Management", *Proceedings of the 5th USA/Europe Air Traffic Management R&D Seminar*, Budapest, Hungary, June 2003.

[22]Kopardekar, P., Magyarits, S., "Measurement and Prediction of Dynamic Density", *Proceedings of the 5th USA/Europe Air Traffic Management R&D Seminar*, Budapest, Hungary, June 2003.

[23]Dyer, M.E., Frieze, A.M. "Planar 3DM is NP-Complete", *Journal of Algorithms*, Vol. 7 (2), pp. 174–184, 1986.

[24]Bloem, M., Kopardekar, P., "Combining Airspace Sectors for the Efficient Use of Air Traffic Control Resources", *Proceedings of AIAA Guidance Navigation and Control Conference*, Honolulu, HI, August 2008.

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775

[25]Bloem, M., Gupta, P., Kopardekar, P., "Algorithms for Combining Airspace Sectors", *Air Traffic Control Quarterly*, Vol. 17, No. 3, September 2009.
[26]Drew, M., "A Method of Optimally Combining Sectors", *Proceedings of AIAA Aviation Technology, Integration and Operations Conference*, Hilton Head, SC, September 2009.
[27]Vranas, P.B., Bertsimas, D.J., Odoni, A.R., "The Multi-Airport Ground-Holding Problem in Air Traffic Control", *Operations Research*, Vol. 42, No. 2, pp. 249–261, March–April 1994.
[28]Bertsimas, D.J., Stock-Patterson, S., "The Air Traffic Flow Management Problem with Enroute Capacities", *Operations Research*, Vol. 46, No. 3, pp. 406–422, May–June 1998.
[29]Bertsimas, D., Lulli, G., Odoni A., "An Integer Optimization Approach to Large-Scale Air Traffic Flow Management", *Operations Research*, Vol. 59, Issue 1, pp. 211–227, 2011.
[30]Landry, S.J., Farley, T., Hoang, T., "A Distributed Scheduler for Air Traffic Flow Management", *Journal of Scheduling*, Vol. 15, Issue 5, pp. 537–551, October 2012.
[31]Tandale, M.D., Wiraatmadja, S., Vaddi,S., Rios, J.L. "Massively Parallel Optimal Solution to the Nationwide Traffic Flow Management Problem", *Proceedings of the AIAA Aviation Technology, Integration, and Operations Conference*, Los Angeles, CA, August 12–14, 2013.
[32]Barnhart, C., Bertsimas, D., Caramanis, C., Fearing, D., "Equitable and Efficient Coordination in Traffic Flow Management", Transportation Science, Vol. 46, No. 2, pp. 262–280, May 2012.
[33]Zhang, Y., Su, R., Li, Q., Cassandras, C.G., Xie, L., "Distributed Flight Routing and Scheduling for Air Traffic Flow Management", *IEEE Transactions on Intelligent Transportation Systems*, Early Access Article, Vol. PP, No. 99, pp.1–12, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7855790&isnumber=4358928, [cited 30 April, 2017].
[34]Johnson, D.S., "Approximation Algorithms for Combinatorial Problems", Journal of Computer and System Sciences, Vol. 9, Issue 3, pp. 256–278, Elsevier, December 1974.
[35]Rosenkrantz, D.J., Stearns, R.E., Lewis, P.M., "An Analysis of Several Heuristics for the Traveling Salesman Problem", *SIAM Journal on Computing*, Vol. 6, Issue 3, pp. 568–581, 1977.
[36]Arora, S., "Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and Other Geometric Problems", Journal of the ACM, Vol. 45, No. 5, pp. 753–782, September 1998.
[37]Håstad, J., "Clique is Hard to Approximate within $n^{1-\varepsilon}$", *Acta Mathematica*, Vol. 182, pp. 105–142, 1999.
[38]Jung, J., Verma, S.A., Zelinski, S.J., Kozon, T.E., Sturre, L., "Assessing Resilience of Scheduled Performance-Based Navigation Arrival Operations", Eleventh USA/Europe Air Traffic Management Research and Development Seminar (ATM2015), Lisbon Portugal, Jun 23–26, 2015.
[39]Swenson, H., Jung, J., Thipphavong, J., Chen, L., Martin, L., Nguyen, J., "Development and Evaluation of the Terminal Precision Scheduling and Spacing System for Off-Nominal Condition Operations", 31st IEEE/AIAA Digital Avionics Systems Conference (DASC), pp. 1–18, Williamsburg, VA, October 14–18, 2012.
[40]Apache Giraph, URL: http://giraph.apache.org [cited 30 April, 2017].
[41]Apache TinkerPop™, URL: http://tinkerpop.apache.org [cited 30 April 2017].
[42]Python NetworkX, URL: https://networkx.github.io [cited 30 April 2017].
[43]Siek, J.G., Lee, L.–Q., *Boost Graph Library, User Guide and Reference Manual*, Addison-Wesley Professional, December 30, 2001.
[44]Kim, J., Papaefthymious, M, Tayyab, A., "An Algorithm for Geometric Load Balancing with Two Constraints", *18th IEEE International Symposium on Parallel and Distributed Processing*, pp. 40–48, April, 26–30, 2004.
[45]Bergey, P.K., Ragsdale, C., T., Hoskote, M., "A Simulated Annealing Genetic Algorithm for the Electric Power Districting Problem", *Annals of Operations Research*, Vol. 121, Issue 1, pp. 33–55, July 2003.
[46]Perreira, N., Langevina, A., Campbell, J.F., "The Sector Design and Assignment Problem for Snow Disposal Operations", *European Journal of Operations Research*, Elsevier, Vol. 189, No. 2, pp. 508–525, 2008.
[47]Curtin, K.M., Hatslett-McCall, K.L., Qui, F., "Determining Optimal Police Patrol Areas with Maximal Covering and Backup Covering Location Models", *Networks and Spatial Economics*, Springer, Vol. 10, Issue 1, pp. 125–145, March 2010.
[48]Altman, M., "The Computational Complexity of Automated Redistricting: Is Automation the Answer?", *Rutgers Computer and Law Technology Journal,* Vol. 23, No. 1, pp. 81–142, 1997.
[49]Howick, R.S., Pidd, M., "Sales Force Deployment Models", *European Journal of Operations Research*, Elsevier, Vol. 48. No. 3, pp. 295–310, October 1990.
[50]Devlin, K. J., *The Millennium Problems: The Seven Greatest Unsolved Mathematical Puzzles of Our Time*, Basic Books, October 2003.
[51]Jaffe, A., Wiles, A., Carlson J., *The Millennium Prize Problems*, American Mathematical Society, June 2006.
[52]Vazirani, V., *Approximation Algorithms*, Springer, March 2004.
[53]Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M., "Proof Verification and the Hardness of Approximation Problems", *Journal of the ACM*, Vol. 45, Issue 3, pp. 501–555, May 1998.
[54]Dinur, I., "The PCP Theorem by Gap Amplification", *Journal of the ACM*, Vol. 54, Issue 3, Article 12, June 2007.
[55]Trevisan, L., "Inapproximability of Combinatorial Optimization Problems", Electronic Colloquium on Computational Complexity, Rev. 1, Report No. 65, February 22, 2010. URL: https://eccc.weizmann.ac.il/report/2004/065 [cited 16, June 2017].
[56]Khot, S., "Inapproximability of NP-Complete Problems", *Proceedings of the International Congress of Mathematicians*, Volume 1, pp. 711–728, Seoul, S., Korea, August 13–21, 2014.

Downloaded by NASA AMES RESEARCH CENTER on July 6, 2017 | http://arc.aiaa.org | DOI: 10.2514/6.2017-3775