MILP global search algorithm for aircraft ground movement optimization

Dual Degree Project
Stage I

by

Pushkar Godbole

09D01005

under guidance of

Prof. Abhiram Ranade
Prof. Rajkumar Pant

Department of Aerospace Engineering
Indian Institute of Technology - Bombay
Mumbai

November 2013

Certificate

This is to certify that this Dual Degree Project - Stage I report titled **MILP global search algorithm for aircraft ground movement optimization** by Pushkar Godbole (Roll no. 09D01005) has been examined and approved for submission.

Date:

Prof. Abhiram Ranade (Guide) **Prof. Rajkumar Pant** (Co-guide)

Acknowledgements

I would like to sincerely thank Prof. Abhiram Ranade for avidly guiding me through out this interdisciplinary CSE-Aerospace project. I thank Prof. Rajkumar Pant for giving realistic insights of airport operations, thus making the work authentic and relevant. I thank the Airport authorities of the CSIA for giving me an opportunity to experience airport operations first-hand. The practical and hands-on nature of this project always keeps me eager and excited.

Pushkar J. Godbole

Abstract

The final objective of the project is to develop and implement an algorithm for optimal real-time routing and scheduling of aircrafts on airports, minimizing the ground delays due to the ever increasing congestion and inefficient decision making. Developing such an algorithm would require a benchmarking method to prove it's quality. Thus there is a need to develop a method to identify a global optimum for the aircraft ground scheduling problem. The aim of Stage I of the project is to design and implement such an exhaustive search algorithm that would guarantee a solution with global optimality. A combination of Depth First Search (DFS) for routing and Mixed Integer Linear Programming (MILP) for scheduling, has been used to develop such an algorithm. This report will describe in detail the design and implementation of the same.

Keywords: Aircraft scheduling, Ground Movement Optimization, Mixed Integer Linear Programming (MILP), CPLEX

Contents

List of Figures

List of Tables

Nomenclature and Abbreviations

MILP: Mixed Integer Linear Programming

CSIA: Chatrapati Shivaji International Airport

ATC: Air Traffic Control

ATCO: Air Traffic Control Officer

CDM: Collaborative Decision Making

ASMGCS: Advanced Surface Movement Guidance and Control

NextGen: Next Generation Air Transportation System

GPS: Global Positioning System

GA: Genetric Algorithm

NP: Non-deterministic Polynomial-time

DFS: Depth First Search

t_{ik}: Time at which aircraft i reaches node k or enters arc k in it's path.

1 Introduction

Aircraft ground movement operations is the continuous and real-time process of routing and scheduling aircraft movements through a complex network of runways and taxiways on an airport, while keeping track of the incoming aircrafts and resource capacities. There has been a steep rise in demand for air transportation in recent times while the airport facilities and operation methodologies have broadly remained constant. This has led to heavy congestions at airports thus increasing ground delays and taxi fuel consumption. Particularly, 12.5% of flight delays in USA in 2012 have been estimated to have come from air traffic in the vicinity of airports [1]. An improvement in airport operations, would not only help minimize these delays but also reduce fuel consumption by approximately 9M tons/year and CO_2 emissions by approximately 28M tons/year [1].

1.1 Motivation

To understand this problem, we first need to grasp it's cause. Delays in departures and arrivals for individual aircrafts, can be caused by a plethora of factors such as weather, visibility, ground crew availability, aircraft readiness, passengers and so on. But, if all airlines follow a fixed predetermined schedule, assuming an ideal scenario (without individual delays), no aircraft should be delayed. But unlike trains or buses, aircrafts do not follow a fixed schedule. They only have a tentative schedule. For instance, if 9:00pm is a high demand time for majority of passengers, all airlines would try to schedule their aircrafts at 9:00pm, irrespective of the airport capacity, so that when a customer searches aircrafts departing at 9:00pm, all of them are listed. But the airport can physically handle only a limited number of take-offs/landings. Thus aircrafts are scheduled and released by the ATC (Air Traffic Control) in real time broadly on a first come-first serve basis.

Figure 1: Manual ATC with automated feedback

Although substantially efficient, the process of ATC at all airports in general and CSIA (Chatrapati Shivaji International Airport) in particular, is majorly manual and empirical in nature. The accuracy, efficiency and automation in the data feedback to the ATCOs (Air Traffic Control Officers), regarding aircraft positions and velocities, both in air and on ground is excellent. But the use of this data for decision making is completely manual and based on the ATCO's experience. This introduces considerable redundancies in the process thus delaying the aircrafts. The introduction of these inefficiencies has a magnified

impact because the delay is propagated through the queue of the following aircrafts and keeps adding up. This is particularly observed in heavy traffic hours which makes the manual decision making significantly difficult to handle. Due to increase in air-transportation demand, heavy traffic has become the plurality of majority airports across the world. Various scheduling strategies and algorithms with varying levels of automation are being investigated to improve airport operations to that end.

1.2 Objective

The aim of this project is to design and implement an aircraft **routing** and **scheduling** algorithm that would:

- provide **testable** near optimal solutions that are **better** than current manually generated ones, to the given aircraft configurations
- be **fast** enough to work on the fly in a **real-time** fashion
- incorporate all necessary operational inputs to generate **realistic** solutions
- aim to **augment** rather than replacing current ATC

Developing such a **testable** algorithm demands the need of a datum to compare to. The solution generated by the algorithm would have to be evaluated quantitatively, in order to establish nearness to global optimality. Thus there is a need for a globally optimal solution to the given problem to start with.
The **aim of stage I** has been, developing and implementing such an algorithm that would generate a schedule with **guaranteed global optimality**. The algorithm developed here in short is an exhaustive search over all possible path-allocation combinations using an MILP model for generating the optimal schedule. With availability of this global optimum, any schedule can now be compared to evaluate its **nearness to optimality**.

With a method for finding out the global optimum established, the objective of **Stage II** would be to develop and implement a **faster** algorithm that would give **near optimal** and **realistic** solutions.

1.3 Report organization

The report is majorly divided into four parts. Chapter 2 summarizes the relevant work done by others in the field of aircraft scheduling, briefly mentioning their advantages and shortcomings. Chapter 3 details the **design** and **implementation** of the MILP based algorithm to narrow down upon a global optimum for a given aircraft-airport configuration. Chapter 4 exhibits the results generated by this algorithm and compares them with results of some existing algorithms to establish improvement. A run on the map of the CSIA is also illustrated in this chapter. Finally Chapter 5 draws conclusions and describes the direction of the future work.

2 Literature review

Various approaches to improve air traffic control have been investigated over the years at both Commercial and Academic level. Although no specific details about the working of the commercial tools are available, the following section will briefly mention those documented.

2.1 Commercial

2.1.1 Collaborative Decision Making

Collaborative Decision Making or CDM is a project launched by EUROCONTROL in 2004 which aims at improving Air Traffic Flow and Capacity Management at airports by reducing delays, improving the punctuality of events and optimizing the utilization of resources [2]. This tool is mainly aimed at efficient assimilation and dissemination of data. It is essentially designed to augment and improve the current decision making system and not participate in it per se. Using CDM, the same data could be accessed and updated by all partners thus reducing the latency in the entire process.

2.1.2 Advanced Surface Movement Guidance and Control Systems

Advanced Surface Movement Guidance and Control Systems or ASMGCS is a tool developed and being validated by EUROCONTROL for routing, guidance and surveillance for the control of aircraft and vehicles in order to maintain the declared surface movement rate under all weather conditions while maintaining the required level of safety [3]. It is the next step after CDM to better utilize all the data assimilated and reduce ground delays. ASMGCS also includes automated routing and scheduling as one of its major objectives besides surveillance.

2.1.3 NextGen

Next Generation Air Transportation System or NextGen [4] is a tool being designed and implemented by the Federal Aviation Administration (FAA) as a one stop solution to all the aviation planning needs for both ground and airspace control. It boasts the use of GPS and satellites to improve the positioning and navigation accuracy of all aircrafts. This information would be shared by all aircrafts and airports. It would use weather predictions to alert partners before hand thus improving their ability to take appropriate decisions. NextGen aims to make aviation decentralized so that every aircraft intelligently and collaboratively plans it's own path both in air and on ground, thus eliminating the redundancies imposed by adherence to extraneous rules such as path and route constraints.

2.2 Academic

This section will go over research work in the field in better detail due to availability of a detailed knowledge base. The approaches to ground movement optimization followed in the research works can be broadly classified into two categories; *macroscopic* and *microscopic*.

