FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Evolutionary Computation methods applied to Operational Control Centers

António José Ferreira de Castro Moura

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Paula Rocha

Second Supervisor: António Castro

July 27, 2015

Evolutionary Computation methods applied to Operational Control Centers

António José Ferreira de Castro Moura

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Daniel Silva
External Examiner: Paulo Novais
Supervisor: Ana Paula Rocha
July 27, 2015

Abstract

Airline companies, as any other company, look forward to maximize their profits. One way of doing it is by minimizing costs. Some of these costs are easier to manage; some others are unpredictable and if they are not solved in a short time window, the ripple effect will increase even more those costs.

In the context of airline companies, when an operational plan gets affected by a disruption, flights tend to get delayed, giving origin to an Irregular Operation (IROP). Operational plans are not only affected by large scale disruptions, such as the interruption of airspace and bad weather, but also and most commonly by minor scale disruptions, like crew sickness and aircraft malfunctions.

Typically, IROPs affect three main dimensions of the operational plan, two related with resources (aircraft and crew), and one other with passengers.

Disruption Management is the process of solving a disruption which is affecting an operational plan, by minimizing the impact and cost that disruption may cause in the original plan.

Solving disruptions is no easy task for the Airline Operations Control Center (AOCC). To ease this task, AOCCs use tools which are able to assist them in managing disruptions. MASDIMA (Multi-Agent System for Disruption Management) is one of those tools that help AOCCs in both analysing the impact of the disruption and solving it. It uses a Multi Agent System paradigm to distribute the expertise related to the three dimensions and automated negotiation as a way of integrate these three perspectives.

In this thesis, three evolutionary computation algorithms (Particle Swarm Optimisation, Ant Colony Optimisation and Genetic Algorithm) will be studied and two of them implemented in MASDIMA on the aircraft dimension of the problem. The goal and main contribution of our work is to provide a comparison regarding both computing time and quality of solutions, with two already implemented algorithms (Hill Climbing and Simulated Annealing).

Resumo

As companhias aéreas, assim como qualquer outra empresa, tentam sempre maximizar os seus lucros. Uma maneira de isso ser feito é minimizando os custos. Alguns desses custos são facilmente controláveis mas, outros, são imprevisíveis e se não forem solucionados num curto espaço de tempo, tendem a aumentar devido ao efeito dominó existente.

No contexto das companhias aéreas, quando um plano operacional é afetado por uma rutura, os voos tendem a atrasar-se, dando origem a uma Operação Irregular (IROP). Os planos operacionais não são somente afetados pelas ruturas de larga escala, tais como interrupção do espaço aéreo e más condições atmosféricas, mas também, pelas mais comuns e de menor escala, tais como absentismo da tripulação ou avaria de aviões.

Tipicamente, as IROPs afetam as três dimensões principais de um plano operacional. Duas relacionados com recursos (avião e tripulação), e a outra com o impacto nos passageiros.

A Gestão de Ruturas é o processo de resolução de uma rutura que afeta o plano operacional, minimizando o impacto e o custos que essa rutura pode causar no plano original.

A resolução de ruturas não é uma tarefa fácil para os Centros de Controlo Operacionais das companhias Aéreas (AOCC). Para facilitar esta tarefa, os AOCCs usam ferramentas que os possam assistir. O MASDIMA (Multi-Agent System for Disruption Management) é uma dessas ferramentas que ajudam os AOCCs a analisar impacto das ruturas e a resolver os problemas causados pelas mesmas. O MASDIMA usa o paradigma dos Sistemas Multi-Agente para distribuir a competência relacionada com as três dimensões, e negociação automática como uma forma de integrar estas três perspectivas.

Nesta tese, três algoritmos de computação evolutiva (Particle Swarm Optimisation, Ant Colony Optimisation e Genetic Algorithm) vão ser estudados e dois deles implementados no MASDIMA. O objetivo e a contribuição principal do nosso trabalho é providenciar uma comparação relativamente ao tempo de execução dos algoritmos e qualidade de soluções, com dois algoritmos já implementados no MASDIMA (Hill Climbing e Simulated Annealing).

Acknowledgements

I would like to express my special appreciation and thanks to my supervisors Professor Ana Paula Rocha and Professor António Castro, for guiding me through this new academic stage, all the motivation towards the subject and last but not least for the trust they placed in me.

To BEST Porto and all the friends I have made there, which made me improve my personal and soft skills and for the fantastic support.

To my friends which without them I would not be who I am today, and these last years would not be the same.

A special thanks to my family, for all they taught me and for all the sacrifices that they have made on my behalf.

António José Ferreira de Castro Moura

"I can't change the direction of the wind,
but I can adjust my sails to always reach my destination."

Jimmy Dean

Contents

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

ACO	Ant Colony Optimisation
AI	Artificial Intelligence
AOCC	Airline Operations Control Center
ARO	Aircraft Recovery
DM	Disruption Management
EA	Evolutionary Algorithm
GA	Genetic Algorithm
HC	Hill Climbing
IROP	Irregular Operation
MASDIMA	Multi-Agent System for Disruption Management
OCC	Operations Control Center
PSO	Particle Swarm Optimisation
SA	Simulated Annealing
SD	Standard Deviation

Chapter 1

Introduction

This chapter will present the context on which the work is going to be performed including the motivation and objectives of the work. The chapter ends with a summary on each of the chapters presented further on this document.

1.1 Context

Airline companies are doing a great effort in order to maximize their revenues while keeping their costs at a minimum. This is no easy task, and due to that, they are investing in tools that optimize their operational schedules. In spite of having an optimal plan, even this one has a strong probability of being affected during the operation, by disruptions as weather changes, aircraft malfunctions or extra maintenance and crew absenteeism, which may lead into delayed flights causing an irregular operation (IROP). In order to manage disruptions the Airline Operations Control Center (AOCC) try to find a solution that will result in a minimum impact either for the flight schedule as well as for the cost. Usually AOCC solve disruptions using a sequential approach, i.e., the process used to solve the disruption is executed in a specific order, in which the three dimensions of the problem - aircraft, crew and passenger - are present in this same order. [CRO13]

While using this sequential approach, different importances are given to the dimensions of the problem that will restrict too much the last dimensions to be solved, making it harder to obtain a good integrated solution.

"MASDIMA (Multi-Agent System for Disruption Management) is capable of monitoring the operational plan and deciding if an event requires or not an action. It is autonomous with decision making capabilities and automates the repetitive tasks; it is adaptive to changes on the environment (includes learning capabilities); it is dynamic and provides solutions in almost real-time and allows the inclusion of a human-in-the-loop to improve the user acceptance of the solutions found automatically by reacting and learning the preferences of this user." [CRO13]

MASDIMA is a Multi-Agent System developed at LIACC that provides the environment where the approached proposal in this thesis will be implemented and tested.

1.2 Motivation and Objectives

One of the problems that the airline industry is facing happens during the day of operations and refers to unexpected events which can disrupt and jeopardise the operational plan, leading to IROPs. Examples of these events can be an adverse change in the weather conditions, unexpected aircraft breakdown leading to a longer maintenance or sick crew, among others. If a proper recovery plan is not taken in a short time, the disruptions can propagate in a large scale over time, yielding new disruptions. One example of this can be if the crew is meant to do another flight and it is stuck at the previous airport, delaying the flight where it was supposed to be, or imposing the need to assign a new crew to that flight. Airlines try to minimize the impact of these events by, for example, using the same crew to perform a set of flights instead of a different crew for each flight. Since the use of expensive recovery actions like ferrying an aircraft, rebook passengers on other airlines and hire a new crew for another flight, increases the operational cost of the recovery plan and reduces the expected revenues the airline company has with the flight, a proper plan must be taken into account. [AENA04]

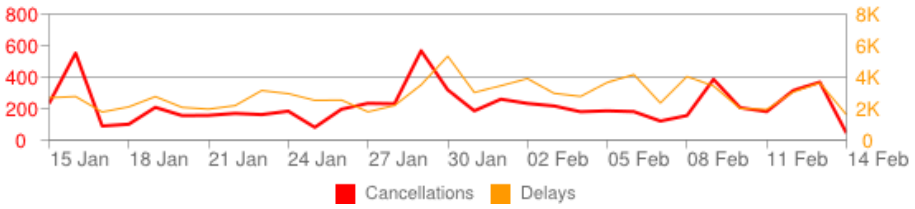

Figure 1.1: European Canceled and Delayed Flights - 2015
[CDF15]

According to the data on the figure 1.1, during a period of thirty days there was a total of 6.861 flights canceled and 88.399 flights delayed only in Europe. The y-axis represents the number of flights, the left y-axis presents the number of canceled flights and the right y-axis the number of delayed flights (this last is using K to infer a scale of 1 to 1.000).

