
UNIVERSITY OF MACEDONIA

POSTGRADUATE PROGRAM

DEPARTMENT OF APPLIED INFORMATICS

# A MATHEURISTIC APPROACH FOR SOLVING THE AIRCRAFT LANDING SCHEDULING PROBLEM

Master Thesis

of

Anastasia Kyriakidou

Thessaloniki, July 2020

# A MATHEURISTIC APPROACH FOR SOLVING THE AIRCRAFT LANDING SCHEDULING PROBLEM

Anastasia Kyriakidou

B.Sc. in Mathematics, Department of Mathematics,

Aristotle University of Thessaloniki, 2017

Master Thesis

submitted as partial fulfillment of the requirements for the

DEGREE OF MASTER OF SCIENCE IN APPLIED INFORMATICS

Supervisor:
Angelos Sifaleras

Approved by examining board on dd/mm/yyyy

| Angelos Sifaleras | Nikolaos Samaras | Dimitrios Hristu - Varsakelis |
|---|---|---|
| ..... | ..... | ..... |

Anastasia Kyriakidou

.....

## Abstract

Aircraft landing scheduling (ALS) is a contemporary NP-hard problem, arising from the continuous growth of the air traffic. The purpose of this thesis is the implementation and application of a matheuristic approach, using VNS algorithm's variants, BVNS, VND and RVNS, along with interior-point method from mathematical programming, as well as the evaluation of solving the static case of this problem, using the Python programming language.

In the first part, the required theoretical background is presented. The description, information and details of the problem, followed by the reference to the VNS, its variants and interior-point method, including the structure and the steps of the algorithms.

In the second part, after a first come first served-based construction method and the application of each algorithm on benchmark problems available on OR Library, the computational results are presented. Finally, the conclusions are mentioned, derived from comparing the results of the above implementation and the optimal values of the problems as well as a statistical analysis of these results.

**Keywords: Aircraft Landing Scheduling, ALS, Matheuristics, Heuristics, Metaheuristics, Optimization, Variable Neighborhood Search, VNS**

## Acknowledgements

I would like to thank my supervisor, Associate Professor Angelos Sifaleras, for the trust, for accepting to supervise my master thesis and for the willingness to provide his knowledge.

I would also like to express my gratitude to my family, my friends and my close ones for their unconditional support and patience during my studies and the implementation of this thesis.

# Contents


1 Introduction 1
- 1.1 Description and significance of the subject . . . 1
- 1.2 Aim and objectives . . . 1
- 1.3 Contribution . . . 2
- 1.4 Basic terminology . . . 3
- 1.5 Outline . . . 5

2 Aircraft Landing Scheduling 7
- 2.1 Introduction . . . 7
- 2.2 ALS problem . . . 11
- 2.3 Mathematical formulation and computational complexity . . . 17
- 2.4 ALS variants . . . 21

3 Heuristics, Metaheuristics & Matheuristics 22
- 3.1 Introduction . . . 22
- 3.2 Metaheuristics classification . . . 24
- 3.3 Historical references . . . 26
- 3.4 Interior-Point Methods . . . 29
- 3.5 Local Search . . . 32
- 3.6 VNS algorithm . . . 33
  - 3.6.1 Introduction . . . 33
  - 3.6.2 VNS description . . . 34
  - 3.6.3 VNS variants . . . 35

4 Literature Review 48

5 Methodology 51
- 5.1 Matheuristic approach for the ALS problem . . . 51
- 5.2 VNS variants implementation for the ALS problem . . . 55
- 5.3 Computational study . . . 57
  - 5.3.1 Benchmarks of OR Library . . . 57
  - 5.3.2 Computational experiments and results . . . 62
    - 5.3.2.1 Matheuristics results . . . 62
    - 5.3.2.2 Gurobi Optimizer results . . . 64
  - 5.3.3 Statistical analysis of the metaheuristic methods . . . 66

6 Conclusions 72
- 6.1 Thesis overview . . . 72
- 6.2 Future work . . . 72

# List of Figures


1 "The busy summer skies" . . . . . 7
2 Delay cause by year, as a percent of total delay minutes . . . . . 8
3 Athens International Airport-Annual flights 2014-2019 . . . . . 9
4 Annual passengers 2003-2018 on all flights to and from the United States 10
5 Evolution of total passenger throughput (in millions) at the 10 largest airports in Asia: 2008-2015 . . . . . 11
6 Variation in cost for a plane within its time window . . . . . 13
7 Area affected by aerodynamic turbulence created by an aircraft . . . . . 14
8 ICAO distance-based wake turbulence separation minima . . . . . 15
9 Categorisation process and criteria for assigning an existing aircraft type into RECAT-EU scheme (*MTOW: Maximum Takeoff Weight*) . . . . . 16
10 RECAT-EU WT distance-based separation minima on approach and departure . . . . . 16
11 RECAT-EU WT time-based separation minima on departure . . . . . 17
12 An example of overlapping time windows . . . . . 20
13 Metaheuristics classification . . . . . 26
14 Graphical representation of the interior-point method . . . . . 31
15 Example of FCFS method . . . . . 52
16 Example of landing times occurring after interior-point method application 52
17 Solution representation . . . . . 53
18 Structure of a solution after a relocate move . . . . . 54
19 Structure of a solution after a 2-opt move . . . . . 54
20 Structure of a solution after a swap move . . . . . 55
21 The results of FCFS, BVNS, VND, RVNS and Gurobi . . . . . 66
22 Friedman test (1) . . . . . 67
23 Descriptive statistics and graphics for BVNS . . . . . 68
24 Descriptive statistics and graphics for VND . . . . . 69
25 Descriptive statistics and graphics for RVNS . . . . . 69

26 Friedman test (2) . . . . . 70

# List of Tables


1 ALS files in OR Library . . . . . 57

2 Computational results of small test datasets . . . . . 63

3 Computational results of large test datasets . . . . . 64

4 Computational results of Gurobi Optimizer . . . . . 65

5 First test set . . . . . 67

6 Wilcoxon test (1) . . . . . 68

7 Second test set . . . . . 70

8 Wilcoxon test (2) . . . . . 71

# 1 Introduction

## 1.1 Description and significance of the subject

Operational Research is a scientific approach to decision making and problem solving, achieved through mathematical models, analytical and stochastic methods. Optimization constitutes a significant part of Operational Research.

The subject of this thesis is a matheuristic approach, with VNS algorithm's variants and the interior-point method, for solving the static case of the Aircraft landing scheduling (ALS) problem. The continuous and rapid increase of civil air traffic in a global level and the limited capability of expanding airport infrastructures has led to the need of developing advanced ways for scheduling the runway operations, which should mainly be able to find a feasible sequence of these operations as well as to minimize their cost.

The usage only of exact algorithms, aiming to find the best solution for similar problems, is usually quite costly in computational complexity and sometimes inefficient in producing the desired results. In addition, some cases do not require an optimal solution, which would consume significant computational time that cannot be provided. Nevertheless, we could accept a lower quality but still feasible and efficient solution, close to the optimal, satisfying the constraints and achieved in a short period of time. Heuristic and metaheuristic methods can contribute to provide the aforementioned solutions, that are acceptable in terms of quality and computational time. ALS problem is one of these cases and a matheuristic algorithm can manage to ensure the smooth operation of runways, which is quite demanding and hard to achieve in busy periods of time.

## 1.2 Aim and objectives

The purpose of this thesis is the development of a matheuristic approach, aiming to provide a solution for some indicative instances of the ALS' problem (from OR Library),

as well as the analytical comparison of the applied methods and their results, in order to lead to safe conclusions about the effectiveness of the algorithms.

Specifically, VNS algorithm's variants (BVNS, VND and RVNS) and the interior-point method are implemented in Python programming language, where these procedures are tested. In addition, the mathematical model, developed also in Python utilizing the Gurobi optimization solver, has been used. The results of these computations are presented and compared. Furthermore, R programming language has been used for a statistical analysis regarding these results and the methods applied.

Moreover, the purpose of this thesis includes the following:

- Brief review of the background and the literature related with the subject of this thesis.
- Analytical presentation and description of the ALS problem and the variations of it.
- Analytical presentation and description of the matheuristic approach.
- Suggestions of future research of the topic.

## 1.3 Contribution

This thesis presents in detail the theoretical and technical elements of scheduling aircraft landings. Moreover, a solution to this problem is analyzed. The awareness of this information regarding the specific problem could lead to a better comprehension of how crucial this problem is nowadays that air transportation industry is growing continuously in a rapid rate.

The approach that is followed will be implemented with a matheuristic method, which includes VNS algorithm's variants and the interior-point method from mathematical programming. Finally, we will evaluate the efficiency of this approach, regarding the aforementioned problem.

Hopefully, the end result of this study will be a helpful reference for future research and development around this field.

## 1.4 Basic terminology

At this point, it is required to mention the basic terminology related to the subject of the thesis.

**Mathematical Optimization or Mathematical Programming**: collection of mathematical principles and methods used for solving quantitative problems in many disciplines, including physics, biology, engineering, economics, and business. In mathematics, computer science and economics, an optimization problem is the problem of finding the best solution from all feasible solutions.

**Combinatorial Optimization**: the process of finding one or more best (optimal) solutions in a well defined discrete problem space. Such problems occur in almost all fields of management , as well as in many engineering disciplines.

**Heuristic Method**: a technique designed for solving a problem rapidly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution.

**Metaheuristic Method**: a higher-level procedure designed to find, generate, or select a heuristic method that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity.

**Matheuristics**: optimization algorithms made by the interoperation of metaheuristics and mathematical programming (MP) techniques.

**Approximation Algorithm**: an algorithm that produces in polynomial time a feasible solution whose objective function value is "close" to the optimal.

**Exact Algorithm**: an algorithm that always solves an optimization problem to optimality.

**Search Space**: the set of all possible solutions for an optimization problem.

**Objective Function**: a mathematical expression describing the relationship of the optimization parameters or the result of an operation that uses the optimization parameters as inputs and evaluates the quality of the solution

**Solution Representation**: encoding of the candidate solutions in a suitable form.

**Variable Neighborhood Search**: a metaheuristic method, proposed by Mladenovic & Hansen in 1997, for solving a set of combinatorial optimization and global optimization problems. It explores distant neighborhoods of the current incumbent solution, and moves from there to a new one if and only if an improvement was made.

**NP**: the class of decision problems that are solvable in polynomial time by a nondeterministic Turing machine.

**NP-Complete**: a problem x that is in NP is also in NP-Complete if and only if every other problem in NP can be transformed into x in polynomial time.

**NP-Hard**: the class of decision problems that are at least as hard as the hardest problems in NP.

## 1.5 Outline

In this subsection we briefly present the structure that is followed and the sections of this thesis.

Firstly, the introductory section includes a presentation and a description of the subject in a general context, aiming to a fundamental acquaintance with the concept and the significance of it. Additionally, this part explains the purpose and the conclusions of this study, along with the required basic terminology of the specific sector.

The next two sections cover analytically the theoretical background of the subject, with regard to the details of the problem as well as the details of the applied computational method. The second section describes the problem of scheduling aircraft landings and the crucial information related with it. In addition, we declare the variations of the problem mainly focused on the static case. Furthermore, this part specifies the mathematical model that is used and the computational complexity of the static case of the ALS problem.

In the third section we present the matheuristic method. We refer to the heuristic and metaheuristic procedures for solving optimization problems and present in detail the individual categories. After mentioning some historical facts about metaheuristic methods, the problems in which they can be applied and a reference to the interior-point method and local search, we present the Variable Neighborhood Search (VNS) algorithm. We describe in depth the details of the method along with its variants (Variable Neighborhood Descent, General VNS, Basic VNS, Reduced VNS, Skewed VNS etc), which are applied for the matheuristic solution of the ALS.

The following section includes the review of some of the most significant literature and related research already conducted.

In the next and critical section of this thesis, we describe in detail the development of the algorithms in Python programming language, both the process of implementation and the application process in some instances from the OR Library. Moreover, this chapter includes the evaluation of the effectiveness, a comparative study and statistical tests of the results regarding the aforementioned algorithms, as well as the

results of the Gurobi optimization solver for the mathematical model.

Finally, in the last section of the thesis, we draw the conclusions of the study and some suggestions for further research upon the topic.

The literature and references related to the subject are mentioned in the end of this thesis.

# 2 Aircraft Landing Scheduling

## 2.1 Introduction

The continuous and rapid increase of civil air traffic in a global level, along with the limited capability of expanding airport infrastructures has led to airports and their runways being the "bottleneck" in airport system. One solution would be to expand airport infrastructures and create a suitable space for smooth operations. Nevertheless, most of the times, this is not a realistic one, due to numerous reasons, such as financial, geographical, environmental, spatial planning and, normally, it is prefered to make the best use of the already existing infrastructure through improved scheduling.

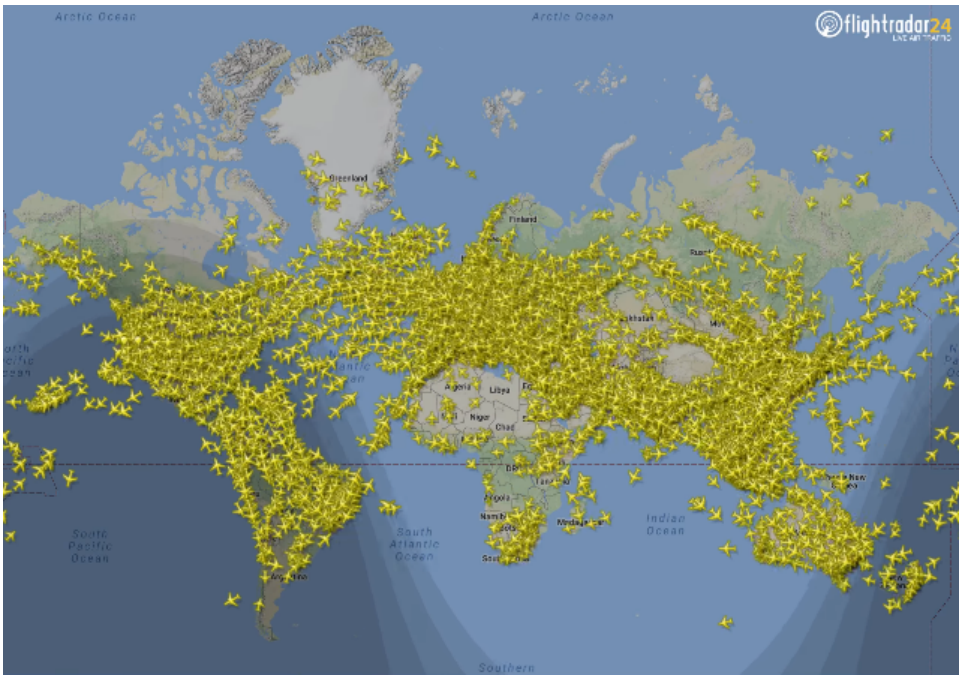


**Figure 1:** "The busy summer skies"
[www.flightradar24.com][51]

According to United States Department of Transportation, the causes of flight delays are classified in the following categories:

1. **Air Carrier:** The cause of the cancellation or delay was due to circumstances within the airline's control (e.g. maintenance or crew problems, aircraft cleaning,

baggage loading, fueling, etc.).

2. **Extreme Weather:** Significant meteorological conditions (actual or forecasted) that, in the judgment of the carrier, delays or prevents the operation of a flight such as tornado, blizzard or hurricane.

