AIAA 2016-3751

AIAA Aviation
13-17 June 2016, Washington, D.C.
16th AIAA Aviation Technology, Integration, and Operations Conference

A Mixed Integer Linear Program for Real-Time Computing the Optimal Push Back Time Windows

William J. Coupe [*] and Dejan Milutinović [†]
Jack Baskin School of Engineering, University of California, Santa Cruz, CA 95064, USA
Waqar Malik [‡]
University of California, Santa Cruz, NASA Ames Research Center, Moffett Field, CA 94035, USA
Yoon Jung [§]
NASA Ames Research Center, Moffett Field, CA 94035, USA

This paper proposes a tool to aid ramp area controllers to meet the scheduled target movement area times by computing the push back time windows for each departing aircraft that ensure conflict free trajectories. The push back window is defined by the range between the earliest feasible push back time and latest feasible push back time. In this work we propose a new mixed integer linear program approach that uses a significantly smaller number of constraints compared to our previous approach. Using fewer constraints, the problem can be solved in less computation time and the mixed integer linear program can be incorporated in real-time decision making. We show that the formulation can be applied to schedules of multiple aircraft without modification. Solutions are provided and the objective function is analyzed to reveal the dependence of the computation time and the quality of solution to a parameter within the objective function.

I. Introduction

Unlike aircraft maneuvers on taxiways, ramp area aircraft movements are typically not confined to well-defined trajectories. The shape and timing of the trajectories are subject to uncertainties resulting from pilots decisions as well as other factors involved in ramp area operations, which can impede upon an optimal taxiway schedule plan. Most of the previous research [1–12] on taxiway scheduling has modeled the FAA controlled runways and taxiways as a graph, i.e. a connected network, and authors have neglected to account for the planning and execution of aircraft maneuvers within the ramp area. Although the surface operations can be improved by adopting an optimal taxiway schedule, its execution ultimately depends on human controllers who control aircraft maneuvers in both the ramp area and taxiways [13]. Ramp area aircraft have been incorporated in [14, 15], but the trajectories are considered to be deterministic.

The main difficulty in the integration of ramp area operations into an optimal taxiway scheduling solution is in addressing uncertainties of ramp area aircraft trajectories. To address the uncertainty, a stochastic model of aircraft trajectories [16, 17] was proposed. The model was used to sample a large number of feasible ramp area aircraft trajectories within the ramp area. A feasible trajectory is any sampled trajectory from the stochastic model that arrives at the target movement area within a predefined range of heading angles. The set of feasible trajectories for each aircraft is sampled in the absence of other aircraft. Therefore, the set of trajectories represent the feasible ways in which the aircraft can push back from their gate and taxi to the target movement area unimpeded by other aircraft.

Using the sampled trajectories, we computed conflict distributions among any two aircraft defined by their relative target movement area schedule [16, 17]. The conflict distributions are defined by a measure

[*] AIAA Graduate Student Member, Graduate Student, Computer Engineering Department, UC Santa Cruz.
[†] Associate Professor, Computer Engineering Department, UC Santa Cruz
[‡] Research Scientist, University Affiliated Research Center, MS 210-8, Moffett Field, CA 94035.
[§] AIAA senior member and Aerospace Engineer, NASA Ames Research Center, MS 210-6, Moffett Field, CA 94035.

of conflict estimated from the ratio of conflicting trajectories to total trajectories for every relative target movement area schedule. The conflict distributions were used to compute conservative conflict separation constraints that were passed to a mixed integer linear program (MILP) [18–20] which incorporated the Spot and Runway Departure Advisor (SARDA) [7–10, 21–23] design approach.

During time periods of heavy traffic, SARDA advises departing aircraft to remain at the gate with engines off, and when cleared, they can proceed straight from the gate to the departure runway queue *minimizing* slowing down or stopping for other traffic [8] and still meet their target movement area and take-off times. This technique has the effect of significantly reducing fuel burn and engine emissions. A Recent study [24] estimated as much as 18% of fuel consumption during taxi operations was due to stop-and-go activity. The study also concluded that under the assumption of 15 knots or greater speed for all unimpeded aircraft, there is the potential to reduce overall fuel consumption on the surface by at least 21%. For departing aircraft to proceed along the route unimpeded, it is critical the aircraft arrive at the target movement area at the scheduled time.

This research proposes a tool to aid ramp area controllers in meeting the scheduled target movement area times by computing the push back time window for each departing aircraft. The push back time window is defined by the range between the earliest feasible push back time and latest feasible push back time. Initiating the push back within these bounds ensures there exists a feasible trajectory that arrives at the target movement area at the required time, which is defined by a higher level optimal taxiway scheduler. The main contribution of the tool is to compute push back time windows that allow for aircraft to taxi unimpeded from the gate to the departure runway queue in the presence of other aircraft and trajectory uncertainties. This allows for ramp controllers to better manage surface traffic, reduce fuel consumption, and execute conflict free ramp area aircraft trajectories.

For relative target movement area schedules resulting in conflict free aircraft trajectories, computing the push back time windows are straightforward and can be estimated from the sampled trajectories. To compute the push back time windows we estimate the maximum trajectory duration and minimum trajectory duration for each aircraft, and then subtract from the scheduled target movement area time for the given aircraft. These computed times represent the start and finish of the feasible push back time window for each aircraft, respectively. When the schedule may lead to aircraft trajectory conflicts, an optimization procedure which solves for push back time windows in the presence of aircraft trajectory uncertainties is needed.

Previously, we proposed a MILP approach to solve for the optimal combination of push back time windows [25]. The optimal combination of push back time windows was defined to maximize the minimum push back time window of a set of all aircraft being scheduled. The solutions were based on conflict points that represent combinations of push back times that result in aircraft trajectory conflicts. We used the idea that no conflict point should be a convex combination [26] of the points in time that define the start and finish of the push back time window. In one-dimension, a convex combination of two points lies in between the two points. The number of constraints that are passed to the MILP is a function of the number of conflict points and for every conflict point, we need five constraints.

The distribution and/or the boundary of the conflict points is not known *a priori*. The MILP that we developed in the previous paper is able to solve for the optimal combination of push back time windows given any distribution of conflict points. We pay for this in terms of the number of constraints and ultimately in the computation time. As the number of aircraft we solve for increases, the number of constraints hinders the solution technique. The resulting algorithm is too slow for real-time decision making and thus we need to develop techniques to reduce the execution time. Here we propose a new MILP approach that uses a significantly smaller number of constraints. Using fewer constraints, the problem can be solved with less computation time and the MILP model can be incorporated in real-time decision making.

This paper is organized as follows. In section II we formulate the problem that we are considering. In section III we describe the techniques that we use to reduce the number of constraints that are passed to the MILP. In section IV we formulate the MILP approach we use to solve for the optimal combination of time windows. Next, in section V, we analyze the multi-criterion objective function that defines our solutions. In the last section, we conclude with a discussion of our findings and provide directions for future work.

II. Problem Formulation

In this section we formulate the problem of computing the optimal ramp area aircraft push back time windows for each departing aircraft. The combination of push back time windows will be constrained to contain zero conflict points. We begin by defining the variables and parameters that we use:

Symbol	**Description**
i	A family of departing trajectories originating from a single gate and characterized by a single left or right push back maneuver pattern
$\mathbb{D}$	The set of all departing trajectory families available to push back from their gate
A	A specific family of trajectories available to push back from gate A
t_i	The scheduled time for family i to arrive at the target movement area
t_i^S	The start of the computed push back sub-window for family i
t_i^F	The finish of the computed push back sub-window for family i
J	The objective function that is being maximized
t_i^{S0}	The start of the feasible push back time window for family i scheduled at $t_i = 0$
t_i^{F0}	The finish of the feasible push back time window for family i scheduled at $t_i = 0$
T_i	The set of all sampled trajectory duration data for departing trajectory family i
PB^i	The push back time of a single aircraft trajectory from family i
κ	A combination of push back times (conflict point) that lead to conflict among family i and j
K	The number of discrete clusters of conflict points
$s(\kappa)$	The silhouette value for conflict point κ
$\bar{s}$	The averaged silhouette value over all conflict points
$\mathcal{A}$	A vertex used to define quadrilateral that bounds conflict points
$\mathcal{Q}$	A Quadrilateral that bounds a set of conflict points (conflict quadrilateral)
$\vec{V}_1$	An edge that defines the conflict quadrilateral $\mathcal{Q}$
$\vec{z}$	A unit vector orthogonal to the x-y plane
$\vec{N}_1$	A normal vector generated from $\vec{V}_1 \times \vec{z}$
$\vec{T}_1^\kappa$	A vector whose tail originates from vertex $\mathcal{A}$ and tip ends at conflict point κ
$\mathbb{C}$	A clustering algorithm used to cluster a set of conflict points
$\mathbb{Q}(\mathbb{C})$	The area of all conflict quadrilaterals using clustering algorithm $\mathbb{C}$
M	A continuous variable representing the minimum push back time window
$\delta_{\min}$	A parameter representing the minimum acceptable push back time window
ϵ	A parameter in the objective function J which influences the shape of the push back time windows
z_{nE}	A binary variable that is one if edge E of the time window is selected as a separating axis, zero otherwise
z_{mE}	A binary variable that is one if edge E of the conflict quadrilateral $\mathcal{Q}$ is selected as a separating axis, zero otherwise
f_{mE}	The function that defines the line for edge E of the conflict quadrilateral $\mathcal{Q}$

A departing aircraft is parked at the gate and scheduled to arrive at the target movement area. In this paper, we assume the target movement area time is provided from a higher level taxiway scheduler. Upon receiving the push back clearance, a tug (operated by ground crew) pushes back the aircraft from the gate. At the end of the push back procedure, the aircraft stops and the tug disengages. This stop period lasts for some time during which the pilot goes through a checklist and then starts the aircraft engine(s). When ready the pilot requests taxi approval, and after the approval, taxies the aircraft until arriving at the target movement area. The target movement area is the point in space where departing aircraft transition from the ramp area into the Federal Aviation Administration (FAA) controlled taxiway and is illustrated in Fig. 1a. During the departure maneuvers the duration of the trajectory, the transitions over the motion phases, and the trajectory path are determined by human operators and are considered to be stochastic in nature.