To get some awareness about the dimension of this problem, consider for instance the American Airlines company, which schedules about 510 aircraft of 14 types to 140 cities covering a total of 2.700 flights and assigns 25.000 crew members to these 2.700 flights. In order to operate a system with such a magnitude and complexity, airlines rely on optimisation techniques for their planning, attempting to get optimal operational plan, making an efficient use of resources, leading to better revenues. [CCZ10]

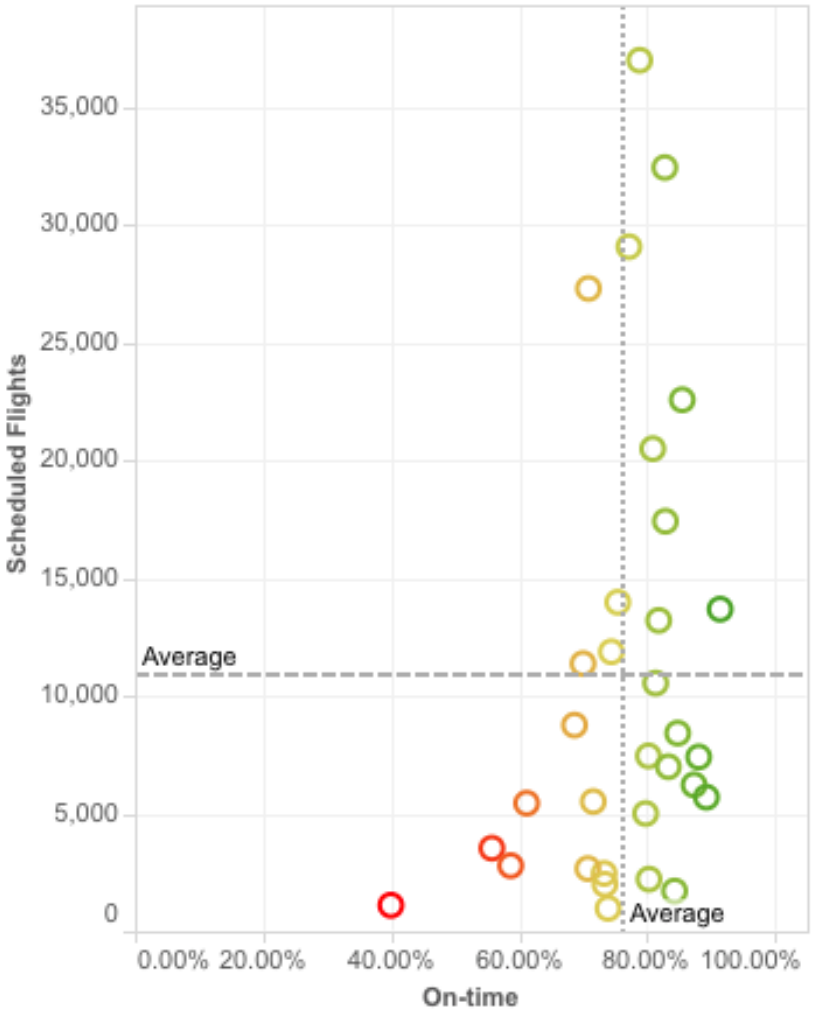

Figure 1.2: Major Airlines from Europe - Arrival Performance, December 2014 [AFS15]

Figure 1.2 represents the percentage of delayed flights in relation to the number of scheduled flights by the major airlines in Europe, during the month of December 2014.

Despite of the average delay among the major Airlines from Europe being 42.05 minutes, only 16% of the flights arrive with delay. According to a study from [CTA04] for each minute of delay at-gate after the first 15 minutes, there is a value of € 72 per minute of delay.

Studies have been made through several airline companies and they indicate that the cost with IROPs can cost up to 3% of the airline annual revenue, [CCZ10]. It was estimated that a better recovery process could result in cost reductions with IROPs of at least 20%, [Irr96]. According to experiments made with TAP Portugal data, authors of [CRO13] show that in the worst case scenario, with a better recovery process it is possible to improve the cost reduction of IROPs between 13% to 17%.

Since there is a way to reduce the costs with IROPs, by improving the recovery process, in this

dissertation it will be presented a study regarding the implementation of methods of evolutionary computation for the Aircraft Recovery (ARO) process in Operation Control Centers (OCCs). The methods will be implemented on MASDIMA and a benchmarking between our methods and the ones previously implemented on the system will be performed.

1.3 Thesis Structure

This dissertation is divided into four more chapters. Chapter 2 reviews the state of the art on the field of Disruption Management (DM) and gives the minimum theoretical information to understand Evolutionary Computation methods used on our approach. Chapter 3 presents the approach taken in the implementation of each method used on the Aircraft Recovery Problem. Chapter 4 presents the results and the benchmarking of all methods. Chapter 5 closes the dissertation, presenting a conclusion and some additional ideas to be taken into account regarding future work.

Chapter 2

Background

This chapter presents background information, related with Disruption Management (DM) in Airline Operational Control Centers (AOCC) as well as with Evolutionary Algorithms (EAs).

2.1 Introduction

This chapter aims to provide some background information regarding DM shortly before or at the day of operations as well as about evolutionary algorithms, a subset of evolutionary computation that we will use in our approach.

During the operational plan, the schedule often has to be revised due to disruptions with distinct sources, such as; nature related, technical problems or crew related. In order to provide some details regarding DM, the following section presents an introduction to DM in airline industry and a description of the planning process, as well as some methods that airlines use in order to create a more flexible schedule.

Since we are going to use evolutionary algorithms during the DM task, a study of Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO) and Genetic Algorithm (GA) is presented in section 2.3.

2.2 Disruption Management

Throughout a given plan, it is normal the existence of disruptions, caused by internal and external factors, leading to a certain deviation from the original plan and potentially affecting its execution. For example, the addition of new restrictions and unpredictable events such as weather changes and terrorist attacks. Thus, any change made in the original plan is called a disruption.

DM according to [YQ04] can be defined as "*after a plan (lying close to an optimal solution for the plan, or even being the optimal solution) have been created, either by optimisation models or using schemes, and during its execution a disruptions occurs. It is possible that the plan moves*

away from the optimal or even became unfeasible. Leading to a need to revise the original plan, reflecting on changes and restrictions implemented by the subsequent disruption, thus minimising the overall impact originated."

The definition of DM goals passes through three essential points, being these to carry out the operational plan, minimize costs and return to the plan as soon as possible. In spite of the first two objectives being part of the DM goals, they are also clearly present in the construction of the original plan, in an attempt to create an optimal plan. Although airlines must manage to balance two types of dimensions (crew and aircraft), it is also imperative that when in the presence of a disruption, it is considered a global view - now considering also the passengers' dimension -, and to fulfilled the operational plan leading them to their final destination at the agreed time, also paying attention to aircraft and tail assignment reducing travel costs. The third objective is common in problem solving and also somehow connected to the other two goals and to all airline resources (as it may compromise the plan outlined for each). [KLL+07]

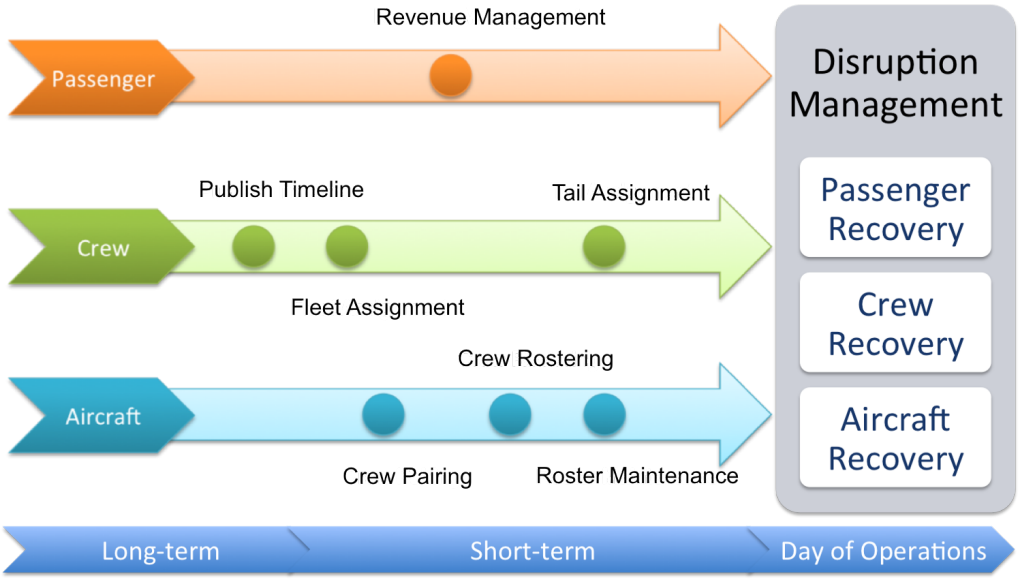

Figure 2.1: Plan creating schema

DM systems face a number of challenges, namely:

- **Timing**, solutions must be generated in a short time when comparing to the time spent at the planning stage.
- **Data**, it is often spread over several databases.
- **Feasibility**, for a solution to be considered as a feasible solution, this one must obey a set of rules, sometimes quite complex.

Alongside with benefits that can be summarized as:

- **Quality of Service**, improvement of quality of service offered to customers.

- **Resource utilisation**, better use of resources.

Airlines companies, as they are already aware of possible disruptions, try to anticipate these. This is done by adding some flexibility in the early stages of the plan creation, as shown in figure 2.1, so it is easier to handle disruptions during the day of operations. The following methods can be incorporated in this process:

- **Add slack in the plans**, instead of planning everything with close limits of the ideal, there is added extra time to the actions.

- **Crew follow each other and the aircraft**, this method preserves some of the various properties of the original plan resulting in an easier recovery and making monitoring an easier operation.

- **Out and back**, If an aircraft does the flight from a hub to a spoke and back to the same hub, these two flights can be canceled without affecting the rest of the aircraft schedule, the same would happen to the crew if they would do both flights.

- **Stand by crew and aircraft**, an extra airplane and its crew can be valuable but are quite expensive resources that can be used in case of disruption.

- **Increase cruise speed**, airplanes have several cruise speeds depending on the altitude at which they travel. Since cruise travel is defined by the speed at which the aircraft has the best efficiency concerning fuel consumption. Usually planes travel at cruise speed, but this one can be increased, since it is usually lower than the top speed from the same plane, thereby saving time that can lead into higher cost brought by disruptions in the three dimensions planes, crew and passengers.

