Analysis of Centralized and Distributed Air Traffic Management Systems via Mixed Integer Linear Programs

Rex K. Kincaid, Cameron Curtis
Dept. of Mathematics, William & Mary
Williamsburg, VA 23187-8795
rrkinc@wm.edu, ccurtis@wm.edu

Logan Wolf
Dept. of Mathematics, William & Mary
Williamsburg, VA 23187-8795
lbwolf@wm.edu

ABSTRACT

Air traffic control, so that aircraft avoid collisions and reach their destinations efficiently, is a complex problem. Centralized control and distributed control are the two most common strategies for managing air traffic. To fully understand the key differences between the two concepts, we focus on the control problem abstractly, as an optimization problem. Mixed integer linear programs (MILPs) are used to model the two control strategies. The performance of the MILPs is compared on simulated data for up to 30 planes needing to resolve conflicts in a 2-dimensional airspace. Results indicate the strengths and weaknesses of each model rather than an individual algorithmic approach.

ABOUT

Rex K. Kincaid is Chancellor Professor of Mathematics at William & Mary. His B.A. in Mathematics is from DePauw University while his M.S. degree in Applied Mathematics and Ph.D. in Operations Research are from Purdue University. His research interests include network location theory and metaheuristics for discrete optimization problems. He has published more than 90 peer-reviewed research articles with more than 40 undergraduate and M.S. degree students as co-authors.

Cameron Curtis is an undergraduate mathematics major at William & Mary graduating in May of 2023.

Logan Wolf is an undergraduate mathematics major at William & Mary graduating in May of 2023. He intends to continue his education at William & Mary with an M.S. in Computer Science with a specialization in Computational Operations Research. He is primarily interested in optimization within the transportation industry.

Analysis of Centralized and Distributed Air Traffic Management Systems via MILPs

Rex K. Kincaid, Cameron Curtis
Dept. of Mathematics, William & Mary
Williamsburg, VA 23187-8795
rrkinc@wm.edu, ccurtis@wm.edu

Logan Wolf
Dept. of Mathematics, William & Mary
Williamsburg, VA 23187-8795
lbwolf@wm.edu

INTRODUCTION

Air traffic management systems address all airborne and ground-based activities of aircraft (e.g., taxiing, takeoff and landing, air traffic control and airspace flow and capacity decisions). The focus here is on air traffic control and, in particular, collision avoidance and efficient route planning. Two competing strategies, centralized control and distributed control, are key components of air traffic control for air traffic management systems. Fundamentally, centralized control involves a single authority collecting and processing information while distributed control involves each aircraft collecting and processing information and exercising trajectory decisions autonomously. Proponents of centralized control emphasize that it allows robust system stability, e.g., a small change in an aircraft's trajectory due to weather would not cause a cascade of changes throughout the entire system. On the other hand, proponents of distributed control emphasize the greater flexibility in trajectory decisions.

Air traffic management requires aircraft path-planning that address multiple aircraft, collision, and obstacle avoidance. Several researchers have developed optimization models (Richards and How, 2002; Cai and Zhang, 2018), as potential solution strategies. Our goal here is to develop an optimization model for both the centralized and distributed control paradigms and evaluate their performance. The results are applicable to both commercial aircraft traffic planning as well as UAV (Unmanned Automated Vehicles) trajectory planning (Xue, 2020).

Trajectory optimization with collision avoidance constraints has been formulated as a mixed integer linear program (MILP) to minimize a cost function regarding flight time(s) (Cai and Zhang, 2018). Centralized control involves a single authority collecting and processing information (e.g., an air traffic controller), while distributed control involves each plane collecting and processing information and exercising trajectory decisions autonomously (i.e., free flight).

MILP MODEL

A MILP is formulated for an aircraft trajectory optimization problem at a specific altitude in the two dimensional (2D) airspace. Trajectories are generated randomly and discretized for an aircraft p given by coordinates $(trajx_{p,0}, trajy_{p,0}), \ldots. (trajx_{p,T}, trajy_{p,T})$ over T time steps. Aircraft must start at their initial trajectory positions and travel within their cone of direction.

Each aircraft p's objective is to minimize the deviation from its projected trajectory, subject to velocity, separation, and movement direction constraints. The objective function penalizes aircraft for deviating from their planned trajectories. To remain a linear programming problem, the objective is formulated with the L_1 norm. The problem is to minimize the sum of the differences in the X and Y directions for each of the N aircraft in each of the T time steps. The constraints indicated below are for all $p, q \in [1, \ldots, N]$ and for all $t \in [0, \ldots, T-1]$.