Modeling ramp area aircraft departure maneuvers as stochastic processes, we sample a large number of departing trajectories from the stochastic model [16, 17]. The sampled trajectories define a family i of feasible trajectories that originate from a single gate and arrive at the target movement area at the scheduled

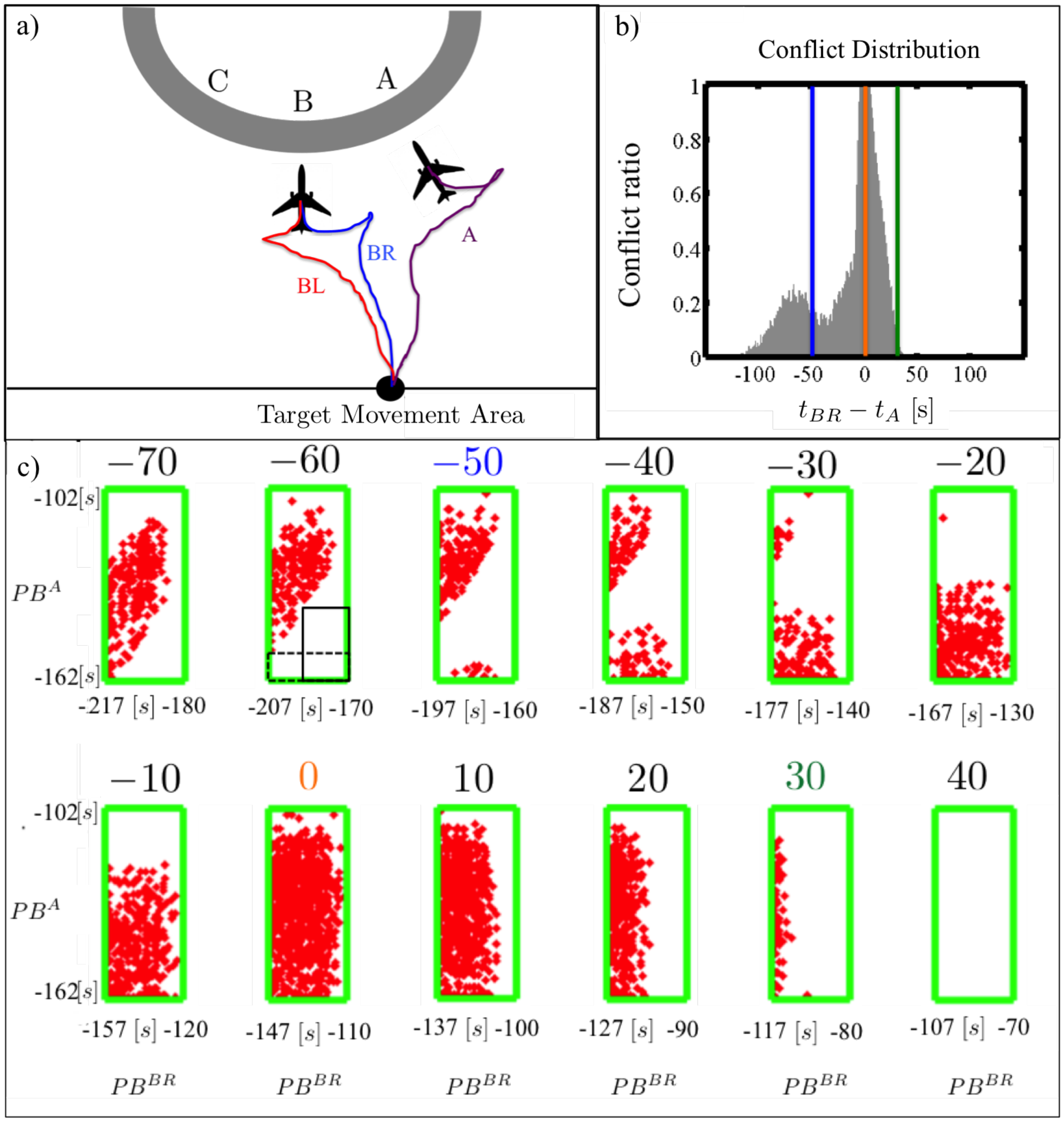

Figure 1. a) Layout of DFW ramp area and an illustration of the A, BL and BR aircraft trajectory. An individual aircraft i, j can be selected from the set of possible aircraft parked at the gate $i, j \in \{A, BL, BR\}$ b) DFW conflict distribution with select cross sections colored for aircraft $i = A$ and aircraft $j = BR$. c) Plot of combinations of push back times (red points) resulting in conflicts between aircraft $i = A$ and $j = BR$ for schedules ranging from $t_{BR} - t_A = -70$ to $t_{BR} - t_A = 40$ at a resolution of 10 [s]. The vertical axis and the horizontal axis represent the push back times of aircraft A and BR, respectively. Each red conflict point is defined as $\kappa = (PB^{BR}, PB^A)$. If we do not account for conflicts the green rectangle represents the feasible push back domain. For the schedule $t_{BR} - t_A = -60$ two feasible push back sub-windows are plotted in black solid and dotted lines.

time t_i. We sample families of trajectories for each unique push back maneuver pattern $i \in \mathbb{D}$ where the set of $\mathbb{D} = \{A, BL, BR\}$ denotes a set of all possible push back maneuver patterns illustrated in Fig. 1a. The figure shows a single trajectory from each unique maneuver pattern that is available to push back. As shown in the figure, the aircraft parked at gate B is available to push back with a left (BL) or a right (BR) push back maneuver.

Using the family of trajectories i and j, we generate a conflict ratio defined by their relative schedule $t_j - t_i$ [17]. A conflict ratio is estimated by fixing the relative schedule of the two families of trajectories and computing the ratio of conflicting trajectories to the total number of trajectories. A conflict is characterized by individual trajectories from the families i and j coming into close spatial proximity along their route. The conflict distribution is estimated by computing a conflict ratio at every whole second, as shown in Fig 1b. In the figure the y-axis represents the ratio of conflicting trajectories to conflict-free trajectories and the x-axis represents the relative target movement area schedule $t_j - t_i$.

For departing trajectory family i with scheduled target movement area time $t_i = 0$, the start of the push back time window is defined by $t_i^{S0} = -\max_i (T_i)$ and the finish of the push back time window is defined by $t_i^{F0} = -\min_i (T_i)$. The variable T_i is the set of all trajectory duration data for family i that is sampled from the stochastic model. For any given relative schedule, the earliest and latest feasible push back times define the green edges of the rectangle that are seen in Fig. 1c. The distribution in trajectory duration is estimated from the robot experiment data which is directly influenced by the human operator [16,17]. We use data from a scaled down robot experiment of the ramp area because trajectory data are not readily available mainly due to the lack of surveillance equipment in the ramp area. Investments in collecting such data are unlikely unless the usefulness of the data in increasing airport efficiency is illustrated.

For the scheduled target movement area time differences that have a non-zero ratio of conflicts, we can store and plot the combination of push back times that lead to conflicts. In Fig. 1c we fix departing trajectory family $i = A$ and family $j = BR$ and the vertical axis represents the push back time of an individual trajectory from family A, PB^A, and the horizontal axis represents the push back time of and individual trajectory from family BR, PB^{BR}. In Fig. 1b we color select cross sections of the conflict distribution to demonstrate the relationship between the ratio of conflicts (Fig. 1b) defined by the difference between their scheduled target movement area times and the set of red conflict points (Fig. 1c) defined by the combination of push back times that lead to conflicts for the given schedule.

The combinations of push back times that lead to conflicts among individual trajectories from family A and family BR are plotted (see Fig 1c) in 10s increments for the target movement area time differences ranging from $t_{BR} - t_A = -70$ to $t_{BR} - t_A = 40$. Given that we are interested in the relative scheduled difference between the two families, we fix the target movement area time of family A such that $t_A = 0$, and the relative schedule difference is defined by the target movement area time of family BR. Associated with each difference in scheduled target movement area time, e.g., $t_{BR} = -70$, is a green rectangle that is defined by the earliest and latest feasible push back times for each family such that the target movement area time schedule is satisfied. Thus, in order to satisfy the target movement area time $t_A = 0$, any individual trajectory from family A must push back within the window $PB^A \in [-162, -102]$ and to satisfy the target movement area time $t_{BR} = -70$ any individual trajectory from family BR must push back within $PB^{BR} \in [-217, -180]$. For -70 there is a set of combination of push back times that lead to conflicts. These combinations are labeled as red conflict points $\kappa = (PB^{BR}, PB^A)$ within the green rectangle (see Fig. 1c).