The **macroscopic** approach treats the problem on a broader scale dealing with airport capacity estimation, delay estimation, weather mitigation, push-back optimization and the like. These algorithms do not directly deal with real-time aircraft routing and scheduling on a one-on-one basis. They mostly tackle it as a **resource-allocation** problem. They

start off by assuming the current ATC system and developing a model based on it.
Microscopic approach on the other hand deals with a point-to-point scheduling of aircrafts on one-on-one basis. They give a precise node by node routing and schedule for all aircrafts as their output. The results generated by such algorithms are invariably better than those generated by macroscopic methods although they lack in the following respects:

- Being highly customized, such schedules are highly dependent on the existing aircraft configuration on the airport. Thus they are subject to change every day. While currently, particular aircrafts have *generally* pre-defined paths and halting patterns. The pilots and ATCs are used to their schedules. Introducing variable schedules daily draws the pilots out of their comfort zones.

- Since the current ground operations heavily rely on on-the-air interactions between the pilots and ATCOs, the instructions need to be co-ordinated manually by the ATCO to the pilot. The radio links between flights and the control towers are highly congested, especially during critical hours. Introducing variable schedules would further raise the demand of pilot-ATCO interaction on an already congested radio link.

- There is a justifiable mistrust among ATCOs in relying entirely upon automated means to deal with ground operations. Heavy testing and safety assurance needs to go into it before it could become broadly acceptable.

The approach of this project is primarily of microscopic optimization. The following paragraphs will briefly go over various research papers in the field as classified above.

2.2.1 Macroscopic optimization

Here two methods of macroscopic nature are illustrated; namely capacity estimation and push-back optimization.

2.2.1.1 Capacity Estimation

The paper by Debashish etal. [5] presents a model that determines the number of aircrafts in the Terminal airspace of an airport that can be safely handled by the Air Traffic Controllers. This model abstracts the model by Janic & Tosic [6] to include arrival modeling and multiple (parallel and cross) runway configurations. The model incorporates the airport layout, aircraft distribution, inter aircraft distance thresholds, arc lengths, inter-arc angles and aircraft velocities to generate the runway and thus the airport capacity. It has been tested for parameters of all major airports in India including Mumbai and Delhi.

2.2.1.2 Push-back Optimization

The PhD thesis by Harshad Khadilkar titled *Networked Control of Aircraft Operations at Airports and in Terminal Areas* [1] deals with generating optimal push-backs for aircrafts at the terminal. This work does not directly deal with routing and scheduling. It is built on the assumption that routing and scheduling decisions are appropriately made by the ATC. It mainly makes use of year long empirical data of the Boston Logan International Airport but can be abstracted to any airport. The model uses the empirical data to setup random variables to stochastically estimate aircraft taxi times. The gates and runway arrival entries are modeled as sources of aircrafts. Parameters are set up to incorporate maximum allowed push-backs, surface surveillance data, separation standards and airport

and aircraft capacities. These and real-time inputs such as incoming (airborne) traffic, surface traffic, push-back requests are fed to the controller. The controller uses a dynamic programming strategy (an abstraction over the k-ctrl strategy: constant traffic at the airport, always), with real-time buffers introduced to temporarily exceed the upper-bound of airport capacity (during heavy traffic) to generate optimal push-backs for all aircrafts currently at the terminals. Introducing these push-backs at the terminal achieves a dual goal of reducing delays and reducing fuel consumption, as the engines of aircrafts do not consume as much fuel when waiting at a terminal as compared to while taxing.

2.2.2 Microscopic optimization

This section will go over some microscopic optimization methods relevant to the project. All microscopic optimization techniques essentially solve a 'snapshot problem'. That is, they take a fixed time frame and a given aircraft distribution to schedule only those aircrafts optimally. Generally the scheduling techniques interpret the airport and network of taxi ways as a graph with the connections as arcs and the intersection and critical points as nodes in the graph. The aircrafts have a given start time, start node and destination node. The objective of the optimization algorithm is to find out a *near optimal*, *conflict free* schedule that conforms to all safety regulations and constraints while maintaining taxi times and stoppages at a low. The Microscopic optimization techniques can be broadly classified into two categories:

2.2.2.1 Stochastic

Aircraft scheduling in domain of microscopic optimization, is essentially a NP-hard problem. Any deterministic method to seek solution would at best take too long to be of practical use. Thus many heuristic methods are employed to solve it. The techniques mentioned here use various methods of stochastic optimization (like Genetic algorithms(GA)) and "intelligent" elimination to narrow down upon a near optimal solution.

Liu etal.:

[7] Liu etal. use a purely stochastic procedure to reach a solution using a *Genetic algorithm*. The objective function to be minimized is the total taxi time of all aircrafts. The algorithm starts off by finding the first 'N' paths for each aircraft. Every arc of each such path for every aircraft is assigned a binary variable x_{kij} where k is the flight no., i is the node-1 of arc $i-j$ and j is the node-2. x_{kij} takes a value of 1 if aircraft k traverses the arc $i-j$ and 0 otherwise. These variables are fed to a genetic algorithm who's chromosome chain is the sequence of all the binary variables x_{kij}. The fitness function of the genetic algorithm is:

$$\sum_{k=1}^{n}\sum_{i=1}^{Q}\sum_{j=1}^{Q} x_{kij}t_{kij}$$

where t_{kij} is the time taken by aircraft k to traverse arc $i-j$ and Q is the number of nodes in that particular path of flight k. The output of the algorithm is the chromosome chain of variables x_{kij} that minimizes the fitness function subject to conflict elimination and regulation constraints. The number of genetic generations has to be limited to restrict the run-time. The optimality of solutions is not testable due to the stochastic nature.

Gotteland etal.:

[8] In their paper Gotteland etal. present three different methods to approach the problem stochastically.
In the first method, aircrafts are scheduled one by one on a first come first serve basis. The path selection for each aircraft at every node is done using the A* algorithm. Arcs are reserved for aircrafts for the time slots during which they are occupied. The next aircraft is scheduled considering this slot reservation information. Thus the problem is in a way solved as a time-dependent network flow optimization problem. The disadvantage of using a first-come-first-serve criterion in prioritizing aircrafts is that, some (faster) aircrafts starting late, might loose out on the time advantage which would further propagate through the entire schedule and increase the total taxi time.
The second method presented is similar to that described by Liu [7]. It uses genetic algorithm to choose paths and the constraint resolution priority order. It essentially replaces the A* algorithm in the first method with GA while eliminating the one-aircraft-at-a-time method of scheduling, thus giving a more holistic solution.
The third method uses a combination in A* and GA to use the strength of each. GA is used for path selection through the path pool while A* is used for prioritizing during conflict resolution.
The GA is run for a fixed number of generations to limit the run-time to realistic levels. It is observed that as the number of aircrafts increase, the delay generated by the first method is the highest while that by the third (mixed) method is lowest.

Baijal etal.:

[9] The paper by Bijal etal. illustrates another stochastic algorithm, Bacterial Foraging, applied to aircraft scheduling problem. The basic premise of the method is similar to that employed by Liu [7]. The algorithm minimizes the total taxi time. The objective function is thus same as that of Liu. The method however uses fixed paths. Dijkstra's algorithm is used to find the shortest paths between origin-destination pairs for each aircraft. These paths are then fed to the Bactrial Foraging (BAFO) algorithm which decides both the priority order and the waiting time for resolving conflicts. The two disadvantages of this method are the fixed paths and arbitrary waiting time allocation. The converged solution although valid might have redundant waiting times.

2.2.2.2 Deterministic & Semi-deterministic

These techniques generally simplify and cut down the solution space to a deterministically solvable problem using some stochastic or empirical techniques. The solution subspace is then deterministically explored for a testable optimum.

Smeltink etal:

[10] The paper by Smeltink formulates the problem as a Mixed Integer Programming (MIP) problem. The algorithm interprets the airport map as a graph and allocates fixed paths to aircrafts. The binary decision variables are then defined according to conflicts. Constraints such as succession, collision, minimum distance, speed are modeled as linear equations with decision variables. Solving the linear equations yields the values of the binary variables such that the cost function is minimized. The cost function is again taken as the total taxi time including the initial waiting time. The MIP is solved using CPLEX to calculate time stamps at each node. The concept of Rolling Horizons has been used to deal with larger problems. 4 different techniques used in Rolling Horizons

are discussed. Essentially the entire problem is split into smaller subproblems with a fraction of aircrafts considered based on a priority criterion. The combined solution of these subproblems gives the solution to the total problem. One major disadvantage of this approach is that the paths are predetermined and fixed through out. This could lead to highly suboptimal solutions.

Marin etal.:

[11] The paper by Marin also employs Mixed Integer Programming but on a broader level. Marin's paper allows aircrafts to be re-routed in case of conflict or congestion. The paper defines two different types of binary decision variables :
E_{it}^{w} : 1 if aircraft w waits at node i at period t, 0 otherwise
X_{ijt}^{w} 1 if aircraft w is routed from node i to j at period t, 0 otherwise
The cost function to be minimized is :

$$F(X,E) = \sum_{w \in W} \sum_{t \geqslant t(w)} P^{w}(\sum t_{ij} X_{ijt}^{w} + \sum E_{it}^{w}) + \sum_{w \in W} \sum r_{i}^{w} E_{i}^{w} T$$

Where P is the priority of aircraft w and r is the shortest path distance between node i and destination of aircraft w. This objective function essentially incorporates two factors : The waiting time and the routing time. The second part of the summation is considered only when the variable E is 1 i.e. when the aircraft traverses that arc. Lower bounds on the aircraft speeds are imposed and the problem is boiled down to a time frame when the scheduling is done. The time is discretized into time slots.

