**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY**

# DESIGN OF REAL-TIME DECISION SUPPORT AND AUTOMATION SYSTEMS FOR ATC AND ATFM

**M.Sc. THESIS**

**Barış BAŞPINAR**

**Department of Aeronautical and Astronautical Engineering**

**Aeronautical and Astronautical Engineering Programme**

**JUNE 2015**

**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY**

# DESIGN OF REAL-TIME DECISION SUPPORT AND AUTOMATION SYSTEMS FOR ATC AND ATFM

**M.Sc. THESIS**

**Barış BAŞPINAR**
**(511131102)**

**Department of Aeronautical and Astronautical Engineering**

**Aeronautical and Astronautical Engineering Programme**

**Thesis Advisor: Assoc. Prof. Dr. Gökhan İNALHAN**

**JUNE 2015**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# ATC VE ATFM İÇİN GERÇEK ZAMANLI ÇALIŞAN KARAR-DESTEK VE OTOMASYON SİSTEMLERİNİN TASARIMI

**YÜKSEK LİSANS TEZİ**

**Barış BAŞPINAR**
**(511131102)**

**Uçak ve Uzay Mühendisliği Anabilim Dalı**

**Uçak ve Uzay Mühendisliği Programı**

**Tez Danışmanı: Doç. Dr. Gökhan İNALHAN**

**HAZİRAN 2015**

**Barış BAŞPINAR**, a **M.Sc** student of ITU **Graduate School of Science Engineering and Technology 511131102** successfully defended the **thesis** entitled **"DESIGN OF REAL-TIME DECISION SUPPORT AND AUTOMATION SYSTEMS FOR ATC AND ATFM"**, which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

| | | |
|---|---|---|
| **Thesis Advisor :** | **Assoc. Prof. Dr. Gökhan İNALHAN**<br>Istanbul Technical University | ..... |
| **Jury Members :** | **Prof. Dr. İbrahim Özkol**<br>Istanbul Technical University | ..... |
| | **Dr. Nazım Kemal Üre**<br>Istanbul Technical University | ..... |

**Date of Submission :** **04 May 2015**
**Date of Defense :** **11 June 2015**

***To my family,***

## FOREWORD

I would like to begin with expressing my gratitude to my thesis advisor Prof. Gökhan İnalhan for his support and guidance throughout this study. I would also like to thank Emre Koyuncu for his guidance and support in this project and Cengiz Paşaoğlu for support with his domain experience.

Throughout my master's education, The Scientific and Technological Research Council of Turkey (TÜBİTAK) has honoured and supported me with its prestigious domestic graduate scholarship. Hence, I would like to thank TÜBİTAK for funding my studies for two years in a row.

Finally, I thank my parents for being source of my motivation.

June 2015

Barış BAŞPINAR
Aeronautical & Control Engineer

**TABLE OF CONTENTS**

**Page**

## ABBREVIATIONS

**ACARE** : Advisory Council for Aviation Research and Innovation in Europe
**ACC** : Area Control Center
**AIP** : Aeronautical Information Publication
**AMAN** : Arrival Manager
**ANSP** : Air Navigation Service Provider
**AOBT** : Actual Off-Block Time
**APP** : Approach Control Office
**ATC** : Air Traffic Control
**ATCO** : Air Traffic Control Officer
**ATFM** : Air Traffic Flow Management
**ATM** : Air Traffic Management
**BADA** : Base of Aircraft Data
**CFMU** : Central Flow Management Unit
**DDR** : Demand Data Repository
**DMAN** : Departure Manager
**DST** : Decision Support Tools
**EREA** : Association of European Research Establishment in Aeronautics
**FMC** : Flight Management Computer
**FMS** : Flight Management System
**GUI** : Graphical User Interface
**HALA** : High Automation Levels in ATM
**IATA** : The International Air Transport Association
**ICAO** : International Civil Aviation Organization
**ILS** : Instrument Landing System
**IOBT** : Initial Off-Block Time
**NextGen** : Next Generation Air Transportation System
**PMM** : Point Mass Model
**ROCD** : Rate of Climb/Descent
**SESAR** : Single European Sky ATM Research Programme
**SID** : Standard Instrument Departure
**STAR** : Standard Terminal Arrival Route
**TBO** : Trajectory Based Operations
**TMA** : Terminal Manoeuvring Area
**TOA** : Time of Arrival
**TRACON** : Terminal Radar Approach Control Facilities
**TWR** : Airport Control Tower
**VOR** : VHF Omni-directional Radio Range

## LIST OF TABLES

**Page**

## LIST OF FIGURES

**Page**

**DESIGN OF REAL-TIME DECISION SUPPORT AND AUTOMATION SYSTEMS FOR ATC AND ATFM**

## SUMMARY

Air transportation has an unrivalled position in the world transportation sector, both in the terms of transportation speed and transportation distance. In this context, while the capacity of the aerial transportation have been increased by 300 per cent in the last 15 years, a) infrastructures (airports and connected land-sea-railway transportation networks, and air traffic control systems), and b) land/flight/air traffic control operations, both of which were caught unprepared to that increase, created a bottleneck in terms of both financial and safety. For example, if no intervention is done to the jams in the US aerial domain, it is thought that it will place a burden of 22 billion $ in 2011, and 40 billion $ annually in 2033 to US economy.

Concerning the modernization of aerial transportation, US and Europe, which has the biggest aerial domain in the world in terms of capacity, made all of the research and development programs from present day to 2025 with their NextGen and SESAR programs respectively. Research and development results of the SESAR program foresees that by 2020, the capacity and safety will increase by a factor of three and ten respectively, and environmental pollution per flight and air traffic management costs will decrease by 10 and 50 per cent respectively. At the same time, a remarkable point is that, although the basic concept of the aviation is clear, how to use these hardware skills and information networks at the implementations are a clear research and development area.

In this scope, the mechanisms, which is compatible with new aviation concept and its infrastructure has been designed. Air traffic flow optimization and air traffic control procedural algorithms that can run on macro scale and real time on the ground, and operator decision-support and automation mechanisms that can work synergistically with these algorithms are topic of this thesis.

The first part of thesis focuses to Air Traffic Control part of Air Traffic Management. In this part, a new hybrid system description of modelling the decision process of the air traffic controllers in en-route and approach operations are presented. The model is based on the domain expertise provided by the state airport authority of Turkey. The emulation of air traffic controller decision process in the hybrid model provides realistic conflict resolution maneuvers and separation assurance in 3D, while being computationally tractable. The algorithm has polynomial iteration complexity in the number of waypoints of the aircraft, which makes it scalable to large-scale ATM scenarios with more than 100 aircrafts. The algorithm is validated on the real air traffic data over Istanbul region extracted from the ALLFT+ dataset provided by EUROCONTROL, which includes over 1000 flights in a 24-hour period. The developed algorithm is also integrated into a Boeing 737-800 flight deck simulator with a custom radar display to demonstrate the applicability to existing avionics systems.

The second part of thesis focuses to Air Traffic Flow Management part of Air Traffic Management. Firstly, the air traffic flow in Europe is analysed under cover of ALLFT+ data. And then, airport based network model of Europe is constructed. After that, a slot allocation algorithm is presented for determination of slots of flights in airports in Europe with arrival and departure demand-capacity balances. With this algorithm, aircrafts will take ground delay if capacities are exceeded. This means that, the workload of ATCO will be decreased and unnecessary fuel consumptions will be prevented at the beginning of flight.

# ATC VE ATFM İÇİN GERÇEK ZAMANLI ÇALIŞAN KARAR-DESTEK VE OTOMASYON SİSTEMLERİNİN TASARIMI

## ÖZET

Hava taşımacılığı ulaşım alanında hız ve kat edilebilen mesafeler açışından rakipsiz bir pozisyona sahiptir. Bununla beraber, hava taşımacılığının kapasitesi son 15 yıl içerisinde %300 oranında artarken; havaalanları, ilgili ulaşım ağları, hava trafik kontrolü gibi altyapılar ve yer/uçuş/hava trafik kontrol operasyonları bu artışa hazırlıksız olarak yakalanmıştır. Bu durum hem ekonomi hem de emniyet açısından darboğaz oluşturmuştur. Örneğin A.B.D hava sahasındaki kilitlenmelere müdahale edilmediği taktirde, bu durumun 2022 yılında yıllık 22 Milyar $, 2033 yılında ise yıllık 40 Milyar $ A.B.D ekonomisine yük getireceği tahmin edilmiştir. Bu ekonomik yük dışında hava taşımacılığı operasyonu ile alakalı havada ve yerde yaşanan kazalar son 15 yıl içinde yine %140 oranında artmıştır.

Havayolu taşımacılığının modernizasyonu amacıyla A.B.D tarafından Nextgen ve Avrupa tarafından SESAR programları başlatılmıştır. Bu programlar ile günümüzden 2025 yılına kadar gerçekleştirilmesi gereken araştırma ve geliştirme projelerinin planlamaları yapılmıştır. SESAR programının hedefi araştırma ve geliştirme faliyetleri sonucu 2020 yılı için; kapasitenin üç kat, güvenlik faktörünün ise on kat arttırılacağı, her bir uçuş için çevresel kirletmenin %10 düşürüleceği ve hava trafik yönetimi maliyetlerinin %50 düşürüleceğidir. Bununla beraber bu yapı içinde dikkat çekici nokta havacılıkta yeni paradigmanın temel taşları belirli olmasına rağmen bu temel donanımsal yeteneklerin ve bilgi ağının uygulamalarda nasıl kullanılacağı ise açık bir araştırma ve geliştirme noktasıdır.

Mevcut durumdaki hava taşımacılığına bakıldığında hava trafiği yönetim yapılarının modernizasyonunda otomasyon seviyesinin arttırılmadan hedeflenen sonuçlara ulaşılamayacağı ve artan trafikle ilgili problemlerin üstesinden gelinemeyeceği algılanabilmektedir. Bu sebeple hem hava trafik kontrolü hem de hava trafik akış yönetimi süreçlerini insanlar açısından işleyişte daha basit ve hem ekonomik hem de çevresel açıdan daha verimli bir hale getirebilecek, alınan kararları hızlandıracak alt sistemlerin tasarlanması gündeme gelmektedir. Tez kapsamında bahsedilen ihtiyaçları karşılayabilecek; yeni hava ulaşımı konsepti ve altyapıları ile uyumlu makro ölçekli ve gerçek zamanda yerde koşabilen yeni hava trafik akış optimizasyonu, hava trafik kontrolü prosedürel algoritmaları ve bunlarla entegre çalışacak operatör karar-destek ve otomasyon mekanizmaları geliştirilmiştir.

Tezin ilk kısmında hava trafik yönetiminin hava trafik kontrol kısmına odaklanılmıştır. Artan hava trafiği ile birlikte hava trafik kontrolörlerinin iş yükünün artacağı ve bu yüzden trafiğin kontrolü ile ilgili oluşacak sıkıntıların giderilmesi için hava trafik kontrolörlerinin sektördeki uçakları ayırma sürecinin daha hızlı ve güvenli yapabilecek alt sistemler tasarlanmıştır. Bu sayede hava trafik kontrolörlerinin iş yükleri azaltılarak problematik durum çözülebilecektir. Genel olarak, hava sahası kullanımının en

temel bariyeri olan hava trafik kontrolörü iş yükü şu iki kaynaktan oluşmaktadır; a) koordinasyon, sesli komut ve iletişim, bilgi yönetiminden kaynaklanan rutin görevler, ve b) taktiksel seviyede ayrışma tespiti, durum değerlendirmesi ve ayrışma çözümünden kaynaklanan görevler. Hava Trafiği artışı ile rutin görev yükleri doğru orantılı olarak artarken, taktiksel görev yükleri karşılıklı ilişkilerden dolayı trafik artışının karesi seviyesinde artış göstermektedir. Bu artışla başa çıkabilmek için kontrolör operasyonlarının rutin çalışma süreçleri belirlenmiş ve iki farklı mod için iki ayrı otomat haline getirilmiştir. Bunun için ayrık durumları ve durumların kendi sürekli uzayları olduğu hibrid modellerden yararlanılmış; Yaklaşma/Kalkış (APP) ve Seyir (ACC) durumları için kontrolör operasyonları modellenmiştir. Oluşturulan bu modeller yazılan algoritmalar sayesinde tüm sektör içerisindeki çatışmaları tespit edip ayırmaları sağlayacak şekilde genişletilmiştir. Algoritmalar sayesinde iki moddaki kontrol operasyonları otonomlaştırılmıştır. Bu modellerin ALLFT+ Avrupa Hava Sahası Uçuş Bilgileri verileri ile validasyonu yapılmış ve başarımı değerlendirilmiştir. Sürecin tasarlanan altsistemlerle çok daha hızlı bir şekilde gerçekleştirilebileceği ve ayırmaların sağlanabileceği gösterilmiştir. Tasarlanan algoritmaların hesaplama zamanlarının polinomsal olduğu gösterilmiştir, bu durum literatürde bulunan çalışmaların çoğuna göre bir avantaj sağlamaktadır. Yüksek yoğunluktaki trafiklerde dahi hesaplama zamanı çok düşük olduğu için tasarımın gerçek hayatta kullanımı kolaylaşmaktadır. Ardından mevcut sistemde kullanılan radar ekranlarının benzeri olan bir simülasyon ekranı tasarlanarak algoritmalar bu sistemin içerisine gömülmüştür. Bu sayede algoritmaların gerçek dünyadaki aviyoniklerle entegre kullanılabilmesine imkan tanınmıştır. Tasarlanan simülasyon ekranın mevcut durumda hava trafik kontrolörlerinin kullandıkları radar ekranları temel alınarak tasarlanmış ve gerektiğinde radar ekranı olarak kullanılabilecek şekilde yazılıma dönüştürülmüştür. Ayrıca simülasyon ekranının mevcut durumda kullanılan ekranlarla benzer olması hava trafik kontrolörleri açısından alışkanlıklarını değiştirmeden kullanımına imkan sağlamaktadır, bu durum tasarımın gerçek süreçlerde kullanılabilirliğini kolaylaştırmaktadır. Bunlara ek olarak, tasarımın kullanıcı tarafından iki ayrı modda kullanılabilmesi bir seçenek olarak sunulmuştur. Tasarım istenildiği taktirde tam otomatik istenildiğinde yarı otomatik olarak kullanılabilmektedir. Bu durum otomasyon seviyesinin kullanıcı yani istenilen prosedür ile belirlenmesine olanak sağlamaktadır. Son olarak, tasarlanan simülasyon ekranı ile oluşturulan otonom sistemin entegrasyonunun ardından; simülasyon ekranı ve oluşturulan trafiğe B737-800 kokpit simülatörü entegre edilerek sahada uçan bir uçak olarak eklenmiştir ve gerçek zamanlı simulasyon sonuçları sunulmuştur. Kokpit simülatorünün sisteme entegre edilme işlemi gerçek uçakların radardan aldıkları verilerin radar ekranına yansıtılmasına benzer bir süreç olduğu için yapılan son çalışmayla tasarlanan altsistemin gerçek süreçte çalışabilir bir yapıya dönüştürülme işlemi tamamlanmıştır.

Tezin ikinci kısmında hava trafik yönetiminin hava trafik akış yönetimi kısmına odaklanılmıştır. Avrupa hava trafik akışının anlaşılabilmesi için ALLFT+ veri seti üzerinden analiz yapılmıştır. Yapılan analizler sonucunda Avrupa Hava Trafiğindeki uçuşların birbirini nasıl etkilediklerinin havaalanı bazlı farklı sezonlara göre çıkarılan ağ modelleri ile modellenebileceği sonucuna varılmıştır. Uçuş fazları düşünüldüğünde, uçuş boyunca en çok problemin yaklaşmada yaşandığı ve bunun da havaalanlarının pist kapasite kısıtlamalarından dolayı olduğu görülmektedir. Bu durum havaalanı bazlı bir ağ modelinin modelleme açısından gerçekçiliğini ortaya koymaktadır. Veri seti incelendiğinde, uçuşların büyük bir bölümünün

azınlıktaki bir havaalanı kümesi arasında gerçekleştiği, havaalanlarının büyük bir çoğunluğunda ise saatlik dört uçuştan daha az bir uçuş gerçekleştiği görülmektedir. Bu çıkarım küçük havaalanı tanımlamasını göndeme getirmekte ve bölgesel küçük havaalanlarının modellemede birleştirilerek tek bir birleşik havaalanı olarak sunulması basitleştirmesine olanak sağlamaktadır. Bu yaklaşımla, Avrupa havaalanı ağ modeli 304 havaalanından oluşacak şekilde modellenmiştir. Ardından, havaalanlarının kalkış ve iniş kapasite-talep dengesini göz önünde bulundurarak tüm ağdaki uçuşların kalkış zamanlarını düzenliyen bir algoritma yazılmıştır. Bu sayede uçuş öncesinde uçaklara müdahale edilip geçikmeler verilerek kontrolörlerin iş yükleri azaltılabilmekte ve gereksiz yakıt tüketiminin önüne geçilebilmektedir. Son olarak, bir durum senaryosu üzerinden algoritmanın doğruluğu ve çalışma şekli denenmiştir. Heatrow Havaalanının kapasitesi yarıya düşürülerek bu kapasite düşümünün diğer havalimanlarında ne kadarlık rotarlara sebep olacağı ALLFT+ uçuş verileri üzerinden incelenmiştir. Algoritmanın kapasite düşümlerinde uçakları yerde tutarak gereksiz yakıt tüketimini ve iş yükü artımını engelleyebileceği ortaya konulmuştur.

# 1. INTRODUCTION

The basis of the current air traffic management (ATM) system was constituted by The International Civil Aviation Organization (ICAO) after The Chicago Convention in 1944. Although the system performed reliably over the years, the steady growth of the air transportation industry calls for fundamental changes in how ATM systems operate. Currently, the number of commercial aircraft flights around the globe is approximately 26 million per year. If this number grows with the expected rate, there will be 48.7 million flights per year in 2030 [2], hence the airspace capacity should also increase accordingly to accommodate the increase in the air traffic volume. By 2050, the number of passengers will increase from 6.5 million to 44 million passengers per day. Having 16 billion passengers and 400 million tons of cargo per year will be another issue in 2050 [3]. In order to cope with such high demand, new infrastructures should be built and new efficient security measures should be designed.

Generally, it can be said that ATM consists of two basic components that are air traffic control (ATC) and air traffic flow management (ATFM). ATC is related to processes that provide tactical separation services, that is, real-time separation procedures for conflict detection and resolution. ATC is usually performed by human controllers who watch over three-dimensional regions of airspace, called sectors, and dictate local movements of aircraft. Their aim is to maintain separation between aircraft while moving traffic as expeditiously as possible and presenting the traffic in an orderly and useful manner to the next sector. As such, ATC actions are of a more tactical nature and primarily address immediate safety concerns of airborne flights. ATFM, on the other hand, refers to processes of a more strategic nature. ATFM procedures detect and resolve demand-capacity imbalances that jeopardize safe separation. By keeping the workload of air traffic controllers to a manageable level, traffic flow management can be viewed as the first line of defense in maintaining system safety. Whereas ATC

generally controls individual aircraft, ATFM usually adjusts aggregate traffic flows to match scarce capacity resources [4].

### 1.1 Purpose of Thesis

Through these objectives, the thesis proposes automation tools to perform routine separation provision tasks of controllers for two different types of flight modes in Chapter 2 and proposes algorithms to allocate slots from perspective of CFMU for flights in European airports in Chapter 3.

Chapter 2 focuses to ATC part of ATM. In Chapter 2, an automation tool is represented to automatize the separation assurance procedure. In this tool, the method utilizes hybrid automata formalism to model controller action obtained from both Arrival/Departure (APP) and En-route (ACC) "what-if" procedures. The hybrid models are envisioned to solve conflicts considering the aircraft performance limitations and environmental model with minimum changing in the current flight plan of the aircraft. It is supposed that, airspace and flow capacity considerations are handled strategically in the context of 4D Reference Business Trajectory (RBT) planning, and aircraft execute their own flight intent trajectories subject to tactical ATC intervenes. The ACC and APP automata ensure that the aircraft maintains a safe separation from other aircraft during both en-route and arrival/departures respectively.

Chapter 3 focuses to ATFM part of ATM. Firstly, the air traffic flow in Europe is analysed under cover of ALLFT+ data. And then, a slot allocation algorithm is presented for determination of slots of flights in airports in Europe with arrival and departure demand-capacity balances.

## 2. HIGH LEVEL AUTOMATION SUPPORT FOR ROUTINE SEPARATION PROVISION TASK

### 2.1 Purpose

In the current ATM operations, air traffic controllers monitor the flight trajectories through the radar screens and make cognitive judgements supported by the automation tools to interpret and resolve conflicts. The workload of the air traffic controllers stems from two sources; a) the routine task load, which is based on the coordination, verbal communication and data management, b) the tactical task load, which is associated with conflict detection, situation monitoring and conflict resolution [5]. As the air traffic volume increases, the routine task load increases proportionally to the size of the traffic volume increase, while the tactical task load increases approximately proportional to the square of the increase in the air traffic volume due to the cross-relations between the flight trajectories.