3. **National Aviation System (NAS):** Delays and cancellations attributable to the national aviation system that refer to a broad set of conditions, such as non-extreme weather conditions, airport operations, heavy traffic volume, and air traffic control.

4. **Late-arriving aircraft:** A previous flight with same aircraft arrived late, causing the present flight to depart late.

5. **Security:** Delays or cancellations caused by evacuation of a terminal or concourse, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 29 minutes at screening areas.

The following figure presents the percentage of total delay minutes for each of the these categories:

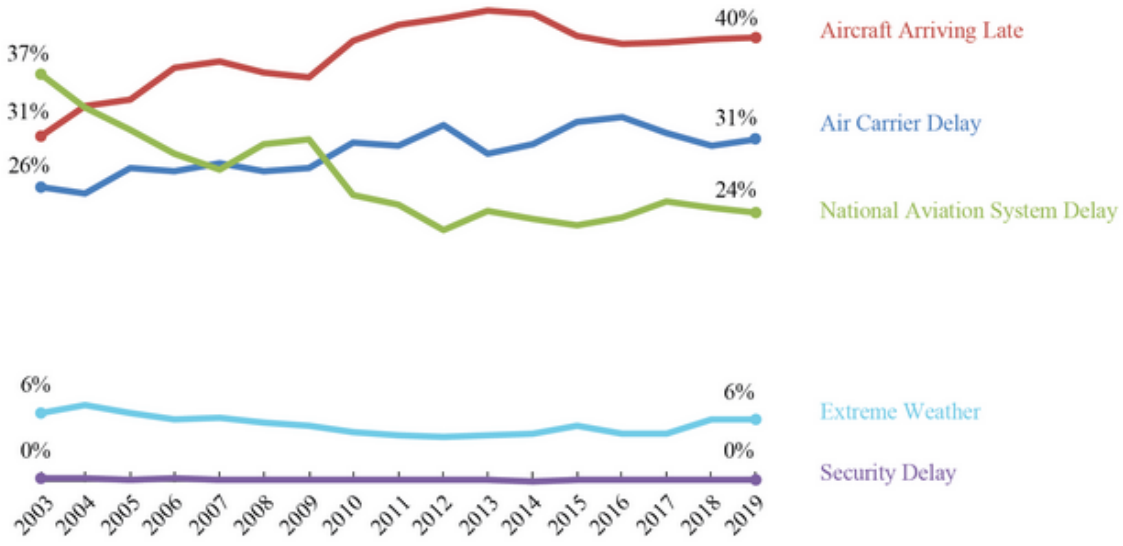


**Figure 2:** Delay cause by year, as a percent of total delay minutes

[www.bts.dot.gov] [48]

As is known, the cost of air transportation is, generally, higher related to other means of transport. The aforementioned bottleneck could affect the flight schedule and cause delays and rescheduling, leading to reduced functionality and an enormous cost. Therefore, the effective utilization of the available capacities is of utmost significance. Considering that the availability of the runways is subject to constraints, consequently arises the problem of aircraft landing scheduling (ALS).

| Year | Domestic flights | International flights | Total flights |
|---|---|---|---|
| 2014 | 67.228 | 87.302 | 154.530 |
| 2015 | 74.740 | 101.416 | 176.156 |
| 2016 | 79.528 | 109.609 | 189.137 |
| 2017 | 81.822 | 114.129 | 195.951 |
| 2018 | 90.530 | 126.564 | 217.094 |
| 2019 | 94.203 | 131.425 | 225.628 |

**Figure 3:** Athens International Airport-Annual flights 2014-2019

[www.aia.gr][46]

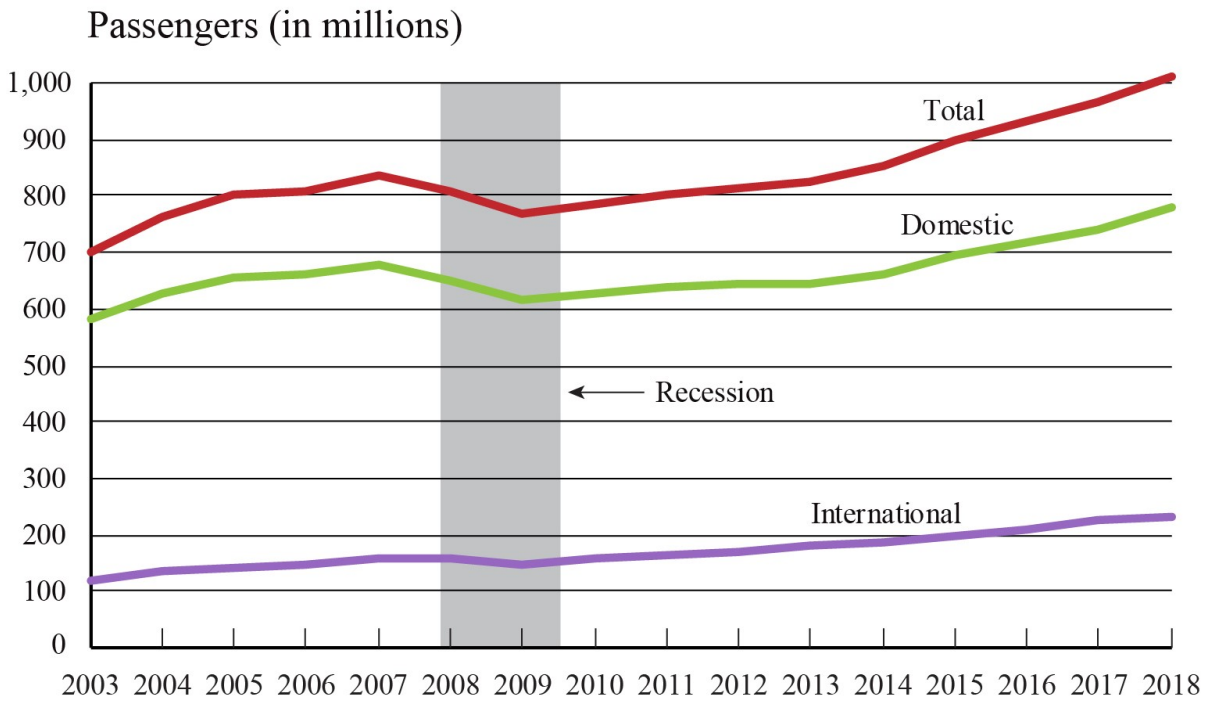


**Figure 4:** Annual passengers 2003-2018 on all flights to and from the United States [www.bts.dot.gov][47]

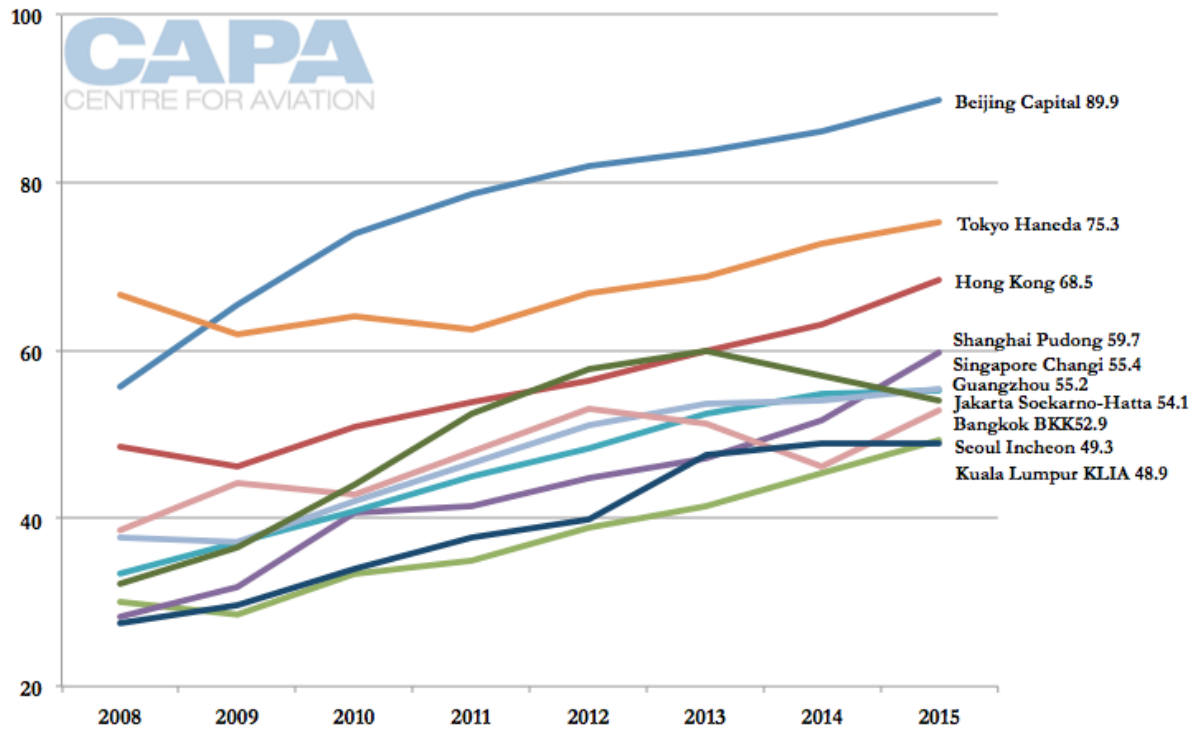


**Figure 5:** Evolution of total passenger throughput (in millions) at the 10 largest airports in Asia: 2008-2015

[centreforaviation.com][43]

The above figures show the general increasing trend in the number of flights and passengers. Regarding the last diagram, it is mentioned that "For some of Asia's 10 largest airports in 2015 100 million annual passengers was a distant target - not in terms of demand but of available infrastructure." [centreforaviation.com][43] An improved and suitable scheduling could, firstly, decrease the cost in time and financial terms and, potentially, lead to the ability of achieving a greater increasement in the number of flights and passengers.

## 2.2 ALS problem

Air Traffic Control (ATC) is responsible for the safety of the operations into and above the runway. As soon as an airplane enters the radar horizon, ATC provide the required directions to the pilot, for the assigned landing time, change of speed,

height or direction. Moreover, if there are more than one runways available, the assigned runway on which to proceed with the landing. "The airport control tower is responsible for ground traffic, and take-off and landing within about 5 nautical miles of the airport and 3000 ft above ground level. The terminal airspace control centre handles departures and arrivals up to 40 nautical miles and 10,000 ft from the airport. Finally, the en-route control centre deals with traffic outside the terminal manoeuvring area." [Bennell et al. (2017)][8] A commonly used strategy for scheduling aircraft landings is "first come first serve" (FCFS) according to the time each aircraft is detected by the ATC radar. Although this method is straightforwardly applied, it could cause unnecessary delays and sometimes be not effective in busy situations.

The landing time of each aircraft is (strictly) bounded between a time window which depends mainly on the specifications and capabilities of the aircraft. The time that an aircraft can perform the landing directly after approaching the airport, while flying at the maximum allowed speed, is the earliest landing time (lower bound). The latest landing time (upper bound) denotes the time that an aircraft can land after flying at its most fuel-efficient airspeed and circling (holding) for the maximum possible time. [Beasley et al. (2000)][5] This limitation is resulting from the finite available fuel that remains to every aircraft while arriving at the destination airport. Admittedly, these time windows constitute hard constraints and a critical factor of scheduling aircraft landings. Ideally, earliest and latest landing time should not be overly close because this could render the problem infeasible.

Moreover, these time windows include the target landing time, which is the preferred landing time produced when the aircraft flies at the most economical speed (cruise speed), and lands without delay. As shown in the next figure, any deviation from the target landing time, either the aircraft is assigned to land earlier or later, will incur an additional cost, which will grow as this deviation grows.

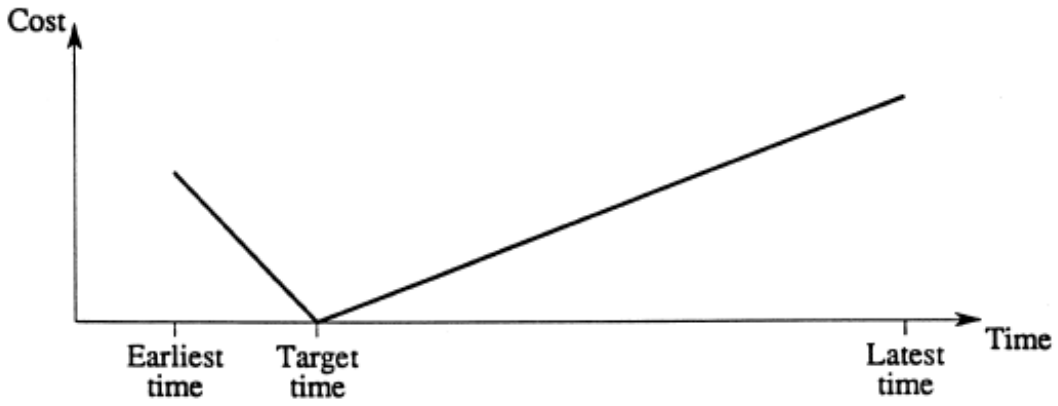


**Figure 6:** Variation in cost for a plane within its time window

[Beasley et al. (2000)][5]

Another critical restriction that has to be taken into consideration while scheduling runway operations is the separation time. The time interval between assigned landing times of a leading and a following aircraft must be greater than a specified minimum to operate safely. "The amount of separation required is determined by collision avoidance, wake turbulence, and other issues. Wake turbulence is generated when an aircraft generates lift. In general, heavier aircraft produce stronger wake turbulence and lighter aircraft are more vulnerable. If following aircraft are not sufficiently separated from the wake of the preceding aircraft, the turbulence could be sufficiently strong enough to result in violent aircraft accelerations. This is one reason why, for take-off and landing, all passengers and crew are seated and belted. In the worst case, wake turbulence could cause the encountering aircraft to lose control and crash. Wake separation standards and their associated procedures are designed to minimize the likelihood of this occurring." [www.aerodefensetech.com][45] The figure below describes the wake turbulence generated by an aircraft.

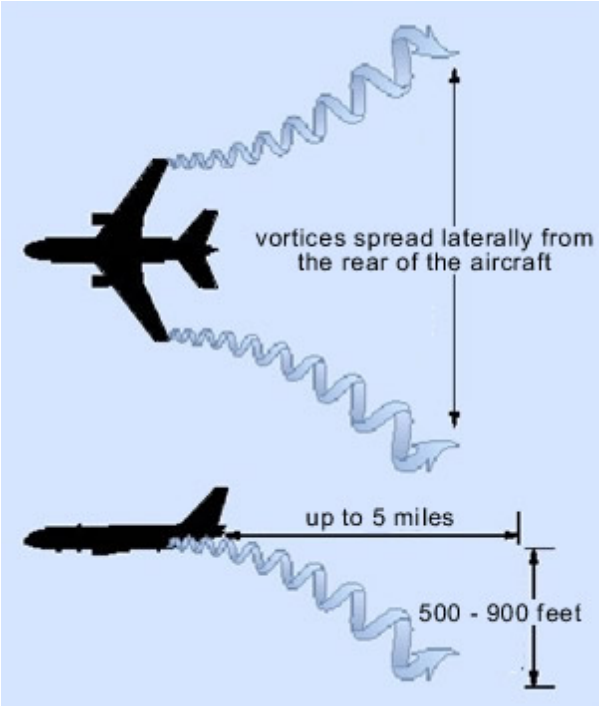


**Figure 7:** Area affected by aerodynamic turbulence created by an aircraft [aviationknowledge.wikidot.com][42]

Therefore, landing a heavy aircraft requires a wider time interval before another aircraft can land. On the other side, the separation time between a light aircraft and the one that follows is (relatively) small as it generates less turbulence. Considering that the separation time issue is evenly significant, several organizations are responsible for the regulation.