Roling etal.:

[12] The paper by Roling and Visser defines the problem as a Mixed Integer Linear Programming (MILP) model in the most open and flexible manner w.r.t. the decision variables. The decision variable defined is :
X_{fdrs} : Is 1 if flight f is delayed by d periods on it's route r in segment s and 0 otherwise. The total taxi time including the waiting time is the cost function to be minimized. The algorithm starts with the shortest paths as default values and executes re-routing and waiting at intermediate nodes to resolve conflicts. The paper doesn't clearly describe the method used to generate paths and achieve the re-routing. A disadvantage of this method is that this method would generate exponentially many decision variables while essentially attempting to solve the NP-hard problem deterministically. The algorithm and MILP model are implemented using CPLEX.

Baik etal.:

[13] The approach by Baik is different from the other deterministic optimization methods. Baik essentially follows the 1 aircraft at a time strategy. This is justified by the real scenario where, in fact the starting time and origin destinations of aircrafts are tentative and the aircrafts have to be scheduled in real-time most generally on a first come first serve basis. Baik thus schedules the aircrafts based on a priority ordering (generally first come first serve) one by one. A time dependent space-time network is created form the graph. After an aircraft has been scheduled, arcs that are occupied are reserved on the time-space network. The next aircraft is routed based on the occupancy information and a Time Dependent Shortest Path is generated. This process is continued in real-time as new aircrafts enter the queue. The major disadvantage of using this method is that it is highly myopic as it schedules one aircraft at a time based on first come first serve priority. A faster aircraft behind in the queue will not only lead to delay in it's schedule but also cause further delay propagation through the entire schedule.

Gupta etal.:

[14] This paper modifies the methodology used by Smeltink [10] to develop a combined approach of MILP and TDSP as used in the paper by Baik [13]. The method starts off by finding a TDSP based routing and schedule of all the aircrafts within the scheduling horizon. With the routes and times known, this schedule is used as a tentative starting point and input to the MILP model. The MILP model for prioritizing aircrafts during conflict resolution is similar to that by Smeltink. The schedule generated by this combined strategy is observed to be better than that generated by the shortest route approach of Smeltink. The model has been implemented in CPLEX.

Testability and optimality:

Diverse approaches have been explored to tackle the aircraft scheduling problem at various levels of flexibility. We can conclude from this survey that approaching the problem with an entirely deterministic and exhaustive approach will be difficult and would lead to unrealistic run-times due to the sheer number of possibilities. Having said that, in order to do better than the current manual approach and to prove so, there is a need of a benchmarking method. This benchmarking irrespective of the efficiency or run-time can be done only with a solution that can not get any better. There is thus a need to find a globally optimal solution to the aircraft scheduling problem, not as a usable option but as a test platform. The aim of the Stage I of the project hence has been developing and implementing such an algorithm that would guarantee global optimality. The algorithm need not find all global optima, but must be such that any other solution would at best be equivalent to that generated by this algorithm. The next chapter will elaborate on the design and implementation of this algorithm.

3 Global Search: Design and Implementation

With the need of a global optimum to benchmark the solutions generated by other heuristic aircraft scheduling strategies, a global search algorithm needs to be designed such that :

- it delivers at least one globally optimal solution such that, any other solution at best, is as good as the one generated by it
- it takes a (possibly long but) finite run-time
- it delivers realistic solutions

The process of finding a global optimum becomes intimidating due to infinite possible choices. This is because, in a globally optimal schedule, an aircraft :

- has **multiple** choices of **routes**
- can wait **anywhere** on the route to solve overlap conflicts
- can wait for **any duration** anywhere on the route to do so

This implies that there can be infinitely many equivalent global solutions. Thus the global optimality of any solution can be proved only conceptually, based on the approach followed and not by exhaustively searching the entire solution space, since that is infinite.

3.1 Design

To cut down the solution space without eliminating the globally optimal solution, four simplifying assumptions have been made :

- As is generally done, the map is discretized into arcs and nodes and interpreted as a graph.

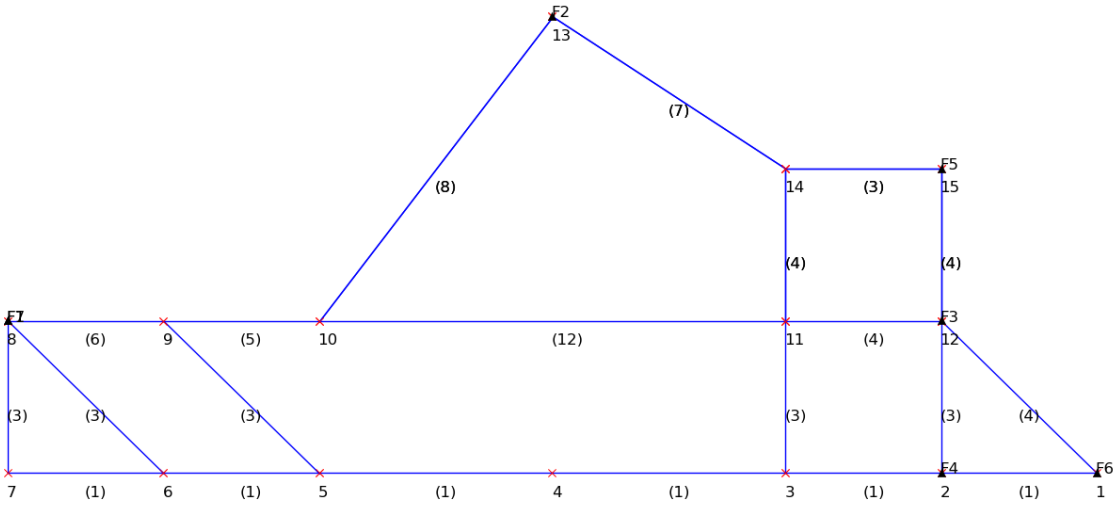

Figure 2: Airport map interpreted as a graph

- Time is discretized into slots (generally 10 secs) as is done in current scheduling methods at airports
- Aircraft waiting is defined on arcs and not at points in the map.

A **B**

Figure 3: Waiting equivalence

  An aircraft waiting at **A** or **B** would be equivalent w.r.t. scheduling time and hence the exact point of waiting on an arc is insignificant.

- Although in real-life scenarios, the starting times and origin-destination pairs of aircrafts are only tentative, exact values are needed in order to reach and justify a global optimum. Hence starting times and origin-destination pairs of all aircrafts are assumed to be fixed; an aircraft enters the scheduling space at it's starting time at it's origin and exits the instant it reaches the destination.

With these assumptions, the solution space is reduced to a finitely solvable one. It needs to be noted that the globally optimal solution generated here would be used as a **post-facto analysis tool** to check how well the other heuristic algorithms performed in face of the variability of parameters. The problem is now approached systematically such that the approach itself ensures global optimality of the solution.

3.1.1 Parameters

3.1.1.1 Inputs

The inputs given to the algorithm include :

- Map : Interpreted as graph with arcs (lengths specified) and nodes
- Aircrafts :
  - Starting time
  - Origin
  - Destination
  - Aircraft speed
  - Trailing separation : This parameter depends on the size and shape of the aircraft. It is necessary to maintain a minimum gap between any two following aircrafts to mitigate the turbulence effects of the leading aircraft. The trailing separation distance of the **leading aircraft** becomes the significant parameter in case of an aircraft following another.

3.1.1.2 Objective

Total time : Summation over the destination times of all aircrafts, weighted by priority

Mathematically,

with t_{ik} : Time when aircraft i reaches node k / enters arc

and P_i: Priority level of aircraft i. Generally landing aircrafts have higher priority over departing aircrafts. It is preferable to not request a landing aircraft to halt midway during taxing. Setting a very high value of P for landing aircrafts ensures this

The objective would be,

Minimize :

$$\sum_{i=0}^{N} P_i t_{in}$$

N = Total no. of flights

n = Last / Destination node in the path of flight i

3.1.1.3 Output

Schedule : Time stamps at every node for each aircraft

Mathematically,

with t_{ik} : Time when aircraft i reaches node k / enters arc

The output would be,

values of t_{ik} for $i \in (0, N)$ & $k \in (0, n)$

N = Total no. of flights

n = Last / Destination node in the path of flight i

3.1.2 High-level design

The overall design of the algorithm can be classified into five major parts as follows:

3.1.2.1 Path finding

With the aircrafts and their origin-destination pairs specified, the first step is to come up with all possible paths for all the aircrafts. This has been done using Depth First Search (DFS). In order to avoid repetition, all origin-destination pairs from the pool of aircrafts are first identified and DFS is applied on each pair. The paths thus generated are then stored for each aircraft accordingly.