2.3 Evolutionary Algorithms

Evolutionary algorithms are part of a subset of Artificial Intelligence (AI), evolutionary computation, where there is a demand for an optimisation of a problem. An evolutionary algorithm can be defined according to the dictionary as “an algorithm which incorporates aspects of natural selection or survival of the fittest. An evolutionary algorithm maintains a population of structures (usually randomly generated initially), that evolves according to rules of selection, recombination, mutation and survival (...)”. [ead15]

Fogel, that introduced the concept of evolutionary computation, defined intelligence as “the capability of a system to adapt its behaviour to meet its goals in a range of environment”. [Fog06]

The following sections present three classes of evolutionary algorithms, namely Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO) and Genetic Algorithm (GA).

2.3.1 Particle Swarm Optimisation

PSO is an evolutionary computing technique introduced by Kennedy (Social Psychologist) and Eberhart (Electrical Engineer) for the first time in 1995 based on a metaphor of social behaviour, [PKB07]. The main objective is to create AI through the study of social interaction, instead of observing the individual skills. The first simulations performed by Kennedy and Eberhart were influenced by the work of Heppner and Grenander, [ES01].

The PSO was developed by observing shoals and flocks of birds in search of food. It is a search algorithm, which aims to find the global solution of the system and not just the local maximum, despite of recording all the local maxima of each and every of its particles, also the algorithm registers the global maximum. Each particle moves in a certain direction based on their own experience, as well as the experience of the entire group. Comparing to other evolutionary algorithms, the main advantages of PSO are its robustness towards the control parameters and its computational efficiency, [Che09].

PSO can be used across a wide area of problems and the applications are numerous and diverse. Examples of applications are video analysis and image, restructuring and design of electrical networks and cargo shipping; electronics and electromagnetism; power generation systems and power; scheduling; architecture and optimisation of communication networks; biological, medical and pharmaceutical; signal processing; robotics; neural networks; military and security. [PKB07]

PSO is based on a number of particles named entities, which are on the search space of the problem, and each entity is responsible for evaluating its own fitness value and knows its current position. Through the analysis of some data, such as its best and current fitness, the positions it occupied in each of the previous iterations, as well as facts of other particles and some random perturbations, the particle then calculates the speed with which will go through the search space. With this data, the algorithm proceeds to the next iteration after all the particles have been moved to their next position. It is expected that in the next iteration, the set of particles - swarm - approaches the optimal solution according to the defined fitness function.

Each individual particle is composed of three N-dimensional vectors, where N is the size of the search space of the problem. The three vectors are the current position, the best previous position and the current speed. The current position is defined by a set of coordinates which represent specific details of the problem, over which the search is based on, and in each iteration the set of coordinates makes a solution which is evaluated as a whole. If the new position is better than any found so far, its coordinates and the value generated by the fitness function are stored, in order to compare with future iterations.

The algorithm does not depend on only one particle, in particular but on a set of particles. A particle alone would not bring any advantages in solving the problem, since the interaction between particles would not exist and thus there was no cooperation among particles. Solving the problem happens with the interaction between the swarm and the analysis of the various individual behaviours, introducing the advantages of group/team work to the problem, mostly known as cooperation, [PKB07].

The interaction between particles follows a particular organisation, which is designated by neighbourhood, that defines the way that two or more particles communicate among themselves. The neighbourhood helps in most cases the algorithm not to get stuck in a local minimum of the fitness function. Examples of neighbourhoods are: single-sighted where each particle communicates only with the following one, ring topology where particles can communicate with the previous and the following one, fully connected topology where each particle has the possibility to communicate with any other particle of the problem and in isolated environments where only a specific number of particles communicates with each other but following the ring topology, [pso15]. Every particle communicates with other particles and they get affected by the best position of a particle that is in its neighbourhood [PKB07].

Listing 2.1 presents the pseudocode for the Particle Swarm Optimisation algorithm, as described in [pso15].

```
For each particle {
   Initialise particle
}

While stopping condition is not met {
   For each particle {
      Calculate particle fitness value
      If the new fitness value is better than the personal Best {
         Update personal Best with the new fitness value
      }
      If the personal Best is better than the global Best {
         Update global Best with the personal Best
      }
   }

   For each particle {
      Evaluate particle Velocity
      Use global Best and Velocity to update the particle Data
   }
}
```

Listing 2.1: Particle Swarm Optimisation Pseudocode (Adapted from [pso15])

In the pseudocode listing 2.1, some parameters that are important to its development have not been mentioned. One of them is the swarm size that can vary greatly with the context of a problem, or even with the complexity of its particles. Another parameter with its own importance is the maximum speed that a particle can get, preventing that particles change a high number of parameters in a single iteration, [pso15]. The opposite can also co-exist, i.e., the minimum speed that a particle can get, which can help the problem, forcing the particles to change their parameters on each iteration.

2.3.2 Ant Colony Optimisation

ACO is a metaheuristic nature inspired, with the main objective of solving Combinatorial Optimisation Problems with a high degree of difficulty. This algorithm was introduced in 1990 by M. Dorigo (Research Director for the FNRS and co-director of IRIDIA) [DB05]. The source of inspiration of ACO was the colonies of ants found in nature, specifically by their foraging, i.e., the search and exploration undertaken by animals in search of food resources, [Blu05].

Ants have a very specific way of performing this foraging, leaving a pheromone trail on the ground through which other ants can know what track they must follow in a certain way, this track is a way of communication between them. In analogy with what has been presented so far, the ACO algorithm is therefore based on indirect communication within a colony of agents (ants) simple motivated by the pheromone trails. Thus ants use the pheromone trails to build probabilistic solutions of the problem and they adapt it, while the algorithm is running, into something that reflects their experience in finding a solution, [DS10].

The ACO algorithm has many possible applications in a lot of areas and can even be considered the top algorithm for various applications. Examples of areas where applications of ACO algorithm have good results are: sequential ordering, planning, probabilistic Travel Salesman Problem, DNA sequencing, [DS10].

There are two different methods used to get solutions, one is based on creating the solution itself, starting from a partial solution and constructing it in the best way possible in order to create the final solution; the other is a typical local search algorithm, moving through the search space and working on solutions already completed, [DS10].

The construction algorithm defines the solution of the problem, step by step as in an incremental way starting from an initial solution (empty) and going through an iterative process by adding components to the solution without using backtracking until a complete solution is made. As a first case, quite simple, the components to be added to the solution are created using a stochastic process. However, there is a way to find better solutions if a heuristic (greedy construction heuristic) is used to estimate the benefit of adding a specific component to the current incomplete solution.

Greedy construction heuristic goes through a step by step algorithm which adds components in order to benefit incomplete solution. It basically starts from an empty solution, and until it is not complete, adds components calculated using the heuristic. In the end, the complete solution is returned. To calculate the component to be added to the solution, is used a heuristic which returns the component with the best heuristic according to the solution which was, at least until then, a partially solution.

A disadvantage of a heuristic running in a greedy basis, is that often the component which is selected is one out of a relatively small and closed set, belonging to the state of the solution at the time the heuristic was performed. Thus, and at an early stage, the solution search space starts to be restricted, as other possible results are taken off from the search space, leading to a reduced number of possibilities when the solution is considered to be close to a complete solution.

Moreover, there are local search based algorithms, which depart from the initial (complete) solution, and try to find a better solution in a neighbourhood of the current solution. The algorithm follows an iterative process and demands for a better solution in the neighbourhood. In the event that a better solution is found, the current solution is replaced by the newly found, and the search continues. This process goes on until no new solution is found and thus the algorithm ends at a local maximum (or perhaps the global maximum) [DS10].

The neighbourhood must be defined based on problem structure and what is desired of it. It is also through the defined structure of the neighbourhood that we can access a given set of other solutions when the problem lies over a particular solution through one step in the algorithm. This is an important issue in the search for local solutions, it is crucial to define a good neighbourhood since the neighbouring solution will replace the current one.

Listing 2.2 presents the pseudocode for the Ant Colony Optimisation, as described in [aco15].

```
1  Create the heuristic solution
2  Evaluate cost of the solution
3  Initiate pheromones
4
5  While stopping condition is not met {
6     For each ant {
7        Construct the solution
8        Calculate ant fitness value
9        If the new cost value is better than the personal Best {
10          Update personal Best with the new cost value
11       }
12       Update local solution and the pheromones trail
13    }
14    Update global solution and the pheromones trail
15 }
```

Listing 2.2: Ant Colony Optimisation Pseudocode (Adapted from [aco15])

One other option which can be included in the listing 2.2, are the Deamon Actions. These are centralized actions which cannot be implemented and performed by ants themselves. Examples of these are the deposit of additional pheromone in paths, either by adding it to the path of an ant (when in local search) or to another path of the problem which can somehow benefit the solution (when in global solution), [dea15].

The update of pheromones is meant to share the good solutions components so that they serve as a possible model for future iterations. In order to do this, we can use two mechanisms. The first one involves increasing the level of pheromone of a certain component of a solution, which is associated with a set of good solutions, where the goal is to convert a certain component in a choice that will be given as more certain for ants to choose (it is not imperative that ants will follow, since the algorithm continues to follow stochastic patterns). The second mechanism prevents an overly conversion to a sub-region of the search space, so the mechanism is implemented in a region where

a certain amount of pheromone will be decreasing in the course of time (iterations) the pheromone deposits left by other ants.

2.3.3 Genetic Algorithms

GAs were formally introduced by John Holland in the 70s in the University of Michigan, United States, [gai15]. These are considered meta-heuristic algorithm, a top-level general strategy which guides other heuristics to search for a solution, which are used efficiently in problems of optimisation and search (Goldberg, 1989; Gen and Cheng, 1997; Parmee, 1999), it is also based in nature, specifically with the process of natural selection, involving concepts such as mutation, recombination and selection. Examples of areas of application of the GAs are: benchmark problems, magnetically leviltated vehicles, optimisation of object shaping and circuit layout [MTK96].