The International Civil Aviation Organization (ICAO) is a UN specialized agency, established by States in 1944 to manage the administration and governance of the Convention on International Civil Aviation (Chicago Convention). ICAO works with the Convention's 193 Member States and industry groups to reach consensus on international civil aviation Standards and Recommended Practices (SARPs) and policies in support of a safe, efficient, secure, economically sustainable and environmentally responsible civil aviation sector. [www.icao.int][53] According to ICAO, wake turbulence separation minima shall be based on a grouping of aircraft types into three categories according to the maximum certificated take-off mass as follows:

1. HEAVY (H) - all aircraft types of 136000 kg or more
2. MEDIUM (M) - aircraft types less than 136000 kg but more than 7000 kg
3. LIGHT (L) - aircraft types of 7000 kg or less

The following distance-based wake turbulence separation minima shall be applied to aircraft being provided with an ATS surveillance service in the approach and departure phases of flight.

| *Aircraft category* | | |
|---|---|---|
| *Preceding aircraft* | *Succeeding aircraft* | *Distance-based wake turbulence separation minima* |
| HEAVY | HEAVY | 7.4 km (4.0 NM) |
| | MEDIUM | 9.3 km (5.0 NM) |
| | LIGHT | 11.1 km (6.0 NM) |
| MEDIUM | LIGHT | 9.3 km (5.0 NM) |

**Figure 8:** ICAO distance-based wake turbulence separation minima [www.icao.int][52]

The Federal Aviation Administration (FAA) is the agency of the United States Department of Transportation responsible for the regulation and oversight of civil aviation within the U.S., as well as operation and development of the National Airspace System. Its primary mission is to ensure safety of civil aviation. [www.skybrary.aero][54] As far as Europe is concerned, European Union Aviation Safety Agency (EASA) is the centrepiece of the European Union's strategy for aviation safety. Its mission is to promote the highest common standards of safety and environmental protection in civil aviation. [www.easa.europa.eu][49] The initial standards set by ICAO are defined based on the worst case in each category and this may lead to over separation in many instances. For this reason, organizations like FAA and EASA reconsider them lately, for a better classification of aircrafts and separations (as these in the following figures), which improves the capacity of the airports.

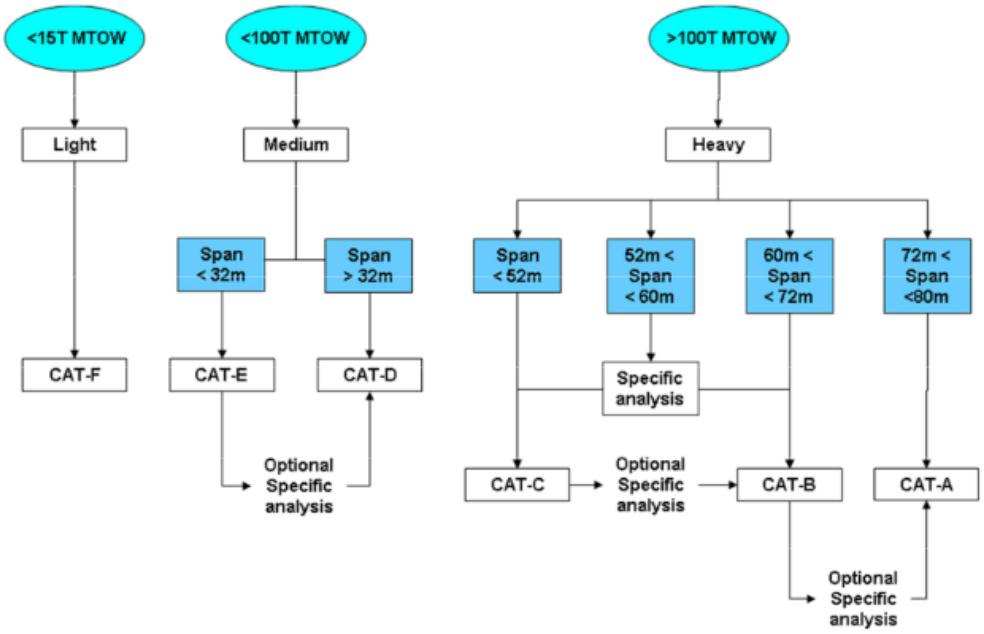


**Figure 9:** Categorisation process and criteria for assigning an existing aircraft type into RECAT-EU scheme (*MTOW: Maximum Takeoff Weight*)

[www.eurocontrol.int][50]

| **RECAT-EU scheme** | | *"Super Heavy"* | *"Upper Heavy"* | *"Lower Heavy"* | *"Upper Medium"* | *"Lower Medium"* | *"Light"* |
|---|---|---|---|---|---|---|---|
| Leader / Follower | | "A" | "B" | "C" | "D" | "E" | "F" |
| *"Super Heavy"* | "A" | **3 NM** | **4 NM** | **5 NM** | **5 NM** | **6 NM** | **8 NM** |
| *"Upper Heavy"* | "B" | | **3 NM** | **4 NM** | **4 NM** | **5 NM** | **7 NM** |
| *"Lower Heavy"* | "C" | | **(*)** | **3 NM** | **3 NM** | **4 NM** | **6 NM** |
| *"Upper Medium"* | "D" | | | | | | **5 NM** |
| *"Lower Medium"* | "E" | | | | | | **4 NM** |
| *"Light"* | "F" | | | | | | **3 NM** |

**Figure 10:** RECAT-EU WT distance-based separation minima on approach and departure

[www.eurocontrol.int][50]

| RECAT-EU scheme<br>Leader / Follower | | "SUPER HEAVY"<br>"A" | "UPPER HEAVY"<br>"B" | "LOWER HEAVY"<br>"C" | "UPPER MEDIUM"<br>"D" | "LOWER MEDIUM"<br>"E" | "LIGHT"<br>"F" |
|---|---|---|---|---|---|---|---|
| "SUPER HEAVY" | "A" | | **100s** | **120s** | **140s** | **160s** | **180s** |
| "UPPER HEAVY" | "B" | | | | **100s** | **120s** | **140s** |
| "LOWER HEAVY" | "C" | | | | **80s** | **100s** | **120s** |
| "UPPER MEDIUM" | "D" | | | | | | **120s** |
| "LOWER MEDIUM" | "E" | | | | | | **100s** |
| "LIGHT" | "F" | | | | | | **80s** |

**Figure 11:** RECAT-EU WT time-based separation minima on departure [www.eurocontrol.int][50]

## 2.3 Mathematical formulation and computational complexity

ALS is doubtlessly a complex and time sensitive problem. For this reason, an analytical model presentation is required, to make more specific the restrictions that must be met. The model is adopted from Beasley et al. [2000][5], for the static case with a single runway. Let the following notation for the parameters:

$P$ = the number of planes

$E_i$ = the earliest landing time for plane $i$ ($i = 1, \ldots, P$)

$L_i$ = the latest landing time for plane $i$ ($i = 1, \ldots, P$)

$T_i$ = the target (preferred) landing time for plane $i$ ($i = 1, \ldots, P$)

$S_{ij}$ = the required separation time ($\geqslant 0$) between plane $i$ landing and plane $j$ landing (where plane $i$ lands before plane $j$), $i = 1, \ldots, P$; $j = 1, \ldots, P$; $i \neq j$

$g_i$ = the penalty cost ($\geqslant 0$) per unit of time for landing before the target time $T_i$ for plane $i$ ($i = 1, \ldots, P$)

$h_i$ = the penalty cost ($\geqslant 0$) per unit of time for landing after the target time $T_i$ for plane $i$ ($i = 1, \ldots, P$)

Therefore, the time window for the landing of plane $i$ is $[E_i, L_i]$, where $E_i \leqslant T_i \leqslant L_i$.

The decision variables are:

$x_i$ = the landing time for plane $i$ ($i = 1, \ldots, P$)

$\alpha_i$ = how soon plane $i$ ($i = 1, \ldots, P$) lands before $T_i$

$\beta_i$ = how soon plane $i$ ($i = 1, \ldots, P$) lands after $T_i$

$$\delta_{ij} = \begin{cases} 1, & \text{if plane } i \text{ lands before plane } j \ (i = 1, \ldots, P;\ j = 1, \ldots, P;\ i \neq j) \\ 0, & \text{otherwise} \end{cases}$$

Without significant loss of generality, we shall henceforth assume that the times $E_i$, $L_i$, and $S_{ij}$ are integers.

The mathematical model is formed as follows:

Objective function:

$$minimize \quad \sum_{i=1}^{P}(g_i\alpha_i + h_i\beta_i) \tag{1}$$

Under the constraints:

$$E_i \leqslant x_i \leqslant L_i, \quad i = 1,2,...,P \tag{2}$$

$$\delta_{ij} + \delta_{ji} = 1, \quad i = 1,2,...,P; j = 1,...,P; j > i \tag{3}$$

$$x_j - x_i \geqslant S_{ij}, \quad i = 1,2,...,P; j = 1,...,P; i \neq j \tag{4}$$

$$x_i = T_i - \alpha_i + \beta_i, \quad i = 1,2,...,P \tag{5}$$

$$\alpha_i \geqslant T_i - x_i, \quad i = 1,2,...,P \tag{6}$$

$$0 \leqslant \alpha_i \leqslant T_i - E_i, \quad i = 1,2,...,P \tag{7}$$

$$\beta_i \geqslant x_i - T_i, \quad i = 1,2,...,P \tag{8}$$

$$0 \leqslant \beta_i \leqslant L_i - E_i, \quad i = 1,2,...,P \tag{9}$$

In more detail, the objective function (1) minimizes the deviation cost of the landing times from the target times. Constraint (2) ensures that each aircraft lands within its time window [$E_i$, $L_i$] and constraint (3) ensures that for each pair ($i$, $j$), either plane $i$ must land before plane $j$ ($\delta_{ij}$ = 1) or plane $j$ must land before plane $i$ ($\delta_{ji}$ = 1). Following, constraint (4) ensures the proper separation, at least $S_{ij}$ time, that must be considered between aircrafts $i$ and $j$. Constraint (5) relates the landing time ($x_i$) of an aircraft $i$ to the time $i$ lands before ($\alpha_i$), or after ($\beta_i$), target ($T_i$). Finally, constraints (6) and (7) ensure that $\alpha_i$ is at least as big as zero and the time difference between $T_i$ and $x_i$, and at most the time difference between $T_i$ and $E_i$. Constraints (8) and (9) are similar constraints for $\beta_i$.

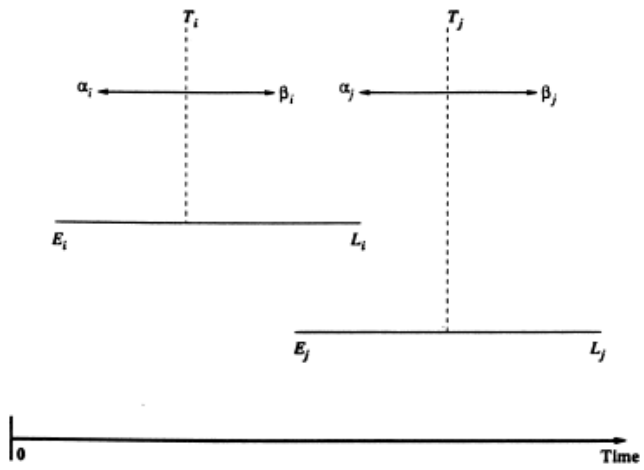


**Figure 12:** An example of overlapping time windows

[Beasley et al. (2000)][5]

If the concepts "aircraft" and "runway" are taken into account as variables in general, the ALS problem can be considered as common with other known scheduling problems. As described by D'Ariano et al. [2012][13], this problem can be viewed as a Job Shop Scheduling (JSS) problem with additional real-world constraints, modelled via alternative graph as a general formulation of the JSS problem, where a job (aircraft/landing operation) must undertake a prescribed sequence of operations on specific machines - with capacity 1 (runways). The target landing time of aircraft $j$ corresponds to the ready time of $r_j$ of job $j$; the landing time of aircraft $j$ corresponds to the starting time $s_j$ of job $j$; the time aircraft $j$ frees the runway corresponds to the completion time $C_j$ of job $j$; the minimum separation time $S_{ji}$ between the landing of aircraft $j$ and the landing of the immediately following aircraft $i$ corresponds to the sequence-dependent processing time $c_{ji}$. Since this formulation is possible, it is straightforward to affiliate the ALS to the category of strongly NP-hard optimization problems, as JSS. Hence, this computational complexity behind the problem restricts the optimal solutions in polynomial time even for medium-sized instances. Additionally, it is highly important to provide a feasible schedule of landings in an acceptable time.

## 2.4 ALS variants

As mentioned in previous sections, this thesis presents the static case of the ALS problem with a single runway. The two main variants of this problem are the static (off-line) and the dynamic (on-line, real-time) case. In the first case, the number of planes, as well as the related times, costs etc and eventually the schedule, are predefined. In the dynamic case, updating rules are constantly applied and it is possible to manage a new plane out of the actual schedule. Both of these variants can be formulated for a single or for multiple runways. Moreover, different objective functions can be applied, regarding the preferred variables, for example time or cost. In addition, it is possible to include airlines' preferences, or increase and decrease the traffic for certain periods of time.

Apart from these, since the model used in this case and in the rest variants can be formulated using time constraints depending on the time windows and is not affected by the operation, it can be applied (or include) also for take-offs. In the case of multiple runways, landings and take-offs are usually separated in different runways (segregated mode), mixed mode is rarely used.

# 3 Heuristics, Metaheuristics & Matheuristics

## 3.1 Introduction

The object of Optimization is to find the best solution from the total of feasible ones for a certain problem. There are several methods suggested to achieve this purpose.

An instance of an optimization problem is a pair $(S, f)$, where $S$ is a finite set of solutions and $f : S \rightarrow R$ is the objective (evaluation) function for the solutions, which is intended to be minimized or maximized, depending on the problem and the requirements. The aim of this process is to find a solution $s' \in S$, so that $f(s') \leqslant f(s)$, $\forall$ $s \in S$, whether it is a minimization problem or $f(s') \geqslant f(s)$, $\forall$ $s \in S$, for a maximization problem, respectively.

Regarding the solution process of optimization problems (especially NP-complete problems), we have some general methods at our disposal, which could be applied in whichever problem and they include the exact algorithms, local search and metaheuristics. However, there are also specific methods available which depend directly and they are designed for the problem they aim to solve. Approximation algorithms and heuristics are included in this method category. Generally, the methods that are commonly applied are exact methods of operational research, of which the most significant examples are the linear programming algorithm Simplex and its variations (Dual Simplex, Big M method etc) as well as interior-point methods.

However, in cases of NP-hard and large-scale optimization problems, for example travelling salesman (TSP), knapsack, scheduling, vehicle routing, bin packing, facility location etc, the conventional methods cannot be, usually, effective enough. The search space is more complex and it is not manageable, most of the times, to find a solution in polynomial time (unless P = NP).