Consider the distribution of conflict points for the scheduled difference of -60 seen in Fig. 1c. We observe that in the bottom right of the green rectangle there is a large area that does not contain any red conflict points. If we restrict an individual trajectory from family A and family BR to push back within the lower right corner of the green rectangle then we can ensure conflict free trajectories. Two potential solutions are shown where the first solution is shown with a solid black line and the second solution with a dotted black line. Among all possible solutions, we define the optimal combination of push back time windows to be the combination where we maximize the minimum time window. By maximizing the minimum time window we compute solutions that ensure any single aircraft's push back time window is not excessively reduced in size to accommodate other aircraft. The optimization problem for aircraft trajectory families i and j is defined as

$$\max_{t_i^S, t_i^F, t_j^S, t_j^F} \quad J := \min\{t_i^F - t_i^S, t_j^F - t_j^S\} \tag{1}$$

$$\text{subject to:} \quad \forall\, \kappa = (PB^j, PB^i) : \left[PB^j \notin [t_j^S, t_j^F] \vee PB^i \notin [t_i^S, t_i^F] \right] \tag{2}$$

where the objective function J is a function of the four variables t_i^S, t_i^F, t_j^S, t_j^F which represent the start and finish of the push back sub-window for departing trajectory families i and j, respectively. The four variables together define a combination of push back sub-windows such as the window labeled with the solid (dotted) black line in Fig. 1c. The optimization problem is subject to the constraints that any given conflict point $\kappa = (PB^j, PB^i)$ can not be contained within the optimal combination of push back sub-windows. For any given relative target movement area schedule, at a resolution of 1[s], we consider computing the optimal combination of push back sub-windows that are constrained to contain zero conflict points.

III. Reducing the Number of Constraints Passed to the MILP

For a real time application we envision a scenario where an optimal taxiway scheduler such as SARDA provides the target movement area time schedule for multiple aircraft. This schedule would be updated once every ten seconds. Our algorithm should return the feasible push back time windows or an infeasible flag at the same rate. The main difficulty in solving for the optimal combination of push back time windows is that the distribution and/or the boundary of the conflict points is not known beforehand. If one were to know the linear boundaries that separate each distinct cluster of conflict points, then a mixed integer linear program could be used in real-time to solve for the optimal combination of push back sub-windows.

In general, solving for the number of distinct clusters K contained in a cloud of points and the linear boundaries that define the distinct clusters can be a challenging task. For some relative target movement area time schedules, the red conflict points form a single cluster whereas for other schedules the single cluster breaks apart into separated clusters, as seen in Fig. 1c. The well separated clusters seem to appear for schedules in-between the two modes of the conflict distribution (e.g. -50, -40 and -30 in Fig. 1c) where there is a potential mixing of two distinct sources of conflict.

We can reduce the overall number of constraints in two steps described in the subsections below. In the first step we use a clustering algorithm to convert the cloud of points into K distinct clusters. Next, in the second step each cluster is bounded within an optimal polygon. The clustering of conflict points (step 1) and the computing of linear boundaries (step 2) can be done offline and the cluster boundaries stored in memory. A small number of boundaries, i.e., the constraints, allows for computing the optimal time windows in significantly less computation time when compared to our previous method. This enables the MILP we develop in this paper to compute the push back time windows online for real-time decision making.

A. Clustering of Conflict Points

The first task is to identify the most natural number of clusters to fit to the data. For our analysis any clustering algorithm is sufficient and we implement the following routines implementing the MATLAB machine learning toolbox: (1) support vector machine (2) naive-bayes estimator (3) k-Means clustering (4) hierarchical clustering and (5) Gaussian mixture model. We also used a clustering algorithm implementing (6) a minimum spanning tree. The minimum spanning tree clustering algorithm is equivalent to the hierarchal clustering algorithm except for the original tree is built utilizing Prim's algorithm [27] while the hierarchal clustering algorithm utilizes Kruskal's algorithm [28] .

We first cluster the data into K distinct clusters. In order to interpret and validate the natural fit of the clusters with the data we use the silhouette [29] method. The silhouette method assumes that the data has been clustered via any technique into $K \geq 2$ separate clusters. For each data point κ, let $a(\kappa)$ be the average squared euclidean distance of κ to all other data points within the same cluster. We can interpret $a(\kappa)$ as how well the point κ is assigned to its own cluster (the smaller the value, the better the assignment).

Let $b(\kappa)$ be the lowest average squared euclidean distance of κ to any other cluster which κ is not a member. The cluster with the lowest average squared euclidean distance is said to be the "neighboring cluster" because it is the next best fit cluster for point κ. We now define the silhouette value $s(\kappa)$:

$$s(\kappa) = \frac{b(\kappa) - a(\kappa)}{\max\{a(\kappa), b(\kappa)\}} \tag{3}$$

which can be written as

$$s(\kappa) = \begin{cases} 1 - a(\kappa)/b(\kappa) & \text{if } a(\kappa) < b(\kappa) \\ 0 & \text{if } a(\kappa) = b(\kappa) \\ b(\kappa)/a(\kappa) - 1 & \text{if } a(\kappa) > b(\kappa) \end{cases} \tag{4}$$

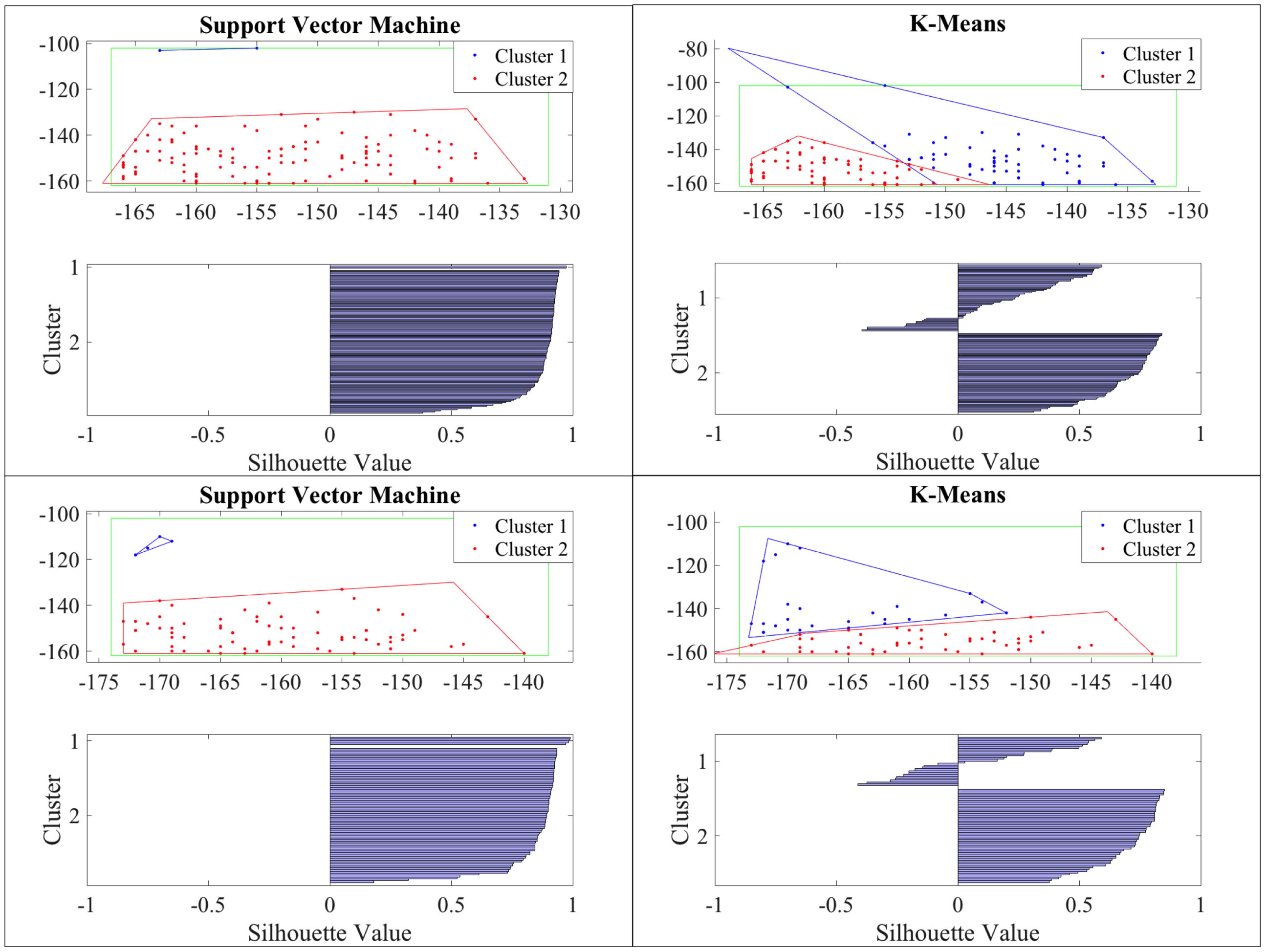

Figure 2. Clusters identified by the support vector machine and k-Means algorithm. The top row is the two methods applied to one distribution of conflict points defined by $t_j - t_i = -20$ and the bottom row is the methods applied to a second distribution of conflict points defined by $t_j - t_i = -27$. The associated silhouette value for each conflict point κ is plotted below the clusters.

From this definition we know that $-1 < s(\kappa) < 1$. For $s(\kappa)$ to be close to 1 we require $a(\kappa) << b(\kappa)$. As $a(\kappa)$ is a measure of how dissimilar κ is to its own cluster, a small value $a(\kappa)$ means it is well matched. A large $b(\kappa)$ implies that κ is badly matched to its neighboring cluster. Thus an $s(\kappa)$ close to one means that the data point κ is appropriately clustered. If $s(\kappa)$ is close to negative one, we see that κ would be more appropriate if it was clustered in its neighboring cluster. An $s(\kappa)$ near zero means that the data is on the border of two natural clusters.