In order to ensure that DFS does not enter into infinite loops, the algorithm is designed to break when a *cycle* is detected. This can be justified realistically by considering the fact that an aircraft circling a set of taxiways to land up at the same place later on would be inefficient besides being illogical both in terms of the delay ensued and the extra fuel consumed in doing so. It would be more appropriate to halt the aircraft in it's place instead of letting it enter a loop.

3.1.2.2 Combinations

With the possible paths for each aircraft identified, the next step is generating all possible combinations of these paths, as we do not know before hand which combination would give the global optimum. Most algorithms that use micro-optimization take the shortest paths as the paths for all aircrafts, but this choice would not be appropriate for all situations and surely not to identify the *global* optimum.

For instance, if there are 3 flights to be scheduled, with each having 3 possible paths, this step will generate 27 flight-path combinations of these aircrafts.

3.1.2.3 Local minima

With all possible path combinations generated, the next step is to find a minimum for every such combination. This is done using a Mixed Integer Programming (MILP) model. All constraints such as aircraft speed, following, collision, trailing separation are incorporated into the MILP using linear equations. The solution to this MILP would be a local minimum for that particular choice of path combinations. The detailed design of the MILP is covered in the next section.

3.1.2.4 Global minimum

The local minima for all possible flight-path combinations generated are compared in this step. The combination with the minimum value of cost function is identified as the **global optimum**. The approach used in this process ensures that the solution generated thus, is indeed a global optimum and no better solution can be identified for the problem.

3.1.2.5 Post processing

With the global optimum identified the output contains schedule with time-stamps for each node in path of each aircraft. The post-processing does the following three things:

- Sanity check: The solution generated is checked for feasibility. Every overlapping arc for a pair of aircrafts is checked for temporal overlap. An error message is generated if an overlap is detected.
- Animation: An animation over the map of the entire scheduling is run to get a better visual understanding.
- Space-time plot: A 3D space-time plot of the schedule is generated. The XY plane is the actual map of the airport while the Z axis is discretized with time slots. The absence of any overlapping/intersecting lines in the space-time plot justifies the feasibility of the solution.

3.1.3 MILP design

This section describes in detail the design of the Mixed Integer Linear Programming (MILP) model used to identify the local minimum for every flight-path combination, going over all the constraints and their mathematical analogs. The constraints imposed include:

- Every aircraft has a maximum speed
- Aircrafts entering an arc from opposite directions must do so exclusively of each other in time
- For an aircraft following another, if the length of the arc is less than the trailing separation of the leading aircraft, the trailing aircraft must enter the arc only after the leading aircraft has exited.
- For an aircraft following another, if the length of the arc is greater than or equal to the trailing separation of the leading aircraft, a minimum distance of the trailing separation of the leading aircraft has to be maintained between the two aircrafts, all throughout the arc.

3.1.3.1 Travel time

Constraint: An aircraft should not traverse an arc at speed greater than the aircraft's speed limit. Travel time on an arc should be greater than or equal to (arc length)/(aircraft speed).

Figure 4: MILP constraint: Travel time

Mathematically,

$$t_{i(k+1)} - t_{ik} \geqslant \frac{length_{ik}}{speed_i}$$

3.1.3.2 Head-on collision

Constraint: Two aircrafts entering an arc from opposite directions must have no temporal overlap. In other words, if two aircrafts A and B enter an arc from opposite directions, either A should enter only after B has completely exited or visa-versa.

It may be noted here that, the subscripts k and l correspond to the k^{th} and l^{th} nodes in the paths of aircrafts i and j respectively. Thus the k^{th} node of aircraft i can be the $(l+x)^{th}$ node of aircraft j; $x \in \mathbb{Z}$. Thus every physical node in the graph has a node number corresponding to each aircraft.

Mathematically,

$$M \times x_a + t_{ik} - t_{j(l+1)} \geqslant 0$$
$$M \times (1 - x_a) + t_{jl} - t_{i(k+1)} \geqslant 0$$

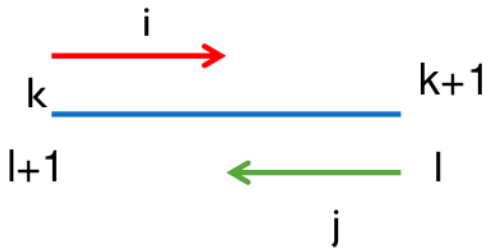

Figure 5: MILP constraint: Head-on collision

$M \gg 0$ & $x_a \in [0, 1]$

It may be noted that, when $x_a = 1$, the first equation is always true as $M \gg 0$ and can be said to be *inactive*. The second equation on the other hand says that entry time of aircraft j into arc l must be greater than or equal to exit time of aircraft i from arc k. When $x_a = 0$, the second constraint becomes inactive while the first constraint says that the entry time of aircraft i into arc k must be greater than or equal to the exit time of aircraft j from arc l. Thus the head-on constraint is satisfied.

3.1.3.3 Common Node 1

Constraint: For an aircraft A following another aircraft B while entering arcs with 1^{st} node common, **if** the length of the arc of B is less than it's trailing separation, then A must enter it's arc only after B has exited it's own arc. **Else**, if length of arc of B is greater than or equal to it's trailing separation, A may enter it's arc only after B is at least at a distance equivalent to it's trailing separation from common node 1, on it's own arc and visa-versa.

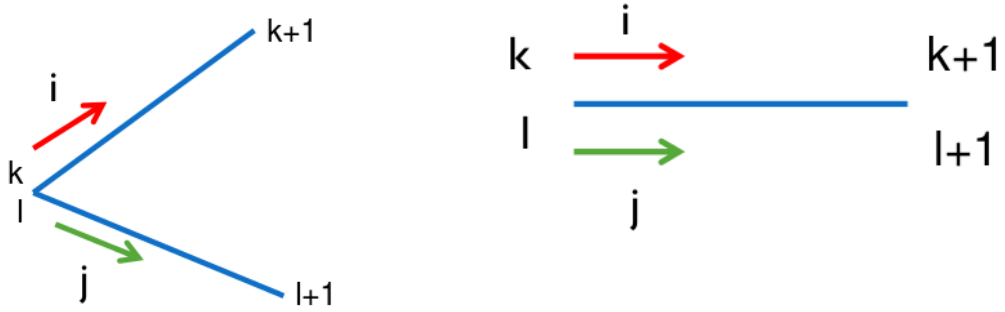

Figure 6: MILP constraint: Common Node 1

Here,

- If j leads and length of arc l is less than sep_j (j's trailing separation), then i enters arc k only after j has exited arc l i.e. reached node $l + 1$.
- If j leads and length of arc l is greater than or equal to sep_j (j's trailing separation), then i enters arc k after j has traversed distance of at least sep_j on arc l

- If i leads and length of arc k is less than sep_i (i's trailing separation), then j enters arc l only after i has exited arc k i.e. reached node $k+1$.
- If i leads and length of arc k is greater than or equal to sep_i (i's trailing separation), then j enters arc l after i has traversed distance of at least sep_i on arc k

Mathematically,

If $Length_l < sep_j$,

$$M \times x_b + t_{ik} - t_{j(l+1)} \geqslant 0$$

Else (if $Length_l \geqslant sep_j$),

$$M \times x_b + t_{ik} - t_{jl} \geqslant \frac{sep_j}{speed_j}$$

If $Length_k < sep_i$,

$$M \times (1 - x_b) + t_{jl} - t_{i(k+1)} \geqslant 0$$

Else (if $Length_k \geqslant sep_i$),

$$M \times (1 - x_b) + t_{jl} - t_{ik} \geqslant \frac{sep_i}{speed_i}$$

$M \gg 0$ & $x_b \in [0, 1]$

It may be noted that only 2 of the above 4 constraints will be applied (1 from (1), (2) and one from (3), (4)) based on the lengths of arcs k and l. Out of those 2, 1 will be active finally based on the value of x_b. $x_b = 0$ would imply j is leading while $x_b = 1$ would imply i is leading.
The value $\frac{sep_i}{speed_i}$ is the time that aircraft i would take to traverse a distance equal to sep_i. Thus the constraints say that if the arc of the leading aircraft is longer than it's trailing separation, the time when the trailing aircraft enters would be later enough for the leading aircraft to traverse at least a distance equivalent to it's necessary separation on it's arc. Thus the common node 1 constraint is satisfied.