In the original Holland's algorithm, a parent is selected according to the fitness of each actual being (these are selected through a stochastic decision, and the better the fitness for each being, the greater the chance of being chosen), and in case of recombination, the other parent will be chosen randomly.

However, there are some variations of the algorithm, such as both parents can be chosen based on the fitness value, different probabilities for the existence of mutation or recombination or population size, since the way the initial population is chosen can have a significant impact on results, [Ree95].

In order to find the optimal solution in the context of a large Combinatorial Optimisation Problem, GAs work on a population of N solutions.

Listing 2.3 presents the pseudocode for the Genetic Algorithm, as described in [gap15].

```
1  Initiate the first generation
2  Evaluate the population of the first generation
3
4  While stopping condition is not met {
5     Create the next generation
6     Selection of individuals
7     Crossover
8     Mutation
9     Evaluate the population
10 }
```

Listing 2.3: Genetic Algorithms Pseudocode (Adapted from [gap15])

The phases contained in the pseudocode of listing 2.3 are explained in next paragraphs, particularly the selection of individuals, crossover and mutation.

Selection of individuals, is based on selecting certain chromosomes, through the fitness value assigned by the fitness function and then these will be part of creating the offspring for the next generation. The connection between the algorithm and natural selection takes place here, since the best are the ones who are more likely to participate in this process. There are several methods of

selection of chromosomes, but the most popular is the "roulette-wheel", which is the analogy of the game itself. Once the chromosomes who will take part in the creation of a new generation are chosen, the process can be repeated.

Regarding the use of genetic operations, in classical GAs usually two are commonly used: crossover and mutation operators, wherein each has a different probability of occurrence, as in nature, in which the probability of each operator is different.

Crossover, the first stage involves selecting pairs of chromosomes that will be the parents of the next generation. This process is done stochastically according to a probability set to crossover. Then, for each pair of parents, it is necessary to decide the crossover point, i.e., the point at which discontinuity exists from the parents information and passes to the other parent to provide the remaining information. To complete this process, two new sprouts must be created, representing the two possible combinations: the first part of the first parent, and the second part of the second parent, leaving with the first part of the second parent and the second portion of the first parent.

Mutation, there is a probability that one or more genes (part of the same chromosome) change its value.

2.4 Summary

Looking into the classical, unorthodox and stochastic ways of both search and optimisation algorithms, there are two different methodologies. In one hand, the classical way which goes as a point-by-point approach through the problem and in each iteration the solution is modified into a different one, hopefully better. In the other hand, using both the unorthodox and stochastic ways, particularly the evolutionary algorithms as stated above are motivated by nature evolutionary principles, which leads the search towards an optimal solution. The difference here is that evolutionary algorithms make use of a population of solutions, and not just one single solution like in the classical algorithms. If there is just one solution, then it is expected to the rest to converge into that solution. If there are multiple solutions, then the algorithm can use multiple optimal solutions as its final solution [Deb01].

In the next chapter, for both PSO and ACO, an approach to the Aircraft Recovery problem will be described.

Chapter 3

Aircraft Recovery Problem

In this chapter the Aircraft Recovery (ARO) Problem will be described as well as our approach used to solve the problem.

3.1 Problem Description

Originally, upon the creation of an initial plan, an aircraft is scheduled to fly a set of routes. A route is a sequence of pairs of airports which are served by an airline, with both departure and arrival times. During the operational plan, disruptions may occur, leading to infeasible routes with a potential increased cost related to that route. To produce recovery plans is a complex task, since more than one dimension must be taken into account, and be re-planned, typically, aircraft, crew and passenger dimensions. The ARO process, responsible to solve the aircraft dimension of the disruption problem, cooperates with the remaining dimensions so that a feasible solution is found and able to be implemented. Additionally the quality of this solution must be estimated so it can be compared to other possible solution and also to the disrupted situation.

ARO needs to provide a solution for this disruption, either by providing a new aircraft to perform the route or by delaying the flight until the disruption is settled, [RJN03]. The ARO arises when a disruption occurs and its main goal is set to restore or recover the initial plan as much as possible in order to minimize the cost and the delay of both the disrupted flight and the operation plan, [LSLC05].

3.2 Approach

In this section, the architecture of MASDIMA will be presented and for both PSO and ACO algorithms a description of the approach used in the Aircraft Recovery problem is provided. For each method or algorithm, the details on how they were implemented and nature metaphors will be complemented with information from the problem itself.

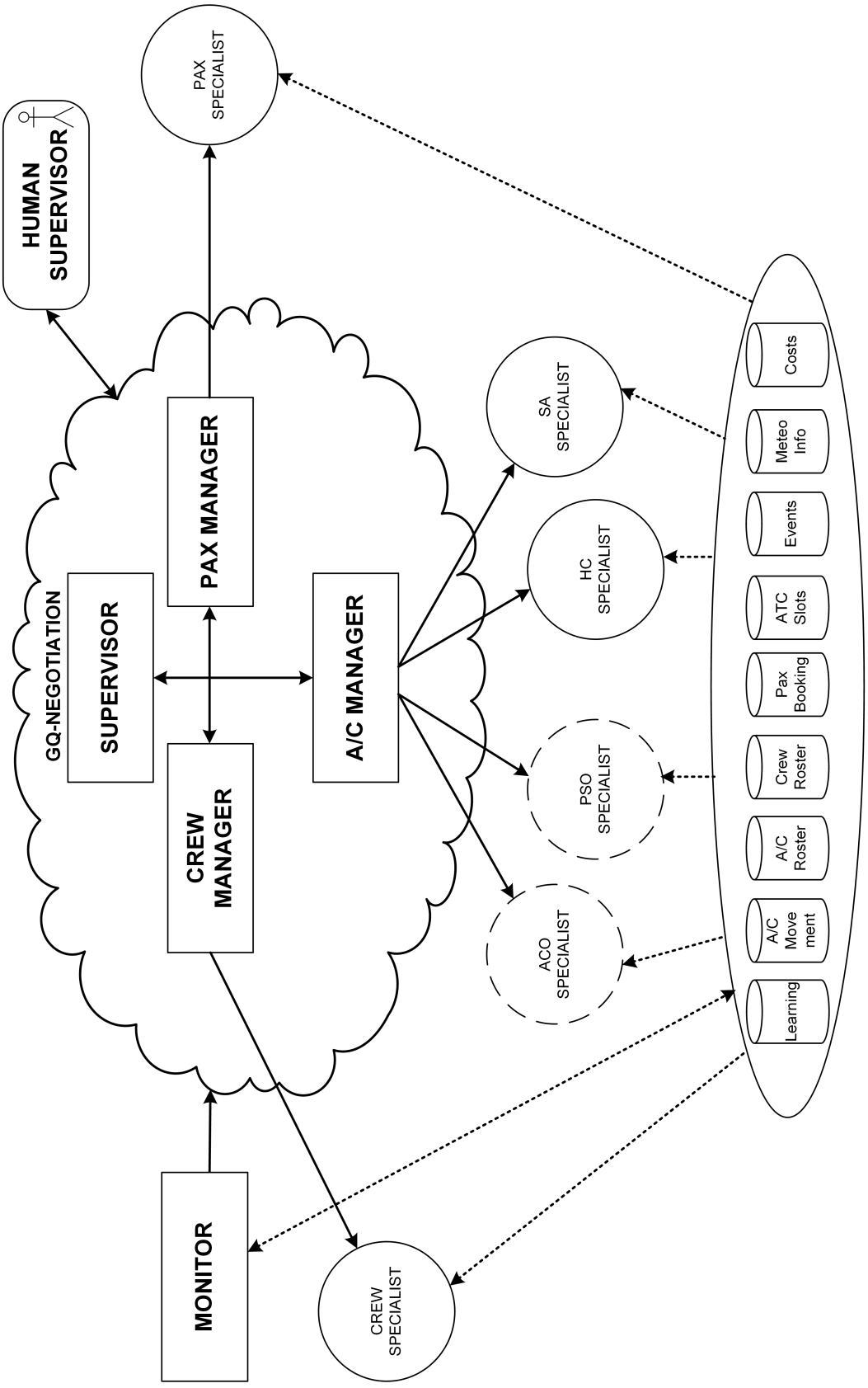

Figure 3.1: MASDIMA Architecture
[CRO13]

Figure 3.1 represents the architecture of MASDIMA. Solid and round cornered boxes represent respectively agents which are implemented and user interface agents. Solid lines represent interactions between agents and the dashed lines actions in the environment. The cylinders inside the ellipse represent the data sources which are part of the environment for agents to consult and to act upon the data.

MASDIMA includes three Manager agents: the Crew Manager, responsible to achieve a solution regarding the crew dimension; the A/C Manager, responsible to achieve a solution regarding the aircraft dimension; and the Pax Manager, responsible to achieve a solution regarding the passenger dimension. Each one of these Manager agents has the help of Specialist agents. A Specialist agent is the expert agent that know effectively how to solve problems concerning their expertise (dimension). Manager agents do not have the knowledge necessary to present an integrated proposal. For that to happen they need to engage into a negotiation protocol with other Managers in order to prepare a proposal to be sent to the Supervisor agent. The Supervisor Agent is the agent who knows the preferences of the user and will select, among the solutions sent by the three Managers, the best one. Again, this selection is the result of a negotiation process. [CRO13] Circles represent Specialist agents, continuous perimeters represent previously implemented agents and dashed perimeters represent the agents implemented on this thesis.

Specialist agents support solving algorithms for the dimension of the Manager they are defined under, and all the partial [1] solutions gathered are sent to this one.