Equation (1), in which $x_{p,t}$ and $y_{p,t}$ denote the decision variables, specifies the initial conditions for each aircraft along its projected trajectory. Equations (2) and (3) impose minimum (v_{min}) and maximum (v_{max}) aircraft velocities, while (4) enforces a minimum separation distance (*minsep*) between each aircraft. Equations (5) and (6) enforce a direction cone of 45 degrees ($\pi/4$) for a change in an aircraft's heading.

$$\min \sum_{t=0}^{T} \sum_{p=1}^{N} \left(\left| x_{p,t} - trajx_{p,t} \right| + \left| y_{p,t} - trajy_{p,t} \right| \right)$$

$$s.t.\ x_{p,0} = trajx_{p,0}\ and\ y_{p,0} = trajy_{p,0}\ (1)$$

$$\left| x_{p,t+1} - x_{p,t} \right| + \left| y_{p,t+1} - y_{p,t} \right| \geq v_{\min} \quad (2)$$

$$\left| x_{p,t+1} - x_{p,t} \right| + \left| y_{p,t+1} - y_{p,t} \right| \leq v_{\max} \quad (3)$$

$$\left| x_{p,t} - x_{q,t} \right| + \left| y_{p,t} - y_{q,t} \right| \geq \min sep\ (4)$$

$$\cos\left(a[p,t] + \frac{\pi}{4}\right) \cdot \left(x_{p,t+1} - x_{p,t}\right) + \sin\left(a[p,t] + \frac{\pi}{4}\right) \cdot \left(y_{p,t+1} - y_{p,t}\right) \geq 0\ (5)$$

$$\cos\left(a[p,t] - \frac{\pi}{4}\right) \cdot \left(x_{p,t+1} - x_{p,t}\right) + \sin\left(a[p,t] - \frac{\pi}{4}\right) \cdot \left(y_{p,t+1} - y_{p,t}\right) \geq 0\ (6)$$

Figure 1. Optimization Model

Linear programming solvers do not allow absolute value signs. Consequently, to solve such a linear program, the absolute values constraints are replaced with four linear constraints. In particular, the four constraints for each of the maximum velocity constraint (3) are given below.

$$x_{p+1} - x_{p,t} + y_{p,t+1} - y_{p,t} \leq v_{max}$$

$$x_{p,t} - x_{p,t+1} + y_{p,t+1} - y_{p,t} \leq v_{max}$$

$$x_{p,t+1} - x_{p,t} + y_{p,t} - y_{p,t+1} \leq v_{max}$$

$$x_{p,t} - x_{p,t+1} + y_{p,t} - y_{p,t+1} \leq v_{max}$$

Each minimum velocity constraint (2) requires additional parameters and decision variables in each of the four replacement constraints. A parameter, M1, is used to represent a large positive integer as well as two binary variables, $B1_{p,t}$ and $B2_{p,t}$. Introducing such additional variables is needed to ensure that every aircraft *p* at every time step *t* has only one of the four constraints satisfied as an equality. A similar set of four constraints is needed to replace constraint (4) and are not shown here.

$$x_{p,t+1} - x_{p,t} + M1 \cdot B1_{p,t} + y_{p,t+1} - y_{p,t} + M1 \cdot B2_{p,t} \geq v_{min}$$

$$x_{p,t} - x_{p,t+1} + M1 \cdot \left(1 - B1_{p,t}\right) + y_{p,t+1} - y_{p,t} + M1 \cdot B2_{p,t} \geq v_{min}$$

$$x_{p,t} - x_{p,t+1} + M1 \cdot \left(1 - B1_{p,t}\right) + y_{p,t} - y_{p,t+1} + M1 \cdot \left(1 - B2_{p,t}\right) \geq v_{min}$$

$$x_{p,t+1} - x_{p,t} + M1 \cdot B1_{p,t} - y_{p,t+1} + y_{p,t} + M1 \cdot \left(1 - B2_{p,t}\right) \geq v_{min}$$

SOLVING CENTRALIZED AND DISTRIBUTED MODELS

Aircraft starting locations and route trajectories were randomly generated by a Python program in a 200 n.m. by 200 n.m. (nautical miles) two-dimensional airspace for a twenty-minute time window (80 time steps of 15 seconds) for each aircraft. A minimum speed (v_{min}) of 1.75 n.m. and a maximum speed (v_{max}) of 2.5 n.m. for each 15 second time step was enforced. (The result is a minimum speed of 420 n.m./hr. and a maximum speed of 600 n.m./hr.) A minimum separation of value 5 n.m. between aircraft was required. All plane characteristics are identical.