3.1.3.4 Common Node 2

Constraint: For an aircraft A following another aircraft B while entering arcs with 2^{nd} node common, **if** the length of the arc of A is less than B's trailing separation, then A must enter it's arc only after B has exited it's own arc. **Else**, if length of arc of A is greater than or equal to B's trailing separation, B must exit it's arc when A is at least at a distance equal to B's trailing separation from the common node 2, on it's own arc.
Here,

- If j leads and length of arc k is less than sep_j (j's trailing separation), then i enters arc k only after j has exited arc l i.e. reached node $l+1$.
- If j leads and length of arc k is greater than or equal to sep_j (j's trailing separation), then j exits arc l i.e. reaches node $l+1$ when the distance of i is at least equal to sep_j from node $k+1$

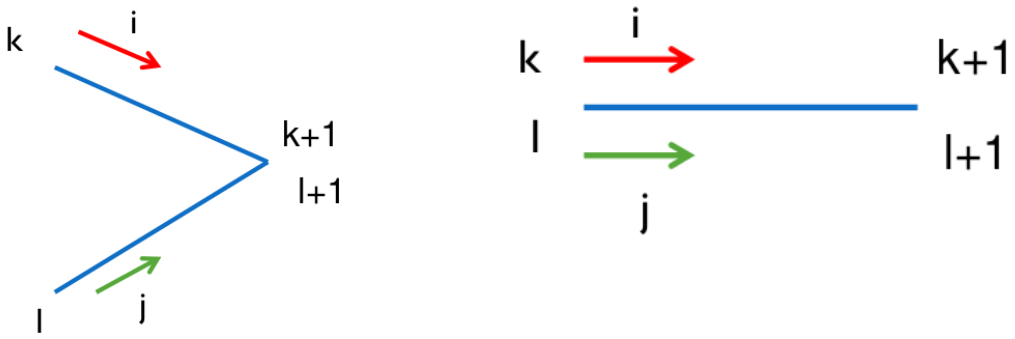

Figure 7: MILP constraint: Common Node 2

- If i leads and length of arc l is less than sep_i (i's trailing separation), then j enters arc l only after i has exited arc k i.e. reached node $k+1$.
- If i leads and length of arc l is greater than or equal to sep_i (i's trailing separation), then i exits arc k i.e. reaches node $k+1$ when the distance of j is at least equal to sep_i from node $l+1$

Mathematically,

If $Length_k < sep_j$,

$$M \times x_c + t_{ik} - t_{j(l+1)} \geqslant 0$$

Else (if $Length_k \geqslant sep_j$),

$$M \times x_c + t_{i(k+1)} - t_{j(l+1)} \geqslant \frac{sep_j}{speed_i}$$

If $Length_l < sep_i$,

$$M \times (1 - x_c) + t_{jl} - t_{i(k+1)} \geqslant 0$$

Else (if $Length_l \geqslant sep_i$),

$$M \times (1 - x_c) + t_{j(l+1)} - t_{i(k+1)} \geqslant \frac{sep_i}{speed_j}$$

$M \gg 0$ & $x_c \in [0, 1]$

It may be noted that only 2 of the above 4 constraints will be applied (1 from (1), (2) and one from (3), (4)) based on the lengths of arcs k and l. Out of those 2, 1 will be active finally based on the value of x_c. $x_c = 0$ would imply j is leading while $x_c = 1$ would imply i is leading.
The value $\frac{sep_i}{speed_j}$ is the time that aircraft j would take to traverse a distance equal to sep_i. Thus the constraints say that if the arc of the trailing aircraft is longer than the trailing separation of the leading aircraft, the time when the leading aircraft exits would be earlier enough for the trailing aircraft to be at a distance at least equal to the leading aircraft's

trailing separation from it. Thus the common node 2 constraint is satisfied.

Note that, the combination of the common node 1 and common node 2 constraints meet the necessary constraint for an aircraft following another on the same arc. In that case both $Length_k$ and $Length_l$ become the same and the constraints are applied at the entry and the exit nodes based on the arc length. Also, the decision variables x_b and x_c would **have to be the same** in such case. (Otherwise there would be overlaps.)

These are all the necessary and sufficient constraints that have to be imposed in order to meet all requirements. We have thus modeled aircraft scheduling operations in face of variable aircraft speeds and separation distances. The speeds and separation distances once defined for an aircraft however remain fixed throughout the optimization. Additionally, the aircrafts are assumed to be **point objects** having **uniform velocities** and **infinite accelerations** while calculating the times. The aircrafts are assumed to **disappear** from the map once they reach their destination.

3.2 Implementation

The implementation of the above algorithm is done partly in Python and partly in C++. Since C++ is much faster than Python, the core of the implementation is done in C++. All the computation and optimization occurs in C++. Python has wonderful tools for data visualization which are very easy to implement. Hence the Post processing is done in Python. Additionally the entire C++ code also is wrapped in Python using Cython (C Python). Cython is a clone of Python with an added feature to write Python like code which is then translated automatically to C or C++. Cython also has the ability to call C++ functions and Python functions simultaneously. Also the functions defined in Cython can be called from Python as modular functions.
The MILP solving is done using CPLEX (a commercial LP solver). CPLEX has a module defined in C++. Hence the problem definition and equation formation for the MILP is done via C++. All the CPLEX options are callable from C++. CPLEX returns it's output to the C++ code which is then post processed by Python.

3.2.1 Inputs

There are in all three input files to be given to the code.

3.2.1.1 Map

The Map given to the code, contains information about the connectivity of taxiways and runways at the airport along with the length of the connections. The `Map` file is essentially a connectivity list of all the nodes in the map (graph) along with the edge lengths. The format of the `Map` file is as follows:

- The line number of the `Map` file is the number of the node in consideration (Numbering starting from 0). Thus the number of lines in the file is equal to the number of nodes on the map. If a node is not connected to any other node, that line is kept empty.
- Every entry in a line has two elements. The connecting node and the corresponding edge length. Thus a node connected to 2 other nodes will have 4 elements in its line. 2 corresponding to each connected node.

- The entries in the map are uni-directional. an entry j k in line number i would imply node i is connected to j with an edge of length k in direction $i - j$. A bi-directional arc would be specified by adding such entries to both the lines i and j.

Here is an example of a typical input `Map` file with the corresponding map.
(The numbering in the input file starts from 0 while that in the figure starts from 1)

```
1 2
0 2 2 2 3 2
1 2 4 2
1 2 4 10
2 2 3 10
```

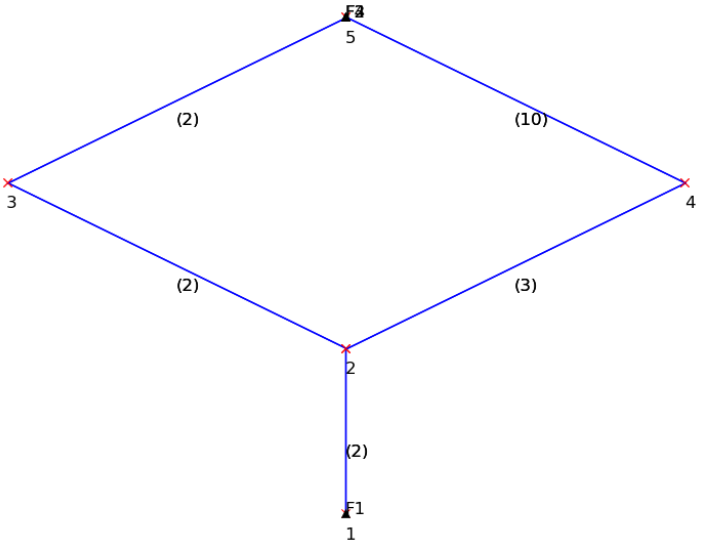

Figure 8: Map example

3.2.1.2 Flights

The next input file is the `Flights` file which contains the information of the flights to be scheduled. Each flight is listed on one line with following 6 parameters:

- Starting node
- Destination node
- Starting time (slot)
- Speed
- Trailing separation
- Priority

Here is an example of a typical `Flights` file for the same map as above:

```
0 4 1 1 3 1
0 3 3 1 3 1
4 0 2 2 3 1
```

3.2.1.3 Map plot

This file is for facilitating the plotting of the airport map. This file merely contains the co-ordinates of the nodes in order to plot the map. The co-ordinates are not to scale, but such that the map plotted is intuitive to understand. Every line corresponds to a node and the two entries per line are the x and y co-ordinates respectively. Here is an example of the `Map plot` file for the same example as above:

```
1 0
1 2
0 4
2 4
1 6
```