One of the crucial and effective methods that have been applied to achieve an acceptable solution in the aforementioned NP-hard problems, from the field of integer programming, is Branch and Bound technique. The basic principle of this approach is that the total set of feasible solutions can be separated into smaller subsets of

solutions which can afterwards be evaluated systematically until the best (optimal) solution is found.

Related research also includes heuristic methods as an approach to find a sub-optimal solution in short computational time. This kind of methods are normally efficient in local search, finding a quality solution in a limited search space. Meta-heuristic methods can gradually guide the search process using heuristic methods and, although they cannot assure an optimal solution, they generally provide quality results.

According to Blum & Roli [2003, p. 270-271][10], in mathematical optimization and computer science, metaheuristic methods are characterized by the following fundamental properties:

1. Metaheuristics are strategies that "guide" the search process.
2. The goal is to efficiently explore the search space in order to find (near-) optimal solutions.
3. Techniques which constitute meta-heuristic algorithms range from simple local search procedures to complex learning processes.
4. Metaheuristic algorithms are approximate and usually non-deterministic.
5. They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.
6. The basic concepts of metaheuristics permit an abstract level description.
7. Metaheuristics are not problem-specific.
8. Metaheuristics may make use of domain-specific knowledge in the form of heuristics that are controlled by the upper level strategy.
9. Todays more advanced metaheuristics use search experience (embodied in some form of memory) to guide the search.

In short, we could say that metaheuristics are high level strategies for exploring search spaces by using different methods. Of great importance hereby is that adynamic balance is given between diversification and intensification. The term diversification generally refers to the exploration of the search space, whereas the term intensification refers to the exploitation of the accumulated search experience.

According to related literature, the word "metaheuristic" was coined by Fred Glover, Professor of Computer Science Applied Mathematics in the University of Colorado [Yang (2011)][40]. As mentioned before, metaheuristic procedures cannot promise a global optimal solution. They commonly operate some kind of stochastic methods and the resulting solution depends on the random variables that have been applied. However, high quality solutions can be achieved in large-scale problems through the search in a large set of feasible solutions.

Related literature involves plenty of experimental computations and results, regarding the advantages, disadvantages and effectiveness of metaheuristic approaches to a variety of problems.

While the aforementioned methods have been widely used, the matheuristic approach has gained attention lately, combining the power of mathematical programming with the flexibility of metaheuristics. Puchinger & Raidl [2005][34] proposed an early survey and classification on combining metaheuristics and exact algorithms in combinatorial optimization. Matheuristics have demonstrated significant results as solution approach for complex optimization problems (routing, scheduling, bin packing, logistics etc).

## 3.2 Metaheuristics classification

There is a wide variety of metaheuristics and a set of properties according to which they can be classified. The most important are the following:

- Population-based metaheuristics and single-point metaheuristics. This classification is based on the number of solutions that are manipulated by the pro-

cedure at any given time. Examples of population-based metaheuristics are Genetic Algorithms, Ant Colony Optimization, Particle Swarm Optimization etc, which manipulate a set of solutions (population) at each iteration. On the other side, Simulated Annealing, Variable Neighborhood Search etc. are single-point metaheuristics, with one solution at each iteration.

- Memory-based metaheuristics and memory-less metaheuristics. This classification is based on the use of memory (search history) or not by the metaheuristic. Tabu search, Scatter Search and Ant Colony Optimization are examples of algorithms for which memory is a crucial element. Variable Neighborhood Search, Iterated Local Search etc. are memory-less metaheuristics.

- Nature-inspired metaheuristics and non-nature-inspired metaheuristics. This classification is based on the origins of a metaheuristic. Examples of nature-inspired metaheuristics are Genetic Algorithms, Particle Swarm Optimization and Ant Colony Optimization. Variable Neighborhood Search, Tabu Search, GRASP etc. belong to the category of non-nature-inspired metaheuristics.

In the following figure are presented the various categories of metaheuristics.

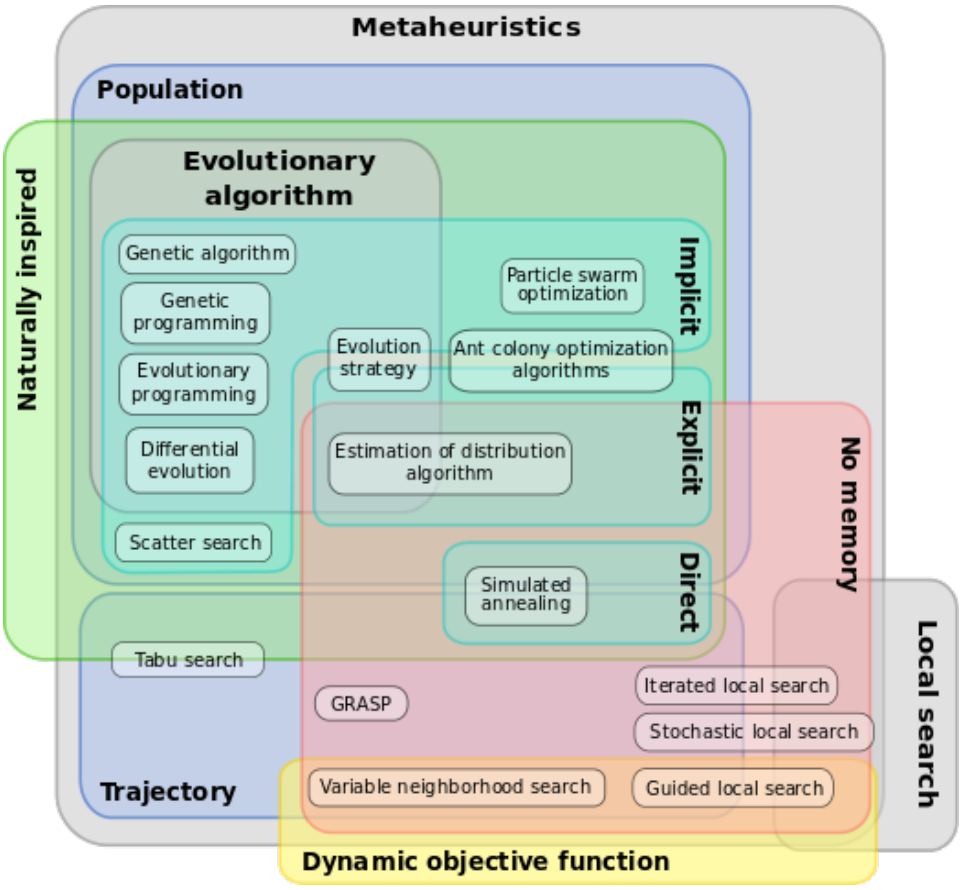


**Figure 13:** Metaheuristics classification

[en.wikipedia.org][44]

## 3.3 Historical references

Apart from the great variety of the existing metaheuristics, new variations are constantly being proposed. A number of the most significant contributions in this field are presented in this section. [en.wikipedia.org][44]

- Stochastic approximation, Herbert Robbins-Sutton Monro, 1951[36]

  As mentioned before, the term "metaheuristc" was referred for the first time by Fred Glover in 1986[17]. Nevertheless, the research in the sector of optimization, that led to results which created a new approach in solving optimization problems through metaheuristic procedures, had started earlier, at least at 1952,

with crucial results, with the research in stochastic approximation by Robbins and Monro.

- Random search, Leonard Andreevich Rastrigin, 1963[35]

  Leonard Andreevich Rastrigin, in 1963, made a premature presentation of random search along with some basic mathematical analysis and proposed it as a method for solving optimization problems.

- Random optimization, J. Matyas, 1965[29]

  In 1965, J. Matyas proposed the method of random optimization. Both random search and random optimization are groups of numerical optimization methods that do not require the slope of the problem to be optimized, and therefore both can be used in functions that are not continuous or differentiable. Such optimization methods are also known as direct search methods, derivative free methods, or black-box methods. Random search works repeatedly in better search positions, which have been sampled by a superspace surrounding the current location, while random optimization works by moving repeatedly to better places in the search space, which have been sampled using, for example, a normal distribution around the current location.

- Evolutionary programming, L. Fogel-A. Owens-M. Walsh, 1966[14]

  In 1966, L. Fogel, A. Owens and M. Walsh, proposed evolutionary programming, one of the four significant examples of evolutionary algorithms. Evolutionary programming is similar to genetic programming, but its structure is stable, while its numerical parameters are allowed to evolve. Fogel first used evolutionary programming in 1960 in the United States in order to use simulated evolution as a learning process aimed at creating artificial intelligence. Fogel used finite state machines as prognostic factors and then developed them. Today, evolutionary programming is a major evolutionary computational dialect without a stable structure, unlike some of the other dialects, and it is becoming increasingly difficult to distinguish it from evolutionary strategies.

- Genetic algorithm, John Henry Holland, 1975[25]

  In 1975, John Henry Holland proposed the Genetic algorithm, a heuristic search that mimics the process of natural selection.

- Scatter search, Fred Glover, 1977[16]

  In 1977, Fred Glover proposed Scatter Search as a technique for combining decision making and constraints to solve integer programming problems. Scatter search systematically combines some “good” solutions (reference points) from previous steps and creates new points by applying additional steps, if it is required, rendering them feasible solutions.

- Genetic programming, Stephen Smith, 1980[39]

  In 1980, we have the description of Genetic programming by Stephen Smith. In artificial intelligence, Genetic programming is a methodology based on an evolutionary algorithm, inspired by biological evolution, which is used to find computer programs that perform a task assigned by the user.

- Simulated annealing, Scott Kirkpatrick-Charles Gelatt-Mario Vecchi, 1983[27]

  In 1983 we have the proposal of Simulated Annealing (SA) by Scott Kirkpatrick, Charles Gelatt and Mario Vecchi. Simulated annealing is a general probabilistic metaheuristic for the total optimization problem of placing a good approach to the total optimal of a given function in a large search space. It is often used when the search space is distinct. For some problems, simulated annealing may be more effective than exhaustive enumeration, under the condition that the purpose is to find an acceptable solution over a certain period of time, rather than finding the optimal solution.

- Tabu search, Fred Glover, 1986[17]

  In 1986, Tabu Search is introduced along with the first mention of the term "metaheuristic" by Fred Glover. Tabu search is a metaheuristic search method that uses local search methods of mathematical optimization. Local searches

(or neighborhood searches) obtain a candidate solution to a problem and explore its neighbors aiming to finding an improved solution. Local search methods tend to "get trapped" to sub-optimal areas or plateaus, where many solutions are equal. Tabu search improves the performance of these techniques by using memory structures that describe the solutions that have been visited or the user-provided set of rules. If a candidate solution has been visited in the past within a certain short period of time or if it has violated a rule, then it is classified as "tabu" (forbidden), so that the algorithm eliminates it.

- Ant Colony optimization, Marco Doringo, 1992[11]

  In 1992, Marco Doringo introduced in his PhD dissertation one of the recent contributions, the Ant Colony Optimization (ACO) method. ACO is a probabilistic technique for solving computational problems, aiming to reduce the intractability of them by discovering "good" paths through graphs.

## 3.4 Interior-Point Methods

Interior-point methods are a certain class of algorithms that solve linear and non-linear convex optimization problems. Linear programming has received significant attention due to the extensive number of applications in science, industry, business, and other fields.

Dantzig proposed the simplex method in 1947, making an initial step for strong research activity in the area of linear programming and optimization in general. The main concept of this algorithm is to "move" from vertex to vertex, along the edge of a feasible region, on which the objective function is decreasing (minimize) or increasing (maximize). Simplex method's efficiency in solving practical problems led to high popularity and years of experiments and applications, resulting gradually in better variants of this algorithm, commonly called pivoting algorithms. Although pivoting algorithms are finite processes, they are not polynomial algorithms. In 1979, Khachiyan

presented the ellipsoid algorithm, the first polynomial algorithm for linear programming. Unfortunately, computational experiments showed that this algorithm, even with various modifications, performs worse than the simplex method on most practical problems. In 1984, Karmakar proposed a new polynomial algorithm for linear programming, quite different than simplex algorithm. It's iterates are calculated in the interior of the feasible region, not on the boundary and makes use of projective transformations and a potential function. Karmarkar's algorithm initiated the field of interior-point methods, high research activity in related areas and a variety of different interior- point methods. It can be shown that Karmarkar's algorithm achieves better iteration complexity than the ellipsoid algorithm.

A generic interior-point algorithm for the linear programming problem is presented below.

Considering a linear programming problem in the standard form:

$$min \quad c^T x \tag{10}$$

$$s.t. \quad Ax = b, \tag{11}$$

$$s \geqslant 0 \tag{12}$$

The interior-point algorithm can be summarized as follows:

1. **Initialization**

   (a) Choose $\beta, \gamma \in (0,1)$ and $(\varepsilon_P, \varepsilon_D, \varepsilon_G) > 0$
   Choose $(x^0, y^0, s^0)$ such that $(x^0, s^0) > 0$ and $||X^0 s^0 - \mu_0 e|| \leqslant \beta \mu_0$
   where $\mu_0 = \frac{(x^0)^T s^0}{n}$

   (b) Set $k = 0$

2. **Step**

(a) Set $r_P^k = b - Ax^k,\ r_D^k = c - A^T y^k - s^k,\ \mu_k = \frac{(x^k)^T s^k}{n}$

(b) Check the termination. If $||r_P^k|| \leqslant \varepsilon_P,\ ||r_D^k|| \leqslant \varepsilon_D,\ (x^k)^T s^k \leqslant \varepsilon_G$, then terminate

(c) Compute the direction by solving the system

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix} = \begin{bmatrix} r_P^k \\ r_D^k \\ -X^k s^k + \gamma \mu_k e \end{bmatrix}$$

(d) Compute the step size

$\alpha_k = max\{\alpha' : ||X(\alpha)s(\alpha) - \mu(\alpha)e|| \leqslant \beta\mu(\alpha),\ \forall\alpha \in [0, \alpha']\}$,

where $x(\alpha) = x^k + \alpha d_x,\ s(\alpha) = s^k + \alpha d_s,\ \mu(\alpha) = \frac{x^T(\alpha)s(\alpha)}{n}$

(e) Update $x^{k+1} = x^k + \alpha_k d_x,\ y^{k+1} = y^k + \alpha_k d_y,\ s^{k+1} = s^k + \alpha_k d_s$

(f) Set $k = k+1$ and go to step 3

The following figure presents the graphical representation of the interior-point method.

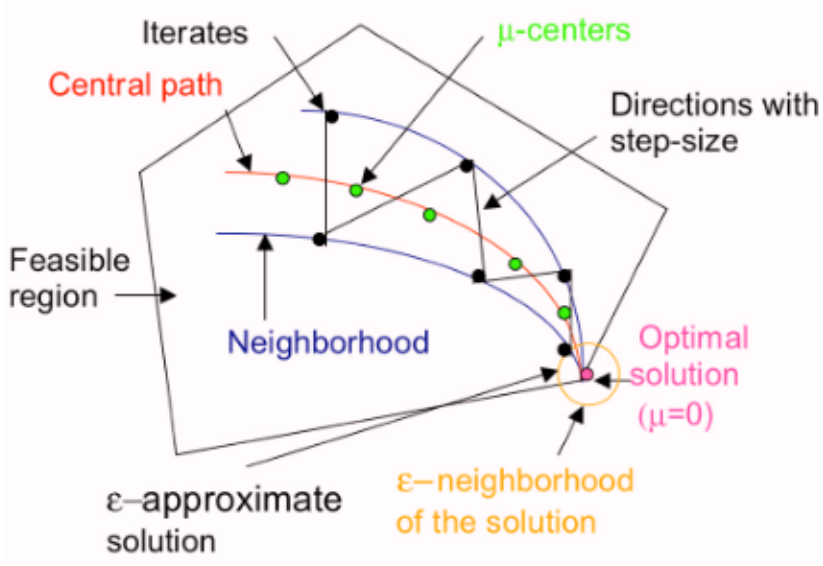


**Figure 14:** Graphical representation of the interior-point method

[Lesaja (2009)][28]

## 3.5 Local Search

Local search is a widely used heuristic method for solving NP-hard optimization problems. The main technique of this method is descending from an initial solution to another within a neighborhood until there in not a descending direction or a stopping condition is met.