A partial solution is defined by a set of parameters that help Managers in their negotiation and to organise integrated solutions. To define a solution firstly there is the need to identify it, usually by giving the current round number of the method, then there is the need to specify the dimension on which the solution will be built upon and the action it intends to perform (as to exchange resources in order to perform different flights), therefore there is also the needed to identify the resource being proposed to perform the action and both the time it needs before executing (also referred as delay) that flight and the cost associated.

The HC Specialist agent implements the *Hill Climbing* algorithm and the SA Specialist agent implements the *Simulated Annealing* algorithm [KGV83]. These agents are the ones that will be compared with the evolutionary algorithms (Particle Swarm Optimisation and Ant Colony Optimisation) implemented in this work.

The optimisation that all Aircraft Specialists pursue, is based on the same objective function (Function 3.5) that evaluates the cost of exchanging the current resource with a different and suitable resource. The cost evaluation used in the objective function passes through a process which extends from getting the readiness of the resource to calculating the cost of resource exchange and the respective penalisation of the exchange.

Function 3.1 presents the penalization for an Aircraft Solution *Sac*.

$$Sac \rightarrow \Re, Penalization(Sac) := \sum_{1}^{n} (\gamma \times fleet_{pen} + \rho \times cap_{pen} + \mu \times pax_{pen}) \tag{3.1}$$

[1] A possible solution to a dimension of the problem.

with

$$n \in N,\ 1 \ \leq \ n \ \leq \ |Sac|,$$

$$\gamma \in \Re,\ 0 \leq \gamma \leq 1,$$

$$\rho \in \Re,\ 0 \leq \rho \leq 1,$$

$$\mu \in \Re, 0 \leq \mu \leq 1,$$

$$fleet_{pen} = \begin{cases} 0 & \text{if } fleet_n = fleet_{fd} \\ 1 & \text{if } fleet_n \neq fleet_{fd} \end{cases}$$

$$cap_{pen} = \begin{cases} 0 & \text{if } sseats_n < tseats_{fd} \\ 1 & \text{if } sseats_n \geq tseats_{fd} \end{cases}$$

$$pax_{pen} = \begin{cases} 0 & \text{if } tseats_n \geq sseats_{fd} \\ 1 & \text{if } tseats_n < sseats_{fd} \end{cases}$$

where:

- *fleet* is the aircraft fleet;
- *sseats* are the sold seats on the flight;
- *tseats* are the total available seats in the flight.

The intention of the penalization function is to penalize solutions if they breach certain conditions. The objective of $fleet_{pen}$ is to penalize solutions that use an aircraft which belongs to a different fleet. As for the the cap_{pen} is to penalize the solutions that use aircraft assign to flights that have more passengers (sseats) than available seats (tseats). And pax_{pen} is used to penalize the solutions that use aircraft assigned to flights that do not have enough available seats for the passengers.

Function 3.2 presents the delay for an Aircraft Solution *Sac*.

$$Sac \rightarrow \Re, Delay(Sac) := \sum_{1}^{n} \frac{(etd_n - std_n)}{\delta} \tag{3.2}$$

with

$$\delta \in \Re, 1 \ \leq \ n \ \leq \ |Sac|,\ \delta \ is \ \max(etd_n - std_n)$$

where:

- *etd* is the expected time of departure of flight;
- *std* is the scheduled time of departure of flight.

Function 3.3 presents the cost for an Aircraft Solution *Sac*, according to function 3.4.

$$Sac \rightarrow \Re, Cost(Sac) := \sum_{1}^{n} \frac{accost(sac_n)}{\theta} \tag{3.3}$$

with

$$\theta \in \Re, 1 \leq n \leq |Sac|, \ \theta \ is \ \max(accost(sac_n))$$

Function 3.4 presents the flight cost for all flights included in a solution *sac*.

$$sac = \sum_{i=1}^{|Fl|} (TkOff_i + Land_i + Park_i + Hand_i + Maint_i + Atc_i + Fuel_i) \tag{3.4}$$

where:

- *TkOff* (Takeoff charges) are applied by airports for each aircraft that takeoff;
- *Land* (Landing charges) identical to TkOff but applied for each aircraft landing;
- *Park* (Parking charges) are applied by airports for parking an aircraft, and it takes into account the time the aircraft is parked as well as the parking place;
- *Hand* (Handling charges) which includes the many service requirements an airline need between the time it arrives at a terminal gate and the time it departs on its next flight;
- *Maint* (Maintenance costs) which includes the maintenance that might be needed to perform during the turn-around of the aircraft at the airport as the aircraft period checks;
- *Atc* (Air Traffic Control charges) it is a charge related to air traffic services during the route of an aircraft;
- *Fuel* (Fuel costs) which includes the cost of fuel needed to go from the origin to the destination, plus any extra fuel required.

Function 3.5 presents the objective function for the solution, according to functions 3.1, 3.2 and 3.3.

$$Sac \rightarrow \Re, OF(Sac) := \alpha_1 \times Penalization(Sac) + \alpha_2 \times Delay(Sac) + \alpha_3 \times Cost(Sac) \tag{3.5}$$

with

$$\sum_{i=1}^{3} (\alpha_i) = 1$$

3.2.1 Particle Swarm Optimisation

As mentioned above in the subsection 2.3.1, PSO is a metaphor to shoals and flocks of birds when in search of food. Through the reading and interpretation of the subsection 2.3.1 and the listing 2.1,

four points are detected which require some explanation on how they will be inserted in the actual context namely the particle, swarm, velocity and neighbourhood.

Particle, it represents a full instance of the actual problem and a pseudocode is shown in listing 3.1. All states of the problem are cloned and instanced on the particle, and so, each particle moves from the beginning of the problem, this is from the aircraft which suffered the disruption, and will make changes accordingly. It also has full access to flight and plan costs, being this last one mostly known as the particle fitness, as to possible replacements to the disrupted aircraft. Since it is an instance of the problem, all the solution generated by this are stored until the end of the global algorithm, where it will share them with its manager.

```
1  class Particle{
2    pBEST, currentBEST //both represent the total cost of the solution. One for the
         previous round and another for the current round
3    velocity //how many changes they particle may suffer per round
4    tailNumber, referenceTailNumber //TN of possible exchange aircraft AND TN of
         disrupted flight aircraft
5    solutions //a set of solutions to send to its Manager
6    operational plan //to be Initialised
7    cost of operational plan //to be Initialised
8    possible exchanges to the disrupted flight //to be Initialised
9
10   Initialise{
11     Set current operational plan  //a Map of all flights with an aircraft assigned
           to them
12     Set current costs of operational plan  //a Map of all aircrafts with a cost
           associated
13     Set possible exchanges to the disrupted flight  //a List of aircrafts suitable
           to replace the disrupted flight aircraft
14   }
15
16   Objective Function{
17     Return Get aircraft delay + Get aircraft exchange costs + Get penalization
18   }
19 }
```

Listing 3.1: Particle Swarm Optimisation - Particle

Swarm, is a set of particles. So it is building different plans as it keeps comparing them with each other and therefore selecting the best solution to be sent to the aircraft manager.

Velocity, defines the ratio on how fast a particles will move through the solution space. Specifically, it is the number of changes a particle will do. A higher velocity means that the particle fitness is low comparing to the swarm best particle. However, for this particular problem, all particles velocity will be set to one, since a higher value would lead to a higher number of solution to be sent to its manager, and the quality of solutions sent would decrease.

Neighbourhood, although particles operate without interaction with each other, it is necessary

for comparing particles and to set new velocities to each round. For this problem the neighbourhood is set as single-sighted, where particles are only able to compare themselves to a better one.

In order to obtain solutions, and according to the listing 2.1, PSO starts by initiating all particles and for every iteration of the algorithm, particles are updated and their fitness value is calculated according to the specified objective function. The update made consists on keep changing the current aircraft chosen to perform the flight, with another aircraft from the suitable resources list, since all the resources present on the list are suitable to replace each others.

3.2.2 Ant Colony Optimisation

ACO in the same way as PSO is based on the foraging of species, but in particularly by ants since they use a pheromones trail so other ants can follow the right path. Through the reading and interpretation of the subsection 2.3.2 and the listing 2.2, there are three points which deserve some explanation on how they will be inserted in the actual context namely the ants, colony and pheromones.

Ant, it is representing an instance of the actual problem as shown in listing 3.2. Similar to a particle, it holds all states of the problem and its own solutions. However, it also saves the trails of visited vertexes, so it is able to apply pheromones after all ants reached their final vertex.

```
1  class Ant{
2    pBEST //represents the total cost of the solution
3    trail //represents all the changes made by the Ant during the current round
4    solutions //a set of solutions to send to its Manager
5    operational plan //to be Initialised
6    costs of operational plan //to be Initialised
7    possible exchanges to the disrupted flight //to be Initialised
8
9    Initialise{
10     Set current operational plan  //a Map of all flights with an aircraft assigned
           to them
11     Set current costs of operational plan  //a Map of all aircrafts with a cost
           associated
12     Set possible exchanges to the disrupted flight  //a List of aircrafts suitable
           to replace the disrupted flight aircraft
13   }
14
15   Objective Function{
16     Return Get aircraft delay + Get aircraft exchange costs + Get penalization
17   }
18 }
```

Listing 3.2: Ant Colony Optimisation - Ant

Pheromones, are the way of ants expressing that a certain path in the colony is a better or worse choice. However, in order to avoid getting stuck in a local maxima, pheromone trails tend to evaporate over the time by a certain ratio.

Colony, defines the search space of the problem, it also references a graph based structure where vertexes are the representation of an aircraft, and each edge is a link between aircrafts suitable for replace one another.

The process that ACO uses to generate solutions, and according to the listing 2.3.2, consists on initiating the pheromones trail and then keep on updating it through the creation of new solutions using ants to travel through the colony and leaving pheromone trails which are proportional to the quality of the solution provided by the ant. As the times goes by and more iterations are made, pheromone trails start to be more trustworthy and provide better solutions.