Figure 2 shows the clusters identified by the support vector machine and the k-Means algorithm. We show these results to illustrate how different the identified clusters can be. The top row shows the results of the different methods applied to one distribution and the bottom row shows the results of the different methods applied to a second distribution. The associated silhouette value for each point κ is plotted below the clusters. As can be seen in the figures, the clusters for the k-Means algorithm seem less natural in comparison to the clusters identified by the support vector machine. This intuition is verified as we see the silhouette values for all points κ are greater than zero for the support vector machine whereas there exist points κ for the the k-Means algorithm that have negative silhouette values. For our application, the worst case scenario is to identify conflict points that have a large gap in space between them to the same cluster because this will cut off otherwise feasible portions of the domain.

The average $s(\kappa)$ over all points κ of a single cluster is a measure of how tightly grouped the data is within the cluster. Define $\bar{s}$ as the average $s(\kappa)$ over all data points of the entire dataset as a measure of how appropriately the data has been clustered overall. For algorithms that can fit data to more than two

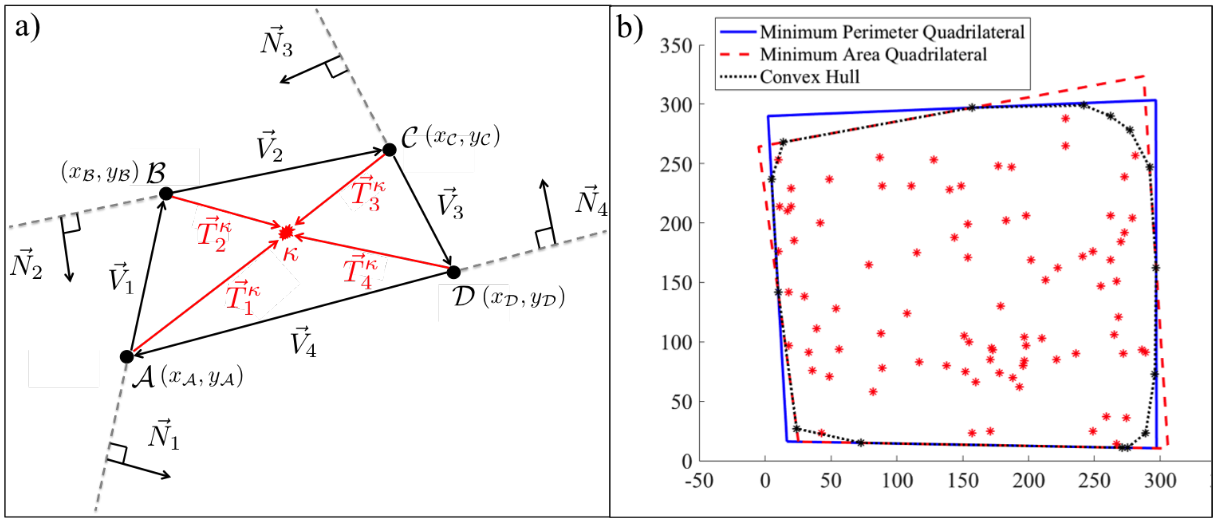
a)
$\vec{N}_3$
$\vec{V}_2$
$\mathcal{C}\,(x_\mathcal{C}, y_\mathcal{C})$
$(x_\mathcal{B}, y_\mathcal{B})\,\mathcal{B}$
$\vec{T}_3^\kappa$
$\vec{V}_3$
$\vec{N}_4$
$\vec{T}_2^\kappa$
κ
$\vec{N}_2$
$\vec{V}_1$
$\vec{T}_1^\kappa$
$\vec{T}_4^\kappa$
$\mathcal{D}\,(x_\mathcal{D}, y_\mathcal{D})$
$\vec{V}_4$
$\mathcal{A}\,(x_\mathcal{A}, y_\mathcal{A})$
$\vec{N}_1$
b)
Minimum Perimeter Quadrilateral
Minimum Area Quadrilateral
Convex Hull
350
300
250
200
150
100
50
0
-50
0
50
100
150
200
250
300

Figure 3. a) Formulation of the optimal quadrilateral. A single red conflict point $\kappa = (PB^j, PB^i)$ is shown as an example how to generate vectors $\vec{T}_1^\kappa$, $\vec{T}_2^\kappa$, $\vec{T}_3^\kappa$, and $\vec{T}_4^\kappa$. b) Solution of minimum area (red) quadrilateral, minimum perimeter (blue) quadrilateral, and convex hull (black) for a random distribution of 100 points.

clusters, we fit the data to $K = 2, 3, 4$ and 5 clusters and select the number of clusters K that provides the maximum silhouette value $\bar{s}$.

We define the optimal number of clusters as one if the averaged silhouette value $\bar{s}$ over all the data points is less than 0.8. If the value $\bar{s}$ is greater than or equal to 0.8 the optimal number of clusters is selected as the number of clusters K that maximized the silhouette value. Using this method, we compute that the optimal number of clusters is two within the interval $t_j - t_i \in [-53, -17]$ for the example in Fig. 1c and one everywhere else. The interval where we find two clusters the most natural fit is consistent with the mixing of distinct conflict sources that are seen in the conflict distribution in Fig. 1b.

B. Conflict Cluster Linear Boundaries

After associating the points to K distinct clusters, the next step is to bound each cluster in an optimal bounding polygon. We would like to bound the conflict points with the convex hull of the cluster of points. In two dimensions, the convex hull of a set of points is the minimum area convex polygon that contains the points [30]. However, each unique side of our polygon will translate into additional constraints and complexity for the MILP. Therefore we consider constraining the bounding polygon to a quadrilateral instead of the convex hull. The minimum area quadrilateral $\mathcal{Q}$ defined by vertices $\mathcal{A} = (x_\mathcal{A}, y_\mathcal{A}), \mathcal{B} = (x_\mathcal{B}, y_\mathcal{B}), \mathcal{C} = (x_\mathcal{C}, y_\mathcal{C})$ and $\mathcal{D} = (x_\mathcal{D}, y_\mathcal{D})$ labelled in a clockwise fashion is formulated as the non-linear optimization problem [26,31]:

$$\min_{A,B,C,D} \quad J := (\vec{\mathcal{AC}} \times \vec{\mathcal{BD}})^2 \tag{5}$$

$$\text{subject to:} \quad \vec{N}_1 \cdot \vec{T}_1^\kappa \geq 0 \tag{6}$$

$$\vec{N}_2 \cdot \vec{T}_2^\kappa \geq 0 \tag{7}$$

$$\vec{N}_3 \cdot \vec{T}_3^\kappa \geq 0 \tag{8}$$

$$\vec{N}_4 \cdot \vec{T}_4^\kappa \geq 0 \tag{9}$$

Define $\vec{V}_1$ as the vector whose tail originates at vertex $\mathcal{A}$ and tip terminates at Vertex $\mathcal{B}$ as seen Fig. 3a. Vectors $\vec{V}_2$, $\vec{V}_3$ and $\vec{V}_4$ are defined in a similar fashion between vertices $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ respectively and $\vec{N}_1$ is the normal vector to $\vec{V}_1$ generated by $\vec{V}_1 \times \vec{z}$. For each conflict point κ we define vector $\vec{T}_1^\kappa$ whose tail originates at vertex A and tip terminates at the conflict point κ. Vectors $\vec{T}_2^\kappa$, $\vec{T}_3^\kappa$ and $\vec{T}_4^\kappa$ are defined in the same way with their tails originating from vertices $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{D}$ respectively. Expression (5) is the objective function to minimize the square of the scalar value associated with the cross product of the vector $\vec{\mathcal{AC}} \times \vec{\mathcal{BD}}$, see Fig. 3a. In two dimensions the absolute value of the cross product can be interpreted as twice the area of the quadrilateral defined by vertices $\mathcal{A}, \mathcal{B}, \mathcal{C}$ and $\mathcal{D}$.

The four constraints (6) - (9) should be generated for each conflict point $\kappa = (PB^j, PB^i)$. Enforcing constraint (6) ensures that the conflict point κ and the normal vector $\vec{N}_1$ are on the same side of vector $\vec{V}_1$, i.e. the conflict point κ is to the right of vector $\vec{V}_1$. Together the four constraints (6) - (9) ensure that the conflict point is to the right of every vector $\vec{V}_1$, $\vec{V}_2$, $\vec{V}_3$ and $\vec{V}_4$ and therefore inside the quadrilateral. Expanding the objective function we have:

$$\begin{aligned} J :=& \Big[\vec{\mathcal{AC}} \times \vec{\mathcal{BD}}\Big]^2 \\ =& \Big[(x_\mathcal{C} - x_\mathcal{A})(y_\mathcal{D} - y_\mathcal{B}) - (x_\mathcal{D} - x_\mathcal{B})(y_\mathcal{C} - y_\mathcal{A})\Big]^2 \end{aligned}$$