In compliance with the need for improving the ATM systems to increase the airspace capacity, the traditional responsibilities of the air traffic controllers based on verbal communication and clearance decisions are aimed to evolve through the use of new functionalities and tools coming from Single European Sky ATM Research (SESAR) and NextGen visions [5–7]. The paradigm shift from clearance-based control to trajectory-based control with Trajectory Based Operations (TBO) functionalities are not only expected to redefine the existing roles of the controllers, but also yield additional responsibilities for them. Therefore the future ATM operations are going to require enhanced and high-level automation support for routine decision-making procedures.

Development of automation tools for large-scale ATM scenarios is a challenging subject. First of all, such system should emulate the decision process of an actual air traffic controller closely in order to generate realistic three dimensional (3D) conflict resolution maneuvers. Secondly, the algorithm should be highly scalable with respect to the number of aircrafts considered for conflict detection and resolution, in order to cope with the increasing air traffic volume. Finally, the algorithm should be verified

on the real air traffic data to demonstrate its applicability to the real world problems. The main objective of this chapter is to develop highly scalable automation tools that addresses the challenges described above for performing routine separation provision tasks of air traffic controllers for approach and en-route flight modes.

### 2.1.1 Related work

A number of projections are made by different organizations for the future mechanisms of Air Navigation Service Providers (ANSPs) and pilots. For instance, in the FlightPath2050 vision document of the High Level Group on Aviation Research of the European Commission [8], the classical roles of the pilot and the air traffic controller remains the same. On the contrary, ACARE (Advisory Council for Aeronautics Research in Europe) envisions free-flight and non-controlled airspace for Air Traffic Control in 2050. The ACARE reports [9, 10] discuss free flight as a viable alternative to full automation. In contrast with the ACARE vision, EREA (The Association of European Research Establishments in Aeronautics) favors a highly or fully automated Air Transport System for the far future [11, 12]. Contrary to these perspectives, Higher Automation Levels in ATM (HALA) suggests that a new role assignment needs to be derived by considering three decision criteria, which are the best time, decision place, and best player. HALA envisions a higher level of automation utilization for the unpredictable events that occur with low available reaction time, and humans using Decision Support Tools (DST) whenever the reaction time permits [13].

In parallel with these different visions, a variety of approaches have been studied in the literature associated with conflict detection and separation assurance problems. In the first group of these approaches, researchers focused on the free flight concept. In this approach a centralized traffic controller does not exist and the conflict detection and resolution are performed airborne. The second class of approaches is centered on the ground operations and development of automation tools to improve the system capacity. The main difference between these two approaches is that the free flight concept does not use the flight path intent information for conflict detection unlike the ground based approach, which utilizes this information in the conflict detection phase.

First we review the works that only consider conflict detection. Most of these works are based on the free flight concept. The work in [14] defines horizontal and vertical

planes as the protected zone of the aircraft, and then uses these zones for conflict detection. The developed algorithm performs a fixed horizon lookahead to propagate these zones and then checks for potential collisions. The propagation process is limited to simplified aircraft dynamics. Shewchun [15] considers more complex dynamics, such as along track and cross track fluctuations, which translates to bearing and acceleration uncertainties. The conflict detection problem is solved using Linear Matrix Inequalities and positive semi-definite programming. There are also approaches that allow computation of conflict probability in free flight. The work in [16] models the trajectory prediction error as a normal distribution, with zero mean and a covariance matrix with eigenvectors in the along-track and cross-track directions. The protected zones are defined according to the minimum separation values and these stochastic error dynamics, which allows computation of conflict probability in the horizontal plane. The work in [17] focuses on the conflict detection for ground based operations in the horizontal plane, but unlike the probabilistic approach authors use the flight plans for conflict detection. Vink et al. [18] also use flight plans for conflict detection. In addition, authors construct uncertainty areas around the trajectories for modeling unpredictable aircraft dynamics, and the conflict detection can be achieved for 3D trajectories. The probabilistic methods have also been applied to the ground-based approaches. For instance, [19] constructs a mathematical model by solving a partial differential equation with Dirichlet boundary conditions to calculate the conflict probability. Note that the aforementioned works are limited to conflict detection, while the automation of the air traffic control system would require algorithms than can perform both conflict detection and resolution.

On the other hand, there exists methods that focus only on the conflict resolution problem or separation assurance. The work in [20] uses the potential field method for conflict resolution in free flight. Although the method is computationally cheap, it is well known that the potential field methods have inherent limitations, such as being stuck in the local minima and oscillating solutions in the presence of narrow passages and dense environments [21]. Hence, the applicability of these methods to separation assurance in realistic ATM scenarios is debatable. Tomlin's work [22, 23], which is based on the free flight concept, develops a hybrid automata framework for conflict resolution. The conflict detection problem is not directly addressed, however the

authors provide ad-hoc definitions of alert zones and protected zones to acknowledge the issue. The conflict resolution is achieved by defining a fixed set of evasion maneuvers as the discrete modes of the nonlinear aircraft dynamics and solving the Hamilton-Jacobi-Bellman (HJB) equation to compute conflict resolution maneuvers. The introduction of hybrid dynamics to air traffic control problem leads to interesting results and provides a natural formal framework for addressing the complex dynamics associated with the problem. However, due to high computational complexity of solving the HJB equation, the case studies use simplified aircraft dynamics and consider only conflict resolution in the horizontal plane. Besides Tomlin's work, Bayen also developed a hybrid automata for separation assurance in the horizontal plane in [24]. Although the aforementioned paper mostly focuses on air traffic flow, the developed automaton is also used for separation assurance. Overall, the high demand of these methods on computation limits their scalability, hence it is difficult to make these algorithms work in a realistic ATM scenario with multiple aircrafts. Moreover, the limitation to conflict detection and resolution to 2D (horizontal plane) is also not a realistic representation of how ATM works. Many separation assurance maneuvers require the aircraft to change the altitude and in many practical situations vertical maneuvers might be the only option to achieve conflict resolution. Hence it is important for an automated ATC system to operate in 3D for realistic applications.

There also a number of methods that combine conflict detection and resolution. Durand [25] models the conflict resolution problem as quadratic program and solves the optimization problem via semi-definite programming combined with a randomization scheme. The algorithm is able to detect and resolve conflicts in 3D and also utilizes flight plans of the aircrafts. However, the algorithm has exponential complexity with respect to the number of aircrafts, which limits its applicability to large-scale automated ATM scenarios. Both [26, 27] follows a similar approach and perform conflict resolution via solving a linear program. The conflict detection is achieved by computing fixed horizon lookahead. These methods are limited to conflict detection and resolution in the horizontal plane and use simplified aircraft dynamics. Overall, it can be observed that the algorithmic approaches to ATM problems either tend to use simplified aircraft dynamics and limit the conflict resolution maneuvers to horizontal plane for the sake of reducing the computational complexity, or tend

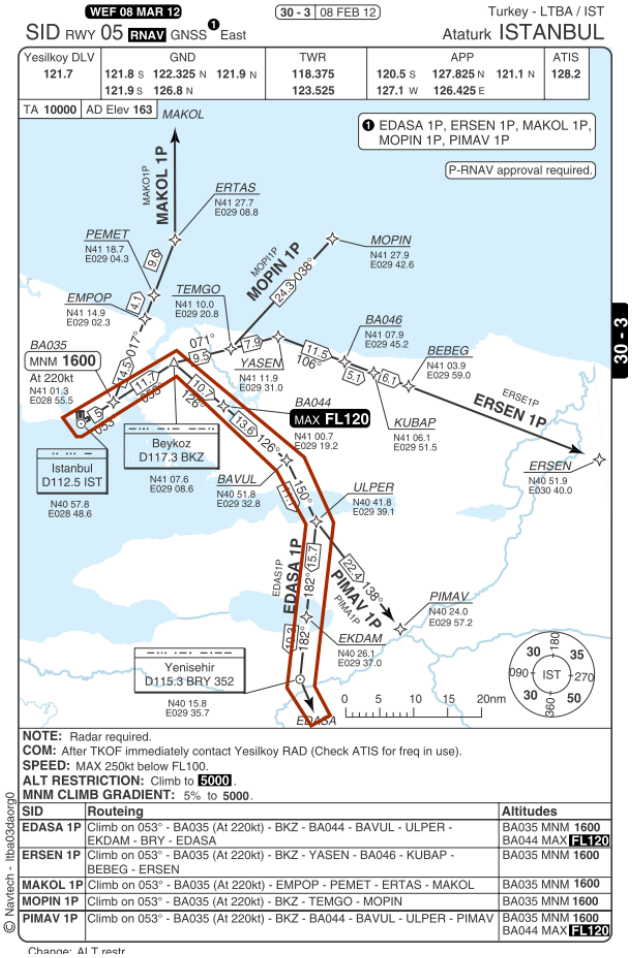
to incorporate more realistic conflict resolution maneuvers and aircraft dynamics by sacrificing computational complexity and hence limiting the scalability.

In conjunction with the algorithmic works above, there are also software tools developed for conflict detection and separation assurance. Most of these tools are semi-automated, that is the software can detect potential conflicts and suggest conflict resolution maneuvers, but the final decision is still provided by the user. The tool develop by Yang [28] is based on the free flight concept. The conflict detection is achieved using the algorithm from [16]. The tool provides the pilot with a probability map of conflicts and suggests maneuvers from a fixed set to resolve the potential conflicts. For the ground-based control systems, NASA's Center-TRACON Automation System (CTAS) [29] and MITRE's URET [30] are developed for providing decision support to air traffic controllers. CTAS consists of three different sub tools, which are traffic management advisor, descent advisor and final approach sequencing tool. These sub tools work together to handle conflicts during the approach and en-route traffic. Similar to CTAS, URET uses flight plans to assist the generation of 4D trajectories for conflict detection in the en-route phase. In addition to these tools, some commercial tools are available for improving the arrival flow, which are named as Arrival Manager (AMAN) products [31].The main features of these products are presented in the Table 2.1. These products mainly focus on arrival sequencing with separation assurance. Advisory actions can also be generated relative to different cost functions to determine the sequenced arrival in some products.

It should be emphasized that these existing tools do not address the conflict detection and resolution of arrival and departure traffic in a joint manner (Note that some of

**Table 2.1**: Properties of the AMAN products.

| | Integrated AMAN /DMAN | Delay absorbance in En-route | Sequenced Arrival | Information about sequence to all ATCo | Traffic Monitoring | Opt. Advisories |
|---|---|---|---|---|---|---|
| MAESTRO | X | X | X | X | | |
| OSYRIS | X | | X | X | X | X |
| 4D PLANNER | | | X | X | | |
| IBP/SARA | | X | X | X | | |
| OPTAMOS | X | X | X | X | | X |
| SELEX AMAN | | | X | X | | X |




| SID | Routeing | Altitudes |
|---|---|---|
| EDASA 1P | Climb on 053° - BA035 (At 220kt) - BKZ - BA044 - BAVUL - ULPER - EKDAM - BRY - EDASA | BA035 MNM 1600 BA044 MAX FL120 |
| ERSEN 1P | Climb on 053° - BA035 (At 220kt) - BKZ - YASEN - BA046 - KUBAP - BEBEG - ERSEN | BA035 MNM 1600 |
| MAKOL 1P | Climb on 053° - BA035 (At 220kt) - EMPOP - PEMET - ERTAS - MAKOL | BA035 MNM 1600 |
| MOPIN 1P | Climb on 053° - BA035 (At 220kt) - BKZ - TEMGO - MOPIN | BA035 MNM 1600 |
| PIMAV 1P | Climb on 053° - BA035 (At 220kt) - BKZ - BA044 - BAVUL - ULPER - PIMAV | BA035 MNM 1600 BA044 MAX FL120 |

**Figure 2.1**: Departure (SID) chart for Ataturk Airport (Runway 05).

the AMAN products are integrated with Departure Managers (DMANs) as shown in the first column Table 2.1, which means that arrival traffic is integrated with departure traffic in the context of runway capability, not in the context of separation assurance). However, there are practical scenarios where these two traffics should be handled together (i.e. an arrival aircraft can have conflict with a departure aircraft and vice versa). For instance, Fig. 2.1 shows EDASA1P, which is a Standard Instrument Departure (SID) route (a route followed after the takeoff) and Fig. 2.2 PIMAV1B is a Standard Terminal Arrival (STAR) route (a route followed during the approach) at the Istanbul Ataturk Airport. These charts show that the arriving and departing aircrafts share a common route, hence potential conflicts can indeed exist in these regions. Thus it is desirable that a fully automated ATC system should treat the conflict resolution in departure and arrival traffics as a joint problem.

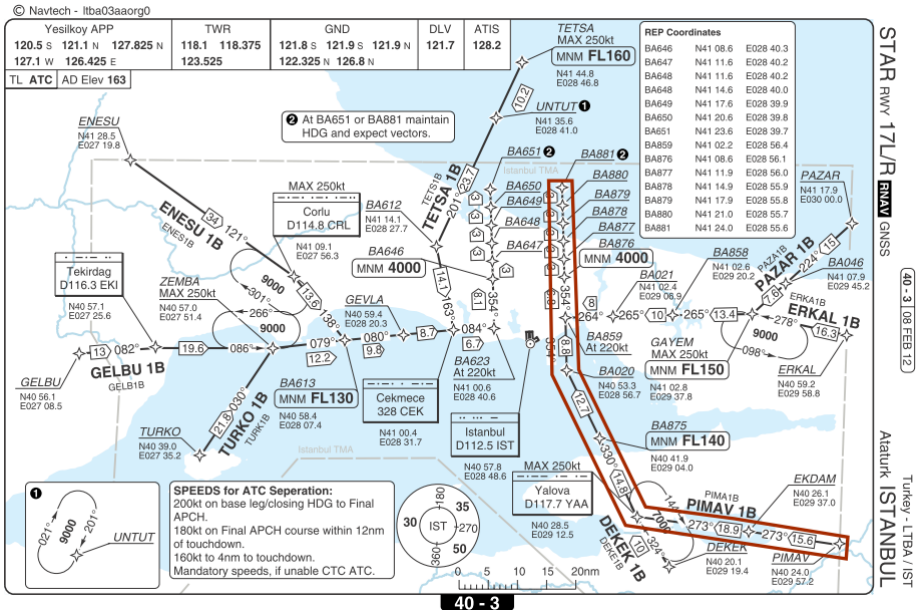



**Figure 2.2**: Arrival (STAR) chart for Ataturk Airport (Runway 17L/R).

### 2.1.2 Overview of the developed approach and summary of contributions

The main objective of this study is to provide a scalable and fully automated ATC system that can be verified on the real air traffic data and can be integrated seamlessly into the existing ATM systems used in the air transportation industry. This objective is achieved by modeling and emulating the decision process of an air traffic controller based on the language of the hybrid automata. The developed control algorithm detects conflicts and ensures separations in horizontal and vertical planes for both en-route and approach phases for the departure and arrival traffic. Compared to the existing approaches in the literature, this algorithm makes the following contributions:

- Compared to the previous algorithmic approaches, such as [23–26], the developed algorithm has better scalability properties and presents a more realistic approach to the automated air traffic control problem that generates separation maneuvers in 3D. This is due to fact that the algorithm is built upon the domain expertise. The developed algorithm emulates the decision process of an actual air traffic controller, which considers 3D separation assurance maneuvers. Hence the algorithm alleviates all the complex search and optimization procedures of the previous algorithmic works and instead uses a deterministic automaton represented by a formal language that models the decision process. It is shown that the algorithm has polynomial complexity in the number of aircrafts and the number of waypoints in their flight plans, hence the algorithm can handle conflict detection

and resolution in large-scale ATM scenarios in 3D. The simulation results show that the algorithm can detect and resolve conflicts for more than a 100 aircrafts in real-time.

- Compared to the existing tools (such as [28–30] and products in Table 2.1), the developed algorithm handles conflict resolution of arrival and departure traffic jointly. Hence the conflicts that arise in the regions outlined in Figs. 2.1 and 2.2 can be handled by the developed algorithm.
- The algorithm is verified on the real traffic data provided by the ALLFT+ dataset. A 24-Hour time window was considered and the simulation results show that the developed algorithm was able to detect and resolve conflicts across approximately 1000 flights.
- Finally, the integration of the algorithm to existing aircraft navigation systems was considered. A radar display GUI embedded with the developed conflict resolution algorithm was developed as a decision support tool for the air traffic controller. The overall system was tested with a piloted Boeing 737-800 flight deck simulator, where the conflict between the piloted aircraft and the simulated air traffic was resolved by the developed algorithm.

## 2.2 Problem Definition

This paper studies the problem of conflict detection and separation assurance for en-route and approach phases, as well as arrival sequencing and the integration of control of arrival-departure traffic. In this context, conflict is defined as a predicted violation of a separation standard. Informally, this definition tells that a conflict exists whenever two aircrafts positions are going to be in a certain distance of each other for some future time $t$. Hence a $4D$ trajectory(location and time) prediction for each aircraft is necessary for the conflict detection. Furthermore, conflicts are usually treated separately for vertical and horizontal dimensions. If the inequality **2.1** holds, the vertical separation is violated. If the inequality **2.2** holds, then horizontal separation is violated. In practice, if the vertical separation is ensured, then horizontal separation is not checked.

$$|z_i(t) - z_j(t)| < v_s \tag{2.1}$$

$$\sqrt{\left(x_i(t)-x_j(t)\right)^2+\left(y_i(t)-y_j(t)\right)^2} < h_s \tag{2.2}$$

In inequalities **2.1** and **2.2**, $i$ and $j$ indicate $i^{th}$ and $j^{th}$ aircrafts, $z$ indicates altitude, $x$ and $y$ represents locations in the plane, $v_s$ is the minimum vertical separation, $h_s$ is the minimum horizontal separation. The objective of the air traffic controller is to send control actions to the aircraft to ensure that separation violations never occur.

## 2.3 Flight Model

For the conflict detection process a performance model of the aircraft is necessary for trajectory propagation. The performance model is also required for the prediction of the aircraft trajectory after a conflict resolution maneuver is suggested. Together with the performance model, a flight management system (FMS) model is also required to establish the link between the aircraft trajectory prediction and the control actions sent by the air traffic controller (ATC). In this section, aircraft performance model and FMS

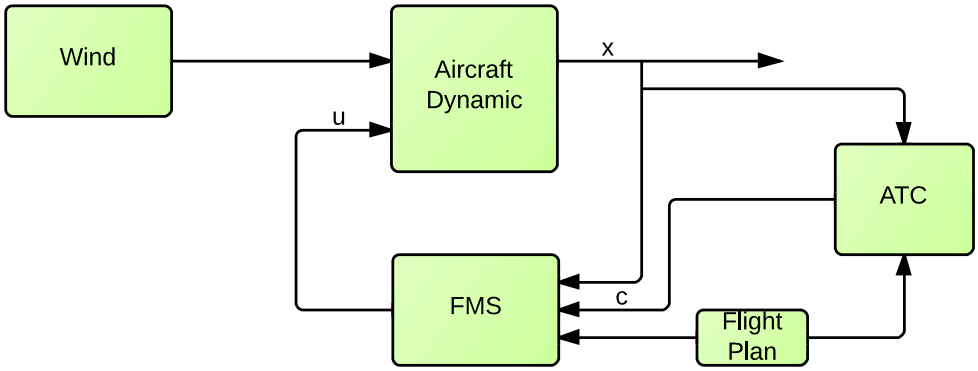



**Figure 2.3**: Block diagram of the flight model components.

models are presented, which are modified version of the models that were previously presented by Lygeros and Glover [32,33]. The Fig. 2.3 shows the modified model used in the paper. In this model, unlike the Lygeros and Glover's model, ATC can affect the FMS directly and get information about the situation of the state variables.

Each flight model has the following parts; the flight plan, the aircraft dynamics, the flight management system (FMS), the wind model and the ATC actions. The overall model is a hybrid dynamical system. The continuous dynamics stem from the aircraft performance model and the discrete dynamics stem from the flight plan and the logic variables embedded in the FMS and ATC actions.

### 2.3.1 Flight plan

The flight plan includes a sequence of way-points, $\{O(i)\}_{i=0}^{M}$ in three dimensions, where $O(i) \in \mathbb{R}^3$. Each waypoint also has a time variable, which represent aircraft's arrival time to the waypoint, $\{t(i)\}_{i=0}^{M}$. The rest of the time variables in the flight plans are ignored, except the sector entrance and exit times. In order to generate the rest of the time variables, BADA flight performance model was used [34] and aircrafts were assumed to have the same speed between two successive waypoints in the en-route phase.