3.3 Summary

To sum up with, both methods are flexible and adapt well enough to the needs of ARP. Nevertheless, both algorithms have their flaws. PSO when in presence of a fairly short sized search space, works worse if a very low velocity maximum is not set, leading PSO to basically lose one of its main characteristics. The ACO running time starts to increase dramatically when the search space exceeds a certain limit.

In the next chapter, the results and analysis of each implemented method will be presented and conclusions will be taken.

Chapter 4

Experiments and Result

In this chapter we present the results from the experiments done with the methods implemented and described on the previous chapter. A comparison between the methods is also presented.

4.1 Experimentation Scenarios

This section describes the environment on which the tests were performed, as well as the generation of test scenarios and how data was handled.

The system used to host the tests was MASDIMA, which according to [CRO13] is holding data from an operational plan from September 2009 of TAP Portugal, since it has properties similar to the average of one year of operation. Data is related to the activity presented on the operational schedule as well as with operational costs. Furthermore, data used contains 49 disruption events, which were randomly selected from the operation plan, including, 49 flights, 31 aircrafts, 286 crew members and 4.760 passengers affected by the 49 disruption events.

For the experimentation scenarios, only five static events were used, since all of these five events represent a disruption in every dimension (aircraft, crew and passenger).

Regarding the metrics to measure the results, they take into account three main indicators:

- **Algorithm running time**, it is a measure of the algorithm efficiency. Also it is based on computing the average time of algorithms.
- **Number of solutions**, it represents the number of solutions a specialist sends to its manager.
- **Cost of the solution**, it is a measure of the solution quality.

Additionally **Utility** is also used as an indicator. It defines the utility of the integrated solution, taking into account the solution plans of all dimensions. However, it is not considered a main indicator, since other dimensions (crew and passenger) were using an automated process during the tests, which returns only randomly generated values.

4.2 Results

The tests as stated above, are based on only five events, since they all represent at least one disruption in every dimension, and all of them are certain to occur. The flights affected by these events have the corresponding flight numbers: 928, 1917, 864, 1614 and 839. All values presented below are an average of twenty test values, which may be consulted in Appendix A, received from each of the flights, which are by their own, and in particular for algorithm running time and number of solutions sent, an average of the first five rounds of negotiation that MASDIMA implements, the remaining aircraft cost and problem utility are just the average of the twenty test final values presented by the supervisor on the final integrated solution.

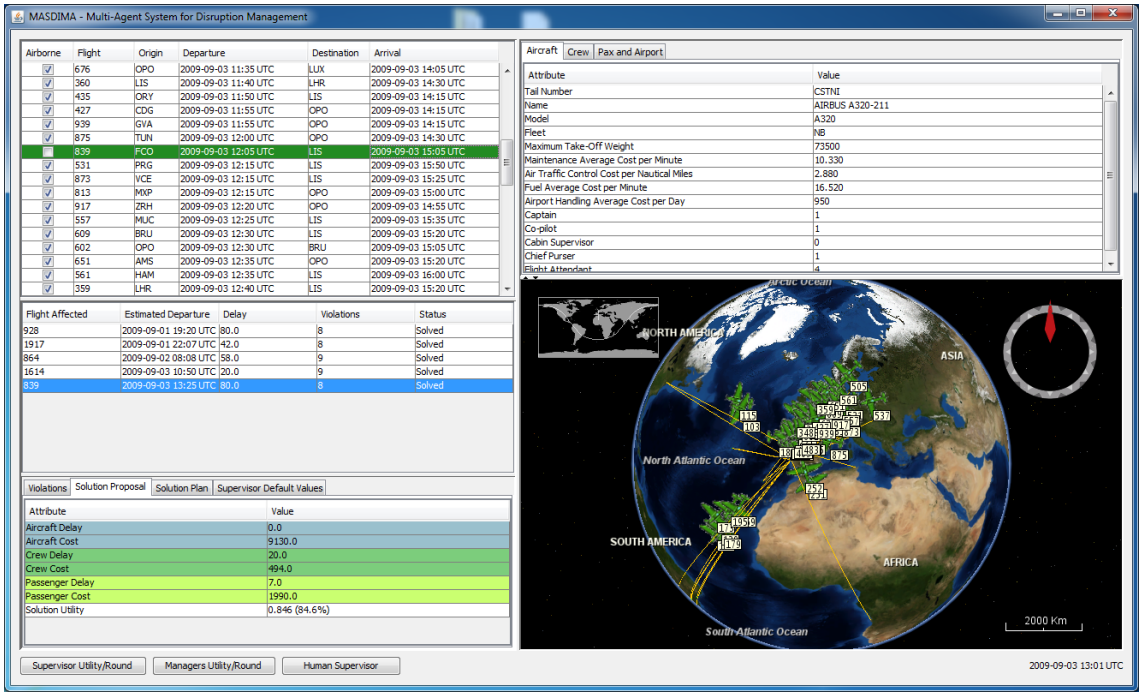

Figure 4.1: MASDIMA User Interface

Figure 4.1 presents the MASDIMA User Interface. Although it is also representing the moment after the fifth event was solved. The center section on the left side of the figure, is the area representing the Problem Resolution Status, where every time an event causes a disruption on a flight, information regarding that flight will appear as the flight number, estimated time of departure, expected delay, the number of violations occurred and the current status, and as the event on flight 839 was the last to appear, it is possible to observe all five events in that area. Also, in the bottom left side section of the figure, is the area representing the Solution Information, where access to more detailed information regarding the solution of the selected flight (for this case flight 839 was selected) like the specific occurred violations through tab violations, solution plan and specific values of the solution are provided [CRO13].

Table 4.1 presents the running time for the algorithms: HC, SA, PSO and ACO.

Table 4.1: Algorithms Running Time

Average Algorithm Running Time (ms)								
	928	1917	864	1614	839	Total	Average	Average SD
HC	770,8	532,1	865,2	87,9	38,5	2294,4	458,9	65,0
SA	77,8	69,8	76,7	72,8	69,9	366,8	73,4	7,6
PSO	168,9	151,7	157,6	87,7	59,7	625,6	125,1	17,1
ACO	1555,4	884,4	1883,0	227,0	327,3	4877,0	975,4	80,5

Table 4.1 Total column shows right at the start that either ACO and Hill Climbing (HC) running times are way over the ones of Simulated Annealing (SA) or PSO. Comparing SA and PSO, we can deduce that SA performs 71% better than PSO in matter of running time.

For a better understanding of the time consumed by these methods, the number of alternative solutions to the disruption flight must be taken into account. In this case, for the affected flights 928, 1917, 864, 1614 and 839, the number of alternatives given are 37, 27, 44, 3 and 2 respectively.

Table 4.2 presents the number of solutions sent by the algorithms: HC, SA, PSO and ACO.

Table 4.2: Number of Solutions sent to Manager

Average Number of Solutions								
	928	1917	864	1614	839	Total	Average	Average SD
HC	111,0	81,0	133,3	12,6	6,0	343,8	68,8	0,9
SA	10,0	10,0	10,1	10,2	10,0	50,2	10,0	0,1
PSO	7,7	7,0	7,9	4,4	3,6	30,5	6,1	0,9
ACO	4,3	4,0	4,6	2,8	4,0	19,7	3,9	0,3

The number of solutions are somehow more relative to compare. But, as stated before, all these events have at least one disruption of each dimension, so if specialists are sending a low amount of solutions to their managers, even if they are sending the top solutions of their dimension, it will result in discarding possible sets of better solutions, since the final solution will be an integrated solution will all dimensions. If a huge amount of solutions are sent, then managers will also need more time to came up with a solution. Table 4.2 shows that either HC and ACO fall into the characteristics provided before, leading SA and PSO to be the ones sending a more reasonable amount of solutions.

Table 4.3 presents the aircraft cost of the solution obtained by the algorithms: HC, SA, PSO and ACO.

Table 4.3: Aircraft Cost presented in Integrated Solution

Average Solution Aircraft Cost								
	928	1917	864	1614	839	Total	Average	Average SD
HC	3160,2	3151,0	3122,7	6612,0	9127,3	25173,2	5034,6	1,5
SA	3728,8	3650,8	4052,6	8235,0	9127,2	28794,3	5758,9	990,2
PSO	3437,7	3696,7	3778,6	7771,3	9126,0	27810,2	5562,0	853,9
ACO	3560,8	3792,5	4054,8	9627,8	9130,0	30165,8	6033,2	1156,1

Table 4.3 shows that HC provides the lowest cost to handle all five events and considering the universe of all aircraft cost values given by every solution generated with HC, the Standard Deviation (SD) is only 1,5. Following HC, we also have PSO, SA and ACO for this same order providing the best solutions and lowest SD.

To be noted that PSO provides solutions with a total cost of only 10% more than HC, while HC takes 267% more than PSO running time.

Table 4.4 presents the utility of the solution obtained by the algorithms: HC, SA, PSO and ACO.

Table 4.4: Solution Utility in Integrated Solution

Average Solution Utility								
	928	1917	864	1614	839	Total	Average	Average SD
HC	0,911	0,861	0,913	0,846	0,866	4,397	0,879	0,020
SA	0,907	0,848	0,914	0,840	0,869	4,378	0,876	0,022
PSO	0,913	0,844	0,907	0,840	0,865	4,368	0,874	0,022
ACO	0,907	0,842	0,905	0,825	0,872	4,350	0,870	0,027

None of the previous tables have shown the interaction and the importance of other dimensions on the integrated solution. Table 4.4 shows the utility of every single integrated solution. Having the best solution from the aircraft dimension does not mean it will be the best solution to fit in the integrated solution.