The selection of the next solution is based on the information that is derived from the evaluation of each neighbor. Commonly, the direction that is used is best improvement. The next solution is chosen after exploring the whole neighborhood. However, in some cases, especially in large search spaces, this direction can be extremely time-consuming and the first improvement technique is preferred to be used. In this direction, the move to a new solution occurs as soon as the first "better" solution is found.

Both of these directions are presented below:

---

**function** BESTIMPROVEMENT ($x$);

  **repeat**

    $x' \leftarrow x$;

    $x' \leftarrow arg\min_{y \in N(x)} f(y)$

  **until** $(f(x) \geqslant f(x'))$;

**end function**

---

**function** FIRSTIMPROVEMENT ($x$);

**repeat**

$x' \leftarrow x$; $i \leftarrow 0$;

**repeat**

$i \leftarrow i+1$;

$x' \leftarrow arg\min\{f(x), f(x_i)\},\ x_i \in N(x)$

**until** $(f(x) < f(x_i)\ or\ i = |N(x)|)$;

**until** $(f(x) \geqslant f(x'))$;

**end function**

[Hansen et al. (2008)][21]

## 3.6 VNS algorithm

### 3.6.1 Introduction

Variable Neighborhood Search (VNS) is a recent metaheuristic method which was introduced by Pierre Hansen and Nenad Mladenovic in 1997, in Computers and Operations Research journal. This method constitutes a framework which provides the capability of developing heuristic algorithms to solve combinatorial and global optimization problems.

VNS manages an iterated local search procedure, exploring the neighborhoods of the current solution (local best) and moves to a new solution if and only if it provides an improvement. The main concept consists of the systematic neighborhood change, which is achieved firstly through the descending phase, aiming to find a local best and secondly the shaking phase to change neighborhood.

According to Hansen & Mladenovic [2003][20], VNS exploits systematically the following observations:

1. A local minimum with respect to one neighborhood structure is not necessary

so for another.

2. A global minimum is a local minimum with respect to all possible neighborhood structures.

3. For many problems local minima with respect to one or several neighborhoods are relatively close to each other.

   This last observation, which is empirical, implies that a local optimum often provides some information about the global one. This may for instance be several variables with the same value in both. However, it is usually not known which ones are such.

### 3.6.2 VNS description

Let $N_k, k = 1, ..., k_{max}$ denote a finite set of pre-selected neighborhood structures and $N_k(x)$ the set of solutions in the $k^{th}$ neighborhood of $x$. The VNS algorithm includes the following steps:

1. **Initialization**

   (a) Select the set of neighborhood structures $N_k, k = 1, ..., k_{max}$, that will be used in the search

   (b) find an initial solution $x$

   (c) choose a stopping condition

2. **Repeat the following until the stopping condition is met**

   (a) Set $k = 1$

   (b) Until $k = k_{max}$, repeat the following steps:

      i. *Shaking.* Generate a point $x'$ at random from the $k^{th}$ neighborhood of $x$ ($x' \in N_k(x)$)

ii. *Local search.* Apply some local search method with $x'$ as initial solution; denote with $x''$ the so obtained local optimum

iii. *Move or not.* If this local optimum is better than the incumbent, move there ($x = x''$), and continue the search with $N_1$; otherwise, set $k = k + 1$

[Hansen & Mladenovic (1999)][19]

The suggested stopping conditions mentioned by Hansen and Mladenovic are maximum CPU time allowed, maximum number of iterations, or maximum number of iterations between two improvements. Furthermore, in the stochastic part of the above steps, the random generated point $x'$, aims to avoid cycling, which could occur whether any deterministic rule was applied.

In addition, a special mention is included by the authors, for the ease of implementation of both the basic version of VNS (with only one parameter $k_{max}$) and various simple extensions of it. Furthermore, some problem specific questions are declared, that need to be answered when developing VNS for solving each particular problem, where we use more than one neighborhood:

1. what $N_k$ should be used and how many of them?
2. what should be their order in the search?
3. what search strategy should be used in changing neighborhoods?
4. what local search routine will be used?

### 3.6.3 VNS variants

VNS has been proved to constitute an efficient approach for obtaining an approximate solution to a variety of demanding optimization problems. However, it still remains incapable to solve major instances. As mentioned in the related literature, the size of the problems considered is in practice more limited by the tools available

to solve them than by the needs of the potential users of these tools. Hence, improvements appear to be highly desirable. Moreover, when heuristics are applied to very large instances, their strengths and weaknesses become clearly apparent. [Hansen et al. (2008)][21] Therefore, Hansen and Mladenovic presented a number of VNS variants to improve its efficiency. These variants are resulting from the aforementioned (in the introduction of this section) observations, used in the following different ways:

- deterministic
- stochastic
- both deterministic and stochastic

[Hansen et al. (2008)][21]
However, a common function of neighborhood change (which is applied only when an improvement occurs) is implemented to the total of variants, which is described below:

---

**function** NEIGHBORHOODCHANGE $(x, x', k)$;
  **if** $f(x') < f(x)$ **then**
    $x \leftarrow x'$; $k \leftarrow 1$ /*Make a move*/;
  **else**
    $k \leftarrow k+1$ /*Next neighborhood*/;
  **end if**
**end function**

---

[Hansen et al. (2008)][21]

Another significant component in VNS is shaking function, which is the stochastic addition in the procedure, as mentioned previously. The following function presents this component:

**function** SHAKE $(x, k, N)$;

*choose* $x' \in N_k(x)$ *at random*;

**return** $x'$

**end function**

[Hansen et al. (2016)][23]

Variable Neighborhood Descent (VND)

VND is the deterministic variant of VNS. According to Duarte et al. [2016][12], VND explores small neighborhoods until a local optimum is encountered. At that point, the search process switches to a different (typically larger) neighborhood that might allow further progress. This approach is based on the fact that a local optimum is defined with respect to a neighborhood relation, such that if a candidate solution $x$ is locally optimal in a neighborhood $N_i(x)$, it is not necessarily a local optimum for another neighborhood $N_j(x)$. Thus, VND explores the solution space using several neighborhood structures either in a (i) sequential, (ii) a nested (or composite), or (iii) mixed nested way. VND process is described below:

**procedure** VND (*Given* $N_k$, $(k = 1,...,k_{max})$ *and some* $x \in X$)

**repeat**

$k = 1$;

**repeat**

$x' \leftarrow localSearch(x, N_k(x))$;

**if** $f(x') < f(x)$ **then**

$x \leftarrow x'$

**else**

$k \leftarrow k + 1$;

**end if**

**until** $k = k_{max}$;

**until** $(some\ stop\ condition\ is\ satisfied)$

**end procedure**

[Gomes et al. (2004)][18]

Reduced VNS (RVNS)

RVNS implements the stochastic neighborhood change by selecting a point $x'$ randomly from $N_k$(x), through the shake function. As described in the neighborhood change function, the change is applied if and only if an improvement occurs. The algorithm is performing iteratively until a termination condition (for example the conditions suggested by Hansen and Mladenovic) is met. The structure of RVNS is the following:

**function** RVNS $(x, k_{max}, t_{max})$;

  **repeat**

    $k = 1$;

    **repeat**

      $x' \leftarrow Shake(x,k)$;

      $NeighborhoodChange(x,x',k)$;

    **until** $k = k_{max}$;

    $t \leftarrow CpuTime()$

  **until** $t > t_{max}$;

**end function**

[Hansen et al. (2008)][21]

In this case, the time condition used for example is ($t_{max}$). Hansen et al. [2008][21] also noted: "RVNS is useful in very large instances, for which local search is costly. It has been observed that the best value for the parameter $k_{max}$ is often 2. In addition, the maximum number of iterations between two improvements is usually used as a stopping condition. RVNS is akin to a Monte-Carlo method, but is more systematic (see, for example, Mladenovic et al.(2003b)) where the results obtained by RVNS were 30% better than those of the Monte-Carlo method in solving a continuous min−max problem)."

Basic VNS (BVNS)

The method that combines the deterministic and stochastic approach of changing neighborhoods is basic VNS. [Hansen et al. (2009)][22] After an initial solution is randomly selected, shake function is applied, followed by local search (best improvement) descending, which is the deterministic part of the process. Furthermore, the last step of the process is neighborhood change, which iterates until a termination condition is met.

**function** BVNS ($p_{max}, t_{max}$);
  $S \leftarrow InitialSolution()$;
  **repeat**
    $p \leftarrow 1$;
    **while** $p \leqslant p_{max}$ **do**
      $S' \leftarrow Shake(S, p)$ /*Shaking*/;
      $S'' \leftarrow LocalSearch(S')$ /*Local search*/;
      $p \leftarrow p + 1$ /*Next neighborhood*/;
      **if** $S''$ *is better than* $S$ **then**
        $S \leftarrow S''$; $p \leftarrow 1$ /*Make a move*/;
      **end if**
    **end while**
    $t \leftarrow CpuTime()$
  **until** $t > t_{max}$;
  **return** $S$;
**end function**

[Amirgaliyeva et al. (2017)][3]

The above function shows the structure of this process.

General VNS (GVNS)

GVNS is similar to VNS method. The only difference found (as presented in the function below) is in the local search, where local search of VNS is replaced with VND. "Using this general VNS (VNS/VND) approach has led to the most successful applications reported." [Hansen et al. (2009)][22]

**function** GVNS $(x, k'_{max}, k_{max}, t_{max})$;

  **repeat**

    $k \leftarrow 1$;

    **repeat**

      $x' \leftarrow Shake(x,k)$;

      $x'' \leftarrow VND(x', k'_{max})$;

      $NeighborhoodChange(x, x'', k)$;

    **until** $k = k_{max}$;

    $t \leftarrow CpuTime()$

  **until** $t > t_{max}$;

**end function**

[Hansen et al. (2008)][21]

Skewed VNS (SVNS)

As mentioned by Hansen et al.[2008][21], "The skewed VNS (SVNS) method addresses the problem of exploring valleys far from the incumbent solution. Indeed, once the best solution in a large region has been found, it is necessary to go some way to obtain an improved one." SVNS does not implement the common neighborhood change function but a specific one that allows a current solution to move to the next (which is randomly chosen) only if it is in a distant neighborhood and of similar quality. SVNS includes the "KeepBest" and "NeighborhoodChangeS" functions, which are introduced below, along with the SVNS function.

**function** KEEPBEST($x, x'$);

  **if** $f(x') < f(x)$ **then**

    $x \leftarrow x'$;

  **end if**

  **return** $x$;

**end function**

**function** NEIGHBORHOODCHANGES($x, x'', k, \alpha$);

  **if** $f(x'') - \alpha\rho(x, x'') < f(x)$ **then**

    $x \leftarrow x''$; $k \leftarrow 1$;

  **else**

    $k \leftarrow k + 1$;

  **end if**

**end function**

**function** SVNS $(x, k_{max}, t_{max}, \alpha)$;

**repeat**

$k \leftarrow 1$; $x_{best} \leftarrow x$;

**repeat**

$x' \leftarrow Shake(x,k)$;

$x'' \leftarrow FirstImprovement(x')$;

$KeepBest(x_{best}, x)$;

$NeighborhoodChangeS(x, x'', k, \alpha)$;

**until** $k = k_{max}$;

$x \leftarrow x_{best}$;

$t \leftarrow CpuTime()$

**until** $t > t_{max}$;

**end function**

[Hansen et al. (2008)][21]

Variable Neighborhood Decomposition Search (VNDS)

VNDS is a variant of BVNS and the main concept of this algorithm is the decomposition of the problem. Having the awareness that for BVNS demanding to solve large-scale problem, this variant is considered as an improved approach. An additional parameter is added, $t_d$, and represents the running time given for solving decomposed (smaller sized) problems by VNS. "For ease of presentation, but without loss of generality, we assume that the solution $x$ represents the set of some elements. In Step 4 we denote with $y$ a set of $k$ solution attributes present in $x'$ but not in $x (y = x' \backslash x)$. In Step 5 we find the local optimum $y'$ in the space of $y$; then we denote with $x''$ the corresponding solution in the whole space $S (x'' = (x' \backslash y) \cup y')$. We notice that exploiting some boundary effects in a new solution can significantly improve the solution quality. This is why, in Step 6, we find the local optimum $x'''$ in the whole space $S$ using $x''$ as an initial solution. If this becomes time-consuming, then at least a few local search iterations

should be performed." [Hansen et al. (2009)][22]

---

**function** VNDS $(x, k_{max}, t_{max}, t_d)$;

  **repeat**

    $k \leftarrow 2$;

    **repeat**

      $x' \leftarrow Shake(x,k)$; $y \leftarrow x' \setminus x$;

      $y' \leftarrow VNS(y,k,t_d)$; $x'' = (x' \setminus y) \cup y'$;

      $x''' \leftarrow FirstImprovement(x'')$;

      $NeighborhoodChange(x,x''',k)$;

    **until** $k = k_{max}$;

  **until** $t > t_{max}$;

**end function**

---

[Hansen et al. (2009)][22]

Parallel VNS (PVNS)

Parallelizing algorithms is normally used for decreasing computational time or increasing the search space. VNS and its variants are included in the methods that can be parallelized and are expected to increase their effectiveness. "Several ways of parallelizing VNS have recently been proposed for solving the p-Median problem. In Garcia-Lopez et al.[2002] three of them are tested: (i) parallelize local search; (ii) augment the number of solutions drawn from the current neighborhood and make a local search in parallel from each of them and (iii) do the same as (ii) but update the information about the best solution found. The second version gives the best results. It is shown in Crainic et al. [2004] that assigning different neighbourhoods to each processor and interrupting their work as soon as an improved solution is found gives very good results. The best-known solutions have been found on several large

instances taken from TSP-LIB Reinelt [1991]. Three Parallel VNS strategies are also suggested for solving the Travelling purchaser problem in Ochi et al. [2001]. See Moreno-Pérez et al. [2005] for details." [Hansen et al. (2009)][22]

Primal-Dual VNS (PD-VNS)

Commonly, in heuristic methods, we are no aware of the optimal solution, therefore, we cannot estimate its difference with the current solution. Whether a lower bound on the objective function could be found, through relaxing the integrality condition on the primal variables, we could determine the performance of the algorithm. Nevertheless, for large scale problems, it is not assured that standard commercial solvers could achieve a solution. Hansen et al. [2009][22] suggest to solve heuristically dual relaxed problems. "The next problem arises if we want to reach an exact solution within a Branch and bound framework, since having the approximate value of the relaxed dual does not allow us to branch easily, e.g., by exploiting complementary slackness conditions. Thus, the exact value of the dual is necessary. In Primal-dual VNS (PD-VNS) [Hansen et al. (2007a)] one possible general way to attain both the guaranteed bounds and the exact solution is proposed." The steps of the algorithms are included in the figure bellow:

---

**function** PD-VNS $(x, k'_{max}, k_{max}, t_{max})$;

$BVNS(x, k'_{max}, k_{max}, t_{max})$ /*Solve primal by VNS*/ ;

$DualFeasible(x, y)$ /*Find (infeasible) dual such that $f_P = f_D$*/;

$DualVNS(y)$ /*Use VNS to decrease infeasibility*/;

$DualExact(y)$ /*Find exact (relaxed) dual*/;

$BandB(x, y)$ /*Apply branch-and-bound method*/;

**end function**

---

[Hansen et al. (2009)][22]

Variable Neighborhood Formulation Space Search (VN-FSS)

A different approach to the problem formulation is introduced in VN-FSS algorithm by Hansen et al. [2009][22]. Instead of one specific formulation, search could be applied to different formulations and jump from one to another. Certainly, we refer to cases were these formulations can be created. As the authors mention, each formulation should lend itself to some traditional search method, its "local search" which works totally within this formulation, and yields a final solution when started from some initial solution. Any solution found in one formulation should easily be translatable to its equivalent in any other formulation. We may then move from one formulation to another, using the solution resulting from the former's local search as an initial solution for the latter's local search. Such a strategy will, of course, be useful only in situations where local searches in different formulations behave differently. This idea was recently investigated in Mladenovic et al. [2005][31] using an approach which systematically changes the formulations for solving circle packing problems (CPP). It is shown there that the stationary point of a non-linear programming formulation of CPP in Cartesian coordinates is not necessarily also a stationary point in a polar coordinate system. A method Reformulation Descent (RD) is suggested which alternates between these two formulations until the final solution is stationary with respect to both. The results obtained were comparable with the best known values, but they were achieved some 150 times faster than by an alternative single formulation approach. In the same paper, the idea suggested above of Formulation space search (FSS) is also introduced, using more than two formulations. Some research in this direction has been reported in Mladenovic [2005][31], Plastria et al. [2005][33], Hertz et al. [2008][24].