### 2.3.2 Aircraft dynamics

The point Mass Model (PMM) is used for modeling the aircraft dynamics from the point of view of ATC. The model is a nonlinear dynamical system with three control inputs and six state variables. The state variables of the aircraft are the horizontal position ($x_1$ and $x_2$), altitude ($x_3$), the true airspeed ($x_4$), the heading angle ($x_5$) and the mass of the aircraft ($x_6$). The control inputs of the aircraft are the engine thrust ($u_1$), the bank angle ($u_2$) and the flight path angle ($u_3$). The wind acts as a disturbance on the aircraft dynamics, which is modeled by the wind speed, $W = (w_1, w_2, w_3)$. The equations of aircraft motion are [33]:

$$\dot{x_1} = x_4 \cos(x_5)\cos(u_3) + w_1 \quad \textbf{(2.3)}$$

$$\dot{x_2} = x_4 \sin(x_5)\cos(u_3) + w_2 \quad \textbf{(2.4)}$$

$$\dot{x_3} = x_4 \sin(u_3) + w_3 \quad \textbf{(2.5)}$$

$$\dot{x_4} = -\frac{C_D S \rho(x_3) x_4^2}{2x_6} - g\sin(u_3) + \frac{u_1}{x_6} \quad \textbf{(2.6)}$$

$$\dot{x_5} = -\frac{C_L S \rho(x_3) x_4}{2x_6}\sin(u_2) \quad \textbf{(2.7)}$$

$$\dot{x_6} = -\eta u_1 \quad \textbf{(2.8)}$$

In the equation set above, aerodynamic lift and drag coefficients are represented by $C_L$ and $C_D$, total wing surface area is $S$, air density is represented as $\rho$ and the thrust-fuel consumption coefficient is represented as $\eta$. These coefficients and the other parameters such as bounds on the speed and mass are provided by the BADA database [34].

### 2.3.3 Flight management system (FMS)

The FMS basically works like a control system for the aircraft. It is responsible for generating the control inputs ($u$) based on the state variables ($x$), flight plan information and the ATC actions. FMS model has 8 discrete modes. These discrete modes are: flight level (FL), way-point index (WP), acceleration mode (AM), climb mode (CM), speed hold mode (SHM),flight phase (FP), reduced power mode (RPM) and troposphere mode (TrM). These modes are defined relative to the BADA [34] database for the calculation of control inputs. Detailed information about these modes can be found in [32, 33]. FMS controller can be divided into two main components. The first component is the vertical and along track motion control with $u_1$ (thrust) and $u_3$ (flight path angle) and the second component is the horizontal motion control with $u_2$ (bank angle).

Speed and the Rate of Climb/Descent (ROCD) are set by the thrust and flight path angle. In our model, FMS is used for tracking the desired speed $V_{nom}$, which depends on the altitude and aircraft type and is determined by the airline. ATC can change this speed by a rate of 2% for increasing or decreasing the aircrafts speed. If aircraft cruises at a constant altitude, the FMS sets the flight path angle to zero, so that the equations produce zero ROCD. Then thrust is used for controlling the speed through the Eq. **2.6**. In climbing or descending motion, the thrust is set to a fixed value. Thus, speed can be controlled via the flight path angle. ROCD is controlled through the Eq. **2.5**. Horizontal position control can be achieved with controlling the bank angle ($u_2$). First, the heading angle is controlled through the Eq. **2.7** and next the horizontal position of the aircraft ($x_1$ and $x_2$) can be adjusted with the heading angle ($x_5$) through the Eqs. **2.3**-**2.4**.

### 2.3.4 ATC actions

ATC can intervene 4 main parameters of this model. ATC can revise the waypoint index, can increase or decrease $V_{nom}$ value and can set flight path angle and bank angle to a fixed value for a time period. The detailed description of these actions is the main subject of this study and will be presented in the further sections.

## 2.4 Decision Process of the Air Traffic Controller

This section provides information on the decision process of an air traffic controller (ATC). Procedural actions of ATC for en-route and approach operations are defined and decision mechanisms are presented. ATCs are responsible for maintaining separation of aircrafts, organizing air traffic flow and providing information to pilots. Controllers usually make a decision based on the following:

- Calculated information based on filed flight plans
- Transmitted information from pilot with voice communications
- Transmitted information from adjacent sector controllers via phone/line
- Perceived information from facilities located on the ground

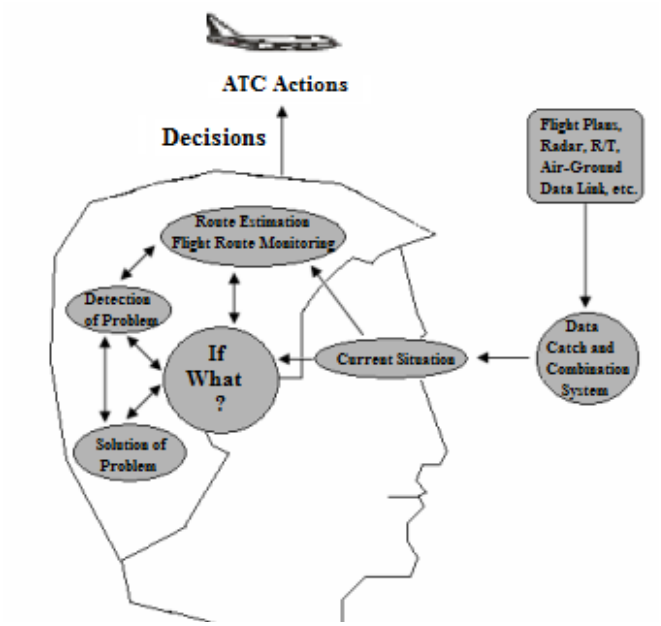



**Figure 2.4**: Decision Process of Air Traffic Controller [1].

This information is transmitted to the controller via a human-machine interface. Decision process of a controller along with the midterm estimations is presented in the Fig. 2.4. The controller evaluates the input information and analyses current situation. Route estimation and flight route monitoring are also included in this process. Afterwards, if the controller detects a conflict, it selects a conflict resolution maneuver and then transmits a corresponding controller action/clearance to the pilot.

### 2.4.1 Decision mechanism of the en-route(ACC) controller

ACC controllers are usually inclined towards not modifying aircrafts routes, unless a safety-critical situation exists (bad weather, conflict detection etc.). Initially, the

ACC controller checks the flight levels of the aircrafts as soon as they enter to his/her sector. If the vertical separation ($v_s = 1000ft$) is ensured (See Eq. **2.2**), controller does not request any actions from the aircraft. For all the other aircrafts, flights in the opposing directions are required to have at least $1000ft$ vertical separation. If the controller chooses an altitude change action between the flights with the same direction and the same flight level, he/she gives climb or descent clearance at least an amount of $2000ft$. The preferred choice of the controller is descent, since the climb action depends on the capability of aircraft at that given time. Next, the controller checks flight routes in the same flight levels. If any conflicts exist between a pair of aircrafts, the controller checks the horizontal separations of the conflicted aircrafts. If any horizontal separation loss is detected by controller, he/she asks a series of if-what questions. If aircrafts do not follow the same route after the detected conflict point, which means the aircrafts are crossing flights, controller considers lateral separation and gives direct routing clearance and bank angle change for the aircraft. Controller also intervenes to the way-point index of aircraft in direct routing action (See Fig. 2.5). For instance, ATC can make the aircraft skip a sequence of waypoints in order to route the aircraft ahead of its original plan.

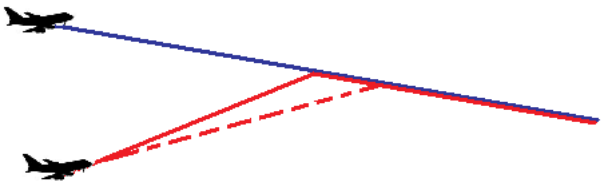


**Figure 2.5**: Direct Routing Action.

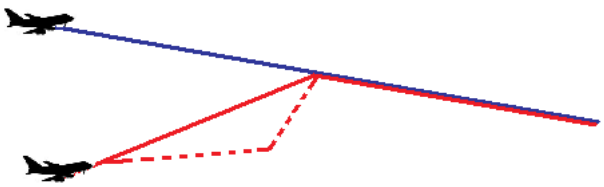


**Figure 2.6**: Delaying Motion with Vector for Spacing.

If aircrafts follow the same route after the conflict point, controller considers longitudinal separation. At this point the controller can select direct routing action, altitude change action or delaying motion, and then gives the corresponding clearance to the pilot. Two different delaying motions are defined; reducing the speed, and vector for spacing (Fig. 2.6). Vector for spacing consists of deviating the aircraft from its original flight plan for a fixed time period.

When lateral separation is ensured and longitudinal separation loss is estimated after the conflict point, controller tries the direct routing action. The routing action covers aircrafts that have the same routes after the conflict point. Controller chooses the altitude change action or delaying motion and then gives the corresponding clearance to the pilot.

### 2.4.2 Decision mechanism of the approach(APP) controller

APP controller is responsible for the flights in the terminal areas, mostly arriving at or departing from one or more aerodromes. Controller takes over departure flights from Tower (TWR) control, separates them from the other departure or arrival flights in order to establish them to their flight routes and hands them over to the ACC control. APP controller also takes over the arrival flights from the ACC controller, separates and sequences these flights for the landing and hands them over to the TWR control.

Standard Instrument Departure (SID) is a procedure for departing flights. If any separation losses are detected for departure flights, controller gives direct routing clearance, delaying motion or horizontal motion at a defined altitude. If any separation losses are detected for the departure flights, controller gives direct routing clearance or delaying motion. After the controller finds a solution for separation losses; he/she transmits this clearance to the pilot swiftly.

Standard Terminal Arrival Routes (STAR) procedures are defined for the arrival flights. Initially, controller sequences the arrival flights relative to the estimated arrival time. Vertical separation ($v_s = 1000ft$), horizontal separation ($3nm$) and longitudinal separation ($5nm$), which is necessary for Instrumental Landing (ILS), must be ensured respectively by the controller.

Detailed information about controller actions, separations, STAR, SID, and ILS can be found in ICAO Doc.4444 [35] and Aeronautical Information Publication of Turkey (AIP) [36].

## 2.5 Automata Representation of the ATC Model

In reality, the controller monitors all flights in his/her sector, compares flight routes of the aircrafts, checks the aircraft′s current states, and then predicts and determines

separation losses. If any separation loss is predicted between two flights, controller requests an action to ensure separation. Controllers have several action types and they go through these actions procedurally. After determining action for a solution to the separation problem, controller gives a clearance to the pilot/FMS.

We constitute two models in this section to represent real air traffic controller decisions in en-route and approach airspaces. These models are used for finding a solution to the separation problem between two aircrafts, and these models will be generalized separation assurances for multiple aircraft (grater than 2) in the next section. Basically, these models emulate the ATC decision procedures that are explained verbally in the previous section.

These models are presented as deterministic automata. An automaton is a formal definition method that accepts an appropriate language with well defined rules; detailed information about automata theory can be found in [37, 38]. An automaton has events and states which are represented as circles and arcs in the directed graph representation. The model has transition functions, which defines the relationship between transitions between states. In the deterministic automata, only one predefined transition is allowed to happen from one state to another. Formally, a deterministic automaton ($G$) is a five-tuple

$$G = (X, E, f, x_0, X_m), \tag{2.9}$$

where

- $X$ is the set of states
- $E$ is the finite set of events
- $f : X \times E \rightarrow X$ is the transition function
- $x_0$ is the initial state
- $X_m$ is the set of final states

### 2.5.1 ACC automaton

Air traffic controller in en-route is defined as the following deterministic automaton:

$$\text{ACC-Automaton} = (X, E, f, x_0, X_m), \tag{2.10}$$

ACC Automaton has eight discrete states that represents the following controller actions.

$$X = \{q_i : i = 0, \ldots, 7\} \tag{2.11}$$

$$x_0 = q_0, \tag{2.12}$$

- $q_0$ is the initial state, which refers to the null action. The controller does not intervene in aircraft's flight plan in $q_0$.
- $q_1$ denotes the direct routing for the first flight.
- $q_2$ denotes the altitude change for the first flight.
- $q_3$ denotes the delaying motion for the second flight with reduced speed.
- $q_4$ denotes the delaying motion for the second flight with vector for spacing.
- $q_5$ denotes the altitude change for the second flight.
- $q_6$ denotes the direct routing for both flights.
- $q_7$ denotes the change in bank angle for both flights.

ACC Automaton can terminate at any states, hence $X_m = X$.

ACC Automaton has eight different events, which are functions of aircraft$'$s states. These functions have boolean outputs which can be either 0 or 1. The finite set of events is:

$$E = \{e_i : i = 1, \ldots, 8\} \tag{2.13}$$

For the definition of events, we define six different functions with $\{0,1\}$. These function$'$s inputs are aircraft$'$s states and flight plans. We refer to these functions are helper functions, which determine aircraft$'$s separation in the flight route. The controller monitors aircraft$'$s current state and flight plan, then predicts when an aircraft goes to which point and determines values of these functions relative to this prediction. $a_0$ is an altitude check function which controls altitudes of two flights. If altitudes of two flights are within the violation tolerance at any point, $a_0$ outputs 1, otherwise 0. $a_1$ is an intersection/conflict check function, which checks routes of two flights. If any two routes are within the violation tolerance, $a_1$ outputs 1, otherwise 0. $a_2$ is a horizontal separation check function which checks the horizontal separation ($5nm$) of two flights. If the separation is ensured, $a_2$ outputs 1, otherwise 0. The longitudinal separation is checked with the $a_3$ function. Another important check is whether two flights follow the same route after the intersection point. If they do so, $a_4$ function outputs 1, otherwise 0. The last function is the horizontal separation check function,

which checks the intervened flights with the all other flights in the same sector and flight level. If the separation is ensured, $a_5$ function outputs 1, otherwise 0.

The relationship between the events and the helper functions are given as follows. Let $\cap$ denotes *and*, $\cup$ denotes *or*, $a$ denotes $a = 1$ and $\bar{a}$ denotes $a = 0$.

$$e_1 = \bar{a}_0 \cup (a_0 \cap \bar{a}_1) \cup (a_0 \cap a_1 \cap a_2 \cap \bar{a}_4) \cup (a_0 \cap a_1 \cap a_2 \cap a_3 \cap a_4)$$

$$e_2 = (a_0 \cap a_1 \cap a_2 \cap a_4 \cap \bar{a}_3) \cup (a_0 \cap a_1 \cap \bar{a}_2 \cap a_4)$$

$$e_3 = (a_0 \cap a_1 \cap a_2 \cap a_4 \cap a_3 \cap a_5)$$

$$e_4 = (a_0 \cap a_1 \cap a_4 \cap \bar{a}_5)$$

$$e_5 = (a_0 \cap a_1 \cap \bar{a}_4 \cap \bar{a}_2)$$

$$e_6 = (a_0 \cap a_1 \cap \bar{a}_4 \cap a_2 \cap a_5)$$

$$e_7 = (\bar{a}_0 \cap a_5)$$

$$e_8 = (a_0 \cap a_1 \cap \bar{a}_4 \cap \bar{a}_5)$$

Directed graph representation of the deterministic automaton of the en-route controller is shown in Fig. 2.7. The transition function $f$ can be read off from this figure.

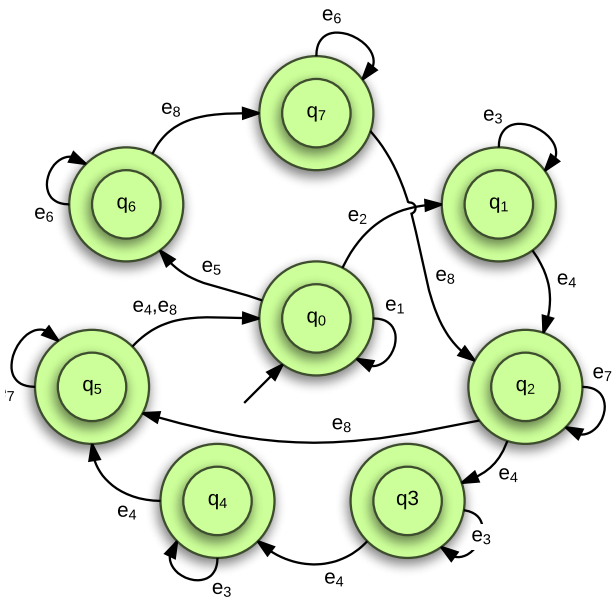



**Figure 2.7**: Deterministic Automaton of En-route Controller.

### 2.5.2 APP automaton

The automaton for the air traffic controller in the approach mode is defined as follows:

$$\text{APP-Automaton} = (X, E, f, x_0, X_m), \tag{2.14}$$

APP Automaton has eight discrete states which represent the defined controller actions.

$$X = \{q_i : i = 0, \ldots, 7\} \tag{2.15}$$

$$x_0 = q_0, \tag{2.16}$$

- $q_0$ is initial state and defined as the null action.
- $q_1$ denotes the direct routing for the second flight (departure flight).
- $q_2$ denotes the delaying motion for the second flight, which is applied with reducing of climb/descent speed (ROCD).
- $q_3$ denotes the horizontal motion for the second flight at a defined altitude. In $q_3$, departure flight starts climbing to a defined altitude, then moving along track and after passing the arrival flight with a vertical separation continues to climb to its original route.
- $q_4$ denotes the direct routing for the first flight.
- $q_5$ denotes the increase of ROCD for the first flight.
- $q_6$ denotes the delaying motion for the second flight with vector for spacing.
- $q_7$ denotes the delaying motion for the second flight with reduced speed

APP Automaton can terminate in any state, hence $X_m = X$.

APP Automaton has nine different events, which are functions of aircraft′s states. These functions have boolean outputs that are either 0 or 1. The finite set of events is:

$$E = \{e_i : i = 1, \ldots, 9\} \tag{2.17}$$

We define eight different helper functions in order to describe the events. The helper function $a_0$ checks the routes of arrival flights and departure flights. If any intersection/conflict occurs in these routes, $a_0$ outputs 1, otherwise 0. $a_1$ checks the routes of the respective departure flights, in order to check the conflict, separation must be also checked between these flights. $a_2$ is defined as the vertical separation (1000ft) check function. $a_3$ is defined as the horizontal separation (3nm) check function. $a_4$ checks separations between two sequenced arrival flights, if separations are ensured, $a_4$ outputs 1, otherwise 0. $a_5$ function ensures separation between the arrival sequenced flights and approach. If the first aircraft is faster than the second aircraft, this function outputs 1, otherwise 0. Last two functions are related to the separation check between all flights. $a_6$ is the function that checks the separation of a departure flight with all

other departure flights. If all separations are ensured, $a_6$ outputs 1, otherwise 0. $a_7$ function makes the same check between an arrival flight and all other arrival flights in the same manner. The relations between the events and helper functions are:

$e_1 = a_0 \cap \bar{a}_2 \cap \bar{a}_3$

$e_2 = ((a_0 \cap a_2) \cup (a_0 \cap \bar{a}_2 \cap a_3)) \cap a_6$

$e_3 = a_1 \cap \bar{a}_2 \cap \bar{a}_3$

$e_4 = ((a_1 \cap a_2 \cap a_5) \cup (a_1 \cap \bar{a}_2 \cap a_3 \cap a_5)) \cap a_6$

$e_5 = \bar{a}_4$

$e_6 = a_4 \cap a_5 \cap a_7$

$e_7 = (a_0 \cap \bar{a}_2) \cup (a_0 \cap a_2 \cap \bar{a}_3) \cup \bar{a}_6$

$e_8 = (a_1 \cap \bar{a}_2) \cup (a_1 \cap a_2 \cap \bar{a}_3) \cup \bar{a}_6$

$e_9 = \bar{a}_4 \cup \bar{a}_7$

Directed graph representation of the deterministic automaton of the approach controller is shown in Fig. 2.8. The transition function $f$ can be read off from this figure.

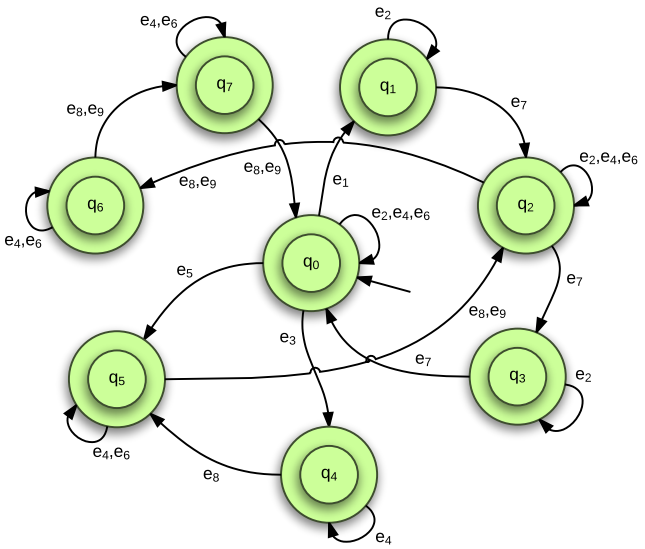



**Figure 2.8**: Deterministic Automaton of the Approach Controller.

## 2.6 Algorithm

In this section, the automaton models from the previous section are generalized to multiple aircraft flying in the same sector. The computational complexity of the en-route and approach controller algorithms are also discussed at the end of the section.