Similarly expanding the constraints we get:

$$\begin{aligned} \vec{N}_1 \cdot \vec{T}_1^\kappa =& (y_\mathcal{B} - y_\mathcal{A})(PB^j - x_\mathcal{A}) - (x_\mathcal{B} - x_\mathcal{A})(PB^i - y_\mathcal{A}) \\ =& PB^j y_\mathcal{B} - x_\mathcal{A} y_\mathcal{B} - PB^j y_\mathcal{A} + x_\mathcal{A} y_\mathcal{A} - x_\mathcal{B} PB^i + x_\mathcal{B} y_\mathcal{A} + x_\mathcal{A} PB^i - x_\mathcal{A} y_\mathcal{A} \\ =& \Big[x_\mathcal{B} y_\mathcal{A} - x_\mathcal{A} y_\mathcal{B}\Big] + x_\mathcal{A} PB^i - PB^j y_\mathcal{A} - x_\mathcal{B} PB^i + PB^j y_\mathcal{B} \\ =& \Big[x_\mathcal{B} y_\mathcal{A} - x_\mathcal{A} y_\mathcal{B}\Big] + PB^i(x_\mathcal{A} - x_\mathcal{B}) + PB^j(y_\mathcal{B} - y_\mathcal{A}) \\ \vec{N}_2 \cdot \vec{T}_2^\kappa =& \Big[x_\mathcal{C} y_\mathcal{B} - x_\mathcal{B} y_\mathcal{C}\Big] + PB^i(x_\mathcal{B} - x_\mathcal{C}) + PB^j(y_\mathcal{C} - y_\mathcal{B}) \\ \vec{N}_3 \cdot \vec{T}_3^\kappa =& \Big[x_\mathcal{D} y_\mathcal{C} - x_\mathcal{C} y_\mathcal{D}\Big] + PB^i(x_\mathcal{C} - x_\mathcal{D}) + PB^j(y_\mathcal{D} - y_\mathcal{C}) \\ \vec{N}_4 \cdot \vec{T}_4^\kappa =& \Big[x_\mathcal{A} y_\mathcal{D} - x_\mathcal{D} y_\mathcal{A}\Big] + PB^i(x_\mathcal{D} - x_\mathcal{A}) + PB^j(y_\mathcal{A} - y_\mathcal{D}) \end{aligned}$$

The minimum perimeter quadrilateral is formulated as a quadratically constrained quadratic program [32,33]

$$\min_{A,B,C,D} \quad J := (x_\mathcal{B} - x_\mathcal{A})^2 + (y_\mathcal{B} - y_\mathcal{A})^2 + (x_\mathcal{C} - x_\mathcal{B})^2 + (y_\mathcal{C} - y_\mathcal{B})^2 + \qquad (10)$$

$$(x_\mathcal{D} - x_\mathcal{C})^2 + (y_\mathcal{D} - y_\mathcal{C})^2 + (x_\mathcal{A} - x_\mathcal{D})^2 + (y_\mathcal{A} - y_\mathcal{D})^2$$

$$\text{subject to:} \quad \vec{N}_1 \cdot \vec{T}_1^\kappa \geq 0 \qquad (11)$$

$$\vec{N}_2 \cdot \vec{T}_2^\kappa \geq 0 \qquad (12)$$

$$\vec{N}_3 \cdot \vec{T}_3^\kappa \geq 0 \qquad (13)$$

$$\vec{N}_4 \cdot \vec{T}_4^\kappa \geq 0 \qquad (14)$$

where expression (10) is the objective function to minimize the perimeter of the quadrilateral defined by vertices $\mathcal{A}, \mathcal{B}, \mathcal{C}$ and $\mathcal{D}$. The four constraints (11) - (14) should be generated for each conflict point κ, and together they ensure that the conflict point is contained within the optimal bounding quadrilateral $\mathcal{Q}$. Figure 3b illustrates the solution of the minimum area quadrilateral with a red dashed line, the minimum perimeter quadrilateral with a blue line and the convex hull with a black line for a random distribution of conflict points.

Let us define the optimal clustering algorithm as the algorithm that minimizes the summation of quadrilateral area (perimeter) over all possible schedules

$$\text{Opt Algorithm } = \min_{\mathbb{C}} \Sigma_k \mathbb{Q}(\mathbb{C}_k) \qquad (15)$$

where $\mathbb{Q}(\mathbb{C}_k)$ is the area of *all* optimal bounding quadrilaterals $\mathcal{Q}$ using cluster algorithm $\mathbb{C}$ for the relative schedule defined by $t_j - t_i = k$. We apply this definition to compute the total area $\mathbb{Q}$ of the optimal bounding quadrilaterals for each algorithm for each relative schedule. Then for each algorithm the total area is summed over all possible relative schedules. Figure 4 illustrates the performance of the clustering algorithms from best to worst: support vector machine, hierarchical / minimum spanning tree, naive bayes, k-Means and lastly Gaussian mixture model.

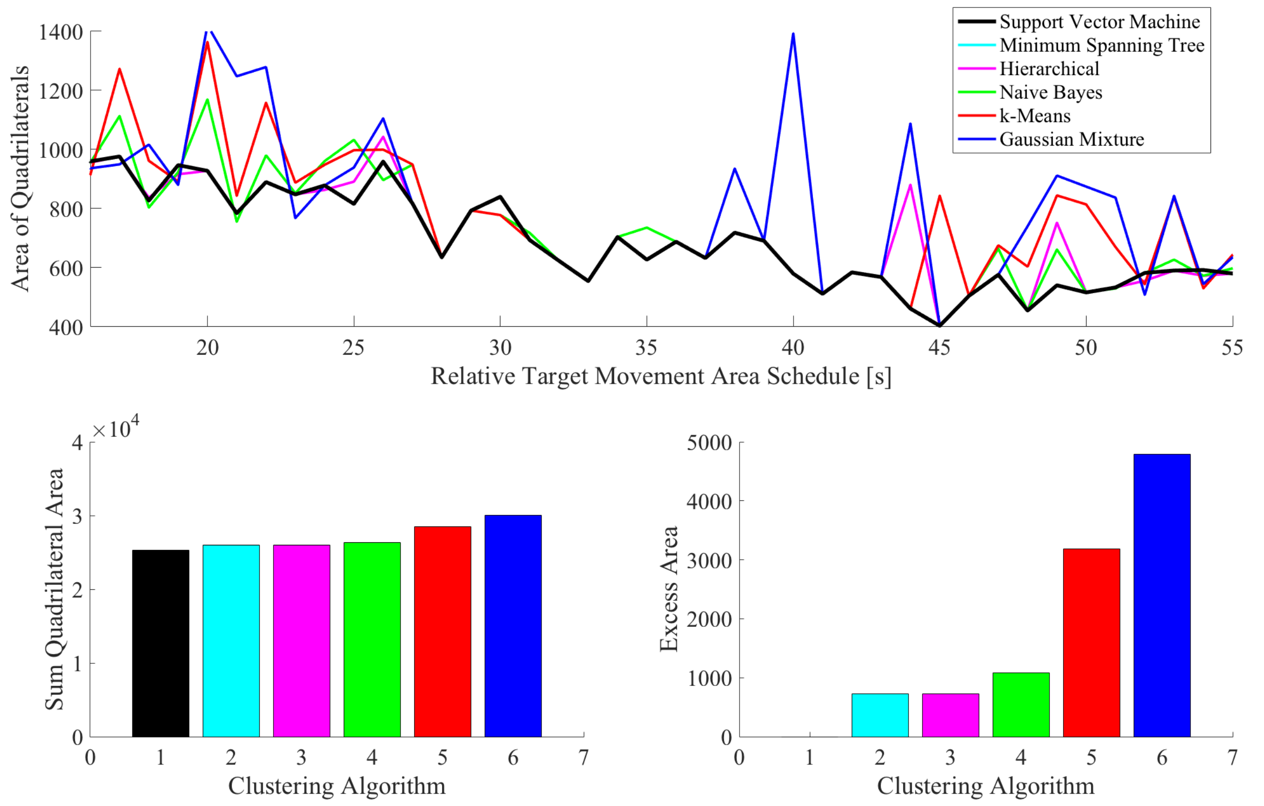

Figure 4. Top: The area of the bounding quadrilaterals for the different clustering algorithms as a function of the relative schedule. Bottom: The summation of total area of all the optimal bounding quadrilaterals.

IV. MILP for Real-Time Computing the Optimal Time Windows

After identifying the linear boundaries of each cluster, we use the boundaries to solve for the optimal combination of push back sub-windows. To ensure that our time windows do not intersect our conflict quadrilaterals we apply the separating axis theorem. The theorem states that in two dimensions, two convex polygons do not intersect if and only if one of the axis of one of the polygons is a separating axis [34].

This implies that if we are given a convex n-gon and a convex m-gon, we can build a MILP that requires $n+m+1$ constraints to ensure the polygons do not intersect. There will be $m+n$ linear inequality constraints; satisfying any one of these particular constraints ensures that the unique edge of the polygon associated with that constraint is a separating axis. The extra constraint will ensure that a minimum of one edge of one polygon is indeed a separating axis.

Here we formulate the MILP that will be used to separate the optimal push back sub-windows from the conflict quadrilaterals. Given two departing aircraft trajectory families $i, j \in \mathbb{D}$, the objective function is defined as

$$\max_{t_i^S, t_i^F, t_j^S, t_j^F} \quad J := \left[M + \epsilon(t_i^F - t_i^S + t_j^F - t_j^S)\right] \tag{16}$$

where the continous value M is equivalent to the minimum time window among both aircraft i and j and ϵ is a scalar valued parameter. For departing aircraft trajectory families $i, j \in \mathbb{D}$ we introduce the two constraints

$$t_i^F - t_i^S - M \geq 0 \tag{17}$$

$$t_j^F - t_j^S - M \geq 0 \tag{18}$$

that ensure the push back time window for family i and the push back time window for family j are both greater than the minimum time window M. We note that the value M is not a fixed value, but a continuous variable passed to the model that is solved for in the optimization problem.

Similarly, for departing aircraft trajectory family $i, j \in \mathbb{D}$ we introduce the two constraints

$$t_i^F - t_i^S - \delta_{min} \geq 0 \tag{19}$$

$$t_j^F - t_j^S - \delta_{min} \geq 0 \tag{20}$$

that ensure the push back time windows for family i and j are both larger than a predefined value δ_{min}. The value δ_{min} is the minimum acceptable push back time window. For example, pilots and ramp controllers could find a schedule that requires aircraft to initiate push back within a time window of 5 seconds too restrictive to consistently execute. In this paper we use the value $\delta_{min} = 25[s]$ when solving for the optimal sub-windows. The correct value should be determined in conjunction by pilots and ramp controllers.