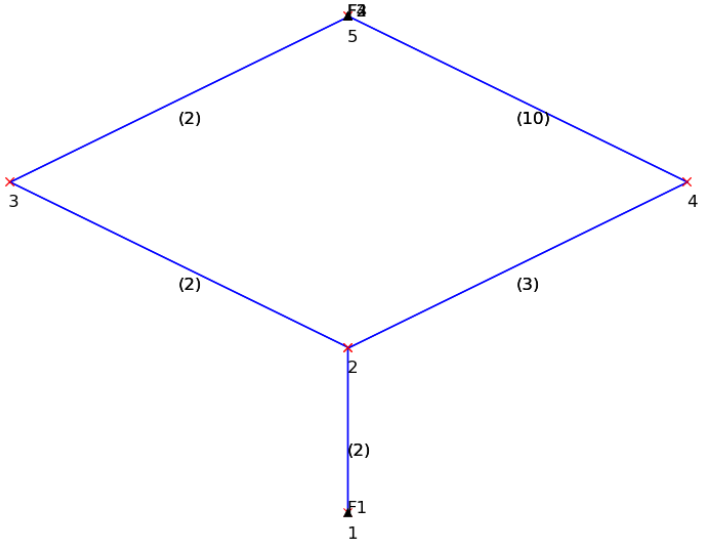

Figure 9: Map plot example

3.2.2 Output

The output file finally generated by the code gives the globally optimal solution identified with the path and time information. Each line in the output file corresponds to each flight scheduled. The entries in each line are in pairs with the first element being the node number while the second one being the time stamp at that node. Here is an example of the solution generated for the example in the input section (Here the node numbers in the plot and the schedule both start from 1):

```
Cost: 24
0: 1 1 - 2 3 - 3 5 - 5 7
1: 1 3 - 2 5 - 4 7
2: 5 7 - 3 8 - 2 9 - 1 10
```

Note how the third (2) flight waits at node 5 as the alternative route of 5 - 4 - 2 - 1 is costlier because the arc length (5-4) is 10 units.

Figure 10 shows the 3D space-time plot of the same solution.

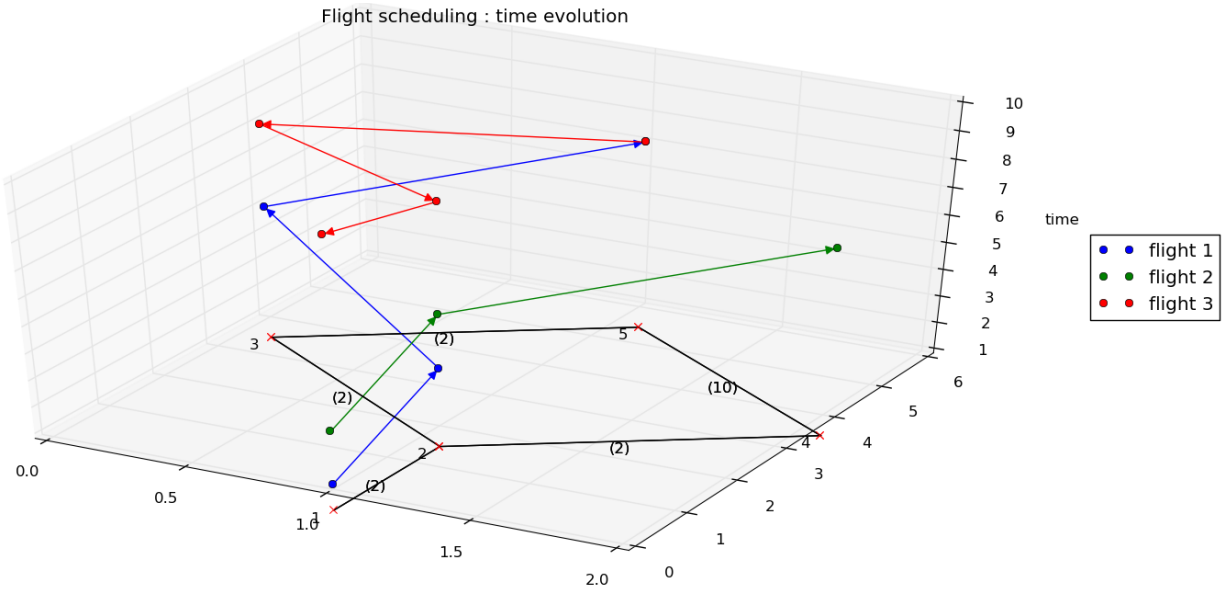

Figure 10: Time evolution plot of the example problem

3.2.3 Process flow

This section will describe how exactly the inputs are translated into the output. The flow of the process in chronological order is as follows:

1. The Python wrapper function calls the C++ function `schedule` which first reads the inputs from the `Map` and `Flights` files
2. `schedule` identifies all the necessary origin-destination pairs and feeds them to the `gen_paths` function
3. `gen_paths` finds all possible paths without cycles for all origin-destination pairs
4. `schedule` then calls the `allcombs` function to generate all flight-path combinations
5. For every combination generated, the `populate` function is called which populates the CPLEX arrays and creates an MILP model
6. `solve` function of CPLEX is called which generates a solution and evaluates the corresponding cost
7. After the MILP model has been solved for every possible flight-path combination and the global minimum identified from the local minima, the final solution is output to the `output` file
8. The `schedule` function exits back to Python which then calls the `feasibility` function that runs a sanity check on the solution to detect spacio-temporal overlaps
9. After deducing the sanity of the solution, the function `animschedule` is called. `animschedule` simulates and animates the schedule in real-time with the help of the `Map plot` file
10. Finally the function `plotschedule` is called which plots the time-evolution of the solution in 3D using the `Map plot` file

4 Results

This section contains the results generated by the global search algorithm on various test cases including problems solved in the referenced papers. An example of the CSIA map is also included in the results.

All results have been generated on an Intel Quad core machine with Xeon E5506 processor and 64Gb ram. CPLEX internally parallelizes the LP solver to the maximum extent. The memory consumption is negligible.

This section contains results illustrated for three problems. The first is a toy problem with only 5 nodes and 4 aircrafts. The second one is a run on the CSIA map with the initial conditions taken from the Bijal etal. [9] paper. The third one is a run on an example from the Roling etal. [12] paper on a bigger map with variable aircraft speeds.

4.1 Toy problem

Flight	Origin	Destination	Start time	Speed	Trailing sep	Priority
1	5	1	0	1	3	1
2	1	5	0	1	3	1
3	1	5	0	2	3	1
4	1	5	0	2	3	1

Table 1: Toy flights

4.1.1 Case 1

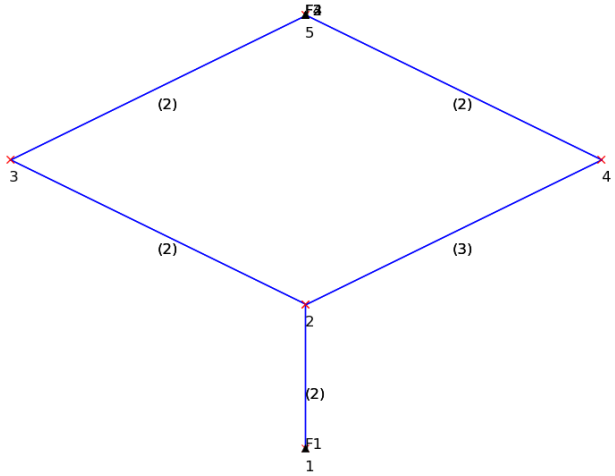

Figure 11: Toy map 1

Time\Flights	1	2	3	4
0	5	1 (W)	1 (W)	1
1	5-4	1 (W)	1	2
2	4 (W)	1	2	3
3	4 (W)	1-2	3	5
4	4	2	5	
5	4-2	2-3		
6	4-2	3		
7	2	3-5		
8	2-1	5		
9	1			

Table 2: Toy solution 1

Cost: 24
Run time: 0.7 seconds

It can be seen that flight 1 takes the longer path (5-4-2-1) with arc 4-2 of length 3 units, as the overall system delay is minimized by this path choice.

Now for the case 2, length of arc 4-5 is changed to 10 units. Which implies the flights should refrain from using this (5-4-2-1) path.