### 2.6.1 ACC algorithm

Psuedocode of the "ACC Controller Algorithm" for the en-route controller is given at the Fig. 2.9.

```
   Input: Flight plans of all enroute flights and current state variables.
   Output: Controller actions and new flight routes of all enroute flights.
1  if a new aircraft comes to the sector then
2      Check separation of all flights in the sector
3      if there are any unseperated flights then
4          foreach Unseperated flight do
5              Set flight1 to oldest aircraft in unseperated flights foreach
               Unseperated flights with flight1 do
6                  Set flight2 to the closest unseperated flight to flight1
7                  Generate controller action from the ACC automaton (Fig. 2.7)
8                  for flight1 and flight2
9                  Set new flight1 and flight2 routes to the new flight routes
```

**Figure 2.9**: ACC Controller Algorithm.

In the algorithm, each flight is compared individually with all the flights in the sector, predicted separation losses are determined and flights with loss of separation are saved into the memory. Next, the flight trajectory with the highest level of conflict is compared pairwise with the other flights with separation loss. These two flights are passed to the ACC Automaton (Fig. 2.7) and the controller action is determined. This procedure is applied between all conflicted flight trajectories. This algorithm is called again when a new aircraft enters to the sector.

### 2.6.2 APP algorithm

Psuedocode of the "APP Controller Algorithm" for the approach controller is given at the Fig. 2.10.

First, arrival flights are sequenced relative to the estimated arrival times. Next, estimated separation losses are determined in three different ways, which are separation losses between two arrival flights, separation losses between two departure flights and separation losses between departure and arrival flights. In the second part of the algorithm, three loops are executed. These three loops find a controller action with APP Automaton for three different ways of separation losses with algorithms in Fig. 2.11 and Fig. 2.12. These three loops are repeated until all separations are ensured in

**Input**: Flight plans of all approach flights and current state variables.
**Output**: Controller actions and new flight routes of all approach flights.

```
1 if a new aircraft comes to the sector then
2     Generate sequence for arrival flights relative to the estimated arrival times
3     Check separation of all flights in the sector while an unseperated flight
      exists do
4         Call SA Algorithm for the same flight phase (Fig. 2.11) in Approach
          (Arrival)
5         Set controller actions and new flight routes
6         Call SA Algorithm for cross flight phases (Fig. 2.12) in Approach
          (Arrival and Departure)
7         Set controller actions and new flight routes
8         Call SA Algorithm for the same flight phase (Fig. 2.11) in Approach
          (Departure)
9         Set controller actions and new flight routes
```

**Figure 2.10**: APP Controller Algorithm.

the sector. This algorithm runs again when a new aircraft enters to the sector. Just like the en-route controller, algorithm always gives priority to the highest level of conflict to begin the conflict resolution process.

**Input**: Flight set.
**Output**: Controller actions and new flight routes.

```
1 if any unseparated flight exists in the flight set then
2     foreach Unsepareted flight in the flight set do
3         Set flight1 to the first coming aircraft in unseparated flights in the flight
          set
4         foreach Flights in the flight set that are unseperated with flight1 do
5             Set flight2 to closest unseparated flight to flight1
6             Generate controller action from APP automata (Fig. 2.8) for flight1
              and flight2
7             Set new flight1 and flight2 routes to the new flight routes
8             Set Controller actions
```

**Figure 2.11**: SA Algorithm for the same flight phase in Approach.

### 2.6.3 Computational complexity analysis

The presented algorithms have two parts, which are conflict detection and separation assurance. We present the computational complexity analysis for both. The main difference is that the complexity of the conflict detection is independent from the number of flights with separation losses, whereas the complexity of conflict resolution is dependent on the number of flights with separation losses in the sector.

**Input**: Flight set 1 and Flıght Set 2.
**Output**: Controller actions and new flight routes.
1 **if** *any unseparated flight exists between the flight set 1 and flight set 2* **then**
2  **foreach** *Unsepareted flights between the flight set 1 and flight set 2* **do**
3  Set flight1 to the first coming aircraft in unseparated flights in the flight set 1
4  **foreach** *Flights in the flight set 2 that are unseperated with flight1* **do**
5  Set flight2 to closest unseparated flight to flight1
6  Generate controller action from APP automata (Fig. 2.8) for flight1 and flight2
7  Set new flight1 and flight2 routes to the new flight routes
8  Set Controller actions

**Figure 2.12**: SA Algorithm for cross flight phases in Approach.

In the conflict detection parts of the en-route and approach algorithm, each aircraft trajectory is checked with other aircrafts trajectories for the determination of conflicts. Hence the conflict detection depends on the number of flights in the sector and the number of waypoints of each aircraft. In the en-route algorithm (Alg. 2.9), the computation time of the conflict detection is proportional to the Eq. **2.18**.

$$\propto \sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{k=1 k\neq j}^{n_i} \left(wp_j \times wp_k\right) \tag{2.18}$$

In Eq. **2.18**; $m$ is the number of flight levels; $n_i$ is the number of flights in flight level $i$; $wp_j$ is number of waypoints in flight $j$ and $wp_k$ is number of way point in flight $k$.

In the approach algorithm, the computation time of the conflict detection is the sum of three conflict detection algorithms, which are ran between arrivals-arrivals, departures-arrivals and departures-departures; each conflict detection algorithm's computation time is proportional to the Eq. **2.19**.

$$\propto \sum_{i=1}^{n} \sum_{j=1 j\neq i}^{m} \left(wp_i \times wp_j\right) \tag{2.19}$$

In Eq. **2.19**; $wp_j$ is the number of waypoints in flight $j$ and $wp_i$ is the number of waypoints in flight $i$ for each conflict detection algorithms. $n$ is the number of flights in arrival, $m$ is the number of flights in departure for conflict detection between arrival and departure. If conflict detection algorithm runs between departures; then $n = m$ and $n$ is the number of flights in departure. If conflict detection algorithm runs between arrivals; then $n = m$, and $n$ is the number of flights in arrival. The algorithms have the same computation times for the same number of arrivals and departures. The scalability of

the algorithms with respect to the number of aircraft and the number of waypoints are shown in Fig. 2.13 for the en-route algorithm and Fig. 2.14 for the approach algorithm. These plots are obtained by running the algorithm on the ALLFT+ dataset. It can bee seen from these figures that algorithm is able to handle a large number of aircrafts (more than 100) in real-time.

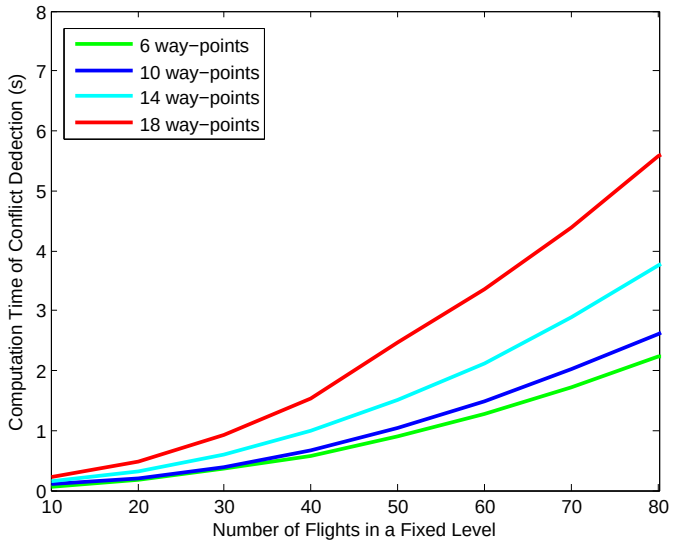



**Figure 2.13**: Computation time of CD algorithm in en-route control (Fig. 2.9).

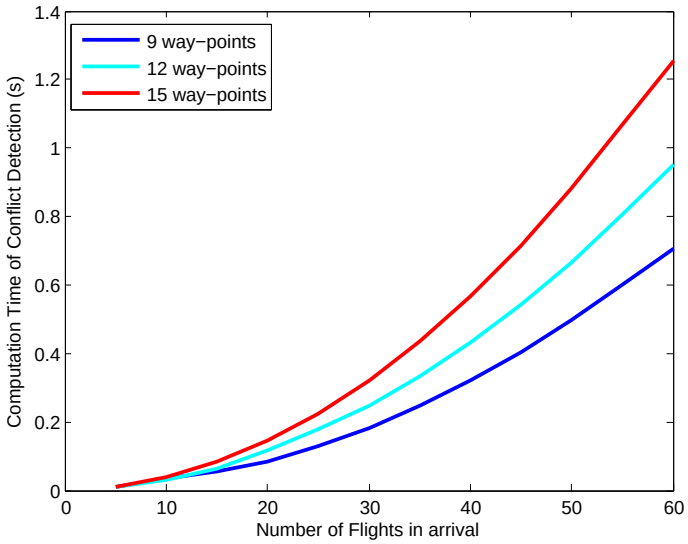



**Figure 2.14**: Computation time of CD algorithm in approach control (Fig. 2.10).

In the conflict resolution part, algorithms computation time is affected by the number of aircrafts that have loss of separation and each aircraft is checked against the other aircrafts for the assurance of separation after controller action. The computation time for the conflict detection performed within the conflict resolution loop for a single aircraft is shown in Fig. 2.15 for the en-route algorithm and Fig. 2.16 for the approach

algorithm. These plots are obtained by running the algorithm on the ALLFT+ dataset. Once again it can be seen that the algorithm can operate in real-time for the large-scale ATM scenarios.

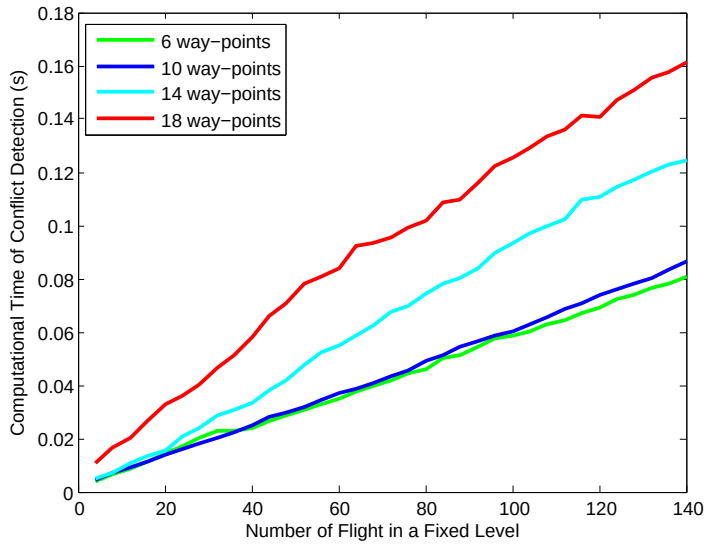



**Figure 2.15**: Computation time of CD in SA for a single aircraft in En-route.

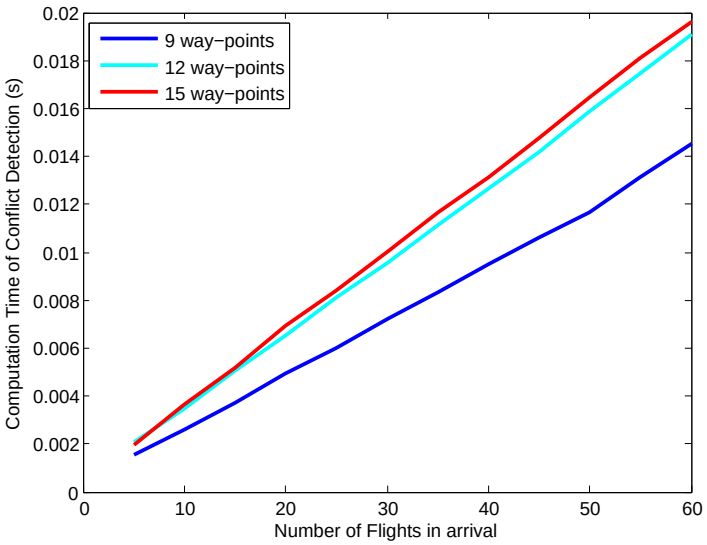



**Figure 2.16**: Computation time of CD in SA for a single aircraft in Approach.

In the en-route algorithm, the computation time of conflict resolution is proportional to Eq.**2.20**.

$$\propto \sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{k=1}^{l} b_k \left( wp_j \times wp_k \right) \tag{2.20}$$

In Eq. **2.20**; $m$ is the number of flight levels; $n_i$ is the number of flights in flight level $i$; $l$ is the number of flights that have separation losses; $wp_j$ is number of waypoints in flight $j$ and $wp_k$ is number of waypoints in flight $k$. In this part of the algorithm, computation time is also proportional to the number of controller actions, which is

represented by $(b_k)$. Note that this number is upper bounded by the number of states in the Fig. 2.7. In the approach algorithm, the computation time of the conflict resolution is proportional to Eq. **2.21**.

$$\propto \sum_{i=1}^{n} \sum_{j=1}^{m} b_j \left( wp_i \times wp_j \right) \tag{2.21}$$

In Eq. **2.21**; if conflict detection algorithm runs between one departure flight and other departure flights; $n$ will be the number of flights in departure, $m$ will be the number of flights in departure which have separation losses. If conflict detection algorithm runs between one arrival flight and other arrival flights; then $n$ is number of flights in arrival, $m$ is the number of flights in arrival which have separation losses.

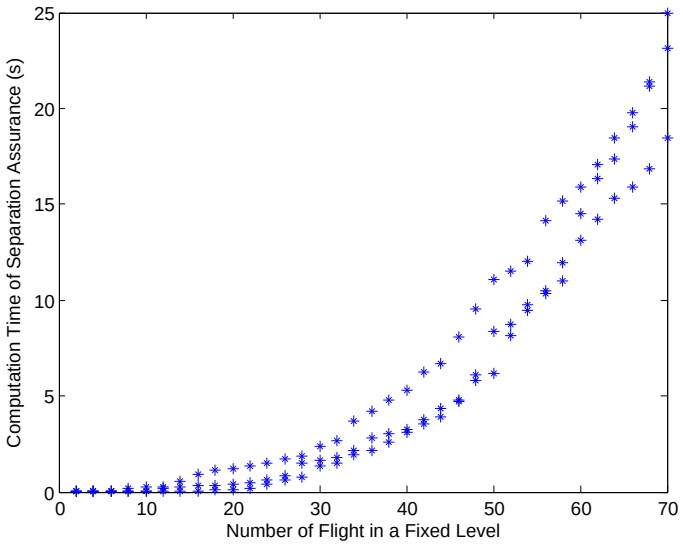



**Figure 2.17**: Computation time of SA for En-route between multiple aircrafts.

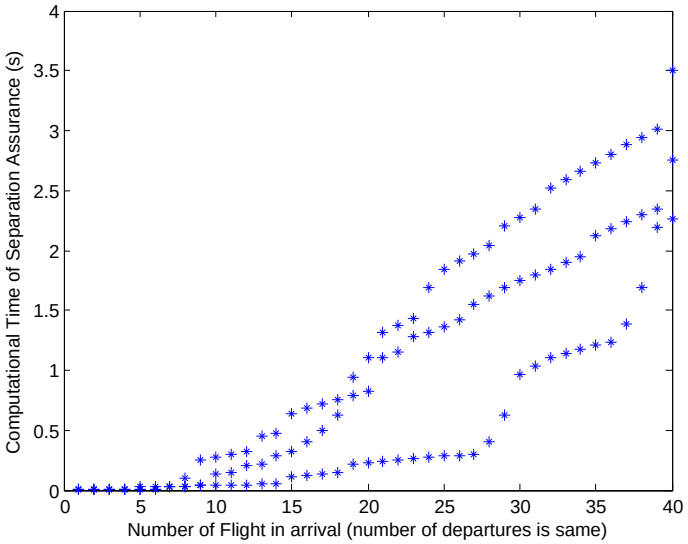



**Figure 2.18**: Computation time of SA for Approach between multiple aircrafts.

Finally, computation times for the multiple aircraft separation assurance is shown in Fig. 2.17 for en-route control and Fig. 2.18 for approach control. These figures are obtained with three different flight sets, with 10 way points each. Overall, the computational complexity analysis shows that the developed algorithm is highly scalable in the number of aircrafts and able to operate in real-time for large-scale ATM scenarios.

## 2.7 ATM Scenario, Control and Simulation Environment: Radar Display

An ATM Scenario, Control and Simulation Environment, which is similar to radar display using by Air Traffic Controllers is designed and produced. This environment, which is shown in Figure 2.19, is used for simulation and control of any traffic scenario, which is defined in a text file. In scenario text file; callsign, sector entrance time, cruise level, route, departure airport and arrival airport for each flight is presented. A flight use these informations as input variables and each flight is realized with these inputs and defined aircraft and FMS models. Besides of these pseudo flights, a flight deck simulator can be integrated to the environment. In this situation, integrated flight deck simulator will use input variables and it don't use aircraft and FMS model of designed simulator environment, it flies with own dynamics. State variables of this flight is updated periodically relative to data which comes from flight deck simulator. Actually, integrated flight deck simulator and simulation environment will be presented as an implementation in result section.

Additional to scenario text file, six different text file are exist for definition of sector that are sector boundary text file, Airways text file, SID routes text file, STAR routes text file, VORs/fixes text file and Airport text file. Sector boundaries are exist as latitude and longitude sets in sector boundary text file. Operated sector are defined relative to this file, in default mode this is the 'IST ACC'. For en-route operation in defined area, en-route airways are exist in Airways text file. Airways is shown as dashed lines in display (Fig.2.19). For approach operation in defined area, SID and STAR routes are exit in related text files. These routes are used in operations, and shown for each flight in approach at related panels.

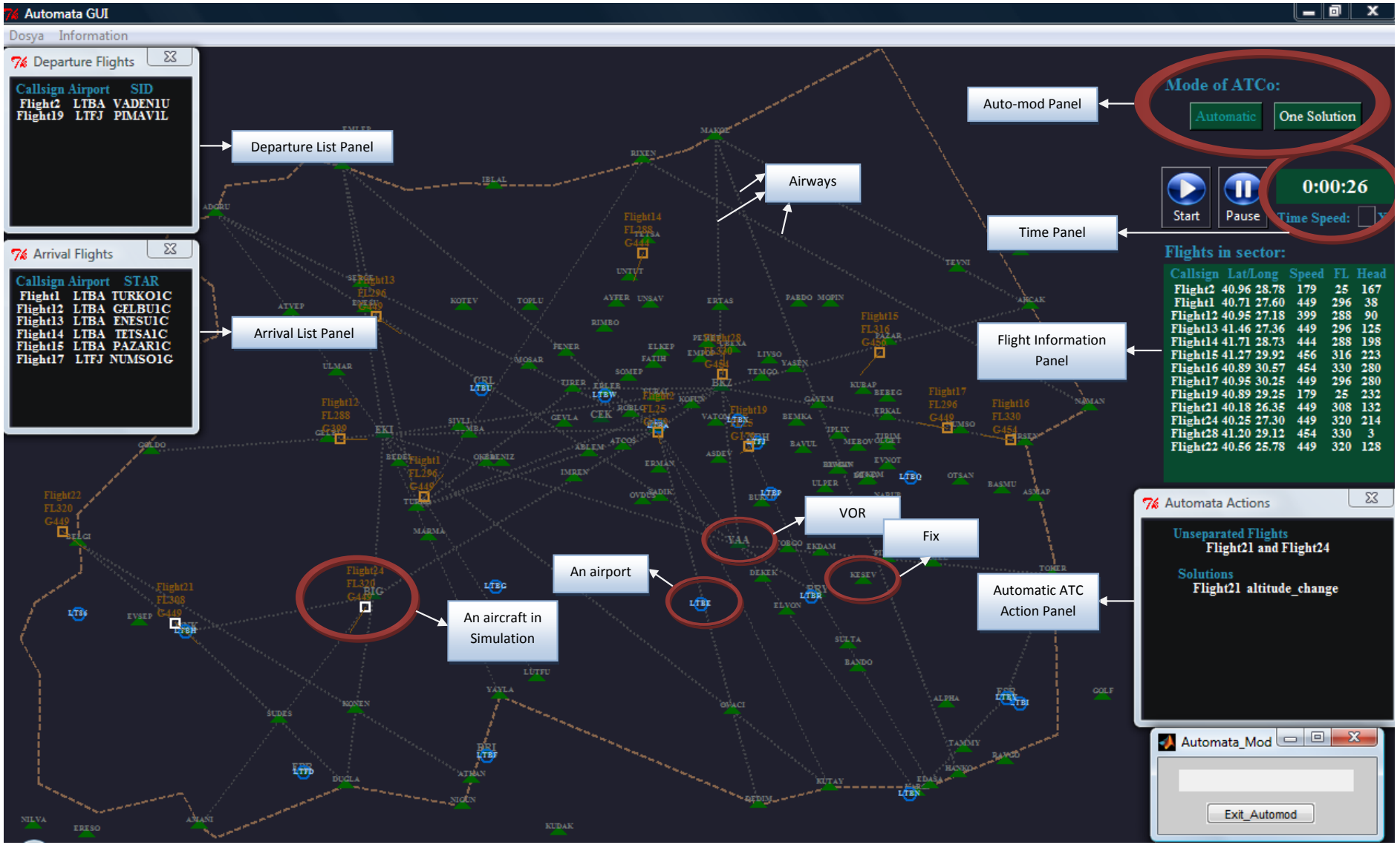



**Figure 2.19**: Simulation Environment: Radar Display.

Departure and Arrival Flights Panels exist at the right top of display. In this panel callsign of arrival flights, destination airport and STAR route are written for all flights in arrival. Similar to this panel, Departure List Panel consist of callsign, airport and SID route of departure flights. For a detailed operation, airports, fixes and VORs in the defined area exist in text files as name, latitude and longitude. These members are taken from files and drawn at display. The airports are drawn as blue circles, fixes are drawn as green triangles and VORs are drawn as dark green triangles, which are shown in Figure 2.19. With mentioned six text files and an additional scenario text file, en-route and approach operations in any area of anywhere can be simulated. Istanbul ACC area is used in our study, which is shown in Figure 2.19.