For departing aircraft trajectory family $i, j \in \mathbb{D}$ we introduce the four constraints

$$t_i^S - t_i - t_i^{S0} \geq 0 \tag{21}$$

$$t_i^F - t_i - t_i^{F0} \leq 0 \tag{22}$$

$$t_j^S - t_j - t_j^{S0} \geq 0 \tag{23}$$

$$t_j^F - t_j - t_j^{F0} \leq 0 \tag{24}$$

Constraints (21) - (24) ensure that for any given combination of target movement time schedules, given by t_i and t_j, the start and end of the push back sub-windows defined by t^S and t^F must be within the bounds defined by the earliest and latest feasible push back times. This implies that there exists a feasible trajectory from family i and j that meets the scheduled target movement area times t_i and t_j without accounting for conflicts. These four constraints describe that the push back time windows that we solve for, which are illustrated in black solid (dotted) lines in Fig. 1, are proper sub-windows of the original green rectangle.

Our problem is defined by a time window with $n = 4$ edges and a quadrilateral with $m = 4$ edges, giving us a total of $m + n = 8$ edges to select from when choosing a separating axis. To separate the optimal time window from a single conflict quadrilateral $\mathcal{Q}$, we introduce the additional nine constraints

$$z_{n1}(t_i^S - \max[y_{\mathcal{Q}}]) \geq 0 \tag{25}$$

$$z_{n2}(t_i^F - \min[y_{\mathcal{Q}}]) \leq 0 \tag{26}$$

$$z_{n3}(t_j^S - \max[x_{\mathcal{Q}}]) \geq 0 \tag{27}$$

$$z_{n4}(t_j^F - \min[x_{\mathcal{Q}}]) \leq 0 \tag{28}$$

$$z_{m1}(f_{m1}) \leq 0 \tag{29}$$

$$z_{m2}(f_{m2}) \leq 0 \tag{30}$$

$$z_{m3}(f_{m3}) \leq 0 \tag{31}$$

$$z_{m4}(f_{m4}) \leq 0 \tag{32}$$

$$z_{n1} + z_{n2} + z_{n3} + z_{n4} + z_{m1} + z_{m2} + z_{m3} + z_{m4} = 1 \tag{33}$$

The variable z_{nE} is a binary variable that is defined as $z_{nE} = 1$ if edge E of the optimal time window is selected as the separating axis, else $z_{nE} = 0$. The variable z_{mE} is a binary variable that is defined as $z_{mE} = 1$ if edge E of the conflict quadrilateral is selected as the separating axis, else $z_{mE} = 0$. The numbering of the edges $nE, mE \in [1, 2, 3, 4]$ of the time window and the conflict quadrilateral is not important as long as each edge has a unique number. Expression (33) ensures that one edge of the optimal time window or one edge of the conflict quadrilateral must be a separating axis.

Expression (25) constrains the bottom edge of the optimal time window to be above the maximum y value of the conflict quadrilateral $\mathcal{Q}$. This provides that the bottom edge of the time window is a separating axis and the quadrilaterals do not intersect. Similarly, Expression (26) constrains the top edge of the optimal time window to be below the minimum y value of the conflict quadrilateral $\mathcal{Q}$. This provides that the top edge of the time window is a separating axis. Expressions (27-28) apply similar reasoning to ensure that the left and right edge of the time window is selected as a separating axis.

Whereas expressions used in (25-28) do not change depending upon the conflict quadrilateral $\mathcal{Q}$, equations (29-32) are dependent upon the unique edges that define the conflict quadrilateral. Select any unique edge mE of $\mathcal{Q}$, to generate the function f_{mE} we need to obtain two pieces of information. First, we need to know if we should constrain the optimal time window above or below this edge, and second, we need the equation of the line that defines edge mE, $\mathcal{Y} = \mathcal{M}\mathcal{X} + \mathcal{B}$.

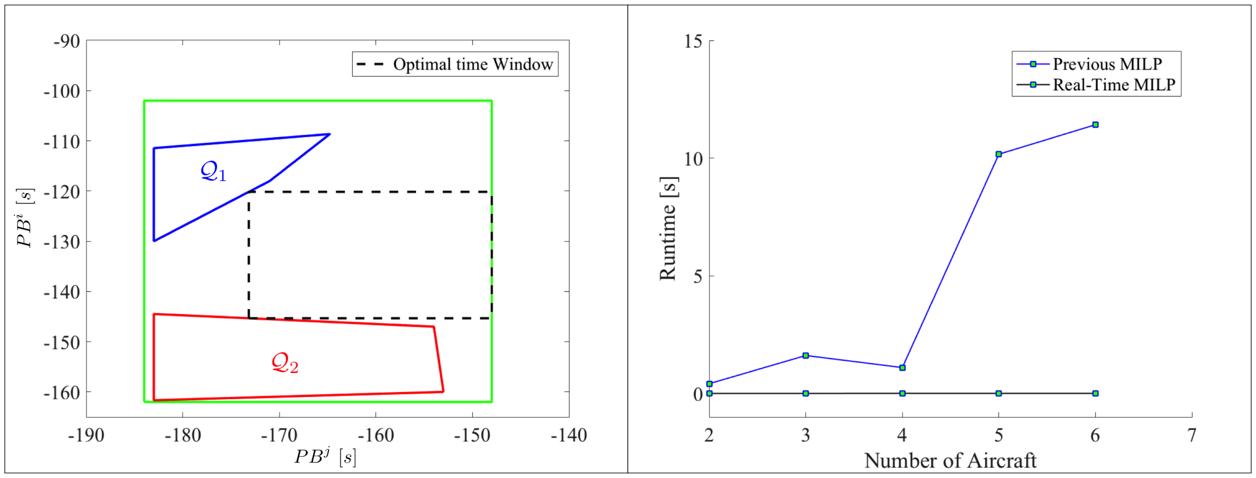

Figure 5. a) Optimal sub-windows using the two conflict quadrilaterals as constraints. b) The reduction in computation time that is achieved by reducing the number of constraints for schedules of $n = 4, 5, 6$ aircraft. The blue line is the MILP defined in our previous paper [25] and the black line is the real-time MILP defined in this paper.

Given these two pieces of information we can generate the functions that ensure the edges of the quadrilateral are a separating axis as

$$f_{mE} = \begin{cases} -t_i^S + \mathcal{M}t_j^F + \mathcal{B} & \text{if } \mathcal{M} \geq 0 \text{ AND above} \\ t_i^F - \mathcal{M}t_j^S - \mathcal{B} & \text{if } \mathcal{M} \geq 0 \text{ AND below} \\ -t_i^S + \mathcal{M}t_j^S + \mathcal{B} & \text{if } \mathcal{M} \leq 0 \text{ AND above} \\ t_i^F - \mathcal{M}t_j^F - \mathcal{B} & \text{if } \mathcal{M} \leq 0 \text{ AND below} \end{cases} \tag{34}$$

The program defined by objective (16) and constraints (17-33) solves for the optimal time window that is constrained to not intersect the conflict quadrilateral $\mathcal{Q}$. For every additional conflict quadrilateral, the nine constraints (25-33) are required to separate the time window. These non-linear constraints can be linearized [25, 35, 36] and the resulting MILP can be solved using the Gurobi Optimizer [37] libraries.

Figure 5a shows the optimal time window solution of the program given the two conflict quadrilaterals shown in blue and red. Figure 5b shows the execution time for schedules of $n = 4, 5, 6$ aircraft for the new MILP compared to the MILP that was defined in our previous paper [25]. As can be seen in the figure, the improvement in computation time is significant. Whereas the previous MILP took roughly 10 seconds to solve for schedules of five aircraft, the MILP developed in this paper can solve for the optimal time windows of five aircraft in less than 0.5 seconds.

V. Analysis of Objective Function

Consider the objective function

$$\max_{t_i^S, t_i^F, t_j^S, t_j^F} \qquad J := \left[M + \epsilon(t_i^F - t_i^S + t_j^F - t_j^S)\right] \tag{35}$$

This objective function formulates a multi-criterion optimization problem. If we set the value of ϵ to zero, we solve for the combination of push back sub-windows where the minimum time window is maximized. In contrast, if we set the value of ϵ sufficiently large, then we solve for the the combination of push back sub-windows where the summation of all push back time windows is maximized.

Figure 6a illustrates the solution of the objective function where ϵ is set to zero. The red and blue parts of the domain represent the conflict quadrilaterals. In Fig. 6b a solution where ϵ is sufficiently large is shown for the same input. Solutions can be quite different depending upon the value of ϵ as can be seen in the figure.