4.1.2 Case 2

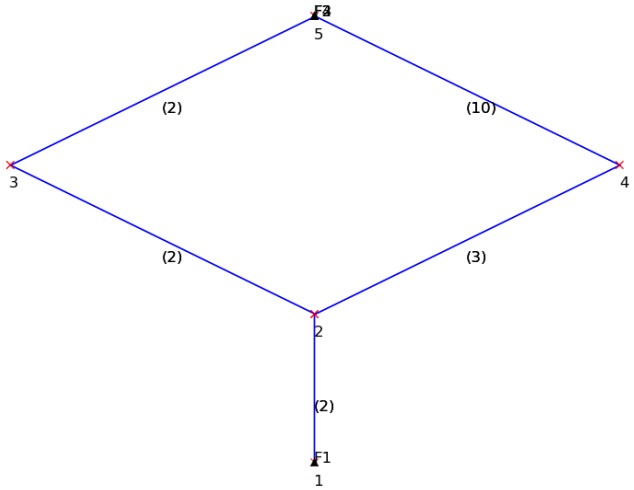

Figure 12: Toy map 2

Time\Flights	1	2	3	4
0	5 (W)	1 (W)	1 (W)	1
1	5 (W)	1 (W)	1	2
2	5 (W)	1	2	3
3	5 (W)	1-2	3	5
4	5 (W)	2	5	
5	5 (W)	2-3		
6	5 (W)	3		
7	5 (W)	3-5		
8	5	5		
9	5-3			
10	3			
11	3-2			
12	2			
13	2-1			
14	1			

Table 3: Toy solution 2

Cost: 29
Run time: 0.7 seconds

As is expected, the flight 1 waits for all other flights to reach node 5 before starting off for node 1 along the same path (5-3-2-1) instead of using the longer path (5-4-2-1) with arc 5-4 (10 units)

4.2 Bijal etal.: CSIA

The initial conditions for this problem have been taken from the paper by Bijal etal. [9]. This problem has 7 flights as follows :

Flight	Origin	Destination	Start time	Speed	Trailing sep	Priority
1	1	8	13	1	3	1
2	1	13	12	1	3	1
3	5	12	6	1	3	1
4	9	2	6	1	3	1
5	1	15	0	1	3	1
6	15	1	0	1	3	1
7	1	8	0	1	3	1

Table 4: Bijal etal. CSIA problem

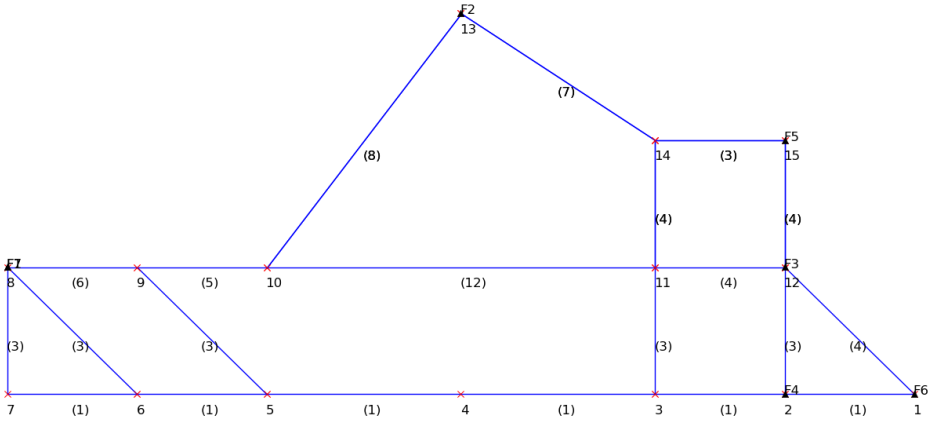

Figure 13: CSIA map

Time Slot	Flt1	Flt2	Flt3	Flt4	Flt5	Flt6	Flt7
0						15--12	1--2
1						15--12	2--3
2						15--12	3--4
3						15--12	4--5
4					1--2	12--1	5--6
5						12--1	6--8
6					2--3	12--1	6--8
7			5--9	9--10		12--1	6--8
8			5--9	9--10	3--4	1	8
9			5--9	9--10			
10				9--10	4--5		
11			9--10	9--10			
12			9--10		5--9		
13		1--2	9--10	10--11	5--9		
14			9--10	10--11	5--9		
15		2--3	9--10	10--11			
16	1--2			10--11	9--10		
17		3--4	10--11	10--11	9--10		
18	2--3		10--11	10--11	9--10		
19		4--5	10--11	10--11	9--10		
20	3--4		10--11	10--11	9--10		
21		5--9	10--11	10--11			
22	4--5	5--9	10--11	10--11	10--13		
23		5--9	10--11	10--11	10--13		
24	5--6	9--10	10--11	10--11	10--13		
25	6--8	9--10	10--11	11--12	10--13		
26	6--8	9--10	10--11	11--12	10--13		
27	6--8	9--10	10--11	11--12	10--13		
28	8		10--11	11--12	10--13		
29		10--13	11--12	12--2	10--13		
30		10--13	11--12	12--2	13--14		
31		10--13	11--12	12--2	13--14		
32		10--13	11--12	2	13--14		
33		10--13	12		13--14		
34		10--13			13--14		
35		10--13			13--14		
36		10--13			13--14		
37		13			14-15		
38					14-15		
39					14-15		
40					15		

Table 5: Bijal etal. CSIA solution

Cost: 186

Time\Flights	1	2	3	4	5	6	7
0					1 (W)	15	1
1					1	15-12	2
2					2	15-12	3
3					3	15-12	4
4					4	12	5
5					5	12-1	6
6			5	9	5 (W)	12-1	6-8
7			5-9	9-10	5 (W)	12-1	6-8
8			5-9	9-10	5 (W)	1	8
9			9	9-10	5		
10			9-10	9-10	5-9		
11			9-10	10	5-9		
12		1	9-10	10-11	9		
13	1	2	9-10	10-11	9-10		
14	2	3	10	10-11	9-10		
15	3	4	10-11	10-11	9-10		
16	4	5	10-11	10-11	9-10		
17	5	5-9	10-11	10-11	10		
18	6	5-9	10-11	10-11	10-13		
19	6-8	9	10-11	10-11	10-13		
20	6-8	9-10	10-11	10-11	10-13		
21	8	9-10	10-11	10-11	10-13		
22		9-10	10-11	10-11	10-13		
23		9-10	10-11	11	10-13		
24		10	10-11	11-12	10-13		
25		10-13	10-11	11-12	13		
26		10-13	11	11-12	13-14		
27		10-13	11-12	12	13-14		
28		10-13	11-12	12-2	13-14		
29		10-13	11-12	12-2	13-14		
30		10-13	12	2	13-14		
31		10-13			13-14		
32		13			14		
33					14-15		
34					14-15		
35					15		

Table 6: MILP CSIA solution

Cost: 166
Run time: 183 minutes

It can be seen that the solution provided by the MILP global search is better than that given by the Bijal etal. paper.

4.3 Roling etal.

This section presents the solution of the problem solved by Roling etal. [12]. This problem has 8 flights as follows :

Flight	Origin	Destination	Start time	Speed	Trailing sep	Priority
1	26	15	7	1	3	1
2	24	15	6	2	3	1
3	25	6	10	2	3	1
4	25	6	8	1	3	1
5	25	6	16	2	3	1
6	24	6	14	1	3	1
7	28	26	0	1	3	1
8	28	26	3	1	3	1