Also during the execution of these tests, Crew and Passengers dimension where using an automated process, which was just generating random values of delay and cost, without any knowledge of their search space or even comparing solution values. Therefore, the data from table 4.4 was only meant to endure the process of creating an integrated solution.

4.3 Summary

To sum up with, both EAs managed to adapt well enough to the ARO problem, and they even produce some interesting results, specially PSO, which is sending a reasonable amount of solutions to its manager so it has enough flexibility to interact with other manager's solutions. Within all four methods it stands in the second best place for both algorithm running time and aircraft cost.

Chapter 5

Conclusion

In this chapter conclusions on the work done will be taken and some ideas for future work will be presented.

5.1 Objective Fulfilment

From the three originally proposed methods to be implemented (Particle Swarm Optimisation, Ant Colony Optimisation and Genetic Algorithm), only two of them were successfully developed (Particle Swarm Optimisation and Ant Colony Optimisation). Either of the two implemented methods showed a quite good adaptability to the Aircraft Recovery problem, in spite of losing some of their characteristics.

Ant Colony Optimisation did not show promising results or even better performance on any of the three main test indicators: running time, number of solutions and aircraft cost. Particle Swarm Optimisation presented a median performance on those same results. Performing better than Hill Climbing regarding the algorithm running time indicator, but worse than Simulated Annealing. Regarding the aircraft cost indicator, Particle Swarm Optimisation performed better than Simulated Annealing but worse than Hill Climbing.

The space state (or search space) of this problem considering the data used on the experiments is also relatively low. Therefore, a local search algorithm (particularly Hill Climbing) will most likely get access to good solutions and probabilistic meta-heuristic algorithms (as Simulated Annealing) will soon find their final solution, due to the low offer on new and better solution. However, this behaviour will be quite different if the search space gets expanded. Hill Climbing will often get stuck in a local maximum not being able to find any other solutions and Simulated Annealing will consume more time to explore the search space considering that better solutions will, eventually, be found. In this case, Particle Swarm Optimisation will become a better candidate to be used as the main method to get better solutions. It will not get stuck in a local maximum and the way it explores the search space (by means of multiple particles) will be an advantage.

5.2 Future Work

Although the results provided by both implemented EAs have demonstrated the effectiveness as well as the adaptability, these can still be further developed in certain ways.

For both, stop conditions may be optimised, and a wider study should be made in order to study the behaviour and influence of changes in the global parameters, which may lead to better performance on future tests.

Particularly for Ant Colony Optimisation, and regarding the structure for the "colony", changes should be thought, since it represents a significant part of the algorithm running time. Currently the "colony" is built on its whole at start, it has one starting and one ending vertex, between both the graph expands by providing alternatives to each of the previous vertexes. This provides a better understanding of the sequence of solutions, but its quite time consuming. A simpler structure might contain several ending vertexes instead of just one, which would lead to fewer edges and consequently less running time. Or by not building the "colony" entirely at start.

Appendix A

Aircraft Specialist Statistics

In this appendix all data gathered from tests is available in the following tables which are sorted by the active method. It should be noted that every value under columns *Time* and *Solution* is already the average value from the first five negotiating rounds of each flight.

Table A.1: Hill Climbing

Execution Times, Solution Numbers and Aircraft Solution per Flight Number																				
928				1917			864			1614			839							
Time	Solutions	A. Cost	A. Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility
755	111	3163	0,916	638	81	3151	0,855	1083	132	3125	0,929	80	13	6612	0,851	43	6	9123	0,856				
1282	111	3162	0,916	582	81	3151	0,835	920	132	3125	0,944	84	12	6612	0,808	40	6	9130	0,857				
856	111	3162	0,907	576	81	3151	0,881	995	132	3121	0,905	77	12	6612	0,865	39	6	9123	0,867				
741	111	3157	0,884	580	81	3151	0,863	853	132	3121	0,894	108	12	6612	0,824	34	6	9123	0,873				
686	111	3162	0,9	475	81	3151	0,803	807	132	3123	0,898	80	13	6612	0,818	35	6	9128	0,848				
840	111	3162	0,882	607	81	3151	0,849	1092	149	3121	0,896	72	12	6612	0,864	40	6	9130	0,874				
714	111	3157	0,908	474	81	3151	0,895	1063	140	3121	0,895	106	13	6612	0,802	44	6	9123	0,835				
790	111	3161	0,907	569	81	3151	0,834	790	132	3121	0,926	116	13	6612	0,878	43	6	9130	0,85				
799	111	3163	0,942	506	81	3151	0,825	826	132	3125	0,898	73	12	6612	0,874	33	6	9129	0,88				
701	111	3161	0,923	626	81	3151	0,876	792	132	3121	0,88	81	13	6612	0,864	43	6	9123	0,862				
793	111	3157	0,902	675	81	3151	0,868	921	132	3121	0,877	82	12	6612	0,881	36	6	9123	0,837				
747	111	3157	0,911	467	81	3151	0,896	781	132	3124	0,929	86	13	6612	0,834	37	6	9130	0,877				
759	111	3162	0,903	469	81	3151	0,884	821	132	3121	0,922	110	13	6612	0,87	37	6	9130	0,874				
702	111	3162	0,923	558	81	3151	0,864	820	132	3125	0,958	85	13	6612	0,854	44	6	9130	0,892				
727	111	3162	0,934	455	81	3151	0,868	777	132	3124	0,937	92	13	6612	0,837	39	6	9129	0,869				
688	111	3163	0,918	453	81	3151	0,872	812	132	3121	0,915	85	13	6612	0,825	39	6	9128	0,876				
711	111	3157	0,915	481	81	3151	0,861	773	132	3124	0,923	82	13	6612	0,857	35	6	9130	0,892				
708	111	3157	0,921	501	81	3151	0,859	804	132	3124	0,935	88	12	6612	0,826	34	6	9130	0,864				
743	111	3160	0,915	485	81	3151	0,849	789	132	3121	0,88	94	12	6612	0,859	36	6	9130	0,842				
674	111	3157	0,899	465	81	3151	0,874	784	132	3125	0,923	76	12	6612	0,825	39	6	9123	0,892				

Table A.2: Simulated Annealing

Execution Times, Solution Numbers and Aircraft Solution per Flight Number																				
928				1917				864				1614				839				
Time	Solutions	A. Cost	A. Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility
83	10	3478	0,887	65	10	3512	0,853	71	10	3612	0,914	54	10	6612	0,868	69	10	9130	0,878
82	10	3657	0,867	69	10	3514	0,811	75	10	4575	0,923	59	10	6612	0,832	69	10	9130	0,816
68	10	6044	0,901	72	10	3478	0,861	74	10	3184	0,89	71	10	6612	0,853	74	10	9130	0,841
79	10	3484	0,901	65	10	3480	0,849	74	10	3611	0,933	72	11	6612	0,813	68	10	9123	0,877
77	10	3509	0,922	65	10	3645	0,872	74	10	3705	0,907	62	10	6612	0,855	64	10	9130	0,877
81	10	3488	0,922	87	10	3482	0,832	76	10	7407	0,918	89	10	11249	0,843	79	10	9123	0,897
78	10	3197	0,913	70	10	3553	0,87	82	10	3121	0,956	68	10	11249	0,796	59	10	9130	0,875
81	10	3514	0,906	72	10	3511	0,875	87	10	3747	0,94	112	10	6612	0,864	78	10	9123	0,859
78	10	3499	0,9	68	10	3650	0,877	80	10	3762	0,935	65	10	6612	0,855	68	10	9130	0,88
66	10	3497	0,912	62	10	3523	0,857	71	10	7565	0,886	76	10	11250	0,83	77	10	9127	0,845
82	10	3197	0,897	66	10	3510	0,877	77	10	3608	0,862	74	10	6612	0,812	73	10	9129	0,896
98	10	3743	0,941	68	10	3570	0,801	74	10	3678	0,932	73	10	6612	0,842	58	10	9123	0,882
83	10	3493	0,922	88	10	3482	0,867	73	10	3662	0,894	76	10	11249	0,843	75	10	9123	0,899
67	10	3161	0,926	65	10	3479	0,842	82	11	3193	0,92	72	10	6612	0,813	72	10	9130	0,856
67	10	3509	0,933	66	10	3523	0,871	80	10	6250	0,901	59	10	11249	0,846	55	10	9129	0,846
70	10	3507	0,9	68	10	5902	0,804	72	10	3758	0,936	75	10	11249	0,831	74	10	9128	0,862
82	10	3746	0,917	65	10	3617	0,86	78	10	3179	0,915	77	11	6612	0,834	73	10	9130	0,878
75	10	5872	0,932	69	10	3519	0,817	76	10	3125	0,925	73	10	11249	0,855	61	10	9123	0,877
79	10	3496	0,884	77	10	3512	0,855	78	10	3184	0,9	75	11	6612	0,861	78	10	9130	0,853
79	10	3484	0,864	68	10	3553	0,805	79	10	3125	0,901	74	10	6612	0,844	73	10	9123	0,88