Centralized Approach

1. Generate an instance with the Python program.
2. Solve the MILP (trajectory.mod and trajectory.dat) with AMPL/Gurobi.

The solution deconflicts all planes simultaneously. Output consists of the (x, y) coordinates for each of the 80 time steps as well as the sum of the trajectory distance variations from the original trajectory at each time step.

To solve the distributed model an aircraft is selected to be *active*. The active aircraft *p*'s trajectory is optimized while holding the other aircraft trajectories fixed. The objective remains the same: minimize trajectory deviation with minimum and maximum velocities while maintaining a minimum separation distance at each time step. After an *active* aircraft *p* finds its optimal trajectory, *p*'s default trajectory is updated, and the process repeats until there are no conflicts. The order in which the aircraft are deemed *active* is an exogenous decision. One approach is to identify the aircraft with the most conflicts in the given time window for each deconfliction MILP problem instance. The outline of these steps is given below.

Distributed Approach

1. Generate an instance with the Python program.
2. Determine an aircraft, *p*, with the maximum number of conflicts over the 80 step time window.
3. If there are no aircraft conflicts, then STOP.
4. Else, solve the MILP for aircraft *p* (all other aircraft trajectories unchanged). Solution deconflicts *p*.
5. Update deconflicted aircraft *p*'s trajectory.
6. Go to step 2.

BASELINE MODEL RESULTS

The aircraft trajectory models were written with the AMPL modeling language (Fourer, Gay, and Kernighan 2003). An AMPL model file contains the objective function, variables, and constraints, while the data and parameters are written to an AMPL data file by a Python script. Gurobi (Gurobi, 2023) optimization software is used to solve the problem.

Several computational experiments were run for 10, 20 and 30 aircraft. In the table below three representative test cases are summarized. In column one, the number of aircraft and model type are listed (C for centralized and D for decentralized). Column two records the number of route changes. Column three lists the objective function value in nautical miles (n.m.) and column 4 gives the computational time needed to find the optimal solution in seconds. For each aircraft row, there is one row for the centralized model (#Aircraft-C) and one row for the distributed model (#Aircraft-D).

Table 1. Results for Centralized and Decentralized Models

#Aircraft-Model	Route Changes	Objective Value (n.m.)	Runtime (seconds)
10-C	3	13.3	1011.5
10-D	2	15.8	7.7
20-C	2	44.1	5555
20-D	5	56.4	56.9
30-C	7	76.5	16624
30-D	8	92.8	75.6

There are three obvious conclusions. Both models make a similar number of route changes, but not necessarily the same changes. The centralized model, in our computational experiments, yielded a smaller deviation from the original route but is not dramatically better than the decentralized model's deviation. The decentralized model, in our computational experiments, is dramatically faster to solve (roughly 50-100 times faster). Although the decentralized model's solve times are much faster than the centralized model the solve times are still too long for a real-world application. Consequently, in the following two sections model extensions are explored to decrease the solve times.

TIME WINDOW ADJUSTMENTS

In an attempt to decrease the solve time, an approach is developed that determines a sequence of smaller time windows centered around a set of conflicts and solves the associated MILP for all aircraft within the smaller time windows in a sequential manner. All aircraft not included in a time window maintain their current trajectories. As in the baseline model, the initial trajectories for all aircraft follow the simplest possible routing: every aircraft proceeds directly and uniformly towards its destination. This often will not be a suitable routing as aircraft may violate separation requirements, but at any given time, the expectation is that few aircraft will violate these requirements. Consequently, each conflict can be treated as its own subproblem involving many fewer aircraft and over a significantly shorter time window. The net effect is a significantly decreased computational load.

The algorithm, given a parameter *w* for a time radius, works as follows: first, initialize every aircraft's trajectory as a direct, constant speed route from its starting point to its end point. At every time *t*, starting with *t=0*, perform the following procedure:

1. Check if any conflicts exist, if not STOP
2. Select a conflict and identify the aircraft involved in the time window *(t-w, t+w)*, for some time radius *w*.
3. Identify all aircraft which may be affected in resolving the conflict.
4. Create a new subproblem and include only the potentially affected aircraft over the time window *(t-w, t+w)*. Each aircraft's new starting point is its existing route's position at time t-w. Its new ending position is its existing route's position at *t+w*.
5. Solve the new subproblem using the MILP laid out in the baseline model. The output will generate updated routes for the affected craft in the time window *(t-w, t+w)*.
6. Increment *t* and return to step 1.