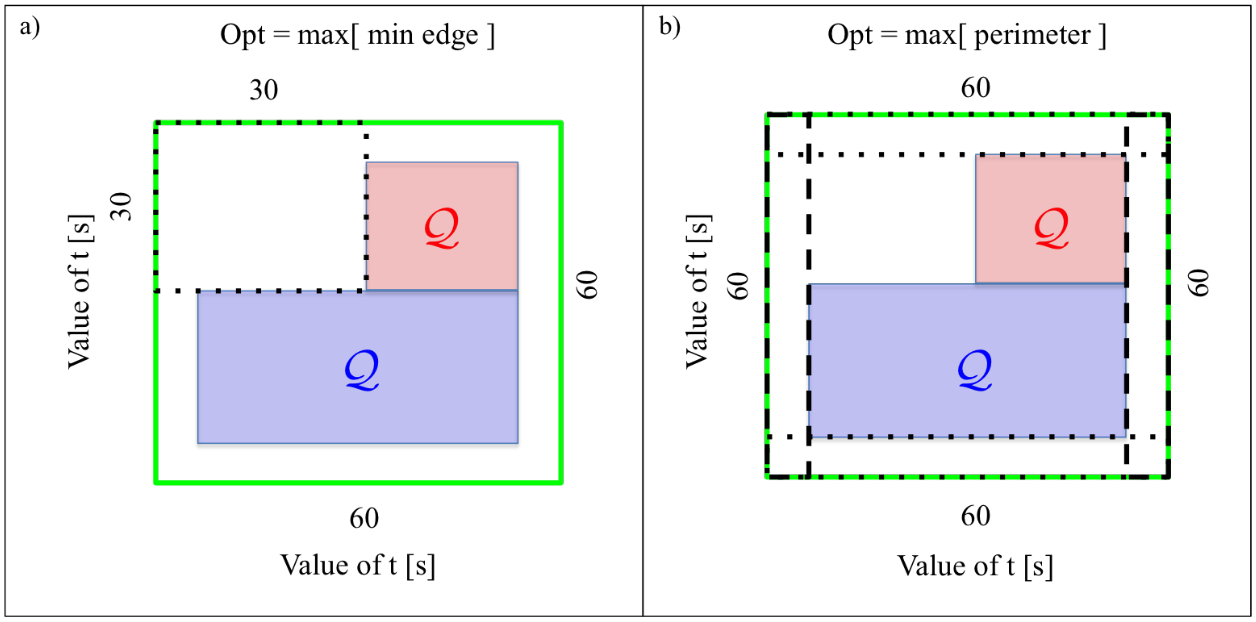

Figure 6. a) The optimal time window defined to maximize the minimum edge is illustrated with a dotted black line. b) The optimal time window defined to maximize the perimeter is not unique. There are four time windows with equivalent perimeter illustrated. Two solutions are represented as tall and skinny time windows with dashed black lines and two solutions are illustrated as short and wide time windows with dotted black lines.

The number of aircraft that we solve for in combination with the value of ϵ can also influence the quality of solution. For example, if we limit ourselves to solutions of two aircraft i and j, the most intuitive solutions are generated with small values of ϵ. This can be seen in Fig. 6 where the maximum perimeter solution (Fig. 6b) for the given input seems quite unappealing compared to the maximum minimum edge solution (Fig. 6a). We believe the solutions seen in Fig. 6b are unappealing compared to the solution in Fig. 6a because the maximum perimeter solutions excessively reduce a single aircraft's push back time window to accommodate the other aircraft's large push back time window.

The inverse is true, however, for schedules of multiple aircraft. If we are to solve for a schedule of 4 aircraft using an objective defined to maximize the minimum edge, then it is easy to generate unappealing solutions. This occurs because reducing a push back time window for a single aircraft can have a dramatic affect on all aircraft, as seen in the black dashed line solutions of Fig. 7. Maximizing the minimum edge will return a solution where the push back time windows for all the aircraft are equal to the minimum push back time window. If we solve the same schedule of 4 aircraft with the objective defined to maximize the summation of push back time windows (equivalent to the perimeter), then the model will try to find solutions where it maximizes the push back time windows of all aircraft systematically, as seen in the grey dotted line solutions of Fig. 7. When we solve for multiple aircraft together, we find solutions that maximize the minimum edge unappealing because a single aircraft's push back time window can dramatically affect the other push back time windows.

In addition to influencing the quality of solution, the value of ϵ has an impact on the computation time. We find that smaller values of ϵ help Gurobi solve the problem in less time. Consider Fig. 8 which shows the computation time as a function of the number of conflict domains that we solve for. A conflict domain is a single pairwise conflict that is being solved for. When we solve for multiple aircraft the coupled conflict domains must be solved together simultaneously to ensure optimal solutions. In this figure we report the computation time using the linear boundaries defined from the conflict quadrilaterals and also report the computation time using the linear boundaries defined by the convex hull. For both approaches, the figure shows that a smaller value of epsilon reduces computation time. As expected, the MILP based on the linear boundaries of the conflict quadrilateral outperforms the MILP based on the linear boundaries of the convex hull regardless of the value of epsilon.

These results show that there is a tradeoff between the quality of the optimal time window solutions and the value of ϵ. When solving for multiple aircraft with value of $\epsilon = 0$, then the solver simply has to find a feasible solution where the minimum edge is maximized. All other edges can be equivalent to the minimum

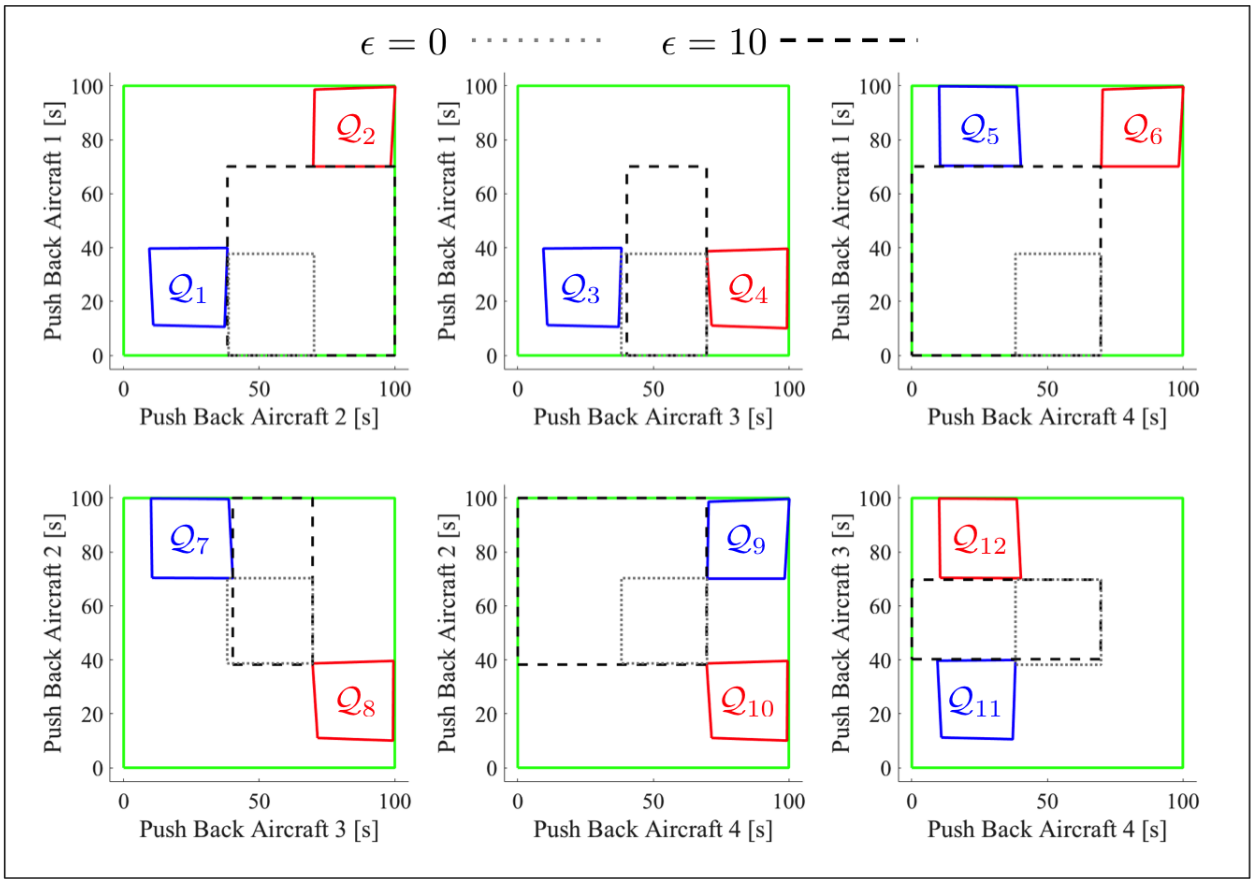

Figure 7. Solutions using a value $\epsilon = 0$ (minimum time window maximized) are shown with a grey dotted line for a schedule of 4 aircraft. Solutions using a value $\epsilon = 10$ (summation of time windows maximized) are shown with a black dashed line for the same input. In each sub-figure the vertical represents the push back time window of aircraft i, PB^i, and the horizontal axis represents the push back time window of aircraft j, PB^j.

edge, generating a relatively unappealing solution as seen in the grey dotted line solutions in Fig. 7. Instead we prefer the model to stretch out the feasible time windows to maximize the time window perimeter. This however, is not a straightforward task, as solving the perimeter problem can be computationally expensive and the solutions for two aircraft can generate relatively unappealing combinations of push back time windows such as the ones seen in Fig. 6b.

VI. Conclusions and Future Work

In this paper, we formulated a MILP to solve for the optimal combination of push back time sub-windows. The main contribution of the MILP is to compute in real-time the push back time sub-windows that allow for aircraft to taxi unimpeded from their gate to the departure runway queue in the presence of other aircraft and trajectory uncertainties. This allows for ramp controllers to better manage surface traffic, reduce fuel consumption, and execute conflict free ramp area aircraft trajectories. The MILP is designed for real-time decision making so the computational runtime is a critical component of the tool.

The MILP formulation is based on a small number of constraints to ensure that its solution can be computed at speeds compatible with the NASA Ames SARDA scheduler. The small number of constraints results from the two-step processing. We first clustered the conflict points and then bounded each cluster of conflict points with linear boundaries. The linear cluster boundaries were used in the MILP where we exploited the separating axis theorem to ensure the push back time sub-windows do not intersect the conflict quadrilaterals. Solutions were provided and the objective function was analyzed to reveal the dependence of the parameter ϵ in the quality of solution and the computation time.