In simulation environment, Aircrafts are presented as orange squares. The callsign, flight level and ground speed of aircrafts are written at above of aircrafts. Besides of these informations, the direction of aircraft is shown with a line and the length of this line shows that aircraft will go to where during 5min, if it goes along track direction. Basic informations as callsign, latitude, longitude, speed, flight level and heading of all aircrafts in defined area are exist in Flight Information Panel in display.

The display is integrated with designed automatic ATCos. It can be used in two different modes owing to auto-mod panel. One of them is automatic mode, which is a full automatic mode and the other one is solution mode, which is a semi-automatic mode. In automatic mode, the current state information is sent to Automatic ATCo environment periodically. In this environment, the predicted trajectories of all flights are generated relative to current state informations and routes by designed aircraft and FMS models. After that, trajectories are used by ACC and APP Algorithms (Figure 2.9 and Figure 2.10) for conflict detection and separation assurance. Conflicts are detected and solutions are sent to the display by the algorithms. In this mode all conflicts are resolved automatically. In the solution mode, the conflicts are still detected by the algorithms but the solutions are generated only whenever ”one solution button” is pushed by the user. Hence this mode acts a semi-automatic decision support systems for the ATC. Both of them sent solutions to the display, detected conflicts and solutions are written in the Auto-mod Action Panel in display. Block diagram of simulation environment at automatic mode is shown in Figure 2.20.

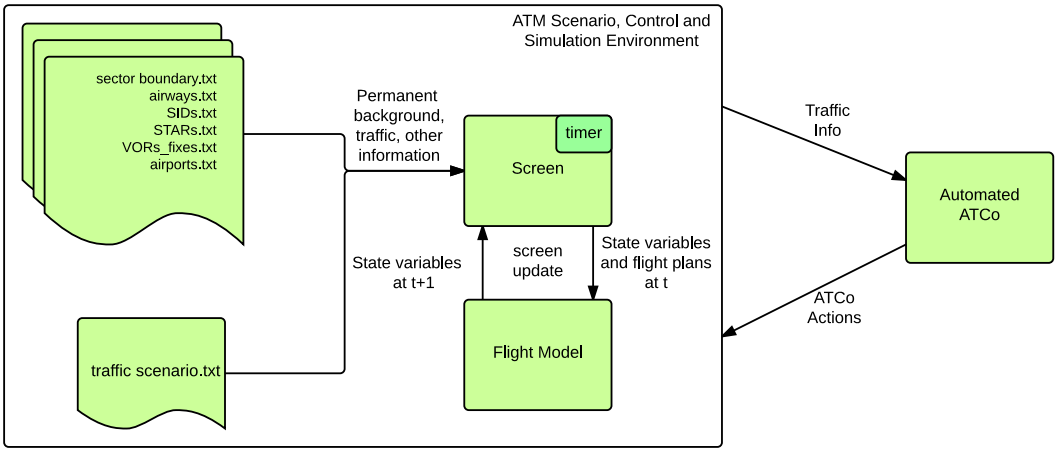



**Figure 2.20**: Block Diagram of Simulation Environment at Automatic Mode.

## 2.8 Results

This section presents simulation and hardware results on several implementations with increasing complexity. First a basic scenario with 8 aircrafts is presented to demonstrate the basic working principles of the algorithm. Next, a 24-hour time window over the Istanbul Ataturk airport is considered to applicability of the algorithm on the real flight data. Finally, the results on integration of the algorithm with a custom radar display and a Boeing 737-800 flight deck simulator are presented.

### 2.8.1 Basic scenario

In the first implementation, a basic problem is solved with the ACC Algorithm (Fig. 2.9). The scenario is shown in Fig. 2.21.

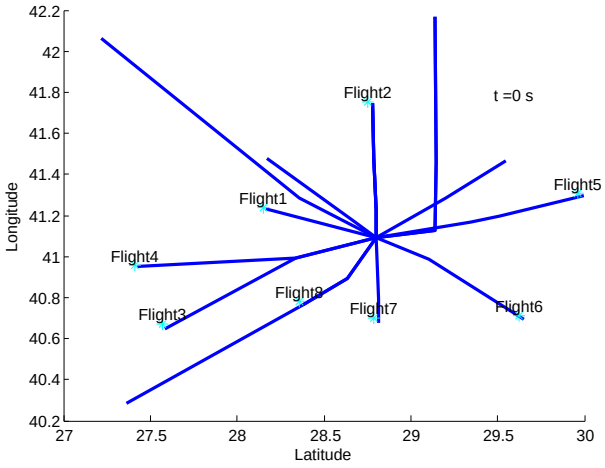



**Figure 2.21**: Basic Scenario in two dimensions at FL320.

In this scenario, 8 aircrafts fly in the $FL320$, which corresponds to $32000ft$ altitude, towards each other. The original flight plans are also shown in Fig. 2.21 with blue

lines. The position of the aircraft on the flight plan is also annotated with text on the figures. The solutions of conflicts and the trajectories after controller action requests are displayed as red lines in Fig. 2.22.

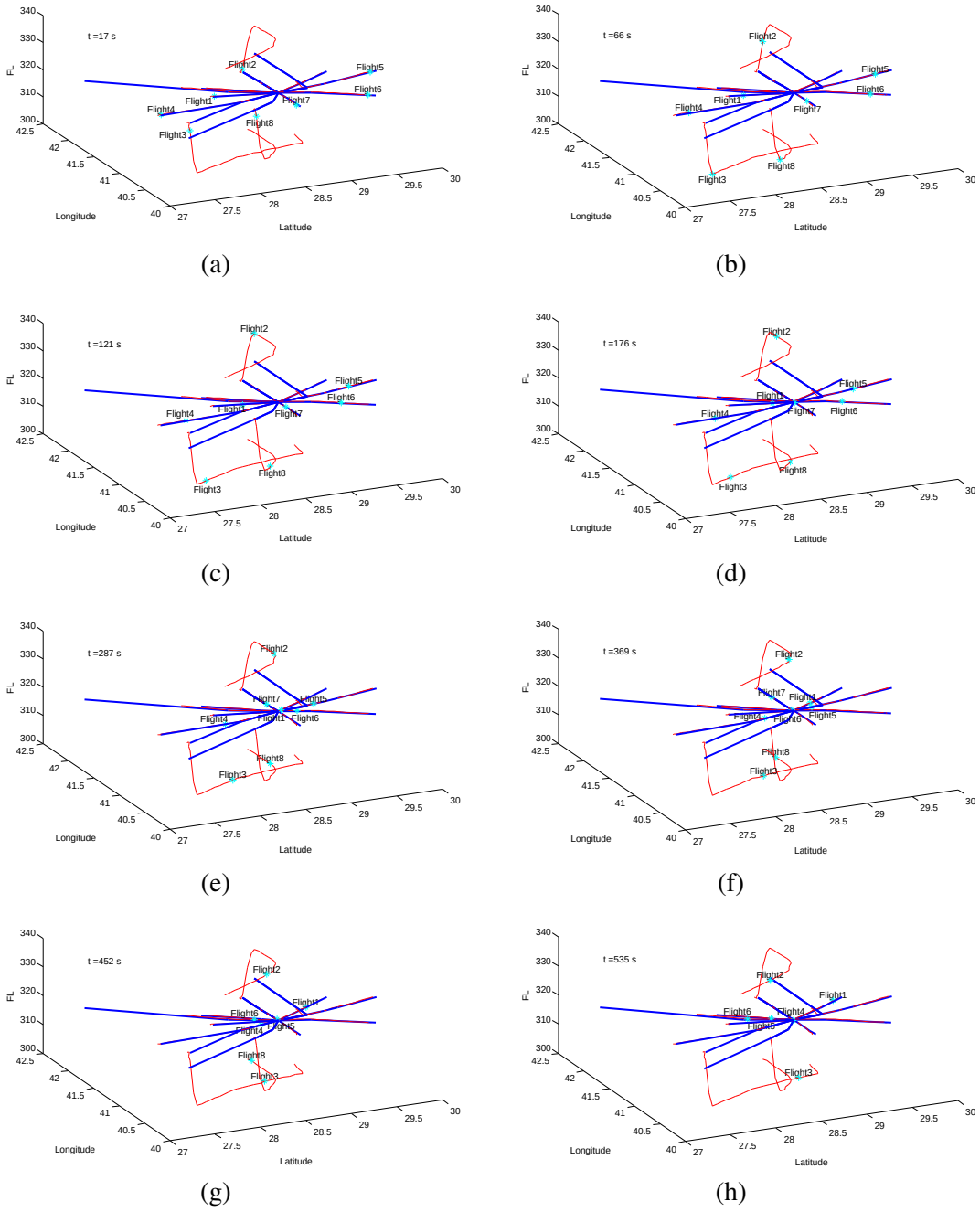



**Figure 2.22**: Basic Scenario in 3D at: (a)17s. (b)66s. (c)121s. (d)176s. (e)287s. (f)369s. (g)452s. (h)535s.

In general, a conflict resolution between two aircrafts might create a new conflict with another aircraft. This basic scenario simulation shows that the developed algorithm presents an elegant solution to this problem. In Fig. 2.22, two aircrafts (Flight7 and Flight8) are close to the center with no horizontal separation in Fig.2.22d, however

their vertical separation is not violated. Note that when the aircrafts are vertically separated, horizontal separation is not needed to be checked. If these aircrafts were not separated, they would have a separation loss with Flight2, which is coming from the opposite direction. Note that these three aircrafts have a conflict in the beginning, however in later stages the algorithm resolves the conflict between Flight7 and Flight8 while assuring that the separation with Flight2 is also ensured. This is because, although the Algorithm 2.9 works by checking conflicts between two aircraft at a time, it ensures separation between all pairs of aircrafts. Conflict between Flight3 and Flight4 are assured with the altitude change, which is shown in Fig. 2.22e. Algorithm 2.9 choses altitude change action. Because if the direct routing action were selected for Flight3, it would have a conflict with Flight5, which can be predicted from Fig. 2.22g. In the final step of the simulation (Fig. 2.22h), it can be seen that all conflicts are resolved.

### 2.8.2 Implementation with ALLFT+ data

#### 2.8.2.1 Implementation for enroute

Real flight data from the ALLFT+ dataset for 24 March 2013 are used for this implementation. The dataset consists of flight plans and information of all flights for one day in Istanbul ACC. In the implementation, En-route controller is responsible for all the flights in the sector, and vertical limits of ACC are 23500 ft and upper. Implementation results are presented in Table 2.2. In the dataset, 1880 flights appeared during the 24 hours in the sector. Simulation results showed that the 134 of these flights them had loss of separation. After the separations are checked, it is seen that one aircraft has loss of separation with at least two aircrafts at the same time. ACC Controller Algorithm (Algorithm 2.9) intervenes to 69 aircrafts and requests 44 of them to make an altitude change action, 19 of them to take direct routing action, 2 of the to takes reducing of speed action and 4 of them to take delaying motion with vector for spacing action. As a result, separation assurance is achieved for the all flights.

#### 2.8.2.2 Implementation for approach

We used two different data sets for approach implementations. The first data set consists of flights for IST APP sector includes arrivals to and departures from Sabiha

**Table 2.2**: Results of Simulations with ALLFT+.

| Variables | ACC | APP (for IST) | APP (for SAW) |
|---|---|---|---|
| # of Flights | 1880 | 514(Arrival), 525(Departure) | 177(Arrival), 180(Departure) |
| # of Flights which have conflict | 134 | 347(Arrival), 278(Departure) | 52(Arrival), 46(Departure) |
| # of Altitude Change | 44 | | |
| # of Direct Route | 19 | 58 | 7 |
| # of Speed Change | 2 | 41 | 2 |
| # of Vector | 4 | 13 | 3 |
| # of Speed (ROCD) Change | | 309 | 43 |
| # of Horizontal Motion | | 3 | 1 |

Gokcen Airport for one day and the second data set consist of flights for IST APP sector includes arrivals to and departures from Ataturk Airport for 1 day. Vertical limits of IST APP are 1500ft and 23500ft. In the first data set, 177 flights appeared in arrival and 180 flights appeared in departure during 24 hours. 52 arrival flights and 46 departure flights have loss of separation. APP Controller Algorithm (Algorithm 2.10) intervenes to 56 aircrafts. In the second data set, 514 flights appeared in arrival and 525 flights appeared in departure during 24 hours. 347 arrival flights and 278 departure flights have loss of separation. APP Controller Algorithm (Algorithm 2.10) intervenes to 424 aircrafts. As a result, separation assurance is achieved for the all flights.

### 2.8.3 Implementation with designed radar display

An aircraft radar display, which has similar GUI used by ATCs is developed, as shown in Fig. 2.23. Designed separation assurance algorithms are then embedded into this display, which demonstrates the applicability of the algorithms to real world avionics systems.

The display shows aircrafts current locations, speeds, headings, altitudes. The followed SID and STAR procedures are also shown in the display. In simulation, flights are emulated to follow defined paths with the flight model and screen is updated at every 0.1s to update the flight information as seen in the radar screen. Display has two different modes; the automatic mode and the solution mode. In the automatic mode, the current state information is sent to ACC and APP Algorithms (Algorithm 2.9 and Algorithm 2.10) periodically. Conflicts are detected and solutions are sent to display

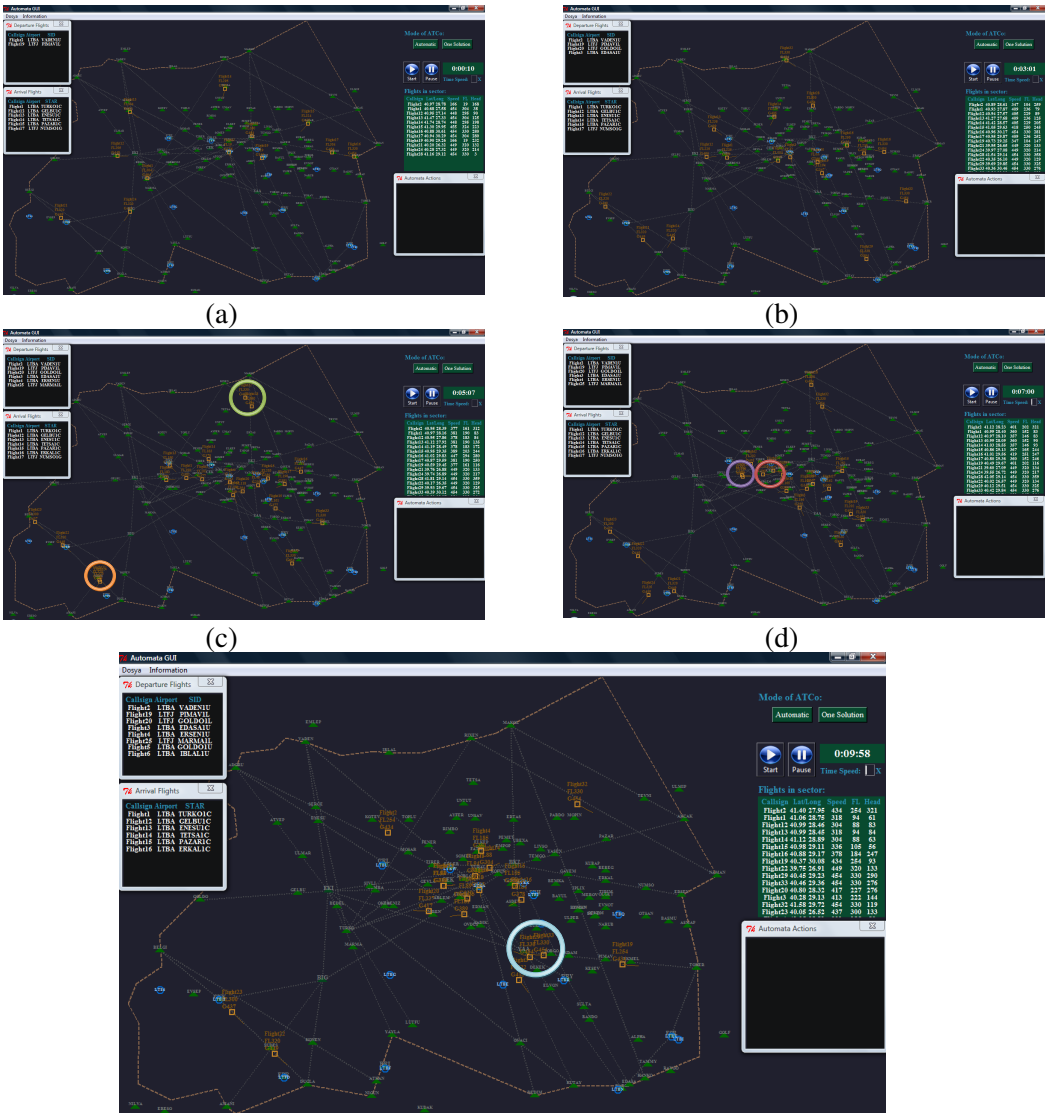


**Figure 2.23**: Radar Display of not intervened situation at: (a)10s. (b)3min. (c)5min. (d)7min. (e)11min.

by the algorithms. In this mode all conflicts are resolved automatically. In the solution mode, the conflicts are still detected by the algorithms but the solutions are generated only whenever ”one solution button” is pushed by the user. Hence this mode acts a semi-automatic decision support systems for the ATC. In display, all flights in ACC and APP are visible in same screen, so the transition between two phases can be done seamlessly.