For a time window *(t-w, t+w),* determining which aircraft may be affected in resolving a conflict requires its own procedure. The simplest algorithm to do so begins by setting a distance cutoff. Every aircraft's speed is at most v_{max}, so over a time window of length $2 \cdot w$ an aircraft can travel at most $2 \cdot w \cdot v_{max}$. If two aircraft are more than $4 \cdot w \cdot v_{max} + minsep$ nautical miles apart, then the separation requirement is guaranteed. Thus, the set of aircraft in the time window for which the separation requirement is *not* guaranteed is easily checked. Note that the original conflicting aircraft is always included in this set. It is possible that not all included aircraft are needed, perhaps some are not party to the original conflict, nor are they forced to change routes to accommodate the new routing, but violating the separation requirement expression provides a sufficient, if not necessary, condition to include an aircraft in our subproblem. As an aside, determining the set of aircraft needed for deconfliction in a time window is equivalent to finding the maximal connected component containing our original aircraft in a "potential conflict" graph.

Another, slightly more selective, approach considers not only the absolute distance between aircraft but also their "maximum closing velocity." All aircraft are restricted to moving in a specific cone of direction pointing from their starting point towards their destination. This information can be used to narrow the list of potential conflicts in the *(t-w, t+w)* time windows. The maximum velocity of an aircraft *p1* toward an aircraft *p2* is found by checking whether the heading toward *p2* is within *p1*'s cone of direction. If so, *p1* can move toward *p2* at a velocity of v_{max}. Otherwise, p1's maximum velocity toward *p2* is to move along one of the edges of its cone of direction. Define $v_{p1\to p2}$ and $v_{p2\to p1}$ as the velocities of *p1* and *p2* toward the opposing aircraft. Consequently, the distance threshold can now be expressed as $2w\left(v_{p1\to p2} + v_{p2\to p1}\right) + minsep$, which is a lower bound on the previous distance threshold of $4 \cdot w \cdot v_{max} + minsep$. Taking the time to compute $v_{p1\to p2}$ and $v_{p2\to p1}$ results in fewer aircraft in each subproblem and, consequently, decreases the solve time for the associated MILP. For instance, consider two aircraft whose initial separation *minsep* $- \epsilon$, but are each heading directly away from each other. Under a strict distance threshold these aircraft would appear as potential conflicting aircraft but would not satisfy the stricter standard which considers their maximum closing velocities.

The smaller time window approach has clear advantages in reducing the overall solve time for deconfliction, but it also requires careful selection of our time window width *w*. Too small of a time window width may miss longer-term route conflicts and may not have enough time steps within the time window to resolve the current conflict. This can eventually create an impossible subproblem in Step 5, even if the original MILP over the complete time window has a feasible solution. One major problem this presents is that if this approximation fails, it tells us nothing about the feasibility of the original problem. That is, the MILP resulting from the smaller time window might lead to an infeasible solution simply because the time window is too small. Lengthening the time window makes it less likely for the procedure to fall victim to this horizon effect and may well make a previously infeasible problem feasible. However, longer time windows require more computational effort since, typically, more aircraft are included for deconfliction.

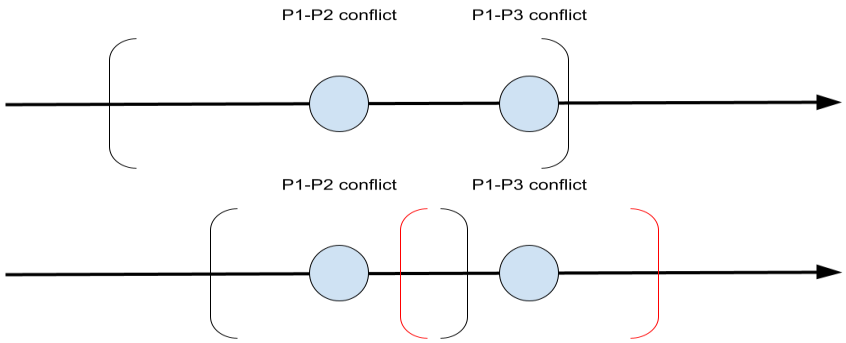

Figure 2. Overlapping time windows.