Now that we understand how to exploit the structure of the conflict point distribution, we can integrate the information of the size of the push back time window into the logic that schedules aircraft at the target

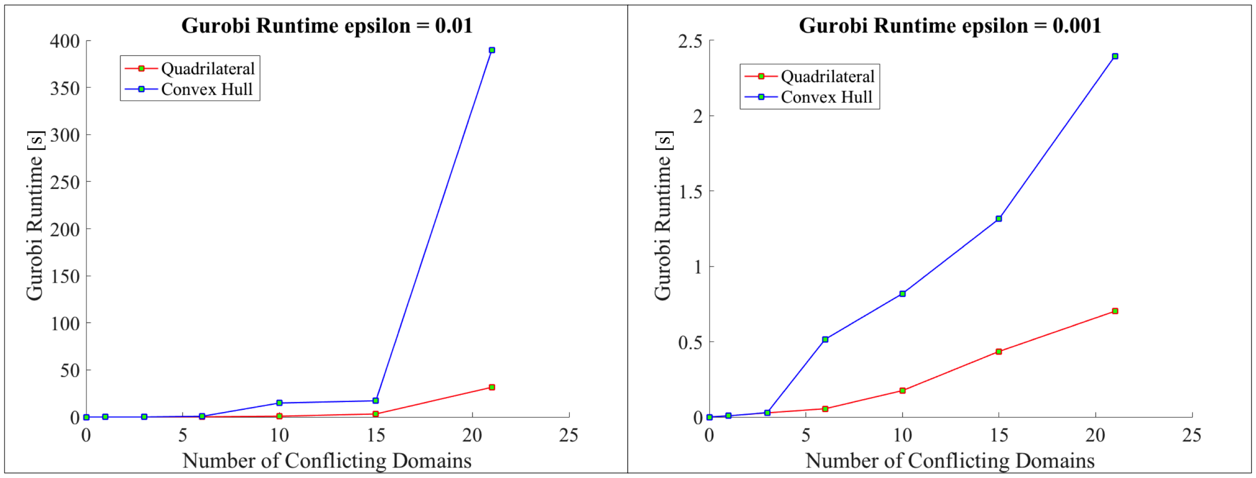

Figure 8. a) Computation time of the MILP with $\epsilon = 0.01$. The red line illustrates the computation time using the minimum area quadrilateral linear boundaries and the blue line illustrates the computation time using the convex hull linear boundaries. b) Computation time of the MILP with $\epsilon = 0.001$. The red line illustrates the computation time using the minimum area quadrilateral linear boundaries and the blue line illustrates the computation time using the convex hull linear boundaries.

movement area. There is a likely tradeoff between the throughput of the schedule and the size of the push back time windows. A MILP approach using a multi-criterion objective function could account for this tradeoff. This can help with the optimal planning and execution of surface operations from the gates all the way to the departure runways.

References

[1]Smeltink, J. W., Sooner, M. J., de Waal, P. R., and van der Mei, R. D., "An Optimisation Model for Airport Taxi Scheduling," *INFORMS Annual Meeting*, 2004.

[2]Herrero, J. G., Berlanda, A., Molina, J. M., and Casar, J. R., "Methods for Operations Planning in Airport Decision Support Systems," *Appl Intell*, Vol. 22, 2005, pp. 183–206.

[3]Marín, A., "Airport Management: Taxi Planning," *Ann Oper Res*, Vol. 143, 2007, pp. 191–202.

[4]Balakrishnan, H. and Jung, Y., "A Framework for Coordinated Surface Operations Planning at DFW International Airport," *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2007.

[5]Rathinam, S., Montoya, J., and Jung, Y., "An optimization model for reducing aircraft taxi times at the DFW international airpiort," *26th International Conference of the Aeronautical Sciences*, 2008.

[6]Roling, P. C. and Visser, H. G., "Optimal Airport Surface Traffic Planning Using Mixed-integer Linear Programming," *Int J Aerospace Eng*, Vol. 2008, 2008, pp. 1–12.

[7]Gupta, G., Malik, W., and Jung, Y. C., "A Mixed Integer Linear Program for Airport Departure Scheduling," *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, 2009.

[8]Malik, W., Gupta, G., and Jung, Y. C., "Managing Departure Aircraft Release For Efficient Airport Surface Operations," *Proceedings of the AIAA Guidance, Navigation, and Control (GNC) Conference*, 2010.

[9]Gupta, G., Malik, W., and Jung, Y. C., "Incorporating Active Runway Crossings in Airport Departure Scheduling," *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2010.

[10]Malik, W., Gupta, G., and Jung, Y. C., "Spot Release Planner: Efficient Solution for Detailed Airport Surface Trffic Optimization," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, 2012.

[11]Clare, G. L. and Richards, A. G., "Optimization of Taxiway Routing and Runway Scheduling," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, No. 4, 2011, pp. 1000–1013.

[12]Anderson, R. and Milutinović, D., "An Approach to Optimization of Airport Taxiway Scheduling and Traversal Under Uncertainty," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 227, No. 2, February 2013, pp. 273–284.

[13]Jung, Y., Hoang, T., Montoya, J., Gupta, G., Malik, W., L., T., and Wang, H., "Performance Evaluation of a Surface Traffic Management Tool for Dallas/Fort Worth International Airport," *9th USA/Europe Air Traffic Management Research and Development Seminar*, 2011, pp. 1–10.

[14]Lee, H. and Balakrishnan, H., "Optimization of Airport Taxiway Operations at Detroit Metropolitan Airport (DTW)." *In AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, 2010.

[15]Lee, H. and Balakrishnan, H., "A Comparison of Two Optimization Approaches for Airport Taxiway and Runway Scheduling." *In Digital Avionics Systems Conference*, 2012.

[16]Coupe, W. J., Milutinović, D., Malik, W., Gupta, G., and Jung, Y., "Robot Experiment Analysis of Airport Ramp Area Time Constraints," *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2013.

[17]Coupe, W. J., Milutinović, D., Malik, W., and Jung, Y., "Integration of Uncertain Ramp Area Aircraft Trajectories and Generation of Optimal Taxiway Schedules at Charlotte Douglas (CLT) Airport," *AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, 2015.

[18]Wolsey, L. A. and Nemhauser, G. L., *Integer and Combinatorial Optimization*, John Wiley & Sons, 2014.

[19]Nemhauser, G. L., "The Age of Optimization: Solving Large-scale Real-world Problems," *Operations Research*, Vol. 42, No. 1, 1994, pp. 5–13.

[20]Klotz, E. and Newman, A. M., "Practical Guidelines for Solving Difficult Mixed Integer Linear Programs," *Surveys in Operations Research and Management Science*, Vol. 18, No. 1, 2013, pp. 18–32.

[21]Jung, Y. C., Hoang, T., Montoya, J., Gupta, G., Malik, W., and Tobias, L., "A Concept and Implementation of Optimized Operations of Airport Surface Traffic," *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Fort Worth, TX*, 2010.

[22]Gupta, G., Malik, W., and Jung, Y. C., "An Integrated Collaborative Decision Making and Tactical Advisory Concept for Airport Surface Operations Management," *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, Indianapolis, IN*, 2012.

[23]Gupta, G., Malik, W., Tobias, L., Jung, Y., Hoang, T., and Hayashi, M., "Performance Evaluation of Individual Aircraft Based Advisory Concept for Surface Management," *Tenth USA/Europe Air Traffic Management Research and Development Seminar (ATM2013)*, Chicago, Il, 2013.

[24]Nikoleris, T., Gupta, G., and Kistler, M., "Detailed Estimation of Fuel Consumption and Emissions During Aircraft Taxi Operations at Dallas Fort Worth International Airport," *Published in the journal Transportation Research Part D: Transport and Environment, Vol. 16D, Issue 4*, June 2011.

[25]Coupe, W. J., Milutinović, D., Malik, W., and Jung, Y., "Optimization of Push Back Time Windows That Ensure Conflict Free Ramp Area Aircraft Trajectories," *AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, 2015.

[26]Bertsekas, D. P., "Nonlinear Programming," 1999.

[27]Prim, R. C., "Shortest Connection Networks and Some Generalizations," *Bell system technical journal*, Vol. 36, No. 6, 1957, pp. 1389–1401.

[28]Kruskal, J. B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem," *Proceedings of the American Mathematical society*, Vol. 7, No. 1, 1956, pp. 48–50.

[29]Rousseeuw, P., "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," *J. Comput. Appl. Math.*, Vol. 20, No. 1, Nov. 1987, pp. 53–65.

[30]Barber, C. B., Dobkin, D. P., and Huhdanpaa, H., "The Quickhull Algorithm for Convex Hulls," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 22, No. 4, 1996, pp. 469–483.

[31]Vavasis, S. A., *Nonlinear Optimization: Complexity Issues*, Oxford University Press, Inc., 1991.

[32]Frank, M. and Wolfe, P., "An Algorithm for Quadratic Programming," *Naval research logistics quarterly*, Vol. 3, No. 1-2, 1956, pp. 95–110.

[33]Pardalos, P. M. and Vavasis, S. A., "Quadratic Programming With One Negative Eigenvalue is NP-hard," *Journal of Global Optimization*, Vol. 1, No. 1, 1991, pp. 15–22.

[34]Gottschalk, S., Lin, M. C., and Manocha, D., "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 1996, pp. 171–180.

[35]Glover, F., "Improved Linear Integer Programming Formulations of Nonlinear Integer Problems," *Management Science*, Vol. 22, No. 4, 1975, pp. 455–460.

[36]Adams, W. P. and Forrester, R. J., "Linear Forms of Nonlinear Expressions," *Operations Research Letters*, Vol. 35, No. 4, 2007, pp. 510–518.

[37]Gurobi Optimization, Inc., "Gurobi Optimizer Reference Manual," 2015.