An example scenario on the radar display is presented in Figs. 2.23 and 2.24. The Fig. 2.23 shows the case where the ATC does not intervene with the situation. In Fig. 2.23c, it is shown that Flight21 and Flight24 have a loss of separation and also Flight28 and Flight32 have loss of separation in en-route. In Figure 2.23d, it is shown

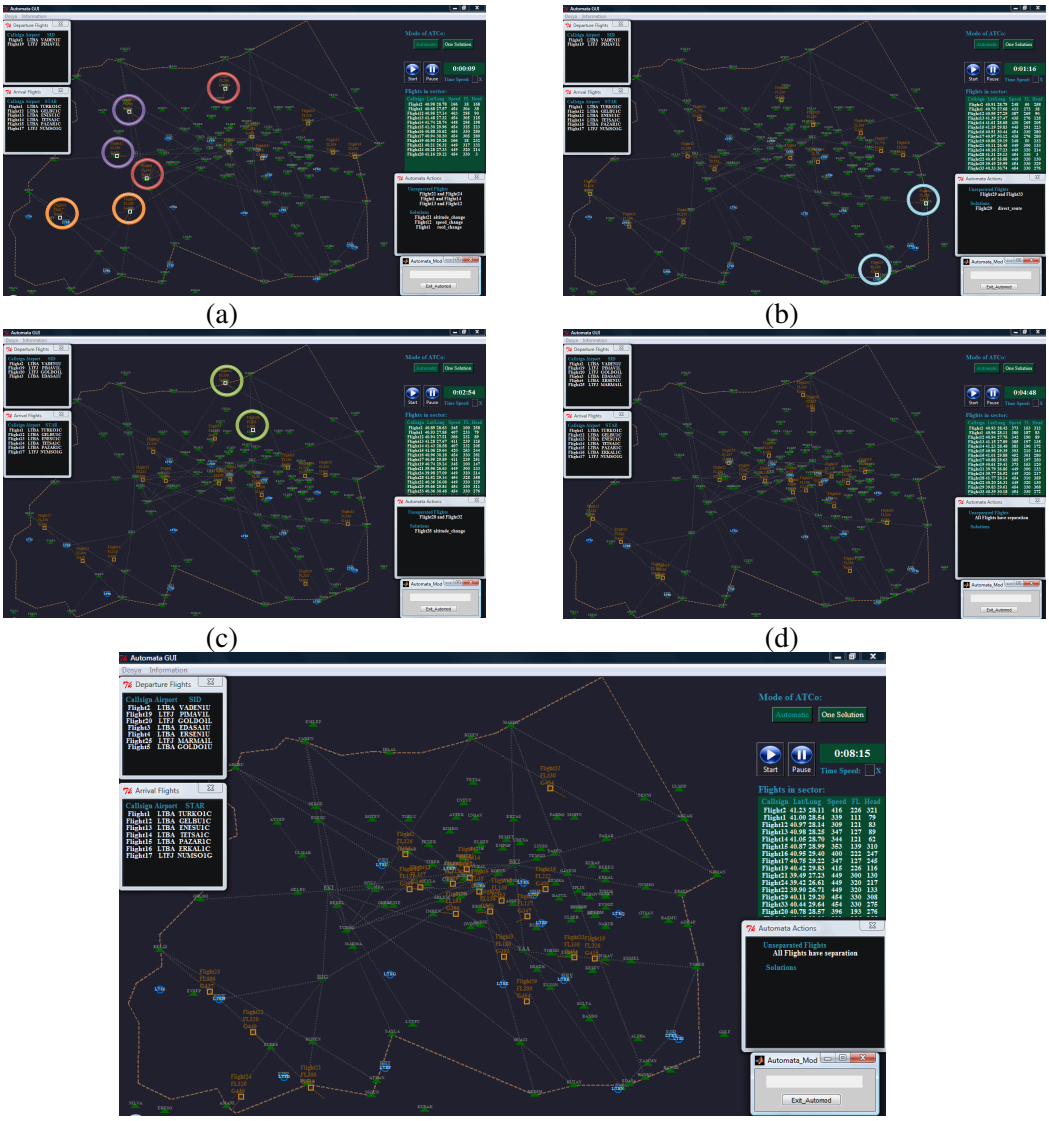



**Figure 2.24**: Radar Display in Automatic Mode at: (a)9s. (b)76s. (c)174s. (d)288s. (e)495s.

that Flight12 and Flight13 have loss of separation and also Flight1 and Flight14 have loss of separation in approach. In Figure 2.23, Flight29 and Flight33 have loss of separation. Note that due to a critical loss of separation aircrafts come dangerously close to each other. If automatic mode is activated in the beginning of simulation, all conflicts can be detected and resolved. This case presented in the Fig. 2.24. The conflict between Flight21 and Flight24 is solved with the altitude change action, the altitude of Flight21 is decreased by Algorithm 2.9 for separation assurance, which is shown in Fig. 2.24a. Flight1 takes the ROCD change action to ensure separation with Flight14, and Flight12 takes the speed change action for separation assurance with Flight13, these actions are generated by Algorithm 2.10, which are shown in Fig.

2.24a. In Fig. 2.24b, it can be shown that conflict between Flight29 and Flight33 at YAA is predicted and direct route action is generated for Flight29 by the Algorithm 2.9. As can be seen in Fig. 2.24d that Flight29 does not go to YAA because of generated action. And Flight28 takes altitude change because of conflict with Flight32, which is shown in Fig. 2.24c.

### 2.8.4 Integration with the Boeing 737 simulator system

The custom radar display is integrated with the flight deck simulator, the block diagram of integrated system is shown in Figure 2.25 and Flight Simulator is shown in the Figure 2.26a.

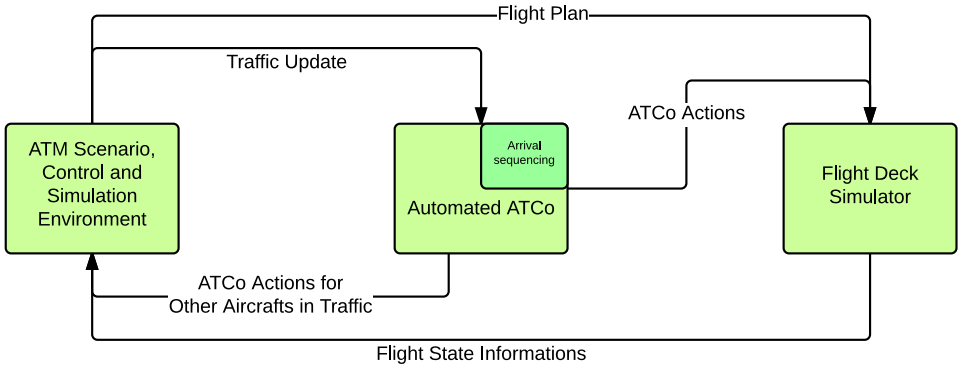



**Figure 2.25**: Block Diagram of Full Integrated System.

In the integrated system, the Boeing 737-800 simulator represent a piloted aircraft flown relative to defined flight plan by a pilot. The callsign of the simulator is named as ”Flight333” in the experiment. In the beginning of the experiment, flight plan of the simulator is converted to 4D trajectory by the flight model and 4D trajectory is sent to FMC by the simulation environment model as latitude, longitude, altitude and speed. FMC is driven automatically with this trajectory, which is shown in the Figs. 2.26d and 2.26e and this trajectory is followed by pilot or autopilot during flight process. Note that FMC can be driven automatically by automated ATCo’s action with permission of pilot during flight, as driven in the beginning of simulation by simulation environment model. Whenever an action, except ”ROCD change”, is requested by the automated ATCo, the 4D trajectory is changed and the new trajectory is updated automatically by FMC relative to ATCo Action with permission of pilot. Whenever ”ROCD change” comes as the requested action; economic, maximum angle or maximum rate modes are used relative to intervene to implement the action automatically. In the experiment scenario, one conflict is detected between the flight deck simulator and one of the

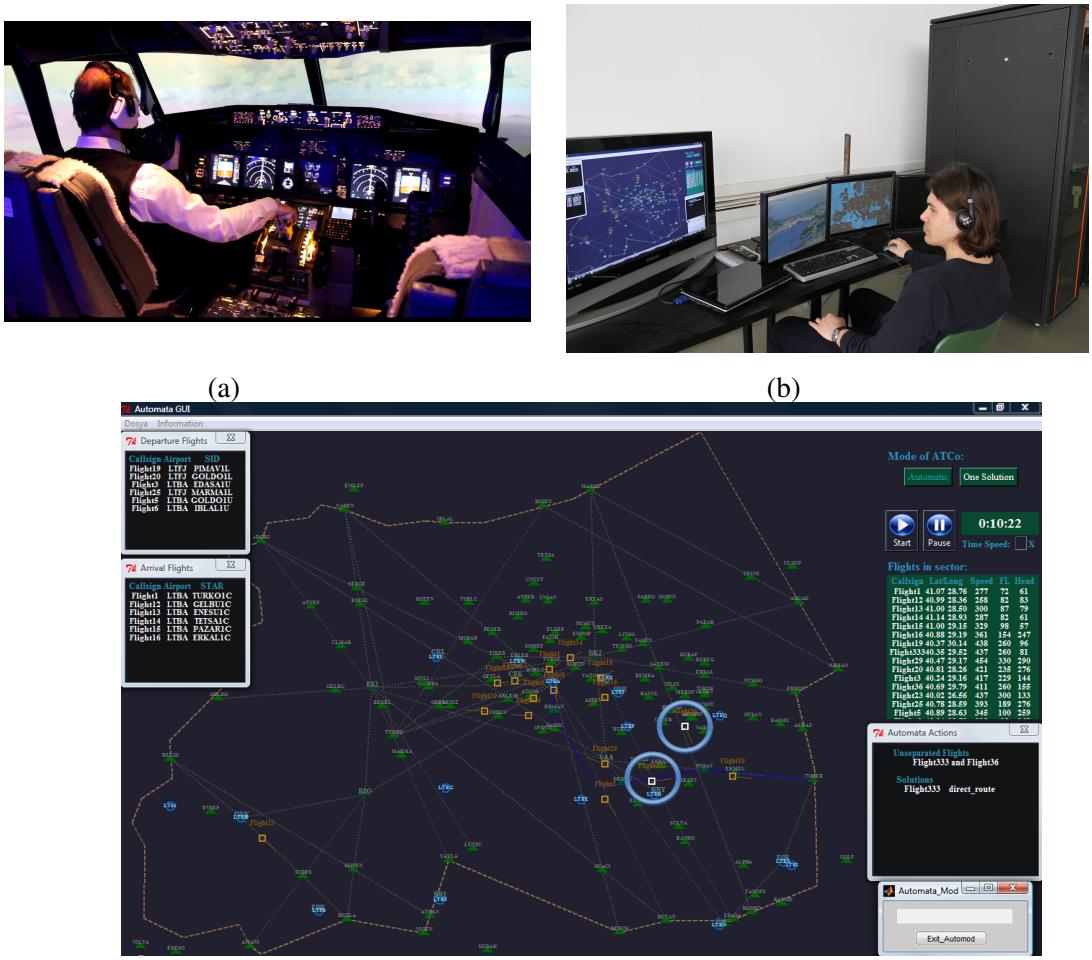



**Figure 2.26**: Full Integrated System: (a)Flight Deck. (b)ATC Desk. (c)Simulation. (d)(e)Automatically Driven FMC.

simulated aircrafts. This conflict is resolved with a direct route action as shown in Fig. 2.26c.

# 3. ANALYSIS OF EUROPEAN AIR TRAFFIC FLOW AND DESIGN OF SLOT ALLOCATION ALGORITHM FROM PERSPECTIVE OF CFMU

## 3.1 Purpose

The purpose of this chapter is understanding of macro-level air traffic flow with data analysis and designing of slot allocation algorithms,which stabilize arrival and departure demand-capacity balance, in the light of analysis.

The main capacity restriction in ATM system results from approach phase of flights. Actually, this restriction is related to runway capacity at landing and take-off. So that, the model, which is designed in this chapter, focuses airports capacities. Firstly, the airports in Europe are analysed from data to construct a realistic network model. And, then a network model is constructed and an algorithm is coded to balance the demands and capacities in European's airports. This algorithm reallocates the departure times of flights with airports capacity restrictions. This means that, an aircraft takes an ground delay when capacity is exceeded. The strategic part of ATM with this algorithm prevents unnecessary holding in airborne, so fuel consumption is decreased with ground action. The workload of ATCO is also decreased with improvement on strategic part of ATM.

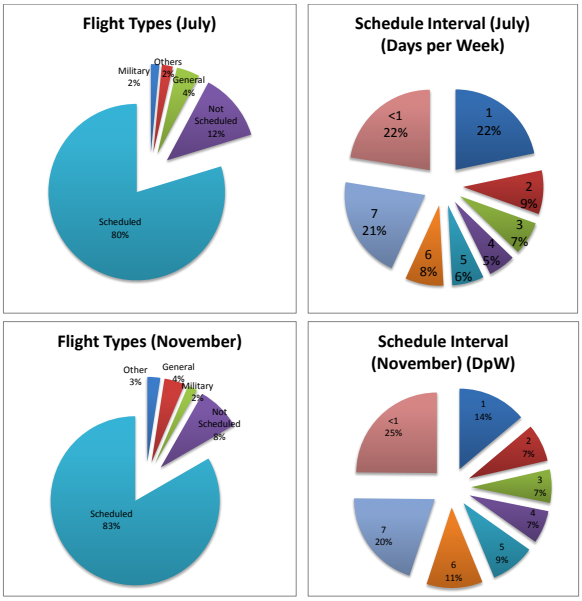



**Figure 3.1**: Flight Types and Schedule Intervals.

### 3.2 Analysis of European Air Traffic Flow Model

The movement in European's airports are observed from ALLFT+ data, and information related to two different month is presented on Table 3.1a and Table 3.1b. In addition, more detailed information about flight types and schedule intervals are presented in Fig. 3.1. The busiest airports in 2011 can be observed on Table 3.1a and Table 3.1b. The order of busiest airports stays nearly constant with minor changes. However the movements per day is higher in summer season compared to winter season. Hence, one can expect higher delays in summer season than winter season under the assumption that the actual capacities are not regulated according to seasons.

**Table 3.1**: Busiest Airports in Europe: (a)November 2011. (b)July 2011.

(a)

| # | Airport | Movement /Day |
|---|---|---|
| 1 | Frankfurt | 1296 |
| 2 | Charles de Gaulle | 1288 |
| 3 | Heathrow | 1242 |
| 4 | Schiphol | 1076 |
| 5 | Madrid | 1073 |
| 6 | Munich | 1058 |
| 7 | Ataturk | 855 |
| 8 | Leonardo da Vinci | 788 |
| 9 | Barcelona | 735 |
| 10 | Vienna | 689 |
| | 42% of all movements in Europe | |

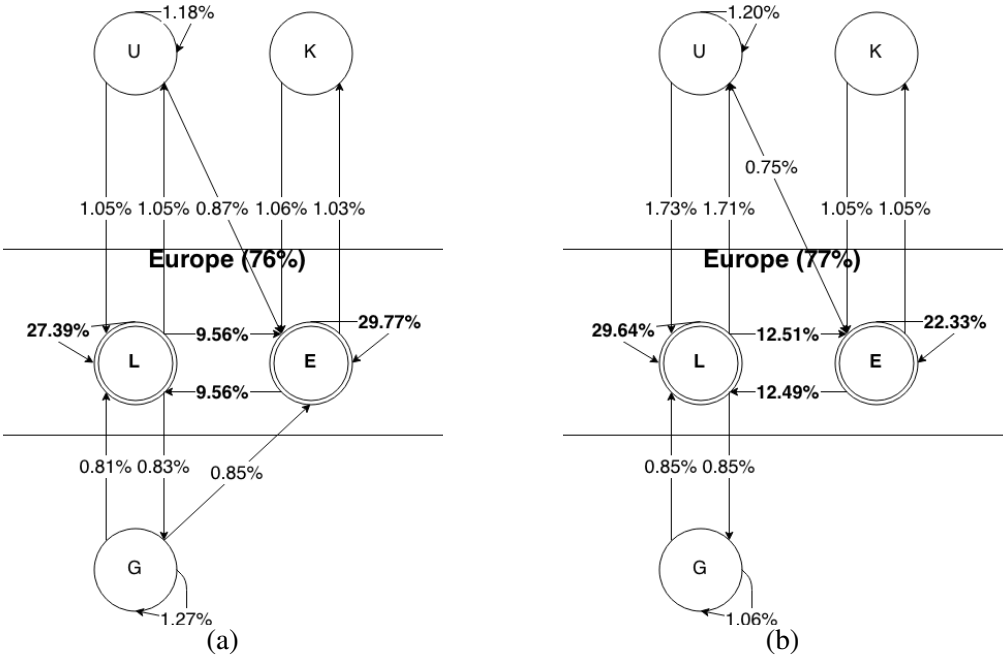
(b)

| # | Airport | Movement/Day |
|---|---|---|
| 1 | Charles de Gaulle | 1500 |
| 2 | Frankfurt | 1380 |
| 3 | Heathrow | 1360 |
| 4 | Schiphol | 1306 |
| 5 | Madrid | 1229 |
| 6 | Munich | 1171 |
| 7 | Leonardo da Vinci | 1002 |
| 8 | Ataturk | 944 |
| 9 | Barcelona | 926 |
| 10 | Gatwick | 792 |
| | 38% of all movements in Europe | |

Moreover, with seasonality the main delay generators, i.e. major airports, change suggesting there should be separate models for each season. Examining air traffic flow between regions and air traffic flow generation and absorption graphs of busiest

airports for each season can further support this hypothesis of needing to use separate models for each season.

In figures 3.2a and 3.2b air traffic flows are presented as weighted directed graphs. Weights on the edges represent the percentage of air traffic flowing from a region to another denoted by ICAO region codes with respect to all traffic volume in July and November 2011.




L: Southern Europe, Israel and Turkey; E: Northern Europe; U: Russia; K: Contiguous United States; G: West Africa and Maghreb

**Figure 3.2**: Flow Distribution: (a)November 2011. (b)July 2011.

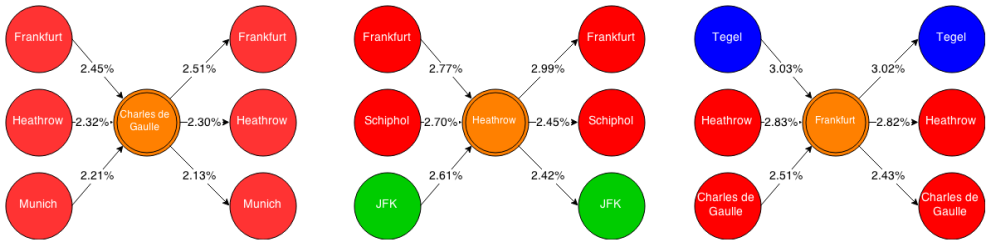
While building the flow distribution graphs only regions with strongest connections have been selected. Connections with less than 0.7% are not considered as significant in order to simplify the model.

It has been observed that air traffic flux varies from one season to another. Since there is not a big variation in total air traffic flow in Europe from season to season, Europe can be assumed as a closed system, i.e. while total flow is constant, rate distribution shifts from one region to another, e.g. air traffic flow between Northern and Southern Europe shifts to inner air traffic flow at Northern Europe in winter season.

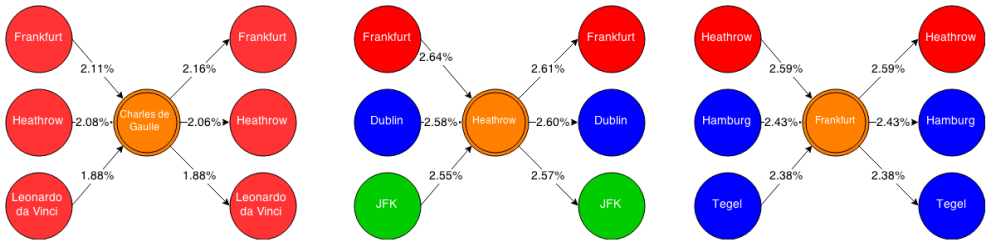
Since the ALLFT+ data consists mainly of European flight data,it is more accurate to focus on Europe region. By implementing a model focused primarily on Europe will represent a stronger model with at least 73% actual flow coverage.

This graph structure also suggests that the traffic sensitivity to delays changes from season to season depending on air traffic flow rates. However, investigating the connections in airport-to-airport level will make presentation of delay sensitivities more accurate.

In Figure 3.3a and 3.3b connectivity graphs of the airports with most movements are demonstrated. Orange airports are the investigated airports, red airports are the busiest airports in Europe, blue airports are airports in Europe that are not in top 10 busiest airports in Europe and green airports are the airports outside Europe.




(a)




(b)

**Figure 3.3**: Connectivity Graphs of Major Airports: (a)November 2011. (b)July 2011.

The color-coding comes in handy because having a connection with a red airport causes more delays because of high traffic volume and having a connection with blue airport causes rather less delays. Having a connection with green airports does not provide an accurate picture caused by lack of data outside the Europe region.

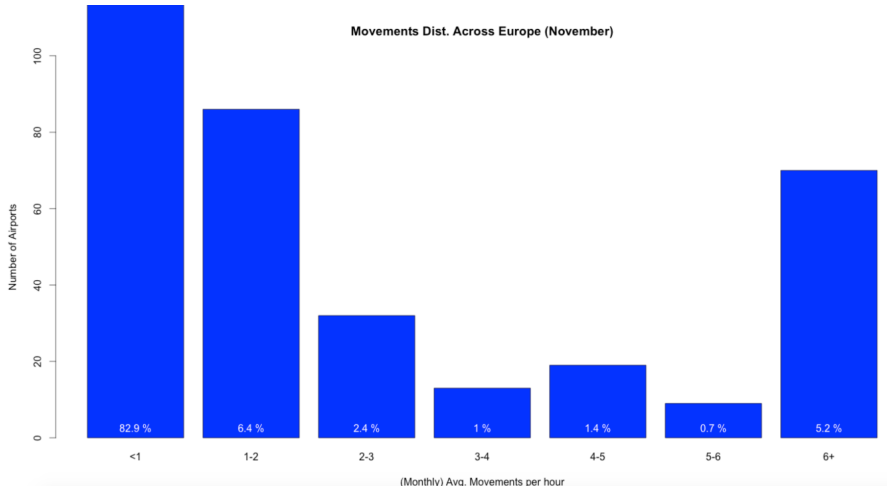
By only looking at color coding one can predict that the delay sensitivity of Charles de Gaulle is higher than Frankfurt airport. Also the delay sensitivity of Frankfurt increases from July to November because of the shift of airport connections.