The algorithm may also struggle when conflicts have overlapping time windows (see Figure 2). If an aircraft has two distinct conflicts within *w* time units (e.g., P1 conflicts with P2 and P3 but in separate conflicts), then the second conflict will be included in the P1's subproblem. Thus, P1's conflict subproblem attempts to resolve the second conflict or, at least, push it later and out of the time window of the first conflict. This could result in unnecessary route changes while postponing the resolution of the actual conflict. Similarly, if an aircraft has two conflicts that are separated by more than *w* but less than *2w* time units, then the later conflict will not be absorbed by the first, but there is an overlapping period for at least one aircraft in which the route is recomputed.

The algorithm was implemented in python using the AMPL Python API to run the subproblems generated in Step 5. When tested against the full centralized MILP, this implementation, as expected, delivered significant speed improvements with only minor increases in the objective value. In one test case, the full MILP yielded an objective of 17.2 and took 170 seconds to run. The time window approach, for the same problem instance, found a route with a slightly worse objective value of 17.9 but the solve time was less than one second. As an aside, we should be mindful that this algorithm will operate best in circumstances where the number of conflicts per time window are limited to a small number.

One potential option to improve the algorithm, and reduce the problems caused by overlapping time windows, is to adaptively adjust the time window around different conflicts. For instance, either lengthen a time window slightly to merge two near conflicts together or, alternatively, shorten two overlapping time windows to eliminate the overlap. Also, we may want to lengthen or shorten the time window based on some estimate of the time required to resolve that conflict. For instance, a conflict with many aircraft in close proximity to each other may require a longer time window to resolve the conflicts. Ultimately, selecting a time window for a given problem instance remains an open issue but could be the key to reducing excess computation while avoiding the creation of infeasible subproblems within otherwise feasible conditions.

DISTRIBUTED MODEL SUBPROBLEMS

A slightly different approach to decreasing solve times is to split the distributed model into subproblems spatially. This is accomplished with the introduction of a detection distance parameter for each aircraft. Unlike the distributed model, in which all aircraft know the trajectory of all other aircraft, this model imposes the restriction that an aircraft only knows the position and trajectory of aircraft within its detection radius. Although restricting information often results in a worse objective value, this method seeks to reduce solve time by taking advantage of the expectation that few aircraft will be in conflict at any given time.

Like the baseline model, a python script first randomly generates position and trajectory data for all the aircraft. Each subproblem is solved with Gurobi optimization software through the AMPL API. Unlike the distributed model, which determines the aircraft *p* with the most conflicts, solves the MILP to resolve those conflicts, and continues in this fashion until all conflicts are resolved, this modification functions more like a time-based simulation. Using the detection distance parameter and aircraft trajectory information, the python script starts at time = 0, and loops through all aircraft in an exogenously determined order. For each aircraft, the trajectory information of any aircraft currently within the detection radius is checked to determine if there is an upcoming conflict. If such a conflict is detected, an AMPL data (e.g., `trajectory.dat`) file is generated including only the currently detected aircraft. Since this method is a modification of the distributed model, the MILP only solves for the optimal trajectory of the current aircraft. The python script then moves to the next aircraft, in order, and when all detected conflicts are resolved at this time step, the code moves on to the next time step, and so on. An outline of the algorithm is as follows:

```
for each time step:
  for each aircraft:
    If ( conflict_detected() ):
      AMPL.solve()
      update_trajectories()
```

In this version of the model, the assumption that all aircraft trajectories are known by each aircraft or by a centralized entity is not required. This change makes the model sequential, with new trajectories being calculated as new information is known or as additional conflicts arise. In the original distributed model, each MILP includes information for aircraft whose route trajectories do not come near the current aircraft, and since the number of constraints grows on the order of the product of the number of aircraft and the number of time steps, this redundancy quickly becomes computationally expensive. Thus, the detection distance parameter is used to determine which aircraft are relevant in each MILP subproblem. Furthermore, since the AMPL data file is only formulated at the time step where a conflict is detected, those earlier time steps in which no conflicts occur are effectively eliminated from the MILP, further reducing the size of each subproblem. A further improvement could be made with principles from the time window adjustments of the previous section to eliminate some of the future time steps from the MILP (after all detected aircraft are out of range). The current implementation, however, solves each MILP for the trajectory from the current time step for all time steps T.

Table 2 provides computational experience with this approach. All parameters are set to the same values as before, with the addition of the detection distance, which is set to 25 n.m. for all aircraft. Each test case was randomly generated, and uninteresting cases with few route changes were excluded. These are not the same test cases used for Table 1.