**function** FORMULATIONCHANGE $(x, x', \phi, \phi', l)$;

**if** $f(\phi', x') < f(\phi, x)$ **then**

$\phi \leftarrow \phi'$; $x \leftarrow x'$ $l \leftarrow l_{min}$;

**else**

$l \leftarrow l + l_{step}$;

**end if**

**end function**

**function** VNFSS $(x, \phi, l_{max})$;

**repeat**

$l \leftarrow 1$ /*Initialize formulation in $F$*/ ;

**while** $l \leqslant l_{max}$ **do**

*ShakeFormulation*$(x, x', \phi, \phi', l)$ /*$(\phi', x') \in (N_l(\phi), N(x))$ at random*/;

*FormulationChange*$(x, x', \phi, \phi', l)$ /*Change formulation*/;

**end while**

**until** *some stopping condition is met*;

**end function**

[Hansen et al. (2009)][22]

# 4 Literature Review

Since the 1990s decade, there have been reports of attempts to solve the ALS problem, as it constitutes a real-life problem with gradually increasing complexity. Beyond the analytical methods, there have also been used heuristic and metaheuristic procedures to seek efficient solutions to acceptable search time for this NP-hard problem. Although exact methods can provide optimal solutions, their computational time increases exponentially as the size of the problem increases. This makes them suitable only for small/medium sized problems.

Although FCFS is the most straightforward and frequently used method for complex landing scheduling scenarios, it does not always provide the optimal solution. Amrahov et al. [2011][4], presented a greedy algorithm and the experimental results show that swapping the order of landings to decrease the delay leads to a 20% increase in the landing rate, compared to FCFS in static cases. Abela et al. [1995][2] described two methods for solving the ALS problem, an approximate solution approach using a Genetic Algorithm (GA) and an exact, BB algorithm based on an exact formulation of the problem as a 0-1 mixed integer programming (MIP) problem. Comparing these methods, BB required less computational time in small cases (up to 15 aircraft) but GA had better results in larger problems.

Beasley et al. [2000][5], presented a mixed-integer 0-1 formulation of the problem for the single runway case and extended the formulation to the multiple runway case. They introduced the benchmark used in this thesis for up to 50 aircrafts and four runways and presented computational results for a LP-based tree search and a heuristic procedure. In 2004, Beasley et al.[6] introduced larger benchmarks, up to 500 aircrafts and 5 runways, and three solution approaches, one optimal and two heuristic, for the displacement problem.

Hu & Paolo [2008][26] used GA based on a binary representation of arriving queues, instead of permutation-representation-based GAs for aircraft arrival sequencing and scheduling. Rather than using the order and/or arriving time of each aircraft in the queue to construct chromosomes for GAs, the authors used the neighboring rela-

tionship between each pair of aircraft, and the resulted chromosome is a 0-1-valued matrix, adopting a highly efficient uniform crossover operator, which is normally not applicable to those permutation representations. Comparative experiments proved the advantages of the applied GA against permutation-representation-based GAs. Bing & Wen [2010][9] introduced an improved artificial fish swarm algorithm (IAFSA), including a mutation operator. The sequence problem of landing aircraft is solved, and the simulation result shows that the IAFSA of ALS can decrease the total delay time by 24.1% compared to the FCFS, for the multiple runway case.

In 2013, Sama et al.[38] introduced a rolling horizon framework to manage busy traffic situations with a large number of delayed aircrafts and compared a Branch and Bound (BB) algorithm with a FCFS rule on practical size instances from Roma Fiumicino and Milano Malpensa airports, in Italy. Computational results demonstrate that, regarding the rolling horizon approach, BB achieved better results in both static and dynamic cases, requiring also less scheduling changes on flights. Salehipour et al. [2013][37] designed a hybrid metaheuristic applying simulated annealing framework, with the objective of minimizing the total deviation of landing time from the target time. The authors applied the proposed approach up to 500 aircrafts and 5 runways, a set of 13 instances introduced by Beasley et al. The computational results demonstrate that the proposed algorithm can obtain the optimal solution for smaller instances (up to100 aircrafts), and also it is capable of finding high quality and comparable solutions for the problems with up to 500 aircrafts and 5 runways in a short time.

Mokhtarimousavi et al. [2014][32] applied two metaheuristic algorithms, multi-objective GA and multi-objective Particle Swarm Optimization (PSO) Algorithm, to solve the ALS problem and determine the landing sequence for a group of 20 aircrafts. Moreover, he shows that the obtained sequence does not follow FCFS law for sequencing. Bencheikh et al. [2016][7] study the dynamic version of the ALP when new aircrafts appear over time and propose a Memetic Algorithm (MA) combining an Ant Colony (ACO) algorithm and a local heuristic. The choice of the ACO algorithm resides in its constructive aspect. The algorithm has been tested on the instances by Beasley et al. as well, involving 10-50 aircraft and 1-5 runways and the results are

compared with three approaches presented by Beasley et al.[2004][6].

Girish [2016][15] proposed a hybrid particle swarm optimization algorithm in a rolling horizon framework to solve the ALS problem. A schedule generation procedure is presented in the paper that generates optimal landing schedules for the given landing sequence and runway allocations. The performance of the proposed algorithm is evaluated on Beasley's benchmark instances involving up to 500 aircrafts and 5 runways. Computational results reveal that the proposed algorithm is effective in solving the problem in short computational time. Finally, Abduallah et al. [2017][1] proposed a Harmony Search (HS) algorithm to solve the multiple runways ALS problem. The performance of the proposed algorithm is evaluated also on the 13 benchmark instances by Beasley et al. The results show that the proposed algorithm works considerably well on small-sized instances.

# 5 Methodology

## 5.1 Matheuristic approach for the ALS problem

The purpose of this section is to present the implementation of the matheuristic approach for the ALS problem, as well as the application and computational results on benchmarks from OR Library, since the background and theoretical details of the topic are mentioned analytically in previous sections. Moreover, this section includes the evaluation of the implementation, a comparison between the results along with the results of the Gurobi optimization solver for the mathematical model and some conclusions resulting from this study. Finally, R programming language has been used for a statistical analysis evaluating the methods applied and their results.

For this matheuristic approach, 3 variants of the VNS metaheuristic are used and compared (BVNS, VND and RVNS) for the sequence problem of the aircrafts. After the sequence is found, interior- point method from mathematical programming is used for finding the landing times of the aircrafts.

As already described in section 3, VNS algorithm includes the following steps, which are repeated until the stopping condition is met:

1. Shaking
2. Local search (improvement)
3. Neighborhood change

Firstly, an initial feasible solution is required, which is obtained using the FCFS method, a simple and intuitive method, as it can always provide a feasible solution. The sequence of the aircrafts' landings is created by pre-ordering them according to non-decreasing arrival times. In the sequence vector, the i-th place contains the aircraft that is assigned to land i-th.

The next figure presents an example of FCFS method:

| Aircraft sequence | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Arrival times | 54 | 120 | 14 | 21 | 35 | 45 | 49 | 51 | 60 | 85 |

| Aircraft sequence | 3 | 4 | 5 | 6 | 7 | 8 | 1 | 9 | 10 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|
| Arrival times | 14 | 24 | 35 | 45 | 49 | 45 | 54 | 60 | 85 | 120 |

**Figure 15:** Example of FCFS method

Since the sequence of the landings has been defined and the model that is presented in subsection 2.3 is linear, interior-point method can be applied to provide the optimal landing time between the time window for each aircraft. An example of this process is presented in the following figure.

| Aircraft sequence | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Earliest landing times | 129 | 195 | 89 | 96 | 110 | 120 | 124 | 126 | 135 | 160 |
| Latest landing times | 559 | 744 | 510 | 521 | 555 | 576 | 577 | 573 | 591 | 657 |
| Lading times | 157 | 258 | 98 | 106 | 118 | 126 | 134 | 142 | 172 | 180 |

**Figure 16:** Example of landing times occurring after interior-point method application

The representation of the solution consists of a vector which contains at the i-th place the assigned lading time of the i-th aircraft, as shown in the next figure.

| Aircrafts (i) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Landing times ($t_i$) | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ | $t_{10}$ |

**Figure 17:** Solution representation

Regarding the local search, three neighborhood structures are used: relocate, 2-opt and swap. The same structures are used in the shaking phase when a local minimum is found, in order to escape from it and a better solution can be achieved. The direction that is being followed is first improvement, as best improvement can be very costly in terms of time, especially for large instances. More specifically, as soon as a better solution is found, it is assigned as the new solution. The stopping conditions applied are 5 iterations between two improvements, as well as a computational time limit of 7200 seconds (2 hours), to prevent excessive runtimes.

Relocate

Relocate method constitutes a straightforward and simple to operate neighborhood structure. Its main operation concerns the selection of an element of the solution vector and its relocation in the position of one of the remaining elements. The elements that are located between the initial and the final position of the aforementioned element are shifted by one position in the opposite direction of it.

The structure of a solution after a relocate move is the following:

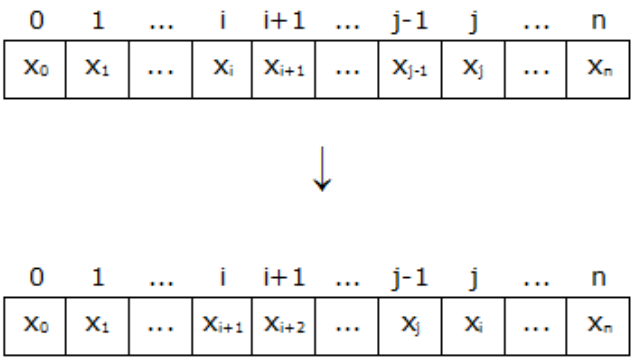


**Figure 18:** Structure of a solution after a relocate move

2-opt

2-opt is one of the most commonly used local search operators, proposed by G.A. Croes 1955. This method consists of dividing the solution vector in two points and then, the element sequence between these specific points is being reversed.

The following figure includes the structure of a solution after a 2-opt move.

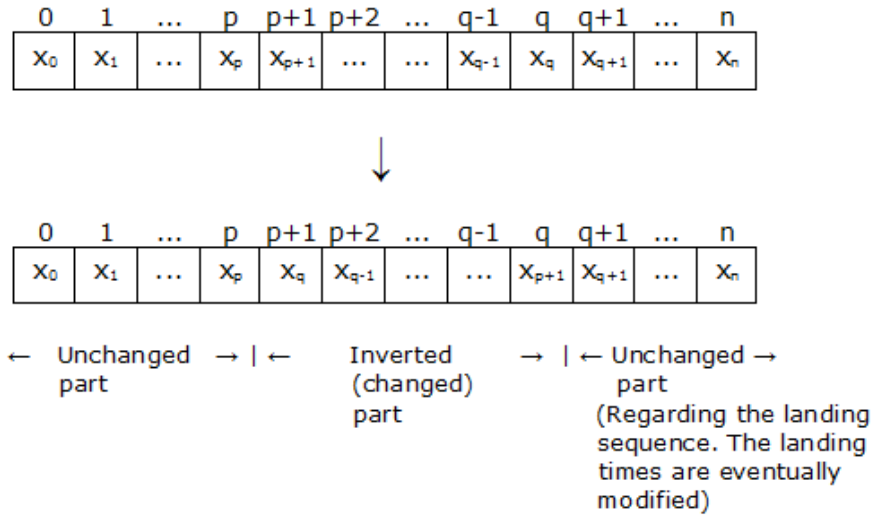


**Figure 19:** Structure of a solution after a 2-opt move

Swap

Swap is another simple to operate neighborhood structure. Two elements of the solution vector are selected and swap their positions and consequently, the resulting solution vector will diverge from the initial vector by only two elements where one replaced the other.

The structure of a solution after a swap move is presented in the figure below:

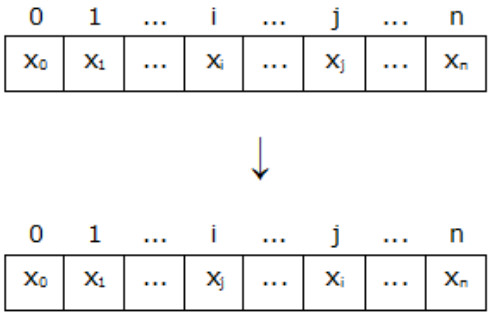


**Figure 20:** Structure of a solution after a swap move

## 5.2 VNS variants implementation for the ALS problem

The VNS variants that are implemented and compared are BVNS, VND and RVNS. Below are summarized the steps for these algorithms, as they are also described analytically in section 3:

BVNS

1. Find an initial feasible solution
2. Shaking
3. Local search

4. Move or not
5. Repeat from step 2 until the stopping condition is met

VND

1. Find an initial feasible solution
2. Local search
3. Move or not
4. Repeat from step 2 until the stopping condition is met

RVNS

1. Find an initial feasible solution
2. Shaking
3. Move or not
4. Repeat from step 2 until the stopping condition is met

Regarding the initial solution, neighborhood structures, direction and stopping condition, the same techniques are implemented for all the algorithms.