Table 7: Roling etal. problem

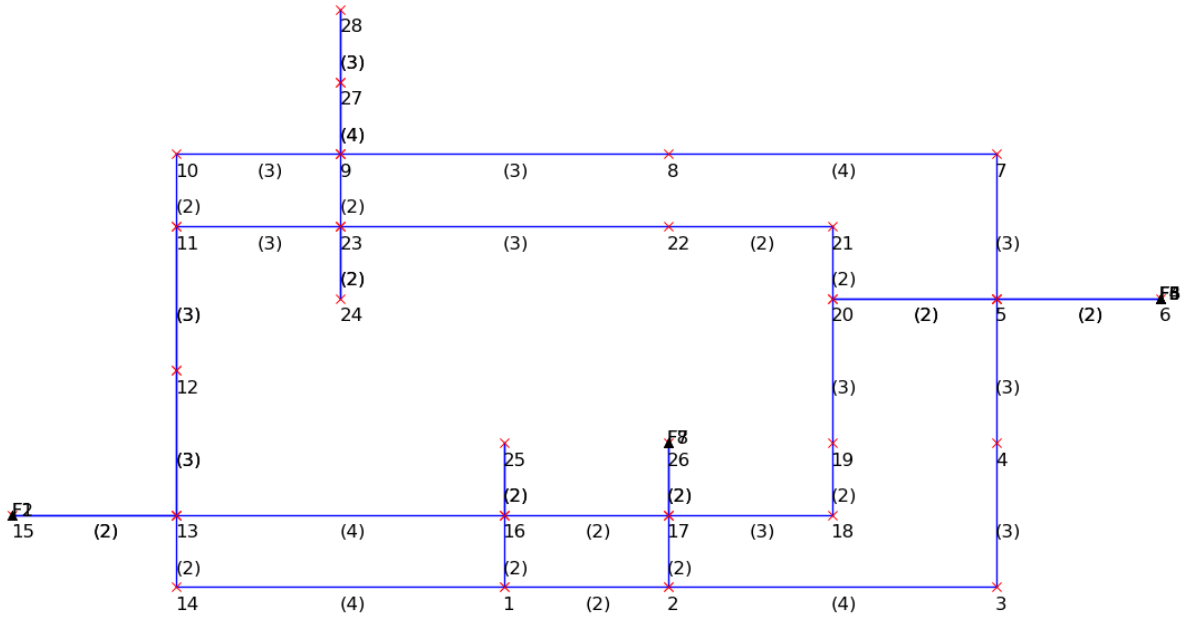

Figure 14: Roling etal. map

Time	Aircraft							
	1	2	3	4	5	6	7	8
0:00							28	
0:10							27-28	
0:20							27-28	
0:30							27-28	
0:40							27-28	
0:50							27-28	
1:00							27-28	28
1:10		24					9-27	27-28
1:20		23-24					9-27	27-28
1:30		23-24					9-27	27-28
1:40		22-23					9-27	27-28
1:50		22-23					9-27	27-28
2:00		22-23		25			9-27	27-28
2:10		21-22		16-25			9-23	9-27
2:20		21-22		16-25			9-23	9-27
2:30		20-21	25	16-25			9-23	9-27
2:40		20-21	16-25	1-16			22-23	9-27
2:50		5-20	16-25	1-16			22-23	9-27
3:00		5-20	13-16	1-16			22-23	9-27
3:10	26	5-7	13-16	1-2			22-23	9-23
3:20	17-26	5-7	13-16	1-2			22-23	9-23
3:30	17-26	5-7	13-16	1-2			22-23	9-23
3:40	17-26	7-8	12-13	2-3			21-22	22-23
3:50	16-17	7-8	12-13	2-3			21-22	22-23
4:00	16-17	7-8	12-13	2-3			21-22	22-23
4:10	16-17	7-8	11-12	2-3			21-22	22-23
4:20	13-16	8-9	11-12	2-3			20-21	22-23
4:30	13-16	8-9	11-12	2-3			20-21	22-23
4:40	13-16	8-9	11-23	2-3			20-21	21-22
4:50	13-16	9-10	11-23	2-3			19-20	21-22
5:00	13-16	9-10	11-23	3-4		24	19-20	21-22
5:10	13-16	9-10	22-23	3-4		23-24	19-20	21-22
5:20	13-16	10-11	22-23	3-4	25	23-24	19-20	20-21
5:30	13-15	10-11	22-23	3-4	16-25	23-24	19-20	20-21
5:40	13-15	11-12	21-22	3-4	16-25	22-23	19-20	20-21
5:50	13-15	11-12	21-22	3-4	1-16	22-23	18-19	19-20
6:00		11-12	20-21	4-5	1-16	22-23	18-19	19-20
6:10		12-13	20-21	4-5	1-2	22-23	18-19	19-20
6:20		12-13	5-20	4-5	1-2	22-23	17-18	19-20
6:30		12-13	5-20	4-5	2-3	22-23	17-18	19-20
6:40		13-15	5-6	4-5	2-3	21-22	17-18	19-20
6:50		13-15	5-6	4-5	2-3	21-22	17-18	18-19
7:00				5-6	2-3	21-22	17-18	18-19
7:10				5-6	3-4	21-22	17-18	18-19
7:20				5-6	3-4	**21**	17-26	17-18
7:30					3-4	20-21	17-26	17-18
7:40					4-5	20-21	17-26	17-18
7:50					4-5	20-21		17-18
8:00					4-5	5-20		17-18
8:10					5-6	5-20		17-18
8:20					5-6	5-20		17-26
8:30						5-6		17-26
8:40						5-6		17-26
8:50						5-6		

Table 8: Roling etal. solution

Cost: 362

Time\Flights	1	2	3	4	5	6	7	8
0							28	
1							28-27	
2							28-27	
3							27	28
4							27-9	28-27
5							27-9	28-27
6		24					27-9	27
7	26 (W)	23					9	27-9
8	26 (W)	23-22		25 (W)			9-23	27-9
9	26	22		25 (W)			23	27-9
10	26-17	21	25	25 (W)			23-22	9
11	17	20	16	25 (W)			23-22	9-23
12	17-16	20-19	1	25 (W)			22	23
13	16	19	2	25			22-21	23-22
14	16-13	18	2-3	25-16		24	21	23-22
15	16-13	18-17	3	16		24-23	21-20	22
16	16-13	17	3-4	16-1	25 (W)	23	20	22-21
17	13	16	4	1	25 (W)	23-22	20-19	21
18	13-15	16-13	4-5	1-2	25	23-22	20-19	21-20
19	15	13	5	2	16	22	19	20
20		13-15	6	2-3	16-13	22-21	19-18	20-19
21		15		2-3	13	21	18	20-19
22				2-3	14	21-20	18-17	19
23				3	14-1	20	18-17	19-18
24				3-4	1	20-5	17	18
25				3-4	2	5	17-26	18-17
26				4	2-3	5-6	26	18-17
27				4-5	3	6		17
28				4-5	3-4			17-26
29				5	4			26
30				5-6	4-5			
31				6	5			
32					6			

Table 9: MILP Roling etal. solution

Cost: 204
Run time: 51 minutes

Even in this case, the solution given by the MILP is better than that by Roling. Although as can be observed, the run time is too high for the exhaustive search to be used on the fly.

5 Conclusions and Future work

The problem being tackled in this project is, routing and scheduling of aircrafts on taxi and runways of the airport for minimizing the delay incurred during congestion, due to suboptimal decision making. The stage I of the project consisted of two major parts. The literature survey and the global optimum identification. An exhaustive search algorithm was implemented using MILP to identify the global best solution for routing and scheduling of aircrafts within a predetermined scheduling horizon. The results generated by this algorithm on problems given in the referenced papers were found to be consistently better than those mentioned therein, thus establishing improvement over other schedules in terms of scheduling quality. However, the algorithm consumes large amount of run time due to the exhaustive nature of it's search.

With the establishment of a method to identify global optimum, the objective of stage II would be to come up with a faster algorithm that could give realistic near-optimal solutions on the fly.

Future work

As has been observed, both deterministic and stochastic approaches have their own shortcomings w.r.t. aircraft scheduling. The deterministic algorithms either have to explore too many possibilities thus raising their run-time to unrealistic levels and making it impossible to use them in a dynamic environment such as ATC. If these algorithms are forced to end prematurely, they tend to use certain greedy approaches (such as first come first serve and one aircraft per iteration) that lead to an early convergence, at best to a local minimum thus providing myopic solutions. Stochastic algorithms on the other hand, although may not get trapped into a local minimum, do not always guarantee a solution, let alone optimality.

Thus a **hybrid approach** which makes use of the best of the two methods must be used to find a near optimal solution, fast. The paper by Gotteland etal. [8] describes such an approach with the combination of Genetic algorithm and A* search, although the paper does not talk about the speed of convergence, it testifies that the hybrid approach indeed gives better results as compared to the individual ones.

Also, instead of dealing with one aircraft at a time (thus fixing it's schedule) and scheduling the remaining aircrafts around the fixed schedules, the concept of *Rolling horizons* as mentioned in papers by Smeltink etal. [10] and Gupta etal. [14] may be explored. **Rolling horizons** essentially extrapolates the first come first serve method to a group of aircrafts. These aircrafts are allowed interact and affect each other's schedules. The aircrafts in the next batch are then scheduled around these aircrafts with some buffer aircrafts between the two sets. These buffer aircrafts' schedules can be affected by either of the batch and by effect, can affect either of the batches.

To incorporate the variability and stochasticity of aircrafts into the schedule, empirical data of the aircrafts scheduled in the past may be used to affect the current schedules. Thus an algorithm that **learns** from the past data and quality of decisions made, could be explored in that regard. The work on lines of Khadilkar [1] but in the domain of micro-optimization could be used as a line of approach.

References

[1] Harshad Khadilkar *Networked Control of Aircraft Operations at Airports and in Terminal Manual ATC with automated feedbackAreas* PhD thesis MIT 2013.

[2] *Airport CDM Implementation Manual*

[3] *Advanced Surface Movement Guidance and Control Systems Manual*

[4] *NextGen implementation plan*, March 2011

[5] Debashish etal. *Enhanced model for Terminal Airspace Capacity Estimation* 2008

[6] Janic. M., Tosic. V. *Terminal Airspace Capacity Model*, Transportation Research, 1982

[7] Changyou Liu, Kaifeng Guo *Airport Taxi Scheduling Optimization Based on Genetic Algorithm*, International Conference on Computational Intelligence and Security, 2010

[8] Gottland etal. *Aircraft Ground Traffic Optimization*

[9] Anant Bijal etal. *Airport Ground Movement Optimization Using Bacterial Foraging Algorithm*, European Journal of Operational Research, 2007

[10] Smeltink etal. *An Optimisation Model for Airport Taxi Scheduling*, Elsevier Science, 2004

[11] Marin A. etal. *Network Design: Taxi planning*, Springer Science, 2008

[12] Roling etal. *Optimal Airport Surface Traffic Planning Using Mixed-Integer Linear Programming*, International Journal of Aerospace Engineering, 2008

[13] Baik H. etal. *Time Dependent Network Assignment Strategy for Taxiway Routing at airports.* Transportation Research, 2006

[14] Prateek Gupta etal. *Generation of Optimized Routes and Schedules for Surface Movement of Aircraft on Taxiways*, 2009