Charles de Gaulle, Heathrow, and Frankfurt have different connectivity characteristics, so one can predict that the delay trends will be different from airport to airport and season to season.

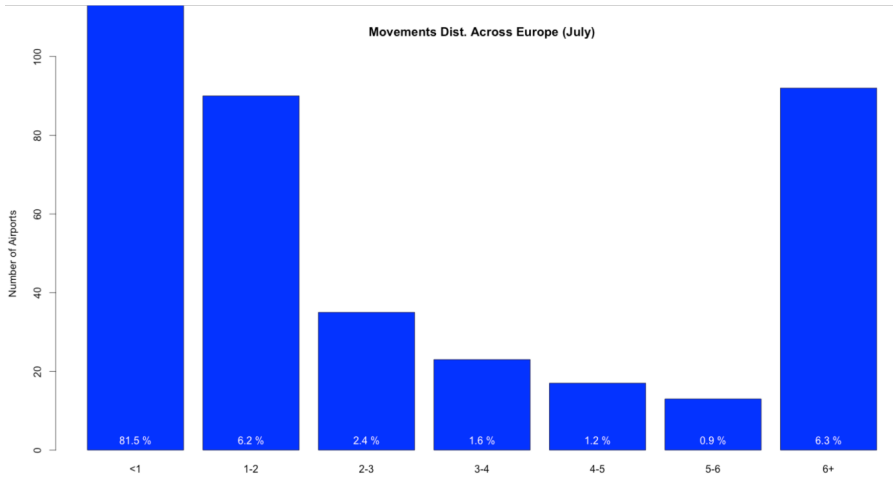
Seasonality has an impact on connection sensitivities. The links get stronger in winter than in summer, so the delay propagation rate is expected to be higher in winter.

### 3.2.1 Airport classification according to air traffic flow rate

Both figures 3.4a and 3.4b suggest that about $82-83\%$ of all airports in Europe have air traffic flow rate less than one movement per hour averaged about one month and this rate reduces at higher flow rates.



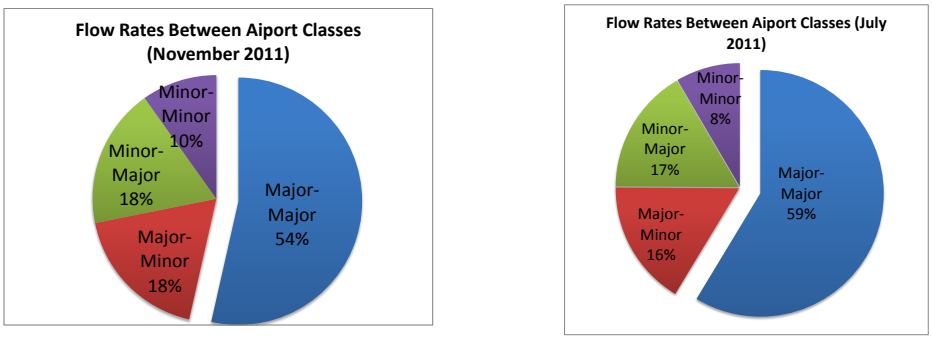

(a)




(b)

**Figure 3.4**: Movements Distribution Across Europe: (a)November 2011. (b)July 2011.

Since major airports are defined as "airports having one or more movements per 15 minutes" ratio of European major airports is only $17-18\%$ among all 1458 airports in Europe. However, the massive rate of air traffic flow is generated by major airports as seen on Figure 3.5a and 3.5b.




(a) (b)

**Figure 3.5**: Air Traffic Volume Ratios: (a)November 2011. (b)July 2011.

Because of this distribution a simplification model has been proposed. Minor airports are grouped by their regions and modelled as aggregated airports. Capacity of flights from those aggregated airports is determined from their cumulated demand with the same capacity calculation procedures used for major airports. With this simplification total number of airports have been reduced from 2057 to 304 consisting of 122 European major airports and 182 aggregated airports (e.g. LFXX, ENXX, EGXX).

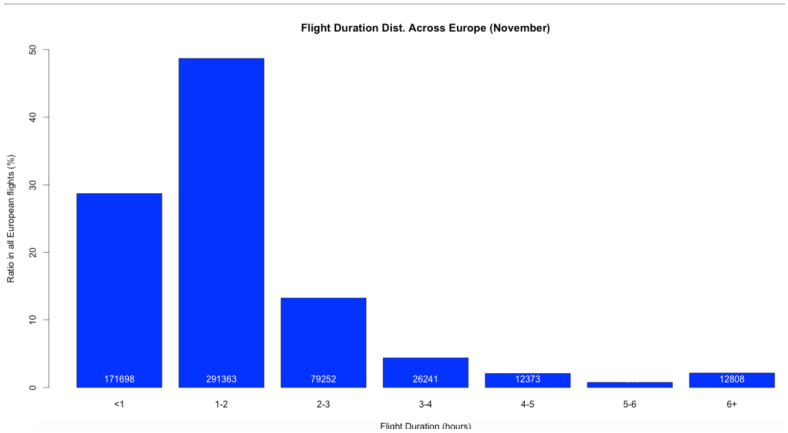
In the simulation, non-European airports and aggregated airports are assumed to have infinite capacity, i.e. their capacity is set to a large number.

Also the distribution of departure movements and arrival movements are very similar thus number of total movements sufficient for representing the movement distribution across Europe.

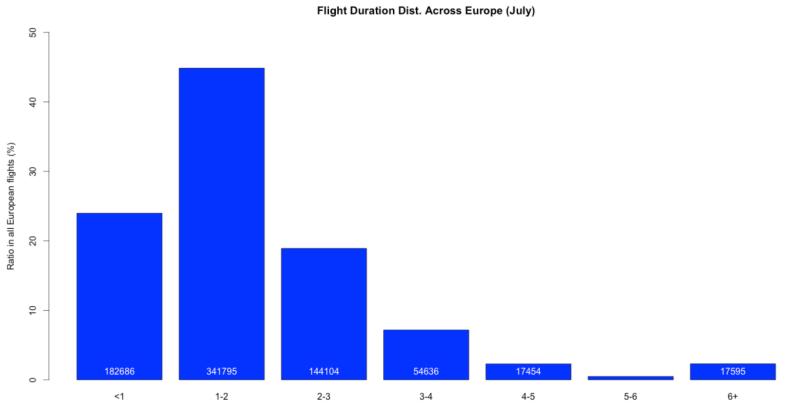
### 3.2.2 Airport classification according to flight durations

Figures 3.6a and 3.6b depict that average flight duration in Europe is distributed normally with a left skew. Also most of the flights take between 1 and 2 hours. This distribution changes slightly with seasonality so it can be modelled as single normal distribution independent from seasonality.

Figure 3.7a and 3.7b clearly show that Marco PoloAirport is very similar to average Europe distribution in figure 8, while Heathrow Airport has a different distribution. Heathrow is a major hub, thus most of the flights are long distance flights and consequently Heathrow's delay characteristics are expected to be different from Marco Polo's.

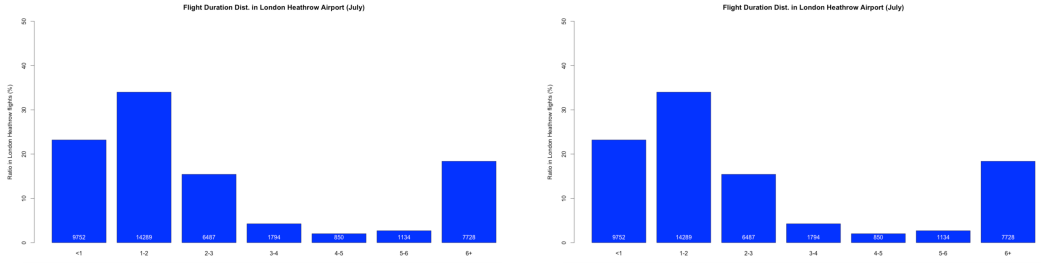



(a)




(b)

**Figure 3.6**: Duration Distribution Across Europe: (a)November 2011. (b)July 2011.




(a) (b)

**Figure 3.7**: Duration Distribution of Specific Airports in July 2011: (a)London Heathrow. (b)Venice Marco Polo.

Moreover, the mean flight duration of European flights are 143 minutes in July and 138 minutes in November. Marco Polo has a mean duration of 97 minutes while Heathrow has 243 minutes in November.

In short, it can be said that flight durations can be defined relative to departure and arrival airports as a normalized flight duration that is extracted from data. And this

durations can be used in algorithm to determine the arrival time of a flight for strategic planning.

### 3.2.3 Nominal capacity rather than declared capacity

In theory, DDR2 capacity declarations data set can be used for determination of an airport's departure or arrival capacity. However, this data set has a problematic situation. Declared values, which is come from DDR2 capacity declarations data set, and real demands values,which is extracted from ALLFT+ data set, sometimes are not matched. For example, in Figure 3.8 arrival movements per 15 minutes (i.e. time window) has been demonstrated. In the figure declared capacity is only 11, while nominal capacity (above 95%) is 14 per 15 minutes. This means that airport can handle 14 aircraft per minutes in reality but only 11 aircraft per minutes is declared in DDR2 capacity declarations data set. It is a problematic situation, nominal capacity (95% of real traffic data) is defined as an alternative capacity to overcome the problem. So, more realistic approaches can be used in algorithms.

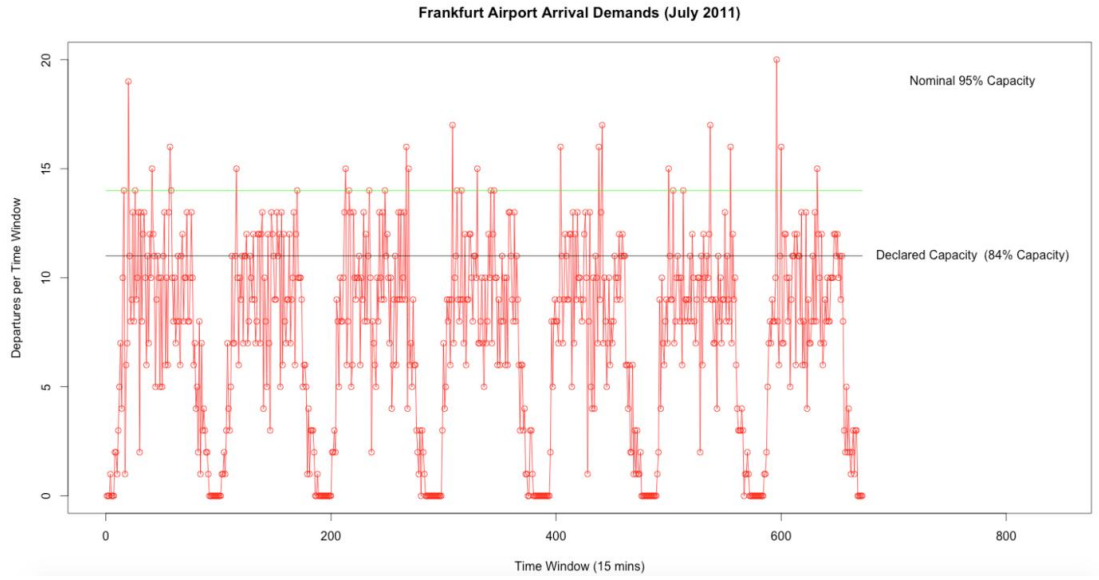



**Figure 3.8**: Arrival Demand of Frankfurt Airport (July 2011).

Another thing is that, according to DDR2 capacity declarations dataset Frankfurt Airport's declared departure capacity is infinity. Since infinity does not provide information about the capacity, infinity capacity set to 99% nominal capacity. And capacity of aggregated airports set to a large number as depicted before.

In theory the utilization rate $\rho$ should not be more than one that is demand should not exceed capacity. With declared capacity, demand exceeds capacity about 16% ($\sim 2.5\sigma$) of the time not indicating a good assumption for the actual capacity value. Thus, a nominal value has been calculated for reducing capacity exceeding rate to 5% ($\sim 3\sigma$) for being acceptable.

### 3.2.4 Inferences from data analysis

The inferences from data analysis are mainly that:

- European traffic flow is mainly related to seasons that are summer and winter seasons. So, two different network model must be constructed.
- Minor airports can be aggregated relative to nationally, this is a realistic simplification.
- Nominal capacity values are used when declared capacity values have contradiction with real traffic capacity.
- Normalized flight durations relative to departure and arrival airport are used for determination of arrival time of flight in algorithm.

## 3.3 Slot Allocation Algorithm

Firstly, the symbols, variables and functions that are used in algorithms are described in Table 3.2 and Table 3.3. Definitions of these parameters can be obtained from these tables.

1 $F_{all} \leftarrow PickScheduledFlights()$
2 $A_{all} \leftarrow GroupAirportsbyRegion()$
3 $(C_{dep,all}, C_{arr,all}) \leftarrow CalculateNominalCapacities()$
4 $D_{all} \leftarrow NormaliseFlightDurations()$
5 $F_{grouped} \leftarrow NormaliseFlightTimes()$
6 $Q_{dep} \leftarrow PopulateDepartureQueues()$
7 $(S_{dep,all}, S_{arr,all}) \leftarrow AllocateSlotsFree()$

**Figure 3.9**: Preprocess.

### 3.3.1 Pre process and assistant algorithms

In Preprocess phase (Fig. 3.9), first of all, scheduled flights are picked from the data by filtering the type of flight. If the flight type is scheduled the flight is included in

**Table 3.2**: Descriptions of Variables.

| Variable | Description |
|---|---|
| $t_{window}$ | Time Window Width (e.g. 15 minutes) |
| $T_{start}$,$T_{end}$ | Start and End Times |
| $A'_{all}$,$A_{all}$ | All Airports, All Normalized Airports |
| $A'_{reg}$ | Regional Airports |
| $C_{dep,all}$,$C_{arr,all}$ | All Departure/Arrival Capacities |
| $Q_{dep}$ | Departure Queue |
| $S_{dep,all}$,$S_{arr,all}$ | All Departure/Arrival Slots |
| f | Departure/Arrival Frequency |
| m | Departure/Arrival Mean Frequency |
| $\Sigma$ | Departure/Arrival Standard Deviation |
| $D_{all}$ | All Flight Durations Between OD Pairs |
| $t_{step}$ | Time Step |
| $t_{slot}$ | Calculated Slot Time |
| $t_{OBT}$ | Off Block Time |
| $t_{TOA}$ | Time of Arrival |

**Table 3.3**: Descriptions of Symbols and Functions.

| Symbols and Functions | Description |
|---|---|
| $a \leftarrow 5$ | $\leftarrow$ Assignment Operator, e.g. 5 is assigned to $a$ |
| 6 // 4 | // Integer Division Operator, e.g. 6 // 4 = 1 |
| NotFound | There is no free space left in time window |
| $D_{all}(A,B)$ | Duration of flight between airports A and B |
| Found() | Return False if NotFound otherwise True |
| Enqueue(Q, A) | Insert a to the end of the queue Q |
| $A \leftarrow Dequeue(Q)$ | Pick the first element from the top of the queue Q |
| $EnqueueAll(Q,A_{list})$ | Insert all elements A in $A_{list}$ to the Q in the same order |
| $R \leftarrow Region(A)$ | Get region code from airport (e.g. A: LTBA, R: LTXX) |
| IOBT(F) | Initial (Scheduled) Off Block Time of flight F |
| AOBT(F) | Actual Off Block Time of flight F |
| ATOA(F) | Actual Time of Arrival of flight F |
| Length(L) | Number of elements in a list L |
| Append(L, A) | Insert A to the end of the list L |
| $AppendAll(L,A_{list})$ | Insert all A in $A_{list}$ to the end of the list L |
| $Available(S(t_{tw}),t_{slot})$ | Checks if slot at index $t_{slot}$ time window $t_{tw}$ is empty |
| $\{\}$ | Empty Dictionary |

simulation. Minor airports are grouped with GroupAirportsbyRegion algorithm (Fig. 3.10) and merged with major airports. After grouping airports departure and arrival capacities are calculated. Since the same flights can have different flight times the minimum duration is picked as nominal time for simulation. Since there are special events in the data the schedule data is not completely normalized. Because of that, days closer to normal picked for every day in a week from a month and a normalized

demands week has been created. After that departure queues are populated with normalized flights according to normalized scheduled off block times. Finally, slots for every airport and every time window are allocated and set free for departure and arrival. In GroupAirportsbyRegion algorithm (Fig. 3.10), major airports are kept as

```
1 foreach A in A'_all do
2     if MovementsPerHour(A) ≥ 4 then
3         Append(A_all,A)
4     else
5         R ← Region(A)
6         Append(A_reg(R),A)
7 AppendAll(A_all,A_reg)
8 return A_all
```

**Figure 3.10**: GroupAirportsbyRegion.

is, while minor airports (movements per time window < 1) are grouped as aggregated airports at their region R.

```
1 foreach A in A_all do
2     f ← Count(Movements(A)), m ← mean(f), Σ ← sd(f)
3     C ← 1.96Σ
4     while Count(Movements (A) < C) / Count(f) < 0.95 do
5         C ← C + 1
6     (C_dep, C_arr) ← C
7     (C_dep,all, C_arr,all) ← Append((C_dep, C_arr))
8     return (C_dep, C_arr)
```

**Figure 3.11**: CalculateNominalCapacities.

In CalculateNominalCapacities algorithm (Fig. 3.11), 95% confidence value of movements in airport for both departure and arrival has been calculated as

$$C = m + 1.96\Sigma \tag{3.1}$$

If the statistical upper level is less then actual (sample) 95% capacity, capacity is increased by one in each iteration, until the actual 95% level is met. Usually the statistical and sample upper levels are equal with some exceptions.

In AssignFlightTimes algorithm (Fig. 3.12), if slot is found successfully, $t_{TOA}$ and $t_{OBT}$ are assigned as Actual Take-off Time and Actual Off-Block Time of F respectively. After that flight F is assigned to departure and arrival slots at these times. Then, queue

**Input**: $t_{TOA}$,$t_{OBT}$
1 $ATOA(F) \leftarrow t_{TOA}$
2 $AOBT(F) \leftarrow t_{OBT}$
3 $S_{dep}(t_{OBT}) \leftarrow F$
4 $S_{arr}(t_{TOA}) \leftarrow F$
5 $Tail_{num} \leftarrow TailNumber(F)$
6 $q_{num} \leftarrow QueueNumber(F)$
7 $LastQueueNumber(Tail_{num}) \leftarrow (q_{num}, t_{TOA})$

**Figure 3.12**: AssignFlightTimes.

number of F is recoded to $LastQueueNumber(Tail_{num})$ as last landed aircraft, this set is used in algorithms to follow the scheduled itineraries, which requires visits to sequences of airports, of a specific aircraft.

**Input**: $t$
1 return (t//$t_{window}$)*$t_{window}$

**Figure 3.13**: GetTimeWindowTime.

A time is rounded up to be a lower factor of $t_{window}$ with GetTimeWindowTime algorithm (Fig. 3.13).

1 **for** *all F* **do**
2 $A \leftarrow Source(F)$
3 $B \leftarrow Destination(F)$
4 $CS \leftarrow Callsign(F)$
5 $D \leftarrow STOA(F) - SOBT(F)$
6 Append($D_{A,B,CS}$, D)
7 **for** *all* $D_{A,B,CS}$ **do**
8 $D_{A,B,CS} \leftarrow statisticalmin(D_{A,B})$
9 return $D_{A,B,CS}$

**Figure 3.14**: NormaliseFlightDurations.

In NormaliseFlightDurations algorithm (Fig. 3.14), flight durations are normalized. For each source-destination-callsign triplet collect the durations by subtracting scheduled off block time from scheduled time of arrival and set the duration of that triplet as the statistical minimum for the set collected. In algorithm, statistical minimum is the least element in the sample, which is higher than the lowest end of the normal curve, i.e. this element is the smallest element in a set, which is not an outlier.

```
for all F do
    A ← Source(F)
    B ← Destination(F)
    CS ← Callsign(F)
    Append(IOBT_{A,B,CS}, IOBT_F)
for all IOBT_{A,B,CS}(F) do
    IOBT_{A,B,CS}(F) ← mode(IOBT_{A,B,CS})
return F
```

**Figure 3.15**: NormaliseFlightTimes.

In NormaliseFlightTimes algorithm (Fig. 3.15), mode of each source-destination-callsign triplet's IOBT is set as all of the flights' IOBT for normalization.