## 5.3 Computational study

### 5.3.1 Benchmarks of OR Library

OR Library[41] is a collection of test datasets for a variety of Operations Research (OR) problems and more specifically, it contains 13 benchmarks for the ALS problem, which are presented in the next table. They were introduced by Beasley et al.[5][6] and are commonly used in literature. The datasets consist of eight small sets, involving from 10 to 50 airplanes, and five large sets, involving from 100 to 500 airplanes. Although optimal results for the small instances are found, for the large instances optimal results do not yet exist.

| File name | Number of aircrafts |
|---|---|
| Airland1 | 10 |
| Airland2 | 15 |
| Airland3 | 20 |
| Airland4 | 20 |
| Airland5 | 20 |
| Airland6 | 30 |
| Airland7 | 44 |
| Airland8 | 50 |
| Airland9 | 100 |
| Airland10 | 150 |
| Airland11 | 200 |
| Airland12 | 250 |
| Airland13 | 500 |

**Table 1:** ALS files in OR Library

The format of these files is the following:

> number of planes, freeze time,
> - for each plane i (i=1,...,p):
>
> appearance time, earliest landing time, target landing time,
> latest landing time, penalty cost per unit of time for landing before target,
> penalty cost per unit of time for landing after target
> - for each plane j (j=1,...p):
>
> separation time required after i lands before j can land

Some examples of OR-library's test datasets are the following:

- _airland1.txt

```
10 10
 54 129 155 559 10.00 10.00
 99999 3 15 15 15 15 15 15
 15 15
 120 195 258 744 10.00 10.00
 3 99999 15 15 15 15 15 15
 15 15
 14 89 98 510 30.00 30.00
 15 15 99999 8 8 8 8 8
```

```
8 8
21 96 106 521 30.00 30.00
15 15 8 99999 8 8 8 8
8 8
35 110 123 555 30.00 30.00
15 15 8 8 99999 8 8 8
8 8
45 120 135 576 30.00 30.00
15 15 8 8 8 99999 8 8
8 8
49 124 138 577 30.00 30.00
15 15 8 8 8 8 99999 8
8 8
51 126 140 573 30.00 30.00
15 15 8 8 8 8 8 99999
8 8
60 135 150 591 30.00 30.00
15 15 8 8 8 8 8 8
99999 8
85 160 180 657 30.00 30.00
15 15 8 8 8 8 8 8
8 99999
```

- _airland2.txt

```
15 10
 54 129 155 559 10.00 10.00
 99999 3 15 15 15 15 15 15
 15 15 3 3 15 15 3
 115 190 250 732 10.00 10.00
 3 99999 15 15 15 15 15 15
 15 15 3 3 15 15 3
 9 84 93 501 30.00 30.00
 15 15 99999 8 8 8 8 8
 8 8 15 15 8 8 15
 14 89 98 509 30.00 30.00
 15 15 8 99999 8 8 8 8
 8 8 15 15 8 8 15
 25 100 111 536 30.00 30.00
 15 15 8 8 99999 8 8 8
 8 8 15 15 8 8 15
 32 107 120 552 30.00 30.00
 15 15 8 8 8 99999 8 8
 8 8 15 15 8 8 15
 34 109 121 550 30.00 30.00
 15 15 8 8 8 8 99999 8
 8 8 15 15 8 8 15
```

34 109 120 544 30.00 30.00

15 15 8 8 8 8 8 99999

8 8 15 15 8 8 15

40 115 128 557 30.00 30.00

15 15 8 8 8 8 8 8

99999 8 15 15 8 8 15

59 134 151 610 30.00 30.00

15 15 8 8 8 8 8 8

8 99999 15 15 8 8 15

191 266 341 837 10.00 10.00

3 3 15 15 15 15 15 15

15 15 99999 3 15 15 3

176 251 313 778 10.00 10.00

3 3 15 15 15 15 15 15

15 15 3 99999 15 15 3

85 160 181 674 30.00 30.00

15 15 8 8 8 8 8 8

8 8 15 15 99999 8 15

77 152 171 637 30.00 30.00

15 15 8 8 8 8 8 8

8 8 15 15 8 99999 15

201 276 342 815 10.00 10.00

3 3 15 15 15 15 15 15

15 15 3 3 15 15 99999

### 5.3.2 Computational experiments and results

#### 5.3.2.1 Matheuristics results

In this subsection are presented the results of the computational experiments, which were executed aiming to evaluate the algorithms described in the previous subsections, implemented in Python 3.7.3 programming language. For this implementation, "Spyder" development environment has been used, which is part of the "Anaconda distribution" and offers a variety of functionalities and capabilities. "linprog" package from "SciPy" library was used for applying the interior-point method. The computational experiments were performed on a computer running Windows 10 64bit with an Intel Core i7-8665U 1.90GHz with 16GB RAM.

The experimental results are separated and presented in two different tables, the first one with the small datasets, for which the optimal value of the objective function and landing schedule has been found, and the second one with the large datasets, where the results are compared with the best value that has been found by Girish [2016][15]. The first three columns contain the file name, the number of aircrafts (N) and the optimal/best value for each dataset. The rest columns provide for each method (FCFS, BVNS, VND and RVNS) the best (min) value (z), based on the objective function, achieved from the experiments, the CPU time (t) and the percentage error (Error %) regarding the difference between the solution found and the optimal solution, as a percentage of the optimal solution, which is calculated as follows:

$$Percentage\ error = 100 * \frac{z - optimal}{optimal}$$

Similarly, for the second table the percentage error is calculated as follows:

$$Percentage\ error = 100 * \frac{z - best}{best}$$

| File name | N | Optimal | FCFS | | | BVNS | | | VND | | | RVNS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | z | t (secs) | Error % | z | t (secs) | Error % | z | t (secs) | Error % | z | t (secs) | Error % |
| Airland1 | 10 | 700 | 1280 | 0.01 | 82.86 | **700** | 0.15 | 0 | **700** | 1.43 | 0 | **700** | 0.09 | 0 |
| Airland2 | 15 | 1480 | 1740 | 0.02 | 17.56 | **1480** | 3.11 | 0 | **1480** | 4.14 | 0 | **1480** | 0.13 | 0 |
| Airland3 | 20 | 820 | 1790 | 0.01 | 118.29 | **820** | 22.46 | 0 | **820** | 8.73 | 0 | **820** | 0.38 | 0 |
| Airland4 | 20 | 2520 | 4490 | 0.01 | 78.17 | **2520** | 67.85 | 0 | **2520** | 21.50 | 0 | **2520** | 1.22 | 0 |
| Airland5 | 20 | 3100 | 6410 | 0.01 | 106.77 | **3100** | 19.85 | 0 | **3100** | 20.32 | 0 | **3100** | 2.59 | 0 |
| Airland6 | 30 | 24442 | **24442** | 0.01 | 0 | **24442** | 3.68 | 0 | **24442** | 0.19 | 0 | **24442** | 0.18 | 0 |
| Airland7 | 44 | 1550 | **1550** | 0.01 | 0 | **1550** | 29.24 | 0 | **1550** | 0.31 | 0 | **1550** | 0.44 | 0 |
| Airland8 | 50 | 1950 | 17980 | 0.02 | 822.05 | **3975** | 1926.37 | 103.85 | 3985 | 2289 | 104.36 | 6425 | 56.04 | 229.49 |

**Table 2:** Computational results of small test datasets

As the results of the small datasets show, although FCFS method can provide a feasible solution in very short time, the quality is limited apart from Airland6 and Airland7, where the optimal landing sequence is the FCFS one. Regarding the rest algorithms, they can all achieve the optimal solution for all the instances except from the last one with 50 aircrafts. RVNS has shown better performance, except from Airland8, but it is worth to be mentioned that since is is based in stochastic computations, the results could exhibit a wide variation. BVNS demonstrated better performance than VND only for three instances, with Airland8 being the most critical where BVNS achieved the best solution of the three algorithms in less computational time.

| File name | N | Best | FCFS | | | BVNS | | | VND | | | RVNS | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | z | t (secs) | Error % | z | t (secs) | Error % | z | t (secs) | Error % | z | t (secs) | Error % |
| Airland9 | 100 | 5611.70 | 17602.63 | 0.06 | 213.68 | 9818.93 | 6970.49 | 74.97 | **7838.85** | 3229.94 | 39.69 | 8202.95 | 240.35 | 46.17 |
| Airland10 | 150 | 12292.20 | 27201.83 | 0.14 | 121.29 | 23800.50 | 6391.74 | 93.62 | 21788.99 | 6835.34 | 77.26 | **17626.07** | 1266.55 | 43.39 |
| Airland11 | 200 | 12418.32 | 33405.36 | 0.34 | 169 | 31606.85 | 5137.31 | 154.52 | 28156.58 | 7012 | 126.73 | **19824.06** | 5483.33 | 59.63 |
| Airland12 | 250 | 16122.18 | 43351.67 | 0.55 | 168.89 | 41664.05 | 6027.86 | 158.43 | 40713.93 | 7191.40 | 152.53 | **30422.52** | 7122.23 | 88.7 |
| Airland13 | 500 | 37064.11 | 91991.72 | 2.82 | 148.2 | 91323.19 | 2289.12 | 146.39 | 89749.32 | 6661.51 | 142.15 | **82620.80** | 1216.31 | 122.91 |

**Table 3:** Computational results of large test datasets

Regarding the large datasets, VND has shown better performance for Airland9. Furthermore, for the last four datasets, RVNS accomplished the best results, as expected, since these are the largest datasets and local search can be significantly time-consuming.

#### 5.3.2.2 Gurobi Optimizer results

The Gurobi Optimizer is a commercial optimization solver for linear programming, quadratic programming, quadratically constrained programming, mixed integer linear programming, mixed-integer quadratic programming, and mixed-integer quadratically constrained programming. Gurobi supports a variety of programming and modeling languages including C, C++, Java, Python, MATLAB, R etc. and includes a number of features to support the building of optimization models.

In the context of this study, the mathematical model presented in section 2.3 was developed in Python programming language using Gurobi Optimizer 8.1.1, also in "Spyder" environment as the previous algorithms. The table below presents the results of the mathematical model using Gurobi Optimizer, achived within the computational time limit of 7200 seconds (2 hours).

| File name | N | Optimal | Gurobi | | |
|---|---|---|---|---|---|
| | | | z | t (secs) | Error % |
| Airland1 | 10 | 700 | 700 | 0.26 | 0 |
| Airland2 | 15 | 1480 | 1480 | 0.85 | 0 |
| Airland3 | 20 | 820 | 820 | 0.40 | 0 |
| Airland4 | 20 | 2520 | 2520 | 5.50 | 0 |
| Airland5 | 20 | 3100 | 3100 | 13.75 | 0 |
| Airland6 | 30 | 24442 | 24442 | 1.82 | 0 |
| Airland7 | 44 | 1550 | 1550 | 2.50 | 0 |
| Airland8 | 50 | 1950 | 1950 | 4.62 | 0 |
| Airland9 | 100 | 5611.70 | 5635.23 | 5985 | 0.42 |
| Airland10 | 150 | 12292.20 | 13130.19 | 6881 | 6.81 |
| Airland11 | 200 | 12418.32 | 13032.62 | 2101 | 4.95 |
| Airland12 | 250 | 16122.18 | 16677.01 | 1054 | 3.44 |
| Airland13 | 500 | 37064.11 | 40193.09 | 7168 | 8.44 |

**Table 4:** Computational results of Gurobi Optimizer

The results of the implementation of the mathematical model with the Gurobi optimization solver showed that Gurobi returns quality solutions when there is enough computational time available, but is not so effective in short periods of time.

The following graph presents the results of the followed methods for all the datasets.

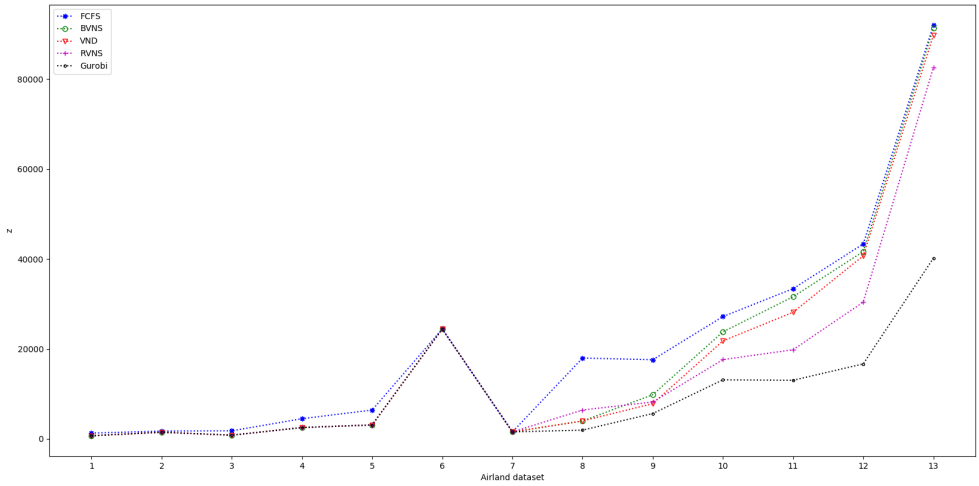


**Figure 21:** The results of FCFS, BVNS, VND, RVNS and Gurobi

### 5.3.3 Statistical analysis of the metaheuristic methods

In this subsection we present a statistical analysis for comparing the metaheuristic methods (BVNS, VND, RVNS) applied for the matheuristic approach. During this analysis, R programming language has been used. The benchmarks are separated in two sets for testing: the first set contains the datasets Airland1-Airland7 and the second set contains the remaining datasets. In the first set, since all the methods have achieved the optimal result, the algorithms are tested regarding the computational time. In the second set, we test the result achieved from these algorithms for each dataset.

The followning table has been used for the first test:

| File name | Method | | |
|---|---|---|---|
| | **BVNS** | **VND** | **RVNS** |
| Airland1 | 0.15 | 1.43 | 0.09 |
| Airland2 | 3.11 | 4.14 | 0.13 |
| Airland3 | 22.46 | 8.73 | 0.38 |
| Airland4 | 67.85 | 21.50 | 1.22 |
| Airland5 | 19.85 | 20.32 | 2.59 |
| Airland6 | 3.68 | 0.19 | 0.18 |
| Airland7 | 29.24 | 0.31 | 0.44 |

**Table 5:** First test set

After applying the Shapiro-Wilk test for normality, the results shown that the null hypothesis (normal distribution) cannot be accepted. Therefore, we used the non-parametric Friedman test instead of the parametric repeated measures ANOVA, since we have three methods to compare. The results of this test are presented in the following figure.

```
        Friedman rank sum test

data:  matrix1
Friedman chi-squared = 8, df = 2, p-value = 0.01832
```

**Figure 22:** Friedman test (1)

According to the null hypothesis, all the group medians are equal and there is not a difference between the methods. The p-value resulting from this test is 0.01832 < 0.05, it is therefore plausible that the three methods have statistically significant different medians.

For further research, we applied the non-parametric Wilcoxon test, with the same

null hypothesis as Friedman test, separating the aforementioned methods in groups of two. The results are shown in the table below.

| Methods | p-value |
|---|---|
| BVNS-VND | 0.2188 |
| VND-RVNS | 0.04688 |
| BVNS-RVNS | 0.01563 |

**Table 6:** Wilcoxon test (1)

According to the above results, BVNS and VND have similar distributions and so the same mean whereas RVNS mean is probably different from the first two.

The same conclusions derived by the following descriptive statistics. RVNS has the smalest median, range etc. and consequently it results in being more effective than the other methods.