Table A.3: Particle Swarm Optimisation

Execution Times, Solution Numbers and Aircraft Solution per Flight Number																			
928				1917				864				1614				839			
Time	Solutions	A. Cost	A. Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility
169	7	3179	0,916	116	6	3478	0,84	117	7	3667	0,914	79	3	11249	0,833	53	3	9123	0,894
185	8	3493	0,901	155	7	3512	0,833	149	7	3613	0,905	88	4	6612	0,813	50	2	9130	0,869
173	8	3742	0,905	121	6	3511	0,86	78	9	3608	0,91	76	3	6612	0,86	47	2	9130	0,875
157	7	3496	0,901	161	6	3482	0,874	168	9	3126	0,871	90	5	6612	0,869	60	3	9123	0,815
189	9	3184	0,924	172	8	3534	0,861	181	8	4634	0,877	87	5	6612	0,869	60	4	9123	0,874
160	7	3512	0,886	158	7	3657	0,855	183	7	3682	0,952	89	5	11249	0,818	59	4	9123	0,847
148	6	3732	0,902	171	8	3518	0,841	147	7	8697	0,885	88	4	6612	0,845	66	4	9123	0,835
172	8	3500	0,937	177	9	3519	0,831	152	8	3708	0,915	89	4	6612	0,845	58	4	9123	0,855
181	8	3746	0,945	158	7	3508	0,796	183	10	3705	0,901	92	5	6612	0,867	61	3	9129	0,853
189	9	3163	0,911	155	6	3519	0,819	183	8	3625	0,929	89	5	6612	0,809	61	4	9123	0,89
169	8	3496	0,898	116	5	3511	0,843	152	7	3121	0,908	90	4	11250	0,859	61	3	9130	0,863
158	8	3489	0,91	167	8	3518	0,839	183	7	3702	0,865	91	5	6612	0,864	58	4	9130	0,871
157	7	3492	0,898	163	7	3511	0,861	186	8	3608	0,897	95	6	6612	0,813	71	5	9130	0,85
160	8	3494	0,893	173	8	3512	0,853	151	7	3126	0,862	89	4	6612	0,836	67	4	9123	0,842
189	9	3514	0,929	173	8	3482	0,863	180	8	3184	0,901	92	5	6612	0,886	70	5	9130	0,875
159	7	3185	0,884	167	8	3523	0,839	168	7	3121	0,942	84	4	6612	0,794	56	3	9123	0,861
179	8	3165	0,925	116	6	3511	0,831	182	7	3611	0,927	94	5	6612	0,809	65	4	9127	0,877
157	7	3179	0,933	171	8	3563	0,856	182	9	3672	0,939	94	5	11249	0,825	58	3	9130	0,886
169	8	3500	0,903	170	7	7087	0,822	148	10	3193	0,924	86	4	11249	0,818	53	3	9123	0,898
158	7	3493	0,961	74	4	3478	0,869	78	7	3168	0,912	72	3	6612	0,859	60	4	9123	0,86

Table A.4: Ant Colony Optimisation

Execution Times, Solution Numbers and Aircraft Solution per Flight Number																				
928				1917				864				1614				839				
Time	Solutions	A. Cost	A. Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility	Time	Solutions	A. Cost	Utility
1548	4	3184	0,94	839	4	3514	0,824	1764	5	3651	0,924	207	2	6615	0,773	329	4	9130	0,898
1734	4	3506	0,924	906	4	3514	0,789	1930	5	3747	0,93	205	2	11250	0,807	287	4	9130	0,887
1598	4	3184	0,916	854	4	3514	0,879	1826	5	12943	0,877	222	3	11250	0,846	264	4	9130	0,845
1585	5	3506	0,928	845	4	3617	0,841	2238	5	3189	0,854	260	3	11250	0,761	326	4	9130	0,865
1469	5	3184	0,89	869	4	3514	0,858	1765	5	3189	0,917	216	3	6615	0,818	285	4	9130	0,825
1908	5	3184	0,925	1105	4	3488	0,887	2106	4	3662	0,93	277	3	11250	0,836	334	4	9130	0,909
1453	4	4447	0,866	850	4	7741	0,813	1854	4	3189	0,902	259	2	11250	0,859	399	4	9130	0,884
1290	4	4447	0,914	837	4	3520	0,832	1709	5	3747	0,919	230	3	6615	0,824	368	4	9130	0,888
1614	4	3184	0,935	909	4	3514	0,846	1833	4	4661	0,884	259	2	11250	0,787	333	4	9130	0,847
1654	5	3645	0,916	882	4	3617	0,805	1751	5	3662	0,919	251	4	11250	0,82	324	4	9130	0,873
1709	4	3645	0,916	883	4	3617	0,805	1960	4	3662	0,919	200	2	11250	0,82	329	4	9130	0,873
1489	4	3184	0,927	930	4	3520	0,898	1996	4	3662	0,924	227	3	11250	0,863	322	4	9130	0,894
1740	4	3506	0,902	877	4	3617	0,858	2102	5	3662	0,913	239	3	6615	0,804	371	4	9130	0,865
1586	5	3740	0,903	888	4	4447	0,821	1872	5	3747	0,827	217	3	11250	0,834	343	4	9130	0,881
1553	4	3506	0,861	894	4	3514	0,861	1813	5	3747	0,931	206	3	11250	0,863	280	4	9130	0,858
1463	5	4505	0,875	840	4	3514	0,861	1807	4	3189	0,923	209	3	11250	0,822	357	4	9130	0,866
1408	4	3645	0,904	895	4	3514	0,851	1903	4	3662	0,917	201	3	6615	0,836	361	4	9130	0,884
1423	4	3184	0,919	834	4	3520	0,855	1810	4	3189	0,877	185	3	6615	0,903	301	4	9130	0,84
1382	4	3184	0,898	925	4	3514	0,796	1769	5	3189	0,89	231	3	6615	0,831	299	4	9130	0,891
1502	4	3645	0,871	825	4	3520	0,859	1851	4	3747	0,915	239	3	11250	0,792	333	4	9130	0,873

References

[aco15] Ant colony system - clever algorithms. http://www.cleveralgorithms.com/nature-inspired/swarm/ant_colony_system.html, 2015. Accessed: 2015-02-10.

[AENA04] Ahmed Abdelghany, Goutham Ekollu, Ram Narasimhan, and Khaled Abdelghany. A proactive crew recovery decision support tool for commercial airlines during irregular operations. *Annals of Operations Research*, 127(1-4):309—310, 2004.

[AFS15] Airline performance reports. http://www.flightstats.com/go/Stats/airlinePerformanceReports.do, 2015. Accessed: 2015-02-06.

[Blu05] Christian Blum. Ant colony optimization: Introduction and recent trends. *Physics of Life reviews*, 2(4):353–373, 2005.

[CCZ10] Xindu Chen, Xin Chen, and Xinhui Zhang. Crew scheduling models in airline disruption management. In *Industrial Engineering and Engineering Management (IE&EM), 2010 IEEE 17Th International Conference on*, pages 1032–1037. IEEE, 2010.

[CDF15] Flightstats global cancellation and delays. http://www.flightstats.com/go/Media/stats.do, 2015. Accessed: 2015-02-06.

[Che09] Po-Hung Chen. *Particle swarm optimization*, chapter Particle swarm optimization for power dispatch with pumped hydro. INTECH Open Access Publisher, 978-953-7619-48-0, 2009.

[CRO13] Antonio JM Castro, Ana Paula Rocha, and Eugenio Oliveira. *A New Approach for Disruption Management in Airline Operations Control*, volume 562 of *Studies in Computational Intelligence*. Springer, Heidelberg, 1 edition, 2013.

[CTA04] Andrew J Cook, Graham Tanner, and Stephen Anderson. Evaluating the true cost to airlines of one minute of airborne or ground delay: final report. *URL: http://www.eurocontrol. int*, 2004.

[DB05] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2):243–278, 2005.

[dea15] Metaheuristics network. http://www.metaheuristics.net/index.php?main=3, 2015. Accessed: 2015-02-09.

[Deb01] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

REFERENCES

[DS10] Marco Dorigo and Thomas Stützle. Ant colony optimization: overview and recent advances. In *Handbook of metaheuristics*, pages 227–263. Springer, 2010.

[ead15] Evolutionary algorithm | define evolutionary algorithm at dictionary.com. http://dictionary.reference.com/browse/evolutionary+algorithm, 2015. Accessed: 2015-02-08.

[ES01] Russell C Eberhart and Yuhui Shi. Particle swarm optimization: developments, applications and resources. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 81–86. IEEE, 2001.

[Fog06] David B Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*, volume 1. John Wiley & Sons, 2006.

[gai15] Introduction to genetic algorithms. http://lancet.mit.edu/mbwall/presentations/IntroToGAs/, 2015. Accessed: 2015-02-10.

[gap15] Genetic algorithms - handout. http://www.cs.ucc.ie/~dgb/courses/tai/notes/handout12.pdf, 2015. Accessed: 2015-02-10.

[Irr96] M. E. Irrang. Crew scheduling models in airline disruption management. In *Airline Irregular Operations*, pages 349–365, 1996.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[KLL+07] Niklas Kohl, Allan Larsen, Jesper Larsen, Alex Ross, and Sergey Tiourine. Airline disruption management—perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162, 2007.

[LSLC05] Michael Løve, Kim Riis Sørensen, Jesper Larsen, and Jens Clausen. Using heuristics to solve the dedicated aircraft recovery problem. *Central European Journal of Operations Research*, 13(2):189, 2005.

[MTK96] Kim-Fung Man, Kit-Sang Tang, and Sam Kwong. Genetic algorithms: concepts and applications. *IEEE Transactions on Industrial Electronics*, 43(5):519–534, 1996.

[PKB07] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[pso15] Introduction to particle swarm optimization. http://mnemstudio.org/particle-swarm-introduction.htm, 2015. Accessed: 2015-02-09.

[Ree95] Colin R Reeves. A genetic algorithm for flowshop sequencing. *Computers & operations research*, 22(1):5–13, 1995.

[RJN03] Jay M Rosenberger, Ellis L Johnson, and George L Nemhauser. Rerouting aircraft for airline recovery. *Transportation Science*, 37(4):408–421, 2003.

[YQ04] Gang Yu and Xiangtong Qi. *Disruption management: framework, models and applications*. World Scientific Publishing Company Incorporated, 2004.