### 3.3.2 Main algorithms

```
Input: T_start, T_end, A_all
Output: Slots
Preprocess() (Alg. 3.9)
t_window ← 15mins, t ← T_start
while t ≤ T_end do
    foreach A in A_all do
        Q ← Q_dep(A,t), Q ← Q_dep,all(A)
        while Length(Q)>0 or Length(S_dep(t)) > 0 do
            F ← Dequeue(Q), B ← Destination(F), S_arr ← S_arr,all(B), D ←
            D_all(A,B)
            SearchSlotDeparture(IOBT(F)) (Alg. 3.17)
        if Length(Q)>0 then
            EnqueueAll(Q_dep(A, t + t_window), Q)
```

**Figure 3.16**: Slot Allocation Algorithm.

The main algorithm is presented in Figure 3.16. This is a recursive algorithm, many algorithms are called by each other with a recursive manner in Figure 3.16.

The simulation starts with preprocessing of airports, demands, queues, and slots (with algorithm presented in Figure 3.9). The step of simulation is 15 minutes and simulation takes place between times $T_{start}$ and $T_{end}$. At every time step, an airport A is picked from all airports and its departure queue Q and its slot $S_{dep}$ are selected for management.

Up until there is neither flight in queue Q nor there is no place in slot left all flights at time t and in departing from airport A are managed. At every management step a flight has been picked (Dequeue) from the beginning of the queue Q. After that the arrival slot $S_{arr}$ and duration D of the flight has been picked from the list $S_{arr,all}$ and $D_{all}$ between A and B, respectively. Consequently SearchSlotDeparture recursive function (Fig. 3.17) starts with the current time window and flight. Recursive function runs until it assigns the flight F to departure and arrival slots or fails to assign if t reaches to $T_{end}$.

After assignment is complete with or without success, the same process repeats for every flight in the queue or until slots are full and for all airports for the particular time window t. If there are still flights left in the queue these flights are assigned to the next time window of the queue of the next time window of the airport. When all assignments are complete for the time window t, t is set to next time window by EnqueueAll function.

**Input**: $t_{sched}$
1 $t_{tw} \leftarrow GetTimeWindowTime(t_{sched})$ (Alg. 3.13)
2 $t_{OBT} \leftarrow SearchSlot(S_{dep}, t_{tw}, t_{sched})$ (Alg. 3.19)
3 **if** *Found*($t_{OBT}$) **then**
4 SearchSlotArrival(tOBT) (Alg. 3.18)

**Figure 3.17**: SearchSlotDeparture Algorithm.

In SearchSlotDeparture algorithm (Fig. 3.17); first of all, calculate $t_{tw}$ that is the start time of the scheduled time $t_{sched}$'s time window. Start searching for a free slot starting from scheduled time in departure slot and continue until a slot is found. Then, if the time $t_{OBT}$ is found, start searching for arrival slot with SearchSlotArrival algorithm (Fig. 3.18) for flight departing at time $t_{OBT}$.

In SearchSlotArrival algorithm (Fig. 3.18), start with calculating time of arrival for given off-block time by adding the duration. Similarly, calculate the start of the time window and search for a slot in arrival. If slot is found in related arrival time window then assign the flight, otherwise continuous with next time window with SearchSlot algorithm (Fig. 3.19). If slot is found, search for slot in departure slots (Fig. 3.17) by calculating the new departure time $t'_{OBT}$. If there is a slot free in the time window then assign the flight to the first available slot in the time window starting from the

```
   Input: t_OBT
1  t_TOA ← t_OBT + D
2  t_tw ← GetTimeWindowTime(t_TOA) (Alg. 3.13)
3  t'_TOA ← SearchSlotInTimeWindow(S_arr, t_tw, t_TOA) (Alg. 3.21)
4  if Found(t'_TOA) then
5      AssignFlightTimes(t'_TOA, t_OBT) (Alg. 3.12)
6  else
7      t'_TOA ← SearchSlot(S_arr, t_tw, t_TOA) (Alg. 3.19)
8      if Found(t'_TOA) then
9          t'_OBT ← t'_TOA − D
10         t''_OBT ← SearchSlotInTimeWindow(S_dep, t_tw, t'_OBT)
11         if Found(t''_OBT) then
12             AssignFlightTimes(t'_TOA, t_OBT) (Alg. 3.12)
13         else
14             t_OBT−NEXT ← t_OBT + t_window
15             SearchSlotDeparture(t_OBT−NEXT) (Alg. 3.17)
```

**Figure 3.18**: SearchSlotArrival Algorithm.

$t'_{OBT}$. If there is no slot left in the time window, continue searching by returning to the SearchSlotDeparture algorithm (Fig. 3.17) with $t_{OBT-NEXT}$.

```
   Input: S,t_tw,t_slot
1  Tail_num ← TailNumber(F)
2  q_num ← QueueNumber(F)
3  if CheckQueueNumber(Tail_num,q_num,t_tw) (Alg. 3.20) then
4      t'_slot ← SearchSlotInTimeWindow(S, t_tw, t_slot) (Alg. 3.21)
5      if Found(t'_slot) then
6          return t'_slot
7  t'_tw ← t_tw + t_window
8  if t'_tw ≤ T_end then
9      return SearchSlot(S,t'_tw,t_slot) (Alg. 3.19)
10 return NotFound
```

**Figure 3.19**: SearchSlot Algorithm.

In SearchSlot algorithm (Fig. 3.19), first, control the last queue number of current aircraft with CheckQueueNumber Algorithm (Fig. 3.20). If value of this function returns to True, this means that scheduled flight before current flight of related aircraft is landed to the airport and then search for a slot in time window. If there is a place return it. Otherwise search slot in next time window starting from the beginning of the time window. If $t'_{tw}$ reach $T_{end}$ return NotFound value returning false for Found function.

```
Input: Tail_num,q_num,t_tw
if q_num == 1 then
    return True
else
    Last_qnum, Last_TOA ← LastQueueNumber(Tail_num)
    if Last_qnum == (q_num - 1) and Last_tw < t_tw then
        return True
    else
        return False
```

**Figure 3.20**: CheckQueueNumber Algorithm.

In CheckQueueNumber algorithm (Fig. 3.20), control the queue number if it equals the one this means that the aircraft is at the beginning of the scheduled flight series and then return the True. Otherwise, take the $Last_{q_{num}}$ and $Last_{TOA}$ of related aircraft that are the queue number and arrival time of last flight of related aircraft respectively. If $Last_{q_{num}} == (q_{num} - 1)$ and $Last_{tw} < t_{tw}$ this mean that the aircraft is in mentioned airport and last flight is performed and aircraft is ready to current flight and, then return True. If conditions are not provided and the return False.

```
Input: S,t_tw,t_slot,S_r
C ← Length(S(t_tw))
NE ← NonEmpty(S(t_tw))
NE_r ← NonEmpty(S_r(t_tw))
if (NE+NE_r) > C then
    return NotFound
else
    t'_slot ← 0
    while t'_slot < t_window do
        if Available(S(t_tw)(t'_slot )) then
            return t'_slot
return NotFound
```

**Figure 3.21**: SearchSlotInTimeWindow Algorithm.

In SearchSlotInTimeWindow algorithm (Fig. 3.21), C is the total capacity, which is summation of departure and arrival capacities, of airport in time window. NE is the loaded slots in related phase which can be arrival or departure. And, $NE_r$ is the loaded slots in reverse phase of related phase. Check total capacities and all slots of airport. If number of loaded slots is smaller than total capacity than continuous. With this approach, arrival and departure capacities of an airport can be shifted to each other.

Check from start of the time window until $t_{window}$. If there are no slots available in the time window return NotFound.

### 3.3.3 Implementation with ALLFT+ data

It is clear that, algorithm do not generate any ground delay when any capacity restriction does not exist. This means that all aircrafts take-off on time at normal conditions. However, algorithm generates ground delays when extreme conditions exist to handle the problem before take-off. An implementation is performed with

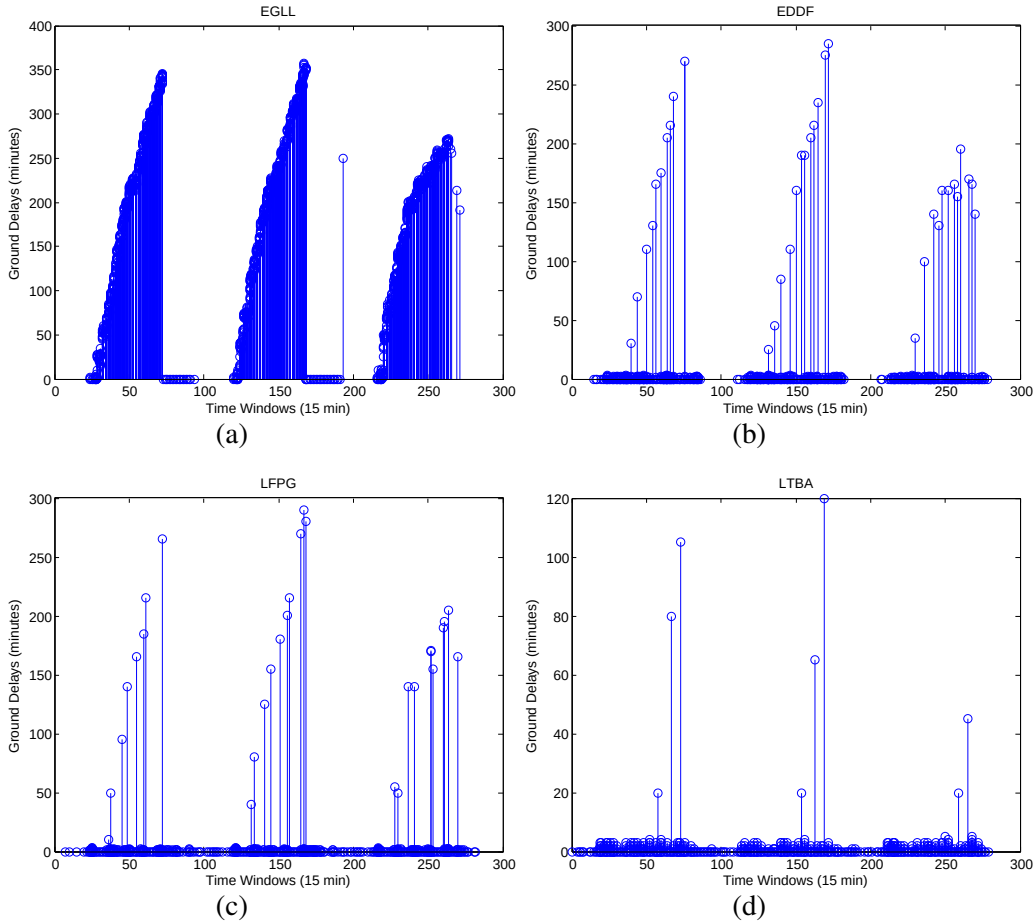



**Figure 3.22**: Ground Delays Because of Capacity Drop at EGLL: (a)EGLL(Heatrow). (b)EDDF(Frankfurt). (c)LFPG(Charles de Gaulle. (d)LTBA(Ataturk).

capacity restriction at Heatrow Airport. Arrival and departure capacity of Heatrow set to half values of real values, and situation is presented in Figure 3.22. This implementation is performed with ALLFT+ data for three days (from 18.03.2013 to 21.03.2013). When capacity of Heatrow is decreased, huge ground delays are generated in Heatrow (Fig. 3.22a). This situation affects EDDF (Frankfurt) (Fig. 3.22b) more than LTBA (Ataturk) (Fig. 3.22d) because of strong connection between

EGLL and EDDF, which is presented before. It is shown that delays are generated periodically. In nights, delays are absorbed due to low demands.

# 4. CONCLUSIONS AND REMARKS

In this thesis, two different study were realized, which are related to ATC and ATFM parts of ATM.

## 4.1 First Part of Study

In the first part of study, two different hybrid system models were presented for automated air traffic control in approach and en-route operations. The algorithm was built upon the domain expertise and emulates closely how an actual air traffic controller decision procedure works. The algorithm was shown to have polynomial complexity with respect to the number of aircrafts and their waypoints. The applicability of the algorithm was demonstrated successfully on real flight datasets and the integration of the algorithm with a radar display and Boeing 737-800 flight deck simulator was also demonstrated. So, workload of ATCO was decreased and sector capacities was enhanced with automated ATCO.

## 4.2 Second Part of Study

In the second part of study, analysis showed that European air traffic flow can be simplified using only flights come from European airports, minor airports can be aggregated relative to region to simplify the network model. An alternative capacity definition was defined as nominal capacity, which was extracted from data set. And, airports based flow models were constructed from data on the light of analysis. An algorithm was designed to reallocates the departure time of flights with arrival and departure demand-capacity balance. Unnecessary holding in airborne was prevented with ground holding using this algorithm, so workload of ATCO was decreased and

fuel consumption was also decreased. Moreover, sector capacities was be used more efficiently with improvement of strategic planning part.

## REFERENCES


[1] **Garcia-Avello, C. and Swierstra, S.** (1997). Human Role in ATM: Support for Decision Making, *NASA*, 19980201667.

[2] **ATAG** (2014). Revolutionising Air Traffic Management.

[3] **IATA** (2011). Vision 2050. Retrived from `https://www.iata.org/about/Documents/vision-2050.pdf`, date retrieved 16.03.2015.

[4] **Vossen, T.W.**, **Hoffman, R. and Mukherjee, A.** (2012). Air traffic flow management, Quantitative problem solving methods in the airline industry, Springer, pp.385–453.

[5] **SESAR** (2007). Concept of Operations.

[6] **JPDO** (2007). Concept of operations for the next generation air transport system. Retrived from `http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA535795`, date retrieved 16.03.2015.

[7] **NextGen** (2011). Delivering the Mid-Term Vision.

[8] **HLG on Aviation Research** (2011). Flightpath 2050 Europe's Vision for Aviation.

[9] **ACARE** (2007). Out Of The Box, Ideas about the Future of Air Transport.

[10] **ACARE** (2010). Aeronautics and Air Transport: Beyond Vision 2020 (Towards 2050).

[11] **EREA** (2010). EREA vision for the future – Towards the future generation of Air Transport System.

[12] **EREA** (2011). EREA study ATS 2050 Phase 2, Towards Full Automation, ATS Vision for the Future.

[13] **SESAR** (2012). HALA! Position paper.

[14] **Kelly III, W.E.** (1999). Conflict detection and alerting for separation assurance systems, Digital Avionics Systems Conference, 1999. Proceedings. 18th, volume 2, IEEE, pp.6–D.

[15] **Shewchun, J.M.**, **Oh, J.H. and Feron, E.** (1997). Linear matrix inequalities for free flight conflict problems, Decision and Control, 1997., Proceedings of the 36th IEEE Conference on, volume 3, IEEE, pp.2417–2422.

[16] **Paielli, R.A. and Erzberger, H.** (1997). Conflict Probability for Free Flight, *Journal of Guidance, Control, and Dynamics*, 20(3), 588–596.

[17] **Sridhar, B. and Chatterji, G.** (1997). Computationally efficient conflict detection methods for air traffic management, American Control Conference, 1997. Proceedings of the 1997, volume 2, IEEE, pp.1126–1130.

[18] **Vink, A.**, **Kauppinen, S.**, **Beers, J. and de Jong, K.** (1997). Medium term conflict detection in EATCHIP phase III, Digital Avionics Systems Conference, 1997. 16th DASC., AIAA/IEEE, volume 2, IEEE, pp.9–3.

[19] **Bakker, G. and Blom, H.A.** (1993). Air traffic collision risk modelling, Decision and Control, 1993., Proceedings of the 32nd IEEE Conference on, IEEE, pp.1464–1469.

[20] **Eby, M.S.** (1994). A Self-Organizational Approach for Resolving Air Traffic Conflicts., *Lincoln Laboratory Journal*.

[21] **Koren, Y. and Borenstein, J.** (1991). Potential field methods and their inherent limitations for mobile robot navigation, Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, IEEE, pp.1398–1404.

[22] **Tomlin, C.**, **Pappas, G.J. and Sastry, S.** (1998). Conflict resolution for air traffic management: A study in multiagent hybrid systems, *Automatic Control, IEEE Transactions on*, 43(4), 509–521.

[23] **Tomlin, C.**, **Mitchell, I. and Ghosh, R.** (2001). Safety verification of conflict resolution manoeuvres, *Intelligent Transportation Systems, IEEE Transactions on*, 2(2), 110–120.

[24] **Bayen, A.M.**, **Grieder, P. and Tomlin, C.J.** (2002). A control theoretic predictive model for sector-based air traffic flow.

[25] **Durand, N.**, **Alliot, J.M. and Granger, G.** (1997). Optimal resolution of en route conflicts, Procceedings of the 1rst USA/Europe Seminar.

[26] **Frazzoli, E.**, **Mao, Z.H.**, **Oh, J.H. and Feron, E.** (2001). Resolution of conflicts involving many aircraft via semidefinite programming, *Journal of Guidance, Control, and Dynamics*, 24(1), 79–86.

[27] **Pallottino, L.**, **Feron, E.M. and Bicchi, A.** (2002). Conflict resolution problems for air traffic management systems solved with mixed integer programming, *Intelligent Transportation Systems, IEEE Transactions on*, 3(1), 3–11.

[28] **Yang, L.C. and Kuchar, J.K.** (1997). Prototype conflict alerting system for free flight, *Journal of Guidance, Control, and Dynamics*, 20(4), 768–773.

[29] **Erzberger, H.**, **Davis, T.J. and Green, S.** (1993). Design of center-TRACON automation system.

[30] **Brudnicki, D. and McFarland, A.** (1997). User Request Evaluation Tool (URET) conflict probe performance and benefits assessment, *MITRE CAASD, MP*, 1130065520.

[31] **EUROCONTROL** (2010), AMAN Status Review.

[32] **Glover, W. and Lygeros, J.** (2004). A stochastic hybrid model for air traffic control simulation, Hybrid Systems: Computation and Control, Springer, pp.372–386.

[33] **I. Lymperopoulos, A. Lecchini, W.G.J.M. and Lygeros, J.** (2007). A Stochastic Hybrid Model for ATM processes.

[34] **EUROCONTROL** (2011). User Manual for the Base of Aircraft Data (BADA) Revision 3.9. Retrived from `http://www.eurocontrol.int/eec/gallery/content/public/document/eec/other_document/2011/EEC-Technical-Report-110308-08.pdf`, date retrieved 09.01.2014.

[35] **ICAO** (2009). Procedures for Air Navigation Services Air Traffic Management.

[36] **DHMI** (2013). Aeronautical Information Publication.

[37] **Cassandras, C.G. and Lafortune, S.** (2008). Introduction to discrete event systems, Springer Science & Business Media.

[38] **Hopcroft, J.E.** (1979). Introduction to automata theory, languages, and computation, Pearson Education India.

## CURRICULUM VITAE


**Name Surname:** Barış Başpınar

**Place and Date of Birth:** Elazığ, June 1990

**Adress:** Controls and Avionics Laboratory, Faculty of Aeronautics and Astronautics, Istanbul Technical University, TURKEY

**E-Mail:** baspinarb@gmail.com


**B.Sc.:** Istanbul Technical University,
Faculty of Aeronautics and Astronautics,
*Aeronautical Engineering*, 2013

Faculty of Electrical and Electronics Engineering
*Control Engineering*, 2013

**Professional Experience and Rewards:**

The Scientific and Technological Research Council of Turkey (TUBITAK)

- *Graduate Research Fellowship* *March 2014 - present*

Istanbul Technical University

- *Honored by ITU for completing Double Major in 4 years* *July 2013*
- *Aeronautical Engineering Second Best Student Award* *July 2013*
- *Control Engineering Best Student Award* *July 2013*

**List of Publications and Patents:**

▪ Baspinar B., Temeltas H. , "EKF Tabanli INS/GPS Entegrasyonu ile Navigasyon", *Otomatik Kontrol Ulusal Toplantisi*, Eylul 2013

**PUBLICATIONS/PRESENTATIONS ON THE THESIS**

▪ Baspinar B., Pasaoglu C. and Inalhan G., "Hybrid Modelling and Automation of Air Traffic Controller Decision Process : Separation Assurance", *6th International Conference on Research in Air Transportation*, May 2014

▪ Baspinar B. et al., "Ayrışma Amaçlı Hava Trafik Kontrol Karar Süreçleri İçin Hibrid Otomat Modeli", *V. Ulusal Havacılık ve Uzay Konferansı*, Eylul 2014

▪ Pasaoglu C., Baspinar B., Ure N.K. and Inalhan G., "Hybrid Modelling and Automation of AirTraffic Controller Decision Process: Separation Assurance", *Proc. IMechE: Part G Journal of Aerospace Engineering*, 2015 (Submitted - Review in Process)