Descriptive Statistics for bvns1

| NAME | VALUE | TYP.ERROR |
|---|---|---|
| number of observations | 7 | |
| number of missing values | 0 | |
| mean | 20.905714 | 8.89217 |
| 0.05 trimmed mean | 20.905714 | |
| flunctuation | 553.494829 | 318.843115 |
| coefficient of variability | 1.165552 | |
| min | 0.15 | |
| max | 67.85 | |
| median | 19.85 | |
| Q1 | 3.395 | |
| Q3 | 25.85 | |
| wide | 67.7 | |
| asymmetry | 1.166064 | 0.92582 |
| kurtosis (corrected) | 0.32287 | 1.85164 |
| check Shapiro-Wilk(p-value) | 0.095092 | |

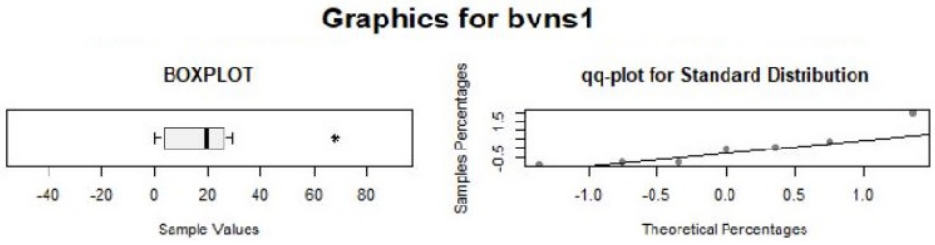


**Figure 23:** Descriptive statistics and graphics for BVNS

Descriptive Statistics for vnd1

| NAME | VALUE | TYP.ERROR |
|---|---|---|
| number of observations | 7 | |
| number of missing values | 0 | |
| mean | 8.088571 | 3.493896 |
| 0.05 trimmed mean | 8.088571 | |
| flunctuation | 85.451181 | 27.357692 |
| coefficient of variability | 1.183661 | |
| min | 0.19 | |
| max | 21.5 | |
| median | 4.14 | |
| Q1 | 0.87 | |
| Q3 | 14.525 | |
| wide | 21.31 | |
| asymmetry | 0.66598 | 0.92582 |
| kurtosis (corrected) | -1.282502 | 1.85164 |
| check Shapiro-Wilk(p-value) | 0.050761 | |

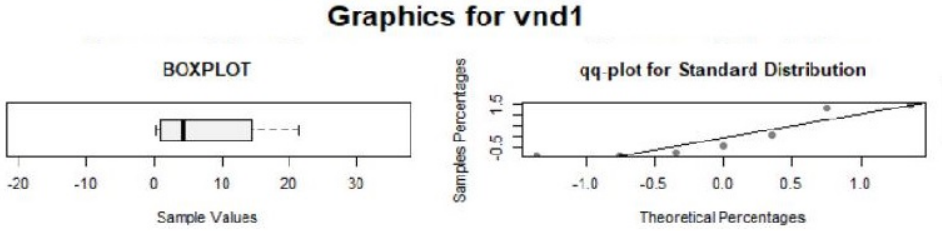


**Figure 24:** Descriptive statistics and graphics for VND

Descriptive Statistics for rvns1

| NAME | VALUE | TYP.ERROR |
|---|---|---|
| number of observations | 7 | |
| number of missing values | 0 | |
| mean | 0.718571 | 0.344255 |
| 0.05 trimmed mean | 0.718571 | |
| flunctuation | 0.829581 | 0.504673 |
| coefficient of variability | 1.312803 | |
| min | 0.09 | |
| max | 2.59 | |
| median | 0.38 | |
| Q1 | 0.155 | |
| Q3 | 0.83 | |
| wide | 2.5 | |
| asymmetry | 1.432217 | 0.92582 |
| kurtosis (corrected) | 0.590599 | 1.85164 |
| check Shapiro-Wilk(p-value) | 0.011054 | |

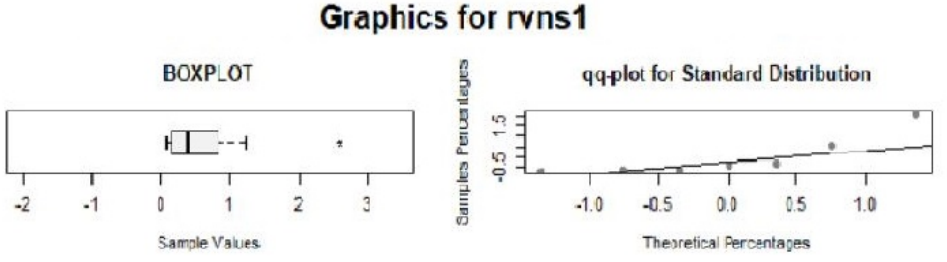


**Figure 25:** Descriptive statistics and graphics for RVNS

Regarding the datasets Airland8-Airland13, we follow the same process for testing the effectiveness of each method, based on the best values achieved. The table used for this test is the following:

| File name | Method | | |
|---|---|---|---|
| | **BVNS** | **VND** | **RVNS** |
| Airland8 | 3975 | 3985 | 6425 |
| Airland9 | 9818.93 | 7838.85 | 8202.95 |
| Airland10 | 23800.50 | 21788.99 | 17626.07 |
| Airland11 | 31606.85 | 28156.58 | 19824.06 |
| Airland12 | 41664.05 | 40713.93 | 30422.52 |
| Airland13 | 91323.19 | 89749.32 | 82620.80 |

**Table 7:** Second test set

The Shapiro-Wilk test indicates again that we cannot accept the hypothesis for normality, hence we continue with the Friedman test. The following figure shows the results of this test.

```
        Friedman rank sum test

data:  matrix2
Friedman chi-squared = 4.3333, df = 2, p-value = 0.1146
```

**Figure 26:** Friedman test (2)

The p-value resulting from this test is 0.1146 > 0.05, therefore we cannot assume that the overall medians differ. Moreover, the Wilcoxon test concludes to the same assumptions, with the p-value of each pair of methods being greater than 0.05, as shown in the table below.

| Methods | p-value |
|---|---|
| BVNS-VND | 0.0625 |
| VND-RVNS | 0.1563 |
| BVNS-RVNS | 0.09375 |

**Table 8:** Wilcoxon test (2)

# 6 Conclusions

## 6.1 Thesis overview

This thesis describes the implementation and application of a matheuristic approach as a solution method for an NP-hard problem, the static case of Aircraft Landing Scheduling problem with a single runway, using Python programming language. This approach is implemented using and comparing VNS algorithm's variants, BVNS, VND and RVNS, along with interior-point method from mathematical programming. The theoretical background and the details of this topic are analytically presented followed by the performance of the implemented algorithms, which is evaluated using 13 benchmarks from OR Library, provided by J.E. Beasley. Computational results for all the test datasets, involving from 10 to 500 airplanes, are presented and compared.

## 6.2 Future work

This thesis constitutes a first attempt to solve an NP-hard problem and especially a scheduling problem. Nevertheless, this study can certainly be subject to improvements and further extensions. The primary task consists of improving or modifying the applied procedures for reducing the runtime. For example, the initial solution can be achieved by sorting the target landing times instead of the arrival times for the initial solution, or even apply a different method for this step. Moreover, additional neighborhood structures could be used, for example 3-opt. Furthermore, the applied algorithms could be parallelized to reduce the computational time and also combined with other metaheuristics (Genetic Algorithm, PSO etc.) for a hybrid implementation.

It is further of special interest to explore the ALS problem's variants. Firstly, this study for the static case can be modified to include more than one runways and, additionally, implement these methods in a dynamic and more realistic environment, both for the single and the multiple runways case. Finally, with minor modifications, these implementations can include the take-offs to complete the realistic solution suggestion.

# References


[1] Abdullah, O.S., Abdullah, S., & Sarim, H.M. (2017). Harmony Search Algorithm for the Multiple Runways Aircraft Landing Scheduling Problem. Journal of Telecommunication, Electronic and Computer Engineering, 9, 59-65.

[2] Abela, J., Krishnamoorthy, M., Silva, A. & Mills, G. (1995). Computing Optimal Schedules for Landing Aircraft. Proceedings of the 12th National ASOR Conference, Adelaide.

[3] Amirgaliyeva, Z., Mladenovic, N., Todosijevic, R., & Urosevic, D. (2017). Solving the maximum min-sum dispersion by alternating formulations of two different problems. European Journal of Operational Research, 260(2), 444-459.

[4] Amrahov, S. E., & Alsalihe, T. A. I. (2011). Greedy algorithm for the scheduling aircrafts landings. 2011 5th International Conference on Application of Information and Communication Technologies (AICT).

[5] Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., & Abramson, D. (2000). Scheduling Aircraft Landings-The Static Case. Transportation Science, 34(2), 180-197.

[6] Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., & Abramson, D. (2004). Displacement problem and dynamically scheduling aircraft landings. Journal of the Operational Research Society, 55(1), 54-64.

[7] Bencheikh, G., Boukachour, J., & El Hilali Alaoui, A. (2016). A memetic algorithm to solve the dynamic multiple runway aircraft landing problem. Journal of King Saud University - Computer and Information Sciences, 28(1), 98-109.

[8] Bennell, J. A., Mesgarpour, M., & Potts, C. N. (2017). Dynamic scheduling of aircraft landings. European Journal of Operational Research, 258(1), 315-327.

[9] Bing, D., & Wen, D. (2010). Scheduling Arrival Aircrafts on Multi-runway Based on an Improved Artificial Fish Swarm Algorithm. 2010 International Conference on Computational and Information Sciences.

[10] Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization. ACM Computing Surveys, 35(3), 268-308.

[11] Dorigo, M. (1992), Optimization, Learning and Natural Algorithms. PhD. Politecnico di Milano.

[12] Duarte, A., Mladenovic, N., Sánchez-Oro, J., & Todosijevic, R. (2016). Variable Neighborhood Descent. Handbook of Heuristics, 1-27.

[13] D'Ariano, A., Pistelli M., and Pacciarelli, D (2012). Effectiveness of Optimal Aircraft Rerouting Strategies for the Management of Disturbed Traffic Conditions. Presented at 91st Annual Meeting of the Transportation Research Board, Washington, D.C.

[14] Fogel, L., Owens, A. & Walsh, M. (1966). Artificial intelligence through simulated evolution.

[15] Girish, B. S. (2016). An efficient hybrid particle swarm optimization algorithm in a rolling horizon framework for the aircraft landing problem, Applied Soft Computing Journal, 44.

[16] Glover, F. (1977). Heuristics for Integer Programming Using SurrogateConstraints, Decision Sciences, 8, 156-166.

[17] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res.13(5), 533-549

[18] Gomes, L.P., Diniz, V.B., & Martinhon, C.A. (2004). An Hybrid GRASP+VNS Metaheuristic for the Prize-Collecting Traveling Salesman Problem.

[19] Hansen, P., & Mladenovic, N. (1999). An Introduction to Variable Neighborhood Search. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, 433-458.

[20] Hansen, P. & Mladenovic, N. (2003). A Tutorial on Variable Neighborhood Search.

[21] Hansen, P., Mladenovic, N., & Moreno Perez, J. A. (2008). Variable neighbourhood search: methods and applications. 4OR, 6(4), 319-360.

[22] Hansen, P., Mladenovic, N., & Moreno Perez, J. A. (2009). Variable neighbourhood search: methods and applications. Annals of Operations Research, 175(1), 367-407.

[23] Hansen, P., Mladenovic, N., Todosijevic, R., & Hanafi, S. (2016). Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization, 5(3), 423-454.

[24] Hertz, A., Plumettaz, M., & Zufferey, N. (2008). Variable space search for graph coloring.Discrete AppliedMathematics,156(13), 2551{2560.

[25] Holland, J. (1975). Adaptation in natural and artificial systems, University of Michigan Press.

[26] Hu, X.-B., & Paolo, E. A. D. (2011). A Ripple-Spreading Genetic Algorithm for the Aircraft Sequencing Problem. Evolutionary Computation, 19(1), 77-106.

[27] Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1983). Optimization by Simulated Annealing, Science220(4598), 671-680.

[28] Lesaja, G. (2009). Introducing Interior-Point Methods for Introductory Operations Research Courses and/or Linear Programming Courses. The Open Operational Research Journal. 3. 1-12.

[29] Matyas, I. (1965). Random optimization. Automation and Remote Control, 26(2), 246-253.

[30] Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. Computers & Operations Research, 24(11), 1097-1100.

[31] Mladenovic, N., Plastria, F., & Urosevic, D. (2005). Reformulation descent applied to circle packing problems.Computers and Operations Research,32, 2419{2434.

[32] Mokhtarimousavi, S., Rahami, H., Saffarzadeh, M. & Piri, S. (2014). Determination of the Aircraft Landing Sequence by Two Meta-Heuristic Algorithms. International Journal of Transportation Engineering, 1(4), 271-284.

[33] Plastria, F., Mladenovic, N., & Urosevic, D. (2005). Variable neighborhood formulation space search forcircle packing. In18th mini Euro conference VNS. Tenerife, Spain.

[34] Puchinger, J. & Raidl, G. R. (2005). Combining Metaheuristics and Exact Algorithms in Combinatorial Optimization: A Survey and Classification. Lecture Notes in Computer Science, 41-53.

[35] Rastrigin, L.A. (1963). The convergence of the random search method in the extremal control of a many parameter system. Automation and Remote Control, 24(10), 1337-1342.

[36] Robbins, H. & Monro, S. (1951). A Stochastic Approximation Method. The Annals of Mathematical Statistics 22, 3, 400-407.

[37] Salehipour, A., Modarres, M., & Moslemi Naeni, L. (2013). An efficient hybrid meta-heuristic for aircraft landing problem. Computers & Operations Research, 40(1), 207-213.

[38] Sama, M., D'Ariano, A., & Pacciarelli, D. (2013). Rolling horizon approach for aircraft scheduling in the terminal control area of busy airports. Transportation Research Part E: Logistics and Transportation Review, 60, 140-155.

[39] Smith, S.F. (1980). A Learning System Based on Genetic Adaptive Algorithms. PhD. University of Pittsburgh.

[40] Yang, X.-S. (2011). Metaheuristic Optimization. Scholarpedia, 6(8): 11472.

[41] (OR Library) http://people.brunel.ac.uk/ mastjjb/jeb/info.html

[42] http://aviationknowledge.wikidot.com/aviation:atmospheric-turbulence

[43] https://centreforaviation.com/analysis/reports/beijing-airport-splits-traffic-hong-kong-seoul-shanghai-race-to-be-asias-100-million-pax-airport-295770

[44] https://en.wikipedia.org/wiki/Metaheuristic

[45] https://www.aerodefensetech.com/component/content/article/adt/features/articles/25559

[46] https://www.aia.gr/company-and-business/the-company/facts-and-figures

[47] https://www.bts.dot.gov/newsroom/2018-traffic-data-us-airlines-and-foreign-airlines-us-flights

[48] https://www.bts.dot.gov/topics/airlines-and-airports/understanding-reporting-causes-flight-delays-and-cancellations

[49] https://www.easa.europa.eu/faq/19224

[50] https://www.eurocontrol.int/sites/default/files/content/documents/sesar/recat-eu-released-september-2015.pdf

[51] https://www.flightradar24.com

[52] https://www.icao.int/MID/Documents/2017/ATM%20SG3/WP27.pdf

[53] https://www.icao.int/about-icao/Pages/default.aspx

[54] https://www.skybrary.aero/index.php/Federal_Aviation_Administration_(FAA)