Occasionally the random generation of starting positions and trajectories places aircraft close together, but just far enough apart so that they do not violate the minimum separation constraint but will within a few time steps. The final line of the above table is one such case, as *p2* started at (100, 200), and *p21* started at (94.6, 200). Thus, while the two aircraft are not in conflict at time step 0, they will be at time step 1. Such cases often yield unusually high solve times and objective values, as the MILP would have had an additional 10 time steps to work with if the conflict did not occur immediately at the beginning. Still, the modifications to the distributed model yielded significant improvements, as even this bad case yielded a significantly faster solve time than the 30 aircraft decentralized model test in Table 1.

Table 2. Results for Modified Distributed Model

#Aircraft	Route Changes	Objective Value (n.m.)	Runtime (seconds)
20	7	32.8	4.43
20	6	21.5	3.28
20	6	53.4	5.19
30	13	94.8	11.7
30	8	69.7	6.54
30	15	172	23.8

CONCLUDING REMARKS

The MILP model developed here provides a means for comparing the performance of centralized versus distributed aircraft collision avoidance systems. The performance metric in the MILP seeks to minimize route trajectory deviations from the initial planned route. The computational results show that the centralized model deconflicts aircraft with the least total deviation from the prescribed routes but at a significant computation cost. The baseline model results for the distributed system dramatically improve the computation performance with a modest increase in the performance metric. However, even the improved computational performance of the distributed model is not enough for real-time decision making in a collision avoidance system.

Two additional approaches for decreasing computational costs were explored. The first approach explored the utility of subdividing the original time window into a series of smaller ones to deconflict the aircraft route trajectories. The width of each small time window must be chosen carefully. If the time window width is too large, the solution time may be prohibitively large and previously optimized trajectories may be needlessly revisited. However, if the time window width is too small then infeasible solutions may result. One potential option to improve the time window partition approach, and reduce the problems caused by overlapping time windows, is to adaptively adjust the time window width around different conflicting routes. A second approach, for decreasing the solve time for route deconfliction, was to split the distributed model into subproblems spatially. The spatial subdivision was accomplished with a detection distance parameter for each aircraft. The computational savings were significant (nearly an order of magnitude). It may also be possible to combine these two approaches.

Many extensions to the current MILP model may be considered. For example, it is straightforward to allow aircraft with individualized characteristics (e.g., individualized maximum velocity, minimum velocity, and direction cones). Replacing the performance metric in the objective function with one that focuses on minimizing fuel consumption and/or travel time may be of more interest than the current objective of minimizing total route trajectory deviations. The current MILP does not allow for trajectory changes due to weather disturbances. If the performance metric is minimizing fuel consumption, then the inclusion of convex shaped avoidance regions (weather fronts) will require the model to allow significant changes to route trajectories than have currently been considered. The addition of a third dimension (changes in altitude) would add greater flexibility for route changes, but it is unlikely to change the conclusions made here indicating the differences in performance between centralized and distributed air traffic control systems.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge Dr. Natalia Alexandrov at NASA Langley Research Center for her help in formulating the problem and feedback on the models presented. Additionally, we acknowledge the contributions of several William & Mary students in 2012-2013 who built and tested an earlier version of the baseline model (Keith Webb, Jennifer Thorne (Schilling) and Matthew Zinn). The authors acknowledge William & Mary Research Computing for providing computational resources and/or technical support that have contributed to the results reported within this paper. URL: https://www.wm.edu/it/rc.

REFERENCES

Cai, J. & Zhang, N. (2018). Mixed Integer Nonlinear Programming for Three-dimensional Aircraft Conflict Avoidance. *PeerJ Preprints 6:327410v1. https://doi.org/10.7287/peerj.preprints.27410v1*.

Fourer, R., Gay, D.M. & Kernighan, B.W. (2003) *AMPL: A Modeling Language for Mathematical Programming*. 2nd Edition. Thompson-Brooks/Cole.

Gurobi Optimization, LLC. (2023) *Gurobi Optimization Reference Manual*. https://www.gurobi.com.

Richards, A.G. & How, J.P. (2002). Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming. *Proceedings of American Control Conference.* ACC02-AIAA1057.

Xue, M. (2020). Urban Air Mobility Conflict Resolution: Centralized or Decentralized? NASA Ames Research Center. AIAA AVIATION 2020 FORUM. June 15-19 VIRTUAL EVENT. https://doi.org/10.2514/6.2